KV CACHE TRANSLATION ACROSS HETEROGENEOUS LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Heterogeneous Large Language Model (LLM) systems increasingly share contexts, retrieved evidence, and multi-agent dialogue histories, yet their internal key-value (KV) caches remain model-specific and cannot be reused across architectures. Consequently, each model must repeatedly prefill or store caches for the same context, limiting the scalability of multi-model reasoning and long-context generation. We propose Mixture-of-Translators (MoT), a cache translation framework that maps context KV caches from a source LLM into the cache space of a target LLM. Unlike prior approaches that depend on a single projection path or global shared latent space, MoT uses multiple translator modules with token-level routing to capture diverse cache translations. We further introduce a Context Correction Loss that aligns the replayed target trajectory with the native target trajectory, reducing residual translation error. Our analysis reveals two competing failure modes: propagation error from early injection and correction-deficit error from late injection. We evaluate heterogeneous translation among Llama-3.2, Gemma-3, and Qwen-3, spanning substantially different architectures and KV-cache spaces. MoT achieves an average accuracy of 57.6% and F1 of 0.42. These scores recover 95.0% and 120.6% of native target performance and reach $1.5\times$ and $3.5\times$ the strongest-baseline scores, respectively. In practical case studies, MoT enables accurate heterogeneous-agent discussion with scale-invariant KV-memory behavior as the number of agents grows, while retaining near-native generation quality in long-context cache-augmented generation. These results demonstrate scalable KV-cache reuse across heterogeneous LLMs. Code and pretrained translator checkpoints are available at `https://anonymous.4open.science/r/MoT-D7E3` and `https://anonymous-hf.com/a/tkelnedcist7`, respectively.

1 INTRODUCTION

A heterogeneous Large Language Model (LLM) system faces a fundamental incompatibility: the context state computed by one model is not directly readable by another. Without translation, each model must reconstruct or store its own version of shared long contexts, retrieved evidence, dialogue histories, or agent-generated states. Thus, the key-value (KV) cache becomes more than an inference artifact; it is a model-specific memory whose reuse depends on transfer across cache spaces. Prompt caching and cache-augmented generation demonstrate the value of context reuse (OpenAI, 2026; Anthropic, 2026; Chan et al., 2025; Lu et al., 2025), but existing mechanisms remain tied to a single model or cache format.

This limitation becomes critical in emerging LLM workflows. Multi-agent reasoning repeatedly processes shared passages, questions, and dialogue histories, causing memory to grow with the number of agents (Li et al., 2023; Du et al., 2024; Wu et al., 2024), while long-context cache-augmented generation avoids repeated prefilling only within a model-specific cache representation (Chan et al., 2025; Lu et al., 2025; Gao et al., 2025). Scalable context reuse therefore requires translating internal KV states across heterogeneous models without sacrificing generation quality.

Recent work explores language-free communication through cache projection, selective KV sharing, hidden-state communication, and shared latent cache alignment (Fu et al., 2026; Shi et al., 2026; Du et al., 2026; Dery et al., 2026). However, fixed projection paths, compatibility assumptions, or shared spaces can be insufficient when source and target models differ in depth, hidden dimension,

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

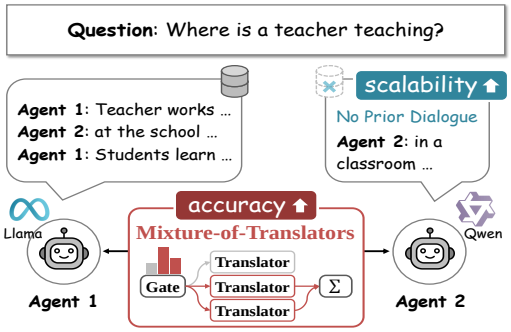


Figure 1: Mixture-of-Translators enables accurate heterogeneous-agent discussion with scalable memory reuse through translated caches.

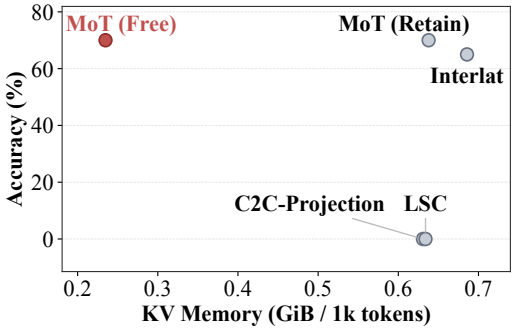


Figure 2: 10-agent performance with alternating Llama-3.2 (3B) and Qwen-3 (4B) agents and heterogeneous cache translation across turns.

attention layout, or KV-head structure. Such heterogeneity may require greater translation capacity than a single universal translator can provide.

We propose MoT (Mixture-of-Translators), which maps source context KV caches into the target cache space using token-level routing over multiple translators. We further introduce a Context Correction Loss that aligns the replayed target trajectory with the native target trajectory. Our analysis identifies two competing errors: early translation can propagate translation shift through many layers, whereas late translation leaves too few upper layers for correction. MoT reduces the translation shift that drives propagation, while context correction directly supervises target-side replay to reduce the remaining last-state shift.

Figure 1 illustrates how MoT enables accurate discussion among heterogeneous agents while reusing translated caches for scalable memory management. Figure 2 shows a clear quality–efficiency advantage in heterogeneous 10-agent reasoning: MoT preserves effective cross-model discussion while substantially reducing KV-memory usage.

We evaluate MoT on closed-set and extractive QA across heterogeneous translations among Llama-3.2 (3B), Gemma-3 (4B), and Qwen-3 (4B). Across these translations, MoT achieves the strongest average accuracy and F1 among the compared methods, with 57.6% accuracy and 0.42 F1. These scores recover 95.0% and 120.6% of native target performance and reach $1.5\times$ and $3.5\times$ the strongest-baseline scores, respectively. Multi-agent reasoning and long-context cache-augmented generation further demonstrate quality-preserving reuse, with scale-invariant cache-memory behavior as agent interactions grow and efficient storage reuse for long contexts.

Key Contributions.

1. **Competing Translation Errors.** We identify propagation error and correction-deficit error as two competing causes of cache-translation failure.
2. **Mixture-of-Translators Architecture.** We introduce MoT, which uses token-level routing over multiple translators to support heterogeneous KV-cache translation.
3. **Context Correction Loss.** We propose a replay-trajectory alignment objective that reduces residual target-side error after cache translation.
4. **Empirical Validation.** Across heterogeneous translations among Llama-3.2, Gemma-3, and Qwen-3, MoT achieves 57.6% average accuracy and 0.42 average F1, recovering 95.0% and 120.6% of native target performance and reaching $1.5\times$ and $3.5\times$ the strongest-baseline scores, respectively.
5. **Practical Case Studies.** Multi-agent reasoning demonstrates quality-preserving cache reuse with scale-invariant KV-memory behavior as interactions grow, while long-context CAG demonstrates effective storage reuse across large context budgets.

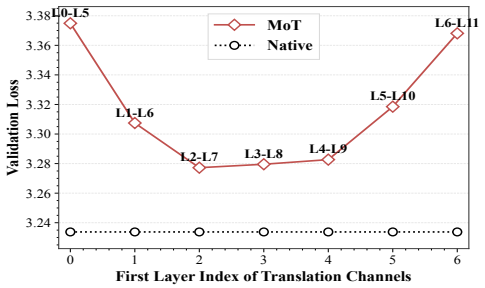


Figure 3: U-shaped loss caused by two competing errors.

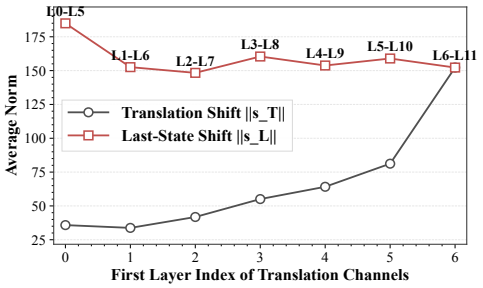


Figure 4: Exponential decay of propagated last-state shift.

2 PRELIMINARIES

2.1 PROBLEM SETUP

We consider context-conditioned generation, where context $\mathbf{x}^{\mathrm{Ctx}}$ and prompt $\mathbf{x}^{\mathrm{Prompt}}$ generate completion y. Let $\mathcal{M}^{\mathrm{Src}}$ and $\mathcal{M}^{\mathrm{Tgt}}$ denote the source and target models. Because their cache spaces differ, the source context cache $\mathbf{KV}^{\mathrm{Src,Ctx}}$ cannot be reused directly by the target. We define **cache translation** as learning $\mathbf{KV}^{\mathrm{Src,Ctx}} \rightarrow \widehat{\mathbf{KV}}^{\mathrm{Tgt,Ctx}}$, which maps the source cache into the target cache space. We view cache translation as directional communication between source and target layers. A connected source-to-target layer pair $c = (i, j)$ defines a **channel**, and the translation channel set is $\mathcal{C} = \{(i_r, j_r)\}_{r=1}^{|\mathcal{C}|}$, where $|\mathcal{C}|$ is the number of channels. Because translation quality depends on cache injection location, we analyze two competing error dynamics induced by channel placement.

2.2 COMPETING DYNAMICS OF TWO TRANSLATION ERRORS

As shown in Figure 3, for `gpt2`→`gpt2` translation, we varied the translation layer position by sliding a channel window of size 6 while increasing the start layer from 0 to 6. Across these positions, we consistently observed a U-shaped loss curve. This pattern suggests that different types of translation error dominate at the two extremes of the layer range: early windows suffer more from propagated translation shifts, whereas later windows leave fewer upper layers available for correction. In this section, we theoretically analyze the competing dynamics of these two errors.

We consider cache translation through the residual recurrence of a Transformer decoder. Motivated by prior work that models perturbation-induced feature shifts across layer positions (Zheng et al., 2025), and by analyses that use residual connections as a useful lens for modeling signal and representation propagation in Transformer-like architectures (He et al., 2023; Qin et al., 2026), we analyze how a translation-induced hidden-state shift behaves as a function of the highest target translation layer.

Definition 2.1. *(Residual Transformer Recurrence)*
For each target layer ℓ, let the layer recurrence be written as

$$\mathbf{h}_\ell = \mathbf{h}_{\ell-1} + F_\ell(\mathbf{h}_{\ell-1}), \qquad 1 \le \ell \le L.$$

where $\mathbf{h}_\ell \in \mathbb{R}^d$ denotes the hidden states after the ℓ-th layer, and F_ℓ denotes the residual branch. The residual branch F_ℓ absorbs the attention, MLP, normalization, and their sequential composition.

Full details including formal assumptions and proofs are provided in Appendix B.

Propagation Error. As shown in Figure 4, we empirically observe that when an error induced by translation at a lower layer propagates to the final layer, the resulting hidden-state shift decays exponentially. In this paragraph, we theoretically model this phenomenon.

Proposition 2.2 (Exponential Decay of Propagated Last-State Shift)**.** *Suppose that the translation-induced shift component formed at the first target translation layer propagates to the final hidden states through the same target residual recurrence. Then,*

$$\|\mathbf{s}_L\| \le (1+\delta)^{L-T_{\mathrm{Start}}} \|\mathbf{s}_{T_{\mathrm{Start}}}\|, \qquad T_{\mathrm{Start}} := \min\{j \mid (i,j) \in \mathcal{C}\}. \tag{1}$$

Thus, the upper bound on the last-state shift $\|\mathbf{s}_L\|$ decreases exponentially as T_{Start} increases.

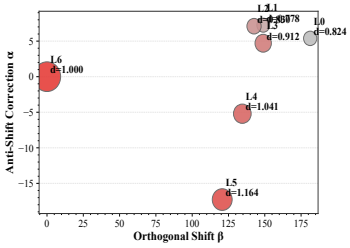


Figure 5: Decomposition of correction coefficients.

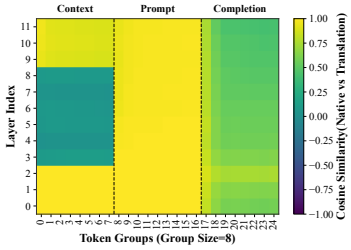


Figure 6: Error propagation and correction.

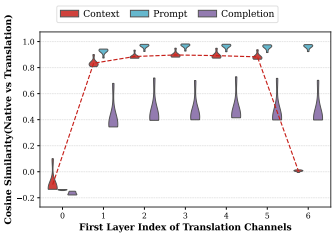


Figure 7: Inverted U-shaped similarity curve.

Correction Deficit Error. As shown in Figure 5, moving the translation window upward leaves fewer upper layers available to correct the translation-induced shift. We summarize this behavior using three quantities: $\alpha_{T:L}$ for anti-shift correction, $\beta_{T:L}$ for orthogonal shift, and $d_{T:L}$ for the remaining shift along the original direction. Empirically, $\alpha_{T:L}$ decreases as the translation layer moves upward, whereas $\beta_{T:L}$ remains nearly unchanged. This indicates that late translation leaves fewer layers available for anti-shift correction, making correction harder. We verify this trend through the increase of the correction-deficit coefficient $d_{T:L}$. Formal definitions are deferred to Definition B.2.

Proposition 2.3 (Increase of Correction-Deficit Coefficient)**.** *Suppose that the translation shift at the highest target translation layer is nonzero, i.e.,* $\mathbf{s}_T \neq 0$*, where* $T := \max\{j \mid (i,j) \in \mathcal{C}\}$*. Then, as* T *approaches the final layer, the remaining correction opportunity decreases, and the correction-deficit coefficient reaches its terminal value* $d_{L:L} = 1$*.*

Competing-Dynamics Translation Objective. Figure 6 compares similarity to the native KV cache when translating the context over layers $[3, 8]$. Native denotes the reference setting where the target model directly prefills $\mathbf{x}^{\mathrm{Ctx}}$ without translation. Across the context, prompt, and completion token groups, we observe both correction and propagation: similarity increases toward later layers for token group 0, while error also grows in the upper-right region, indicating accumulated propagation. Figure 7 further shows similarity trends while sliding a 6-layer translation window from layer 0 to layer 6. Unlike the prompt and completion regions, the context region exhibits an inverted U-shaped similarity curve, consistent with the U-shaped loss in Figure 3. This suggests that insufficient correction is especially pronounced for translated context caches.

Together, these observations show that translation-layer selection induces two competing errors. The first is the translation shift $\mathbf{s}_T$ accumulated inside the channel window, whose bound can increase as the highest translated layer T moves upward. The second is the final last-state shift $\mathbf{s}_L$, which remains when upper layers after translation cannot sufficiently correct the shift. Thus, translating too early can amplify propagation error, whereas translating too late can increase correction-deficit error.

We now decompose the components that constitute $\mathbf{s}_L$.

Proposition 2.4 (Last-State Shift Decomposition)**.** *Assume that* $\mathbf{s}_T \neq 0$*. For the correction deficit coefficient* $d_{T:L}$ *and the orthogonal shift coefficient* $\beta_{T:L}$ *defined in Definition B.2,*

$$\|\mathbf{s}_L\|^2 = d_{T:L}^2 \|\mathbf{s}_T\|^2 + \beta_{T:L}^2 \tag{2}$$

holds. Therefore, the last-state shift can be decomposed into the correction deficit component that remains along the translation-shift direction and the orthogonal shift component.

Proposition 2.4 shows that the last-state shift is not determined solely by the translation shift. Even when the translation shift $\|\mathbf{s}_T\|$ is small, the last-state shift $\|\mathbf{s}_L\|$ can remain large due to correction deficit $d_{T:L}$ or orthogonal shift $\beta_{T:L}$. Conversely, reducing only the last-state shift does not necessarily prevent a large translation shift from forming.

Therefore, translation should jointly minimize the translation shift $\|\mathbf{s}_T\|$ and the last-state shift $\|\mathbf{s}_L\|$. The former captures error accumulated within the selected channel set, while the latter captures residual error after upper-layer correction. Optimizing only one is insufficient; robust cache translation requires controlling both competing error dynamics.

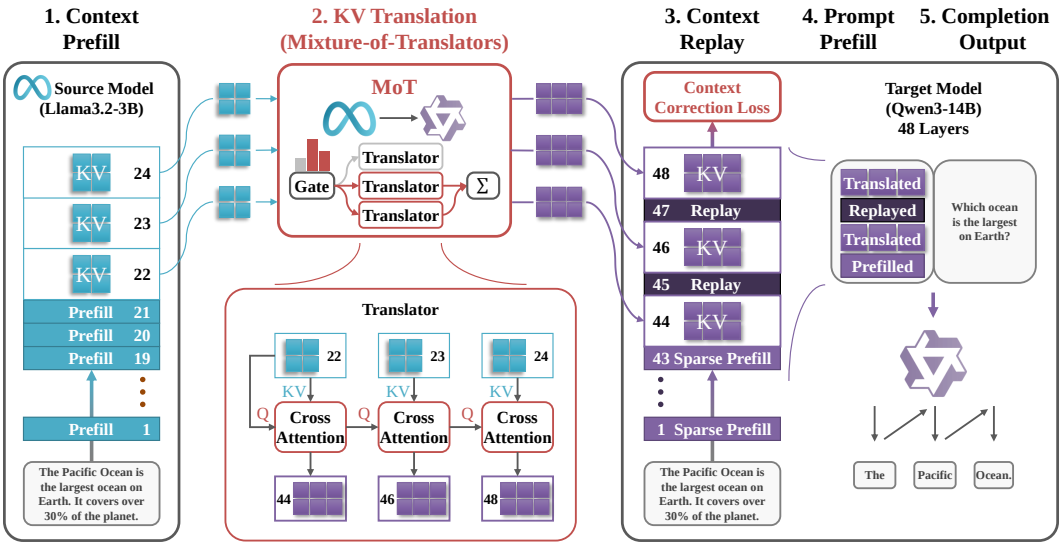


Figure 8: Overview of Mixture-of-Translators and Context Correction Loss.

3 Proposed Method: Mixture-of-Translators

The theoretical analysis in the preliminaries shows that cache translation must jointly control two error dynamics: the translation shift $\|\mathbf{s}_T\|$ formed inside the channel window and the last-state shift $\|\mathbf{s}_L\|$ remaining after insufficient upper-layer correction. Based on this analysis, we propose two learning components. First, Section 3.1 introduces **MoT (Mixture-of-Translators)**, which reduces token-level translation mismatch inside the channel window through gated translator routing. Second, Section 3.2 introduces **Context Correction Loss**, which aligns the replayed target trajectory with the native target trajectory after translation and directly targets the remaining last-state shift.

These two components are used within the overall cache-translation pipeline shown in Figure 8. The pipeline consists of five stages: ① **Context Prefill**, which computes the source context cache and selects translation channels; ② **KV Translation**, where MoT maps the source cache into the target cache space; ③ **Context Replay**, which reconstructs the target-side context cache with context-correction supervision; ④ **Prompt Prefill**, which conditions the target model on the reconstructed cache; and ⑤ **Completion Output**, which generates the final response. Channel mapping and cache replay are detailed in Appendix C.4 and Appendix C.5.

3.1 Mixture-of-Translators

Each translator in MoT uses the backbone architecture described in Appendix C.3. A single backbone translator applies one translation rule to the entire cache, which can limit its ability to capture token-dependent source-to-target correspondences. To reduce the translation shift $\|\mathbf{s}_T\|$, MoT performs token-level routing over multiple translators, allowing different tokens to exploit distinct translation specializations. We formalize this effect and its connection to the translation shift in Appendix C.1.

Definition 3.1 (Mixture-of-Translators). *For a fixed channel set $\mathcal{C}$, let $\mathbf{z}_u$ denote the source context-cache input at token u. We introduce N_{Tr} translators $\{\mathrm{Tr}_m\}_{m=1}^{N_{\mathrm{Tr}}}$ and a gating function that produces nonnegative token-level scores $g(\mathbf{z}_u) = (g_1(\mathbf{z}_u), \ldots, g_{N_{\mathrm{Tr}}}(\mathbf{z}_u))$. Here, Top-$K$ denotes the number of translators used for each token. Let the selected translator indices at token u be $\mathcal{T}_{\text{Top-}K}(u) :=$ Top-K $(g(\mathbf{z}_u))$. The selected scores are renormalized as*

$$\widetilde{g}_m(\mathbf{z}_u) := \frac{g_m(\mathbf{z}_u)}{\sum_{r \in \mathcal{T}_{\text{Top-}K}(u)} g_r(\mathbf{z}_u)}, \qquad m \in \mathcal{T}_{\text{Top-}K}(u).$$

Then, the output of Mixture-of-Translators is defined as

$$\mathrm{Tr}_{\mathrm{MoT}}(\mathbf{z}_u) := \sum_{m \in \mathcal{T}_{\text{Top-}K}(u)} \widetilde{g}_m(\mathbf{z}_u)\ \mathrm{Tr}_m(\mathbf{z}_u).$$

3.2 CONTEXT CORRECTION LOSS.

MoT reflects token-specific translation difficulty and thereby reduces the translation shift $\|\mathbf{s}_T\|$ formed inside the channel window. However, even if $\|\mathbf{s}_T\|$ becomes small, the last-state shift $\|\mathbf{s}_L\|$ may still remain when the upper layers after translation fail to provide sufficient correction. Therefore, it is necessary to directly constrain the target model after translation to follow the native hidden-state trajectory. To this end, we introduce the Context Correction Loss.

For a fixed channel set $\mathcal{C}$, let $\widehat{\mathbf{h}}_\ell^{\mathrm{Tgt}}$ denote the hidden states obtained by replaying the target model with the translated context cache, and let $\mathbf{h}_\ell^{\mathrm{Tgt}}$ denote the hidden states obtained when the target model natively prefills the same context. The ideal objective of the Context Correction Loss is to directly align the hidden-state trajectory after translation with the native trajectory:

$$\mathcal{L}_{\mathrm{CC}}^{\mathrm{Src}\to\mathrm{Tgt}}(\mathcal{C}) = \sum_{\ell=T_{\mathrm{Start}}+1}^{L} \left\| \widehat{\mathbf{h}}_\ell^{\mathrm{Tgt}} - \mathbf{h}_\ell^{\mathrm{Tgt}} \right\|^2 .$$

Since the loss includes the final-layer term $\left\| \widehat{\mathbf{h}}_L^{\mathrm{Tgt}} - \mathbf{h}_L^{\mathrm{Tgt}} \right\|^2 = \|\mathbf{s}_L\|^2$, it directly reduces the last-state shift. Implementation details, including the KV-based heuristic, are provided in Appendix C.2.

4 EXPERIMENTS

We evaluate MoT on downstream QA benchmarks with a primary focus on heterogeneous cache translation among Llama-3.2, Gemma-3, and Qwen-3. Section 4.1 summarizes the experimental setup, and Section 4.2 reports the main heterogeneous translation results. Section 4.3 analyzes the effects of correction behavior, loss design, and the MoT architecture. Additional translation results, ablations, and scalability analyses are provided in Appendix E.

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate two QA benchmark groups: **closed-set QA**, including BoolQ, PubMedQA, and MMLU-Redux, reported by average accuracy; and **extractive QA**, including SQuAD-v1.1 and NewsQA, reported by average F1. For case studies, we additionally use StrategyQA for multi-agent reasoning and HotpotQA-E for long-context CAG in Section 5.

Methods. We compare MoT with representative internal-state communication methods. LSC (Dery et al., 2026) maps layer-wise KV caches into a global shared latent space through model-specific adapters; C2C-Project (Fu et al., 2026) uses the projection-based C2C variant to directly map source KV states into the target cache space; Interlat (Du et al., 2026) maps a sender's last-layer hidden-state sequence into the receiver's embedding space; and KVComm (Shi et al., 2026) selectively shares important KV layers between structurally compatible models without training.

Models. Our main evaluation uses a heterogeneous model pool consisting of Llama-3.2 (3B), Gemma-3 (4B), and Qwen-3 (4B). These model families differ in depth, hidden dimension, attention configuration, and KV-cache structure, resulting in substantially different cache spaces. See Appendix E.3 for the shared translator training configuration.

4.2 RESULTS

Main Results. Table 1 shows that MoT consistently achieves the strongest cache-translation performance. Across heterogeneous translations, MoT obtains an average cosine similarity of 0.98, accuracy of 57.6%, and F1 of 0.42. These scores recover 95.0% and 120.6% of Native accuracy and F1 and reach $1.5\times$ and $3.5\times$ the strongest-baseline scores, respectively; the strongest baseline, Interlat, achieves 39.1% accuracy and 0.12 F1. The same advantage is consistently observed for homogeneous translation. The lower baseline performance may partly reflect their respective design constraints. LSC requires source-to-shared and shared-to-target mappings through a common translation space, which can make heterogeneous translation more difficult. C2C-Project relies on a direct projection, potentially limiting its capacity to bridge substantial cache-space mismatch. Interlat communicates only last-layer hidden states rather than the full token-wise KV cache, which may limit the information available for reconstructing the target cache.

Table 1: Cache translation results across nine translations among Llama-3.2 (3B), Gemma-3 (4B), and Qwen-3 (4B). Native denotes the unshaded target model without translation. Darker green indicates higher recovery relative to Native.

Source Model	Target Model	Method	Cosine Sim	Closed-set QA (Accuracy)				Extractive QA (F1)		
				BoolQ	PubMedQA	MMLU-Redux	Avg	SQuAD	NewsQA	Avg
Llama-3.2	Llama-3.2	Native	N/A	74.0	71.7	56.5	67.4±4.9	0.41	0.48	0.44±0.01
		LSC	0.01	0.0	0.0	0.0	0.0±0.0	0.00	0.01	0.00±0.00
		C2C-Project	0.11	54.7	**62.3**	21.1	46.0±1.1	0.08	0.02	0.05±0.03
		Interlat	0.84	62.7	60.0	42.7	55.2±2.3	0.21	0.13	0.17±0.03
		MoT	**1.00**	**73.7**	70.0	**56.2**	**66.6**±6.2	**0.51**	**0.54**	**0.53**±0.02
	Gemma-3	Native	N/A	64.3	42.3	23.5	43.4±2.5	0.49	0.35	0.42±0.04
		LSC	0.03	29.0	12.7	0.4	14.0±1.3	0.09	0.01	0.05±0.03
		C2C-Project	0.77	**58.7**	**39.3**	0.2	32.8±2.1	0.17	0.05	0.11±0.04
		Interlat	0.24	35.7	23.3	2.4	20.5±3.0	0.17	0.02	0.10±0.04
		MoT	**0.99**	**63.3**	43.7	**22.9**	**43.3**±3.2	**0.43**	**0.32**	**0.38**±0.03
	Qwen-3	Native	N/A	83.0	59.3	72.4	71.6±1.2	0.26	0.21	0.24±0.03
		LSC	-0.01	48.7	23.0	36.9	36.2±1.9	0.15	0.00	0.08±0.01
		C2C-Project	0.74	51.3	47.7	27.2	42.1±1.9	0.09	0.03	0.06±0.01
		Interlat	0.64	49.7	44.0	41.0	44.9±3.1	0.20	0.04	0.12±0.01
		MoT	**0.99**	**86.0**	**56.0**	**71.7**	**71.2**±1.1	**0.40**	**0.25**	**0.32**±0.03
Gemma-3	Llama-3.2	Native	N/A	74.0	71.7	56.5	67.4±4.9	0.41	0.48	0.44±0.01
		LSC	0.00	0.0	0.0	0.0	0.0±0.0	0.01	0.01	0.01±0.00
		C2C-Project	0.09	51.0	44.7	24.5	40.1±1.1	0.15	0.01	0.08±0.07
		Interlat	0.57	55.7	**59.0**	29.2	48.0±1.4	0.23	0.05	0.14±0.07
		MoT	**0.96**	**72.3**	44.3	**55.0**	**57.2**±3.6	**0.52**	**0.44**	**0.48**±0.05
	Gemma-3	Native	N/A	64.3	42.3	23.5	43.4±2.5	0.49	0.35	0.42±0.04
		LSC	0.04	43.0	3.3	1.6	16.0±0.8	0.12	0.01	0.07±0.02
		C2C-Project	0.02	19.3	**32.7**	0.0	17.3±1.2	0.09	0.03	0.06±0.02
		Interlat	0.21	37.0	17.3	**20.4**	24.9±1.4	0.24	0.04	0.14±0.06
		MoT	**0.99**	**65.7**	38.0	10.8	**38.1**±1.4	**0.52**	**0.36**	**0.44**±0.03
	Qwen-3	Native	N/A	83.0	59.3	72.4	71.6±1.2	0.26	0.21	0.24±0.03
		LSC	0.01	0.0	0.0	0.0	0.0±0.0	0.00	0.01	0.00±0.00
		C2C-Project	0.67	60.7	49.0	40.3	50.0±2.0	0.17	0.05	0.11±0.01
		Interlat	0.67	51.7	36.7	40.4	42.9±1.6	0.18	0.04	0.11±0.02
		MoT	**0.97**	**83.7**	**45.0**	**71.2**	**66.6**±1.4	**0.54**	**0.27**	**0.41**±0.03
Qwen-3	Llama-3.2	Native	N/A	74.0	71.7	56.5	67.4±4.9	0.41	0.48	0.44±0.01
		LSC	0.01	0.0	0.0	0.0	0.0±0.0	0.00	0.01	0.01±0.00
		C2C-Project	0.09	48.7	56.0	27.9	44.2±2.3	0.13	0.02	0.07±0.06
		Interlat	0.58	57.0	**60.0**	31.4	49.5±2.7	0.22	0.05	0.13±0.06
		MoT	**1.00**	**74.3**	68.7	**55.8**	**66.2**±6.2	**0.57**	**0.56**	**0.57**±0.02
	Gemma-3	Native	N/A	64.3	42.3	23.5	43.4±2.5	0.49	0.35	0.42±0.04
		LSC	0.03	42.7	10.3	0.2	17.7±2.2	0.11	0.01	0.06±0.04
		C2C-Project	0.01	21.3	14.7	0.0	12.0±0.6	0.10	0.02	0.06±0.02
		Interlat	0.21	43.3	25.3	17.9	28.8±1.8	0.20	0.03	0.12±0.06
		MoT	**0.99**	**60.0**	**44.0**	**18.9**	**41.0**±3.5	**0.45**	**0.32**	**0.38**±0.03
	Qwen-3	Native	N/A	83.0	59.3	72.4	71.6±1.2	0.26	0.21	0.24±0.03
		LSC	0.00	0.0	0.0	0.0	0.0±0.0	0.00	0.00	0.00±0.00
		C2C-Project	0.01	61.3	58.0	40.9	53.4±2.9	**0.22**	0.05	0.13±0.02
		Interlat	0.34	44.0	43.7	41.7	43.1±2.0	0.11	0.03	0.07±0.01
		MoT	**0.99**	**83.3**	**59.3**	**72.4**	**71.7**±1.5	0.24	**0.18**	**0.21**±0.03

The recovery distributions are also strongly separated: the lower quartile (Q1) of MoT reaches 93.4% for accuracy and 95.1% for F1, substantially exceeding the upper quartile (Q3) of all baseline recoveries at 66.2% and 31.3%, respectively. This gap indicates that MoT's advantage is not driven by a small number of favorable translations, but is consistently maintained across model pairs and tasks. The same separation is further reflected across MMLU-Redux categories in Figure 9.

4.3 Root Cause Analysis

Shifts and Correction. Figure 10 shows that, compared with the Single Backbone & Prompt LM Loss setting in Figure 4, the proposed MoT & Context Correction Loss sharply reduces both the translation shift $\|\mathbf{s}_T\|$ and the last-state shift $\|\mathbf{s}_L\|$ across translation-layer positions. This shows that the two components suppress their respective target error dynamics. We further compare correction trends between Figure 5 and Figure 11. Overall, $\alpha_{T:L}$ slightly decreases and rises only at a few later positions, while $\beta_{T:L}$ drops substantially from early translation layers and is further reduced in later layers. Although $d_{T:L}$ slightly increases, its magnitude remains limited. Thus, from Eq. equation 2, the reduction in $\|\mathbf{s}_L\|$ is better attributed to the joint decrease of $\|\mathbf{s}_T\|$ and $\beta_{T:L}$, rather than to a substantial improvement in correction deficit. Additional results are provided in Appendix E.5.

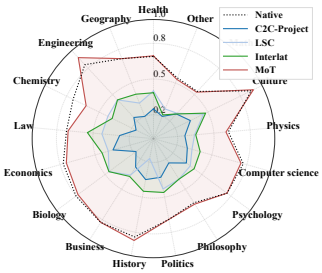


Figure 9: MMLU-Redux.

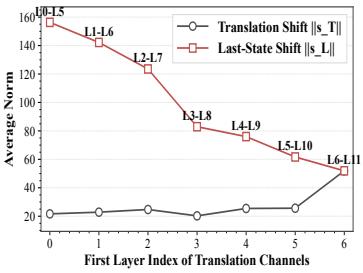


Figure 10: Improved shifts.

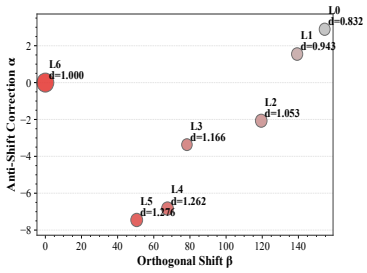


Figure 11: Improved correction.

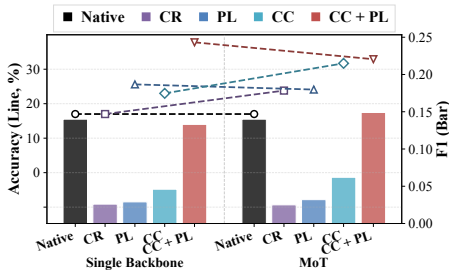


Figure 12: Loss ablation.

Method	Acc	F1	Translator Usage	
			CV^2	Entropy
MoE	48.1	0.41	0.82	0.15
MoT-Uni	49.0	0.41	0.00	0.69
MoT	51.0	0.43	0.32	0.51

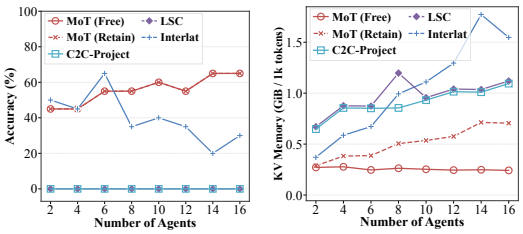
Table 2: MoT architecture ablation.

Loss Ablation. Figure 12 shows that the proposed CC (Context Correction) loss remains effective independently of MoT. Even in the Single Backbone setting, without routing, CC yields more stable accuracy and F1 gains than the prior CR (Context Reconstruction) and PL (Prompt LM) losses (Dery et al., 2026). This suggests that CC directly reduces the last-state shift by aligning the translated cache with the native target trajectory, rather than relying on translator routing. The same trend holds under MoT, supporting our analysis that reducing the translation shift with MoT and the last-state shift with CC are complementary. Individually, CR and PL provide limited recovery, whereas CC gives stronger gains. Combining CC with PL achieves the best accuracy and F1 by coupling context-level KV correction with prompt-conditioned generation supervision, so we adopt CC+PL as the final heuristic loss.

MoT Architecture Ablation. For this controlled architecture ablation, we use Qwen2.5 (7B)→Qwen2.5 (0.5B) translation and compare translator selection in MoT under the simplest configuration, $(\text{Top-}K, N_{\text{Tr}}) = (1, 2)$. As baselines, we use MoE (Mixture-of-Experts), which splits the internal Dense MLP into experts, and MoT-Uni, which always uses translators uniformly. The results are summarized in Table 2. Usage Imbalance is $\text{CV}^2 = \text{Var}_e(I_e)/\text{Mean}_e(I_e)^2$, where I_e is the cumulative selection count of translator e, and Usage Entropy is $-\sum_e p_e \log p_e$ with $p_e = I_e / \sum_j I_j$. Higher entropy indicates more balanced translator usage. Table 2 shows that MoE usage is highly skewed, while MoT-Uni is balanced but lacks input-specific selectivity. In contrast, MoT uses both translators selectively and achieves higher Acc and F1, supporting that token-wise translator selection helps reduce the translation shift $\mathbf{s}_T$.

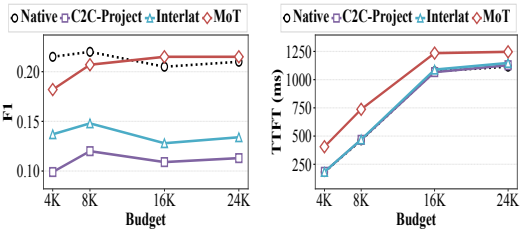
5 Case Studies

Memory Reuse: Multi-Agent Reasoning. For multi-agent reasoning, we evaluate StrategyQA using a fixed MALLM configuration with Expert persona generation, Simple response generation, the Memory discussion paradigm, and Supermajority Consensus (Becker et al., 2025). We define the first agent as the Hub Agent and compare two cache-management modes: Retain, which keeps all resident agent caches, and Free, which retains the Hub Agent cache while releasing non-hub caches after offload to support scalable memory reuse.



(a) Accuracy (b) KV Memory

Figure 13: Agent scalability.



(a) F1 by Budget (b) TTFT by Budget

Figure 14: Long-context scalability.

We consider two complementary settings: heterogeneous 10-agent reasoning as a practical cross-model case study, and homogeneous agent-count scaling to isolate memory scalability. Figure 2 shows the heterogeneous 10-agent result, where MoT exhibits a clear quality–efficiency advantage by preserving accurate cross-model discussion while substantially reducing KV-memory usage.

Figure 13 evaluates homogeneous scalability from 2 to 16 agents. Because unsuccessful discussions can terminate early and produce shorter histories, we normalize KV usage by the tokens generated during successfully verified turns, preventing early termination from appearing artificially memory-efficient. As the number of agents increases, MoT shows an overall improvement in reasoning accuracy while MoT (Free) keeps normalized KV memory nearly constant. In contrast, Retain and the baselines accumulate memory as the interaction grows. Together, these results demonstrate **scale-invariant KV-memory behavior**: cache translation preserves useful discussion history while cache eviction suppresses redundant model-local state.

Detailed experimental configurations, cache-management procedures, normalization, training costs, and memory breakdowns are provided in Appendix F.1.

Storage Reuse: Long-Context CAG. This case study evaluates long-context CAG (Cache-Augmented Generation), where long contexts are stored as KV caches and reused. This setting tests whether translated caches can preserve useful long-context information as the reusable context grows. On `HotpotQA-E`, we vary the context budget over 4K, 8K, 16K, and 24K. Native represents the RAG setting that prefills the original context for every query, while the other methods reuse stored caches in the CAG setting. Figure 14(a) shows that C2C-Project and Interlat remain in a low-F1 regime even as the context budget increases. In contrast, MoT improves substantially with larger budgets, especially at 16K and 24K, indicating that it can effectively reuse long-context information stored in the KV cache. Figure 14(b) shows that TTFT (Time to First Token) increases with context budget. Native incurs larger prefill cost, while CAG methods incur cache conversion and injection overhead. MoT shows only marginally higher TTFT due to the additional use of MoT. Overall, MoT achieves F1 close to Native under large budgets, showing stronger cache-reuse quality for long-context CAG. Details are provided in Appendix F.2.

6 Conclusion

We introduced MoT (Mixture-of-Translators), a cache translation framework for reusing KV caches across heterogeneous LLMs. MoT combines multiple translator modules with token-level routing and a Context Correction Loss that aligns the replayed target trajectory with the native target trajectory. Our analysis identified two competing error sources, propagation error and correction-deficit error, which motivate this joint design. Experiments on closed-set QA and extractive QA demonstrate that MoT preserves downstream quality under heterogeneous cache translation among Llama-3.2, Gemma-3, and Qwen-3, achieving 57.6% average accuracy and 0.42 average F1. These scores recover 95.0% and 120.6% of native target performance and reach $1.5\times$ and $3.5\times$ the strongest-baseline scores, respectively. The practical case studies further show that translated caches enable accurate heterogeneous-agent discussion with scale-invariant KV-memory behavior as agent interactions grow, while preserving near-native quality for long-context CAG. Together, these results establish heterogeneous KV-cache translation as a practical foundation for scalable multi-model LLM systems.

REPRODUCIBILITY STATEMENT

We provide the assumptions and complete proofs for the theoretical results in Appendix Section B. Detailed descriptions of the translator architecture, Context Correction Loss, channel mapping, cache replay, and the hidden-state translation variant are provided in Appendix Section C. Benchmark prompt formats are specified in Appendix Section D. Training configurations, additional benchmark results, ablations, and scalability analyses are reported in Appendix Section E, and the complete experimental configurations for the multi-agent reasoning and long-context CAG case studies are provided in Appendix Section F. Dataset, implementation, and model-license information is documented in Appendix Section G.

The code used in this work is available at `https://anonymous.4open.science/r/MoT-D7E3`, and the pretrained translator checkpoints are available at `https://anonymous-hf.com/a/tkelnedcist7`. Together, these materials provide the theoretical derivations, implementation details, experimental configurations, evaluation prompts, and trained artifacts needed to reproduce the main results.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing; retrieve and discover relevant literature; support research ideation and execution; draft portions of the paper; and assist with method implementation and software code. We did not use generative AI tools to generate synthetic datasets or to prove mathematical claims.

All AI-assisted outputs were reviewed by the authors. In particular, suggested literature was checked against the cited sources, research suggestions were independently evaluated before use, and AI-assisted implementation and code were inspected and tested by the authors. Mathematical claims and proofs were independently checked by the authors. The authors take responsibility for the final content of this work, including its text, claims, code, and other artifacts produced with the aid of generative AI.

REFERENCES

Anonymous. Towards a collaborative memory for agentic workflow: Breaking the prefix barrier with segment-level KV cache sharing. OpenReview, 2026. URL `https://openreview.net/forum?id=kgzBkyqg6Z`. ICLR 2026 submission; public record remains anonymous; accessed 2026-09-19.

Anthropic. Prompt caching. Claude Platform Documentation, 2026. URL `https://platform.claude.com/docs/en/build-with-claude/prompt-caching`. Living documentation; citation year reflects the version accessed on 2026-09-19.

Jonas Becker, Lars Benedikt Kaesberg, Niklas Bauer, Jan Philip Wahle, Terry Ruas, and Bela Gipp. MALLM: Multi-agent large language models framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 418–439, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-demos.29. URL `https://aclanthology.org/2025.emnlp-demos.29/`.

Brian J. Chan, Chao-Ting Chen, Jui-Hung Cheng, and Hen-Hsen Huang. Don't do RAG: When cache-augmented generation is all you need for knowledge tasks. In *Companion Proceedings of the ACM Web Conference 2025*, pp. 893–897. Association for Computing Machinery, 2025. doi: 10.1145/3701716.3715490. URL `https://doi.org/10.1145/3701716.3715490`.

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. *arXiv preprint arXiv:2606.13594*, 2026a.

Weijian Chen, Shuibing He, Haoyang Qu, Ruidong Zhang, Siling Yang, Ping Chen, Yi Zheng, Baoxing Huai, and Gang Chen. IMPRESS: An importance-informed multi-tier prefix KV storage system for large language model inference. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*, pp. 187–201, Santa Clara, CA, February 2025. USENIX Association.

ISBN 978-1-939133-45-8. URL https://www.usenix.org/conference/fast25/presentation/chen-weijian-impress.

Xiaoyu Chen, Fengge Wu, Zhao Junsuo, and Yun Fan. CondenseFlow: Scalable latent space collaboration via semantic compression for multi-agent systems. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13694–13712, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.669. URL https://aclanthology.org/2026.findings-acl.669/.

Zihan Chen, Zeshen Li, Howard H. Yang, Tony Q. S. Quek, and Jihong Park. Federated inference for heterogeneous LLM communication and collaboration, 2026c. URL https://arxiv.org/abs/2603.28772. Position paper.

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding the mixture-of-experts layer in deep learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23049–23062, 2022.

Lucio M. Dery, Zohar Yahav, Henry Prior, Qixuan Feng, Jiajun Shen, and Arthur Szlam. Latent space communication via K-V cache alignment, 2026. URL https://arxiv.org/abs/2601.06123.

Nishanth Dikkala, Nikhil Ghosh, Raghu Meka, Rina Panigrahy, Nikhil Vyas, and Xin Wang. On the benefits of learning to route in mixture-of-experts models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9376–9396, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.583.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27106–27129, San Diego, California, USA, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1248. URL https://aclanthology.org/2026.acl-long.1248/.

Timothy W. Finin, Richard Fritzson, Donald P. McKay, and Robin McEntire. KQML as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management*, pp. 456–463. Association for Computing Machinery, 1994. doi: 10.1145/191246.191322. URL https://doi.org/10.1145/191246.191322.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29, pp. 2145–2153, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html.

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://proceedings.iclr.cc/paper_files/paper/2026/hash/474ada926b331d78f06d95e8913111cc-Abstract-Conference.html.

Shiwei Gao, Youmin Chen, and Jiwu Shu. Fast state restoration in LLM serving with HCache. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 128–143. Association for Computing Machinery, 2025. doi: 10.1145/3689031.3696072. URL https://doi.org/10.1145/3689031.3696072.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 325–338, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/a66caa1703fe34705a4368c3014c1966-Paper-Conference.pdf.

Bobby He, James Martens, Guodong Zhang, Aleksandar Botev, Andrew Brock, Samuel L. Smith, and Yee Whye Teh. Deep transformers without shortcuts: Modifying self-attention for faithful signal propagation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2302.10322.

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.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Hyesung Jeon, Hyeongju Ha, and Jae-Joon Kim. LRAgent: Efficient KV cache sharing for multi-LoRA LLM agents, 2026. URL https://arxiv.org/abs/2602.01053.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data, 2018. URL https://arxiv.org/abs/1811.11479.

Ruofan Jia, Weiying Xie, Jie Lei, Jitao Ma, Haonan Qin, and Leyuan Fang. HeteroTune: Efficient federated learning for large heterogeneous models, 2024. URL https://arxiv.org/abs/2411.16796.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era, 2020. URL https://arxiv.org/abs/2006.02419.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Daliang Li and Junpu Wang. FedMD: Heterogenous federated learning via model distillation. In *FL-NeurIPS 2019: International Workshop on Federated Learning for User Privacy and Data Confidentiality*, 2019. URL https://arxiv.org/abs/1910.03581.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008, 2023. doi: 10.52202/075280-2264. URL https://proceedings.neurips.cc/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

Sichu Liang, Zhenglin Wang, Chujiajia, Pengfei Xia, Hui Zang, and Deyu Zhou. When KV cache reuse fails in multi-agent systems: Cross-candidate interaction is crucial for LLM judges. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7205–7230, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.327. URL https://aclanthology.org/2026.acl-long.327/.

Xiaoze Liu, Ruowang Zhang, Weichen Yu, Siheng Xiong, Liu He, Feijie Wu, Hoin Jung, Matt Fredrikson, Xiaoqian Wang, and Jing Gao. The vision wormhole: Latent-space communication in heterogeneous multi-agent systems, 2026. URL https://arxiv.org/abs/2602.15382.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, and Esha Choukse. Droidspeak: KV cache sharing for cross-LLM communication and multi-LLM serving, 2024. URL https://arxiv.org/abs/2411.02820.

Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. TurboRAG: Accelerating retrieval-augmented generation with precomputed KV caches for chunked text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6588–6601, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.334. URL https://aclanthology.org/2025.emnlp-main.334/.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

OpenAI. Prompt caching. OpenAI API Documentation, 2026. URL https://developers.openai.com/api/docs/guides/prompt-caching/. Living documentation; citation year reflects the version accessed on 2026-09-19.

Zaifeng Pan, Ajjkumar Patel, Yipeng Shen, Zhengding Hu, Yue Guan, Wan-Lu Li, Lianhui Qin, Yida Wang, and Yufei Ding. KVFlow: Efficient prefix caching for accelerating LLM-based multi-agent workflows. In *Advances in Neural Information Processing Systems*, volume 38, pp. 139912–139931, 2025. doi: 10.52202/085713-4208.

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

Zhen Qin, Jinxin Zhou, Jiachen Jiang, and Zhihui Zhu. On the convergence of gradient descent on learning transformers with residual connections. *IEEE Signal Processing Letters*, 33:1661–1665, 2026. doi: 10.1109/LSP.2026.3684818. URL https://doi.org/10.1109/LSP.2026.3684818.

Xingyu Qu, Siyuan Lu, Zhiyu Chen, Sheng Wang, and Tao Lin. Cachebridge: Efficient cross-model kv cache transfer. *arXiv preprint arXiv:2609.00891*, 2026.

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 51094–51116. PMLR, 2025. URL https://proceedings.mlr.press/v267/ramesh25a.html.

Maximillian Rossi, Prajwal Raghunath, and Eugene Wu. Latent cache flow: Model-to-model communication without text. *arXiv preprint arXiv:2605.22863*, 2026.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12974–12990, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.701. URL https://aclanthology.org/2024.acl-long.701/.

Xiangyu Shi, Marco Chiesa, Gerald Q. Jr. Maguire, and Dejan Kostic. KVComm: Enabling efficient LLM communication through selective KV sharing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=F7rUng23nw.

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, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.518. URL https://aclanthology.org/2025.emnlp-main.518/.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h0ZfDIrj7T.

Zijian Wen, Tao Zhang, Shuangwu Chen, Shenghao Ye, Yu Guo, Qirui Chen, Jingxian Shuai, Yunpeng Hou, Huasen He, and Jianyang. SpecCache: Speculative KV cache reuse for efficient RAG serving. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18861–18871, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.859. URL https://aclanthology.org/2026.acl-long.859/.

Sunghyeon Woo, Jaeeun Kil, Hoseung Kim, Minsub Kim, Joonghoon Kim, Ahreum Seo, Sungjae Lee, Minjung Jo, Jiwon Ryu, Baeseong park, Se Jung Kwon, and Dongsoo Lee. ICarus: Identical cache reuse for efficient multi-model inference. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qrMo6R7lOS.

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.

Huan Yang, Renji Zhang, Mingzhe Huang, Weijun Wang, Yin Tang, Yuanchun Li, Yunxin Liu, and Deyu Zhang. KVShare: An LLM service system with efficient and effective multi-tenant KV cache reuse, 2025a. URL https://arxiv.org/abs/2503.16525.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. In *Advances in Neural Information Processing Systems*, volume 38, pp. 148312–148339, 2025b. doi: 10.52202/085713-4461.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. Chunkattention: Efficient self-attention with prefix-aware KV cache and two-phase partition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11608–11620, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.623. URL https://aclanthology.org/2024.acl-long.623/.

Xinye Zhao and Spyridon Mastorakis. SemShareKV: Efficient KVCache sharing for semantically similar prompts via token-level LSH matching. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 432–445, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-ijcnlp.25. URL https://aclanthology.org/2025.findings-ijcnlp.25/.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a774503daed55eb53c634847ae071ec7-Abstract-Conference.html.

APPENDIX

Appendix Contents

A Related Work **15**
B Theoretical Analysis **17**
C Proposed Method Details **20**
D Benchmark Prompts **25**
E Additional Results **30**
F Details of Case Studies **40**
G License **44**
H Limitations and Broader Impacts **44**

A RELATED WORK

A.1 COMMUNICATION BETWEEN MODELS

Model communication predates LLMs and spans symbolic messages, parameter or model-update sharing, logit distillation, and learned vector protocols (Finin et al., 1994; McMahan et al., 2017; Jeong et al., 2018; Li & Wang, 2019; Foerster et al., 2016; Lazaridou & Baroni, 2020). These paradigms primarily target training-time aggregation, output-level transfer, or task-specific protocols rather than transmitting context-conditioned internal states. LLM systems instead commonly communicate through text via role-based collaboration, mixture-of-agents, debate, routing, and collaborative decoding (Li et al., 2023; Wu et al., 2024; Hong et al., 2024; Wang et al., 2025; Du et al., 2024; Ong et al., 2025; Shen et al., 2024). Although model-agnostic, text communication discretizes internal representations, introducing a semantic bottleneck and decoding overhead, while routing does not directly transfer internal understanding. Embedding-, activation-, and latent-state methods (Pham et al., 2024; Ramesh & Li, 2025; Tang et al., 2025) provide continuous alternatives but remain tied to particular representations or language trajectories; heterogeneous federated methods such as HeteroTune (Jia et al., 2024) instead focus on training-time knowledge sharing.

Recent work therefore communicates context-specific internal states directly. KVComm (Shi et al., 2026) approximates KV-cache offsets for training-free cross-context reuse; LSC (Dery et al., 2026) aligns model caches in a global shared latent space; C2C (Fu et al., 2026) directly projects and fuses source caches into the receiver space; and Interlat (Du et al., 2026) compresses last-hidden-state "thought" representations for language-free communication. Related studies further explore semantic compression and aligned latent communication (Chen et al., 2026b; Rossi et al., 2026; Chen et al., 2026a). These methods extend communication beyond text, logits, and parameters, but KVComm mainly targets RoPE-based cross-context reuse, LSC requires shared-space alignment, C2C can incur pairwise fuser or translator costs as the model pool grows, and Interlat does not explicitly resolve KV-format mismatch. MoT instead uses a gated mixture of translators to increase translation capacity for heterogeneous KV caches without relying on a single translator or fixed shared space.

A.2 MEMORY REUSE FOR AGENTIC REASONING

Agentic workflows repeatedly reuse system prompts, tool instructions, retrieved contexts, and shared workspaces across agents or branches (Du et al., 2024). Prefix-cache reuse removes redundant prefilling for identical prefixes, but model-specific KV formats force heterogeneous models to maintain separate copies. Cache reuse can also affect downstream judging when cross-candidate interaction is required (Liang et al., 2026), whereas we focus on memory redundancy. We classify compatibility as **Homogeneous** (same model), **Intra-Family** (same family), or **Cross-Family** (different families); Appendix E.1 gives the full taxonomy.

Same-format reuse ranges from token-exact or structured prefix sharing (ChunkAttention, PromptCache) to workflow-aware or multi-tier caching and semantic/segment-level reuse (KVFlow, IMPRESS, SemShareKV, KVShare, CrossKV) (Ye et al., 2024; Gim et al., 2024; Pan et al., 2025; Chen et al., 2025; Zhao & Mastorakis, 2025; Yang et al., 2025a; Anonymous, 2026); all assume a common

Table 3: Comparison of shared-prefix cache reuse across M participating models or agents.

Category	Heterogeneity Level	Cache Format	Memory Redundancy
ChunkAttention(Ye et al., 2024), PromptCache(Gim et al., 2024), KVFlow(Pan et al., 2025), SemShareKV(Zhao & Mastorakis, 2025), KVShare(Yang et al., 2025a), CrossKV(Anonymous, 2026)	Homogeneous	KV Cache	$O(M)$
DroidSpeak(Liu et al., 2024), LRAgent(Jeon et al., 2026)	Intra-Family	KV Cache	$O(M)$
ICaRus(Woo et al., 2026)	Intra-Family	KV Cache	$O(1)$
C2C(Fu et al., 2026), LSC(Dery et al., 2026)	Cross-Family	KV Cache	$O(M)$
Vision Wormhole(Liu et al., 2026)	Cross-Family	Visual Tokens (Non-KV)	N/A
MoT (Proposed)	**Cross-Family**	**KV Cache**	$O(1)$

Table 4: Comparison of long-context storage reuse across M models.

Category	Heterogeneity Level	Storage Redundancy
RAG(Lewis et al., 2020)	Cross-Family[†]	$O(1)$ (Text)
CAG(Chan et al., 2025), HCache(Gao et al., 2025), TurboRAG(Lu et al., 2025), KVLink(Yang et al., 2025b), SpecCache(Wen et al., 2026)	Homogeneous	$O(M)$
MoT (Proposed)	**Cross-Family**	$O(1)$

[†]RAG shares model-agnostic text rather than stored model-dependent internal states.

model or KV format and thus retain multi-model redundancy. Cross-model methods such as C2C, LSC, and FedRefine, together with closed-form or calibrated mappings (Fu et al., 2026; Dery et al., 2026; Chen et al., 2026c; Heo et al., 2026; Qu et al., 2026), enable state transfer across models or devices but introduce pairwise mapping, shared-space alignment, or federated communication costs. Under stronger compatibility assumptions, DroidSpeak selectively recomputes layers across same-architecture variants, LRAgent separates shared-backbone and LoRA cache components, and ICaRus freezes a common cache-generating encoder to enable identical-cache reuse (Liu et al., 2024; Jeon et al., 2026; Woo et al., 2026); these mechanisms are difficult to extend to arbitrary cross-family models. Vision Wormhole (Liu et al., 2026) instead communicates visual latent tokens through VLM interfaces and is therefore related but not a prefix-KV reuse method.

MoT stores shared prefixes once in the hub-model format and translates them to target formats on demand, reducing cache multiplicity to $O(1)$ rather than scaling with the number of participating models.

Table 3 separates cross-model compatibility from cache multiplicity: cross-model transfer can still retain $O(M)$ model-local caches, whereas MoT retains only the Hub Agent cache.

A.3 Storage Reuse for Long-Context RAG

RAG retrieves query-relevant documents and shares model-agnostic text across heterogeneous models without additional training (Lewis et al., 2020), but each request still requires prefilling and KV-cache construction. Internal-state reuse avoids this repeated work by precomputing representations offline: TurboRAG stores KV caches for retrieved chunks (Lu et al., 2025); CAG preloads a collection into offline caches for query-only online generation (Chan et al., 2025); HCache restores KV caches from stored hidden states (Gao et al., 2025); KVLink uses trainable special tokens and training for chunk-cache reuse (Yang et al., 2025b); and SpecCache selectively reuses cached states while recomputing critical tokens identified by a lightweight speculative model (Wen et al., 2026). However, these stored KV or hidden states remain architecture- and tokenizer-specific, so the same item must be stored separately when the model changes (Lu et al., 2025; Chan et al., 2025; Gao et al., 2025; Wen et al., 2026).

MoT instead stores each shared prefix once as a hub-format KV cache and performs one-hop hub-to-target translation at inference, avoiding per-model prefix-state stores.

Table 4 distinguishes model-agnostic text sharing from model-dependent internal-state reuse: existing cache-based methods require per-model states, whereas MoT retains a single hub-format prefix cache across heterogeneous models.

B Theoretical Analysis

B.1 Assumptions

The following assumption specifies the local regularity condition used in our analysis of translation-error propagation and correction dynamics.

Assumption B.1. *(Local Lipschitz Residual Branch)*
There exists a constant $\delta \geq 0$ such that, for every layer ℓ, the residual branch F_ℓ is locally Lipschitz on the hidden-state region. That is, for any two hidden states $\mathbf{x}$ and $\mathbf{x}'$ in this region,

$$\|F_\ell(\mathbf{x}) - F_\ell(\mathbf{x}')\| \leq \delta \|\mathbf{x} - \mathbf{x}'\|.$$

We call δ the uniform residual-branch Lipschitz constant.

B.2 Shift Delta and Correction Deficit Coefficient

Definition B.2. *(Shift Delta and Correction Deficit Coefficient)*
For the highest target translation layer T, let $\mathbf{s}_T$ denote the translation shift accumulated within the channel set, and let $\mathbf{s}_L$ denote the final shift observed at the last hidden state. We define the shift delta additionally induced by the remaining upper layers after T as $\Delta\mathbf{s}_{T:L} := \mathbf{s}_L - \mathbf{s}_T$. That is, the shift delta is a vector that describes how the residual dynamics after T modify the translation shift.

When $\mathbf{s}_T \neq 0$, we define the anti-shift correction coefficient as

$$\alpha_{T:L} := -\left\langle \Delta\mathbf{s}_{T:L}, \mathbf{u}^{\|} \right\rangle, \qquad \mathbf{u}^{\|} := \frac{\mathbf{s}_T}{\|\mathbf{s}_T\|}.$$

Here, $\mathbf{u}^{\|}$ denotes the unit vector in the translation-shift direction. We also define the orthogonal shift coefficient as

$$\beta_{T:L} := \left\| \Delta\mathbf{s}_{T:L} - \left\langle \Delta\mathbf{s}_{T:L}, \mathbf{u}^{\|} \right\rangle \mathbf{u}^{\|} \right\|.$$

Finally, the correction deficit coefficient is defined as

$$d_{T:L} := \frac{\langle \mathbf{s}_L, \mathbf{u}^{\|} \rangle}{\|\mathbf{s}_T\|}.$$

This value is a normalized projection coefficient that quantifies how much of the translation-shift direction accumulated within the channel set remains in the last hidden state.

B.3 Proof of Proposition 2.2

Proof. Let $\mathbf{h}_\ell$ and $\widehat{\mathbf{h}}_\ell$ denote the hidden states after the ℓ-th layer induced by the native target cache and the translated target cache, respectively, and define $\mathbf{s}_\ell := \mathbf{h}_\ell - \widehat{\mathbf{h}}_\ell$. The shift formed at T_{Start} is $\mathbf{s}_{T_{\text{Start}}} = \mathbf{h}_{T_{\text{Start}}} - \widehat{\mathbf{h}}_{T_{\text{Start}}}$.

This proposition tracks how much the translation-induced shift component generated at T_{Start} can be amplified as it passes through subsequent layers. For each layer $\ell = T_{\text{Start}} + 1, \ldots, L$, assuming that the two hidden-state trajectories propagate through the same target residual recurrence, we can write

$$\mathbf{h}_\ell = \mathbf{h}_{\ell-1} + F_\ell(\mathbf{h}_{\ell-1}), \qquad \widehat{\mathbf{h}}_\ell = \widehat{\mathbf{h}}_{\ell-1} + F_\ell(\widehat{\mathbf{h}}_{\ell-1})$$

Therefore,

$$\mathbf{s}_\ell = \mathbf{s}_{\ell-1} + \left(F_\ell(\mathbf{h}_{\ell-1}) - F_\ell(\widehat{\mathbf{h}}_{\ell-1}) \right).$$

Taking the norm on both sides and applying the triangle inequality together with Assumption B.1, we obtain

$$\|\mathbf{s}_\ell\| \leq \|\mathbf{s}_{\ell-1}\| + \delta\|\mathbf{s}_{\ell-1}\| = (1+\delta)\|\mathbf{s}_{\ell-1}\|.$$

Applying this recursively for $\ell = T_{\text{Start}} + 1, \ldots, L$ gives

$$\boxed{\|\mathbf{s}_L\| \leq (1+\delta)^{L - T_{\text{Start}}} \|\mathbf{s}_{T_{\text{Start}}}\|.}$$

This proves Eq. equation 1. Since L and $\|\mathbf{s}_{T_{\text{Start}}}\|$ are fixed, increasing T_{Start} reduces the exponent $L - T_{\text{Start}}$, and hence the bound on the propagated last-state shift decreases exponentially. □

B.4 Proof of Proposition 2.3

This appendix gives the detailed derivation behind Proposition 2.3. The key point is that, as the highest target translation layer T moves toward the final layer L, fewer upper layers remain available to correct the translation shift. Consequently, the correction-deficit coefficient approaches its terminal value.

Proof. Let $\mathbf{h}_\ell$ and $\widehat{\mathbf{h}}_\ell$ denote the hidden states after the ℓ-th layer induced by the native target cache and the translated target cache, respectively, and define $\mathbf{s}_\ell := \mathbf{h}_\ell - \widehat{\mathbf{h}}_\ell$. For a channel set $\mathcal{C}$, define the target translation layer set and its boundary layers as

$$\mathcal{I}^{\mathrm{Tgt}}(\mathcal{C}) := \{j \mid (i,j) \in \mathcal{C}\}, \qquad T_{\mathrm{Start}} := \min \mathcal{I}^{\mathrm{Tgt}}(\mathcal{C}), \qquad T := \max \mathcal{I}^{\mathrm{Tgt}}(\mathcal{C}).$$

The shift formed at T_{Start} is $\mathbf{s}_{T_{\mathrm{Start}}} = \mathbf{h}_{T_{\mathrm{Start}}} - \widehat{\mathbf{h}}_{T_{\mathrm{Start}}}$.

For each layer $\ell = T_{\mathrm{Start}} + 1, \ldots, T$, assuming that the two hidden-state trajectories propagate through the same target residual recurrence, we can write

$$\mathbf{h}_\ell = \mathbf{h}_{\ell-1} + F_\ell(\mathbf{h}_{\ell-1}), \qquad \widehat{\mathbf{h}}_\ell = \widehat{\mathbf{h}}_{\ell-1} + F_\ell(\widehat{\mathbf{h}}_{\ell-1}).$$

Therefore,

$$\mathbf{s}_\ell = \mathbf{s}_{\ell-1} + \left(F_\ell(\mathbf{h}_{\ell-1}) - F_\ell(\widehat{\mathbf{h}}_{\ell-1})\right).$$

Taking the norm and applying the triangle inequality together with Assumption B.1, we obtain

$$\|\mathbf{s}_\ell\| \le \|\mathbf{s}_{\ell-1}\| + \delta\|\mathbf{s}_{\ell-1}\| = (1+\delta)\|\mathbf{s}_{\ell-1}\|.$$

Applying this recursively for $\ell = T_{\mathrm{Start}} + 1, \ldots, T$ gives

$$\|\mathbf{s}_T\| \le (1+\delta)^{T - T_{\mathrm{Start}}} \|\mathbf{s}_{T_{\mathrm{Start}}}\|. \tag{3}$$

Thus, when T_{Start} and $\|\mathbf{s}_{T_{\mathrm{Start}}}\|$ are fixed, the upper bound on the translation shift $\|\mathbf{s}_T\|$ can increase as T increases.

We now bound the correction opportunity provided by the remaining upper layers after T. By Definition B.2, $\Delta\mathbf{s}_{T:L} = \mathbf{s}_L - \mathbf{s}_T$. Also,

$$\mathbf{s}_L - \mathbf{s}_T = \sum_{\ell=T+1}^{L} (\mathbf{s}_\ell - \mathbf{s}_{\ell-1}),$$

and therefore

$$\|\Delta\mathbf{s}_{T:L}\| \le \sum_{\ell=T+1}^{L} \|\mathbf{s}_\ell - \mathbf{s}_{\ell-1}\|.$$

For each term,

$$\mathbf{s}_\ell - \mathbf{s}_{\ell-1} = F_\ell(\mathbf{h}_{\ell-1}) - F_\ell(\widehat{\mathbf{h}}_{\ell-1}),$$

so by Assumption B.1,

$$\|\mathbf{s}_\ell - \mathbf{s}_{\ell-1}\| \le \delta\|\mathbf{s}_{\ell-1}\|.$$

Moreover, applying the recurrence bound starting from T, we have

$$\|\mathbf{s}_{\ell-1}\| \le (1+\delta)^{\ell-1-T}\|\mathbf{s}_T\|.$$

Therefore,

$$\|\Delta\mathbf{s}_{T:L}\| \le \sum_{\ell=T+1}^{L} \delta(1+\delta)^{\ell-1-T}\|\mathbf{s}_T\|.$$

Computing the geometric sum yields

$$\|\Delta\mathbf{s}_{T:L}\| \le \left((1+\delta)^{L-T} - 1\right)\|\mathbf{s}_T\|.$$

By Definition B.2, the anti-shift correction coefficient is

$$\alpha_{T:L} = -\left\langle \Delta\mathbf{s}_{T:L}, \mathbf{u}^{\parallel} \right\rangle.$$

Thus, by the Cauchy–Schwarz inequality,

$$\alpha_{T:L} \le \left| \left\langle \Delta \mathbf{s}_{T:L}, \mathbf{u}^{\parallel} \right\rangle \right| \le \|\Delta \mathbf{s}_{T:L}\|.$$

Hence,

$$\alpha_{T:L} \le \Big((1+\delta)^{L-T} - 1 \Big) \|\mathbf{s}_T\|. \tag{4}$$

Substituting Eq. equation 3 into this bound gives

$$\alpha_{T:L} \le \Big((1+\delta)^{L-T_{\mathrm{Start}}} - (1+\delta)^{T-T_{\mathrm{Start}}} \Big) \|\mathbf{s}_{T_{\mathrm{Start}}}\|.$$

Therefore, when T_{Start} and $\|\mathbf{s}_{T_{\mathrm{Start}}}\|$ are fixed, this upper bound decreases to 0 as $T \to L$. This formalizes the intuition that late translation leaves fewer remaining upper layers for anti-shift correction.

Finally, we derive the correction-deficit bound. Since $\mathbf{s}_L = \mathbf{s}_T + \Delta \mathbf{s}_{T:L}$ and $\mathbf{s}_T = \|\mathbf{s}_T\| \mathbf{u}^{\parallel}$, we have

$$\langle \mathbf{s}_L, \mathbf{u}^{\parallel} \rangle = \|\mathbf{s}_T\| + \left\langle \Delta \mathbf{s}_{T:L}, \mathbf{u}^{\parallel} \right\rangle = \|\mathbf{s}_T\| - \alpha_{T:L}.$$

Dividing both sides by $\|\mathbf{s}_T\|$, we obtain $d_{T:L} = 1 - \alpha_{T:L}/\|\mathbf{s}_T\|$. Using Eq. equation 4, we get

$$d_{T:L} \ge 1 - \Big((1+\delta)^{L-T} - 1 \Big) = 2 - (1+\delta)^{L-T}.$$

Thus, when L is fixed, the lower bound on $d_{T:L}$ increases as T increases.

When $T = L$, there are no remaining upper layers, so $\Delta \mathbf{s}_{L:L} = \mathbf{s}_L - \mathbf{s}_L = 0$. Therefore, $\alpha_{L:L} = 0$ and $\beta_{L:L} = 0$. Moreover, if $\mathbf{s}_L \neq 0$, then $\mathbf{u}^{\parallel} = \mathbf{s}_L/\|\mathbf{s}_L\|$. Hence, $d_{L:L} = \langle \mathbf{s}_L, \mathbf{s}_L/\|\mathbf{s}_L\| \rangle / \|\mathbf{s}_L\| = 1$. Thus, at the final hidden state, no upper-layer correction opportunity remains, and the correction-deficit coefficient reaches its terminal value. □

B.5 Proof of Proposition 2.4

Proof. By Definition B.2,

$$\alpha_{T:L} = -\left\langle \Delta \mathbf{s}_{T:L}, \mathbf{u}^{\parallel} \right\rangle$$

and

$$\beta_{T:L} = \left\| \Delta \mathbf{s}_{T:L} - \left\langle \Delta \mathbf{s}_{T:L}, \mathbf{u}^{\parallel} \right\rangle \mathbf{u}^{\parallel} \right\|.$$

Therefore, when $\beta_{T:L} > 0$, for some $\mathbf{u}^{\perp} \perp \mathbf{u}^{\parallel}$, we can write

$$\Delta \mathbf{s}_{T:L} = -\alpha_{T:L} \mathbf{u}^{\parallel} + \beta_{T:L} \mathbf{u}^{\perp}$$

When $\beta_{T:L} = 0$, the same expression holds by choosing any $\mathbf{u}^{\perp} \perp \mathbf{u}^{\parallel}$.

By definition, $\mathbf{s}_L = \mathbf{s}_T + \Delta \mathbf{s}_{T:L}$ and $\mathbf{s}_T = \|\mathbf{s}_T\| \mathbf{u}^{\parallel}$. Thus,

$$\mathbf{s}_L = (\|\mathbf{s}_T\| - \alpha_{T:L}) \mathbf{u}^{\parallel} + \beta_{T:L} \mathbf{u}^{\perp}.$$

Since $\mathbf{u}^{\parallel} \perp \mathbf{u}^{\perp}$,

$$\|\mathbf{s}_L\|^2 = (\|\mathbf{s}_T\| - \alpha_{T:L})^2 + \beta_{T:L}^2.$$

Moreover, by Definition B.2,

$$d_{T:L} = \frac{\langle \mathbf{s}_L, \mathbf{u}^{\parallel} \rangle}{\|\mathbf{s}_T\|} = \frac{\|\mathbf{s}_T\| - \alpha_{T:L}}{\|\mathbf{s}_T\|}.$$

Therefore, $\|\mathbf{s}_T\| - \alpha_{T:L} = d_{T:L} \|\mathbf{s}_T\|$. Substituting this into the previous expression gives Eq. equation 2. □

C Proposed Method Details

C.1 Mixture-of-Translators

Prior work on mixture-of-experts has shown that input-dependent specialization and learned routing can improve over applying a single expert uniformly across inputs (Jacobs et al., 1991; Chen et al., 2022; Dikkala et al., 2023). Following this perspective, we analyze how token-level translator routing affects cache translation error and, in turn, the translation shift studied in the main text.

For a fixed channel set $\mathcal{C}$, let $\mathbf{y}_u$ denote the native target-cache block for token u, and let $\widehat{\mathbf{y}}_{u,m} := \mathrm{Tr}_m(\mathbf{z}_u)$ be the output of translator m. We write the corresponding error as $e_{u,m} := \|\widehat{\mathbf{y}}_{u,m} - \mathbf{y}_u\|^2$.

A single translator must use the same translator for every token, giving the best achievable error $\mathcal{E}_{\mathrm{Single}} := \min_m \sum_u e_{u,m}$. In contrast, token-wise selection can choose the best translator separately for each token. We define the resulting specialization gain as

$$\Gamma_{\mathrm{Spec}} := \mathcal{E}_{\mathrm{Single}} - \sum_u \min_m e_{u,m} \geq 0.$$

Thus, $\Gamma_{\mathrm{Spec}} > 0$ whenever different tokens benefit from different translators.

Let $w_{u,m}$ denote the normalized routing weight assigned by MoT to translator m for token u, with zero weight for unselected translators. The learned router need not always select the best translator. We therefore define its routing gap as

$$\mathcal{R}_{\mathrm{Route}} := \sum_u \left(\sum_m w_{u,m} e_{u,m} - \min_m e_{u,m} \right) \geq 0.$$

Proposition C.1 (Token-Wise Translator Specialization)**.** *For a fixed translator pool, the cache mismatch produced by Mixture-of-Translators satisfies*

$$\left\| \widehat{\mathbf{Y}}_{\mathrm{MoT}} - \mathbf{Y} \right\|_F^2 \leq \mathcal{E}_{\mathrm{Single}} - \Gamma_{\mathrm{Spec}} + \mathcal{R}_{\mathrm{Route}}. \tag{5}$$

Consequently, if $\mathcal{R}_{\mathrm{Route}} < \Gamma_{\mathrm{Spec}}$, token-level routing achieves lower cache mismatch than the best single translator from the same translator pool.

Proof. For each token, $\widehat{\mathbf{y}}_u^{\mathrm{MoT}} - \mathbf{y}_u = \sum_m w_{u,m}(\widehat{\mathbf{y}}_{u,m} - \mathbf{y}_u)$. Since the squared Euclidean norm is convex,

$$\left\| \widehat{\mathbf{y}}_u^{\mathrm{MoT}} - \mathbf{y}_u \right\|^2 \leq \sum_m w_{u,m} e_{u,m}.$$

Summing over tokens and using the definitions of Γ_{Spec} and $\mathcal{R}_{\mathrm{Route}}$ gives Eq. equation 5. If $\mathcal{R}_{\mathrm{Route}} < \Gamma_{\mathrm{Spec}}$, its right-hand side is strictly smaller than $\mathcal{E}_{\mathrm{Single}}$. □

We next connect this cache-space result to the translation shift $\mathbf{s}_T$ used in the main analysis. Let Φ_T denote the fixed target computation that maps the translated cache to the hidden state at the highest target translation layer $T := \max\{j \mid (i,j) \in \mathcal{C}\}$. Let κ_T denote its local sensitivity, such that $\|\Phi_T(\mathbf{Y}_1) - \Phi_T(\mathbf{Y}_2)\| \leq \kappa_T \|\mathbf{Y}_1 - \mathbf{Y}_2\|_F$ over the relevant cache region.

Corollary C.2 (Translation-Shift Bound for Mixture-of-Translators)**.** *The translation shift induced by MoT satisfies*

$$\|\mathbf{s}_T\|^2 \leq \kappa_T^2 \left(\mathcal{E}_{\mathrm{Single}} - \Gamma_{\mathrm{Spec}} + \mathcal{R}_{\mathrm{Route}} \right). \tag{6}$$

Under Assumption B.1, this shift further propagates as

$$\|\mathbf{s}_L\| \leq (1+\delta)^{L-T} \kappa_T \sqrt{\mathcal{E}_{\mathrm{Single}} - \Gamma_{\mathrm{Spec}} + \mathcal{R}_{\mathrm{Route}}}. \tag{7}$$

Proof. By the local sensitivity of Φ_T, $\|\mathbf{s}_T\| \leq \kappa_T \|\widehat{\mathbf{Y}}_{\mathrm{MoT}} - \mathbf{Y}\|_F$. Combining this with Proposition C.1 gives Eq. equation 6. After layer T, the two trajectories follow the same target residual recurrence, so Proposition 2.2 gives $\|\mathbf{s}_L\| \leq (1+\delta)^{L-T} \|\mathbf{s}_T\|$, which yields Eq. equation 7. □

Combining Eq. equation 6 with Proposition 2.4 further gives

$$\|\mathbf{s}_L\|^2 \le d_{T:L}^2 \kappa_T^2 \left(\mathcal{E}_{\text{Single}} - \Gamma_{\text{Spec}} + \mathcal{R}_{\text{Route}}\right) + \beta_{T:L}^2. \tag{8}$$

Thus, token-level routing reduces the translation-shift contribution when its specialization gain exceeds its routing gap. The remaining error depends on the upper-layer correction dynamics, motivating the separate Context Correction Loss.

C.2 Context Correction Loss

In practice, instead of explicitly storing and comparing hidden states, we use a heuristic that compares the KV cache formed during replay with the native KV cache. Thus, in implementation, we use the following KV-based form:

$$\sum_{\ell=T_{\text{Start}}+1}^{L} \left(\left\| \widehat{\mathbf{K}}_\ell^{\text{Tgt}} - \mathbf{K}_\ell^{\text{Tgt}} \right\|^2 + \left\| \widehat{\mathbf{V}}_\ell^{\text{Tgt}} - \mathbf{V}_\ell^{\text{Tgt}} \right\|^2 \right). \tag{9}$$

Proposition C.3 (Last-State Shift Reduction by Context Correction Loss)**.** *Suppose that the stacked final-layer key/value projection of the target model has full column rank. Then, reducing the final-layer term of Eq. equation 9 provides a training signal that reduces an upper bound on the last-state shift* $\|\mathbf{s}_L\|$.

Proof. Let the final-layer key and value projection matrices be denoted by $\mathbf{W}_L^K$ and $\mathbf{W}_L^V$, respectively, and define the stacked final-layer key/value projection as

$$\mathbf{W}_L^{KV} := \begin{bmatrix} \mathbf{W}_L^K \\ \mathbf{W}_L^V \end{bmatrix}.$$

The assumption in Proposition C.3 means that $\mathbf{W}_L^{KV}$ has full column rank.

At each layer, the native key and value caches are fixed linear projections of the native hidden state:

$$\mathbf{K}_\ell^{\text{Tgt}} = \mathbf{W}_\ell^K \mathbf{h}_\ell^{\text{Tgt}}, \qquad \mathbf{V}_\ell^{\text{Tgt}} = \mathbf{W}_\ell^V \mathbf{h}_\ell^{\text{Tgt}}.$$

Similarly, the replayed key and value caches are given by

$$\widehat{\mathbf{K}}_\ell^{\text{Tgt}} = \mathbf{W}_\ell^K \widehat{\mathbf{h}}_\ell^{\text{Tgt}}, \qquad \widehat{\mathbf{V}}_\ell^{\text{Tgt}} = \mathbf{W}_\ell^V \widehat{\mathbf{h}}_\ell^{\text{Tgt}}.$$

By the definition of the last-state shift, $\mathbf{s}_L = \widehat{\mathbf{h}}_L^{\text{Tgt}} - \mathbf{h}_L^{\text{Tgt}}$. Therefore, at the final layer,

$$\widehat{\mathbf{K}}_L^{\text{Tgt}} - \mathbf{K}_L^{\text{Tgt}} = \mathbf{W}_L^K \mathbf{s}_L, \qquad \widehat{\mathbf{V}}_L^{\text{Tgt}} - \mathbf{V}_L^{\text{Tgt}} = \mathbf{W}_L^V \mathbf{s}_L.$$

Therefore, the final-layer KV matching error is

$$\left\|\mathbf{W}_L^K \mathbf{s}_L\right\|^2 + \left\|\mathbf{W}_L^V \mathbf{s}_L\right\|^2 = \left\| \begin{bmatrix} \mathbf{W}_L^K \\ \mathbf{W}_L^V \end{bmatrix} \mathbf{s}_L \right\|^2.$$

Since

$$\begin{bmatrix} \mathbf{W}_L^K \\ \mathbf{W}_L^V \end{bmatrix}$$

is fixed by the target model and has full column rank, any nonzero shift $\mathbf{s}_L \neq 0$ cannot vanish in the final-layer key/value space.

Furthermore, because this projection is a fixed linear map that does not change during training, the proportional relationship between the final-layer KV matching error and the last-state shift also remains fixed throughout training. Therefore, reducing the final-layer KV matching error is equivalent to reducing a fixed upper bound on $\|\mathbf{s}_L\|$. Since Eq. equation 9 includes this final-layer term, the KV-based implementation is justified as a heuristic for reducing the last-state shift. □

C.3 BACKBONE TRANSLATOR

The backbone translator directly maps the source context $\mathbf{KV}$ cache into the target cache space for a selected channel set. We adopt the cross-attention-based translation pattern of prior work (Dery et al., 2026), but unlike shared-latent-space approaches, we perform direct source-to-target cache translation without introducing an additional shared latent space.

Keys and values are translated independently. The translator first projects each source cache block into a common translator space, combines information across the selected channels through recurrent cross-attention, and projects the result into the target cache space. The same formulation applies to contiguous and non-contiguous channel sets.

Given a fixed channel set $\mathcal{C}$, let $\mathbf{X}_{\mathcal{C}}^{\mathrm{Src,Ctx}} = (\mathbf{X}_1^{\mathrm{Src,Ctx}}, \ldots, \mathbf{X}_N^{\mathrm{Src,Ctx}})$ denote the source-side cache block selected by $\mathcal{C}$, where $N = |\mathcal{C}|$. Here, the index $r \in \{1, \ldots, N\}$ refers only to the slot order inside the selected channel window, not to an absolute model layer index. Key and value caches are translated independently, so the following description applies to each $X \in \{K, V\}$.

First, each source cache slot is normalized and projected into the translator hidden space:

$$\mathbf{z}_r^X = \phi\big(\mathbf{W}_{\mathrm{In}}^X \, \mathrm{LN}_{\mathrm{In}}^X \left(\mathbf{X}_r^{\mathrm{Src,Ctx}}\right)\big), \qquad r = 1, \ldots, N,$$

where ϕ denotes the GELU activation. The translator then performs recurrent cross-attention over the projected window. It maintains a running hidden state $\mathbf{h}^X$, initialized as $\mathbf{h}_0^X = \mathbf{z}_1^X$. For each slot r, the running hidden state attends to the corresponding projected source cache slot:

$$\mathbf{h}_r^X = \mathrm{CAB}_r^X \left(\mathbf{h}_{r-1}^X, \mathbf{z}_r^X\right), \qquad r = 1, \ldots, N.$$

The cross-attention block is defined as

$$\mathbf{a}_r^X = \mathrm{CrossAttn}\left(\mathrm{LN}_Q(\mathbf{h}_{r-1}^X), \mathrm{LN}_C(\mathbf{z}_r^X), \mathrm{LN}_C(\mathbf{z}_r^X)\right),$$

$$\widetilde{\mathbf{h}}_r^X = \mathbf{h}_{r-1}^X + \mathbf{a}_r^X,$$

$$\mathbf{h}_r^X = \widetilde{\mathbf{h}}_r^X + \mathrm{FFN}\left(\mathrm{LN}_F(\widetilde{\mathbf{h}}_r^X)\right).$$

Thus, in the notation $\mathrm{CrossAttn}(Q, K, V)$, the query Q is the normalized running hidden state, while both the key K and value V are the normalized projected source cache slot. Intuitively, the running hidden state accumulates information from previously visited slots, and each cross-attention step lets it retrieve the information needed from the current source slot. This cross-layer scan can be repeated for multiple translator stages, following the recurrent translator pattern of LSC (Dery et al., 2026).

After the final scan, the hidden states collected across the channel window are concatenated and projected into the target cache space:

$$\mathbf{F}^X = \mathrm{Concat}\left(\mathbf{h}_1^X, \ldots, \mathbf{h}_N^X\right),$$

$$\widehat{\mathbf{X}}_{\mathcal{C}}^{\mathrm{Tgt,Ctx}} = \phi\big(\mathbf{W}_{\mathrm{Out}}^X \, \mathrm{LN}_{\mathrm{Out}}^X \left(\mathbf{F}^X\right)\big), \qquad X \in \{K, V\}.$$

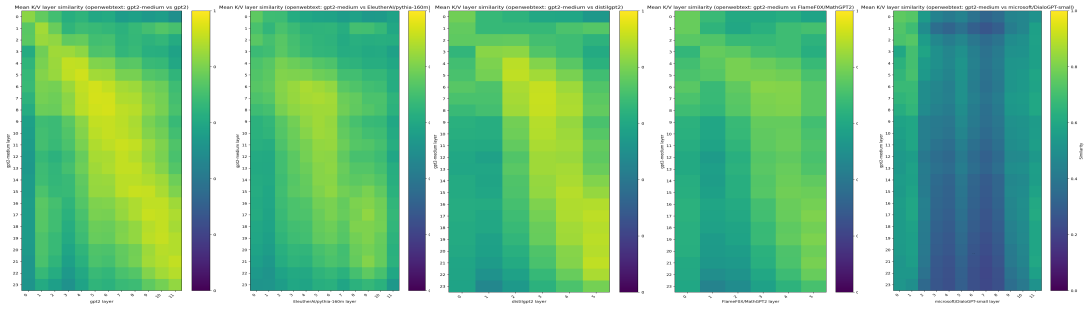
The translated key and value blocks are then combined as

$$\widehat{\mathbf{KV}}_{\mathcal{C}}^{\mathrm{Tgt,Ctx}} = \left(\widehat{\mathbf{K}}_{\mathcal{C}}^{\mathrm{Tgt,Ctx}}, \widehat{\mathbf{V}}_{\mathcal{C}}^{\mathrm{Tgt,Ctx}}\right).$$

This formulation abstracts away implementation-level details such as tensor layout changes and reshaping. The important point is that the backbone translator uses recurrent cross-attention to fuse information across the selected source cache window and directly emits the corresponding target cache block. It serves as the basic translation unit used by MoT: each expert in MoT has the same backbone structure, while the routing function determines how the outputs of multiple backbone translators are selected or combined for each token.

C.4 PRE-TRANSLATION PHASE: CHANNEL MAPPING

Channel mapping is the pre-translation stage that determines the set of layer pairs $\mathcal{C}$ over which cache translation is performed between the source model and the target model. The main focus of this work is to learn accurate cache translation for a given channel set, while the bi-objective problem of jointly



(a) `gpt2-medium` ↔ `gpt2` (b) `gpt2-medium` ↔ `pythia-160m` (c) `gpt2-medium` ↔ `distilgpt2` (d) `gpt2-medium` ↔ `MathGPT2` (e) `gpt2-medium` ↔ `DialoGPT-small`

Figure 15: Preliminary layer-mapping analysis. Layer-wise similarity between heterogeneous model architectures, computed from the mean key/value cache representations obtained on the same OpenWebText samples.

optimizing the size of the channel set is left outside the scope of this paper. Therefore, we assume that $|\mathcal{C}|$ is given, and discuss the corresponding parameter search in the experimental section.

Before specifying the channel-mapping rule, we first conduct a preliminary layer-mapping analysis to examine whether layers of heterogeneous models exhibit a meaningful correspondence. Specifically, we prefill the same OpenWebText samples into both models, compute the mean representations of the key and value caches produced at each layer, and measure the layer-wise similarity for all source-to-target layer combinations. Figure 15 visualizes the resulting similarity matrices for several heterogeneous model pairs.

Figures 15(a)–(d) show that, even when the source and target models have different numbers of layers, high similarity often appears between layers with similar relative depths. In other words, lower-depth source layers tend to form mean KV representations that are more similar to lower-depth target layers, while higher-depth source layers are more similar to higher-depth target layers. This trend appears as a spectral diagonal pattern in the similarity matrix, suggesting that relative depth provides a useful first-order principle for constructing channel mappings.

In particular, panel (b) of Figure 15 shows the result for `gpt2-medium` and `pythia-160m`. Although the two models do not share the same training data or architecture family, they still exhibit high similarity between layer pairs with similar depth ratios. This suggests that the depth-ratio relationship is not merely a phenomenon within the same model family, but may also hold under more general forms of model heterogeneity.

By contrast, Figure 15(e) illustrates an exceptional case. When models differ substantially in their instruction-following behavior or training data distribution, the spectral diagonal pattern can become less pronounced. In such cases, a fixed depth-ratio rule may not always yield the optimal channel mapping. We therefore treat depth-ratio mapping as a simple and effective default strategy, while leaving adaptive channel mapping under stronger data or instruction mismatch as future work.

Motivated by this preliminary analysis, we use Depth-Ratio Mapping (Fu et al., 2026) as the default channel-mapping rule. Intuitively, even when two models differ, lower layers tend to encode lower-level representations, whereas higher layers tend to encode higher-level representations. Therefore, it is natural to connect layer pairs that have similar relative depths.

Depth-Ratio Mapping connects source layers and target layers in order according to their relative depth. That is, it aligns the source model from bottom to top with the target model from bottom to top, so that lower-depth source layers are connected to lower-depth target layers, and higher-depth source layers are connected to higher-depth target layers. Let the resulting ordered channel sequence be $\mathcal{C}_{\mathrm{DR}} = (c_1, c_2, \ldots, c_m)$. Here, each c_r denotes the r-th depth-ratio layer pair, and $m = \min(L^{\mathrm{Src}}, L^{\mathrm{Tgt}})$ is the length of the sequence.

Given $|\mathcal{C}|$, we do not search over all possible subsets of $\mathcal{C}_{\mathrm{DR}}$. Instead, we use only contiguous channel windows as candidates:

$$\mathfrak{C}_{\mathrm{DR}}(|\mathcal{C}|) = \left\{ \{c_q, c_{q+1}, \ldots, c_{q+|\mathcal{C}|-1}\} \,\middle|\, 1 \le q \le m - |\mathcal{C}| + 1 \right\}.$$

This design is motivated by Proposition 2.2 and Proposition 2.3, which show that translating at too early layers can increase propagation error, whereas translating at too late layers can reduce correction opportunity. Therefore, instead of selecting sparse and disconnected channels, choosing a single contiguous window allows us to move the start and end points of the window together and more stably find a balance between the two error dynamics.

The final channel set is selected as the one that minimizes the validation loss among the sliding-window candidates. However, training the full MoT for every candidate channel window is computationally expensive. Therefore, during the channel optimization stage, we use a single backbone translator as a proxy to quickly evaluate the validation loss of each candidate. Since this proxy uses the same backbone structure as the translators inside MoT, it is suitable for measuring the translation difficulty induced by channel location. The detailed structure of the backbone translator is provided in Appendix C.3:

$$\mathcal{C}_\star^{\mathrm{Src}\to\mathrm{Tgt}} = \arg\min_{\mathcal{C}'\in\mathfrak{C}_{\mathrm{DR}}(|\mathcal{C}|)} \mathcal{L}_{\mathrm{Val}}^{\mathrm{Src}\to\mathrm{Tgt}}(\mathcal{C}').$$

C.5 Post-Translation Phase: Cache Replay

Immediately after translation, we need to construct a cache replay procedure so that the selected translation result can be used in the actual long-context setting. After the translation phase, the source model's context cache has been converted into the cache space of the target model, and this translated result should be usable as the actual context cache of the target model. However, the translated cache provides only the context $\mathbf{KV}$ cache corresponding to the target translation layer set $\mathcal{I}^{\mathrm{Tgt}}$. Therefore, in order to obtain a target-side context $\mathbf{KV}$ cache that can be used consistently during subsequent prompt prefill and completion generation, a cache replay procedure that passes the context through the target model again is required.

The most direct approach is to perform full attention at each target layer ℓ, while injecting the translated $\mathbf{KV}$ cache instead of the native $\mathbf{KV}$ cache only at the translated layers. Let N_{Ctx} denote the context length, L^{Tgt} the number of target layers, and d the attention dimension. Since full-attention-based replay allows each context query token to attend to all previous context key-value tokens, its computational complexity is approximately $\mathcal{O}(L^{\mathrm{Tgt}} N_{\mathrm{Ctx}}^2 d)$. In particular, replay is performed before prompt prefill in order to reconstruct the target-side context cache. Thus, for long contexts, the N_{Ctx}^2 term dominates the cost, and the need to pass through multiple target layers makes this procedure expensive.

To reduce this cost, we use Top-S Source-Guided Sparse Attention. Here, S_{Sparse} denotes the number of source-guided key positions to be attended to at each context query position. The key idea is that, during target replay, we do not attend to all context key-value tokens again. Instead, we select only the token positions that the source model regarded as important in the given context and reuse them for target attention. In other words, the selection criterion for sparse attention is not newly learned inside the target model, but is derived from the attention pattern of the source model.

Let the context $\mathbf{KV}$ cache used for replay at target layer ℓ be denoted by $\overline{\mathbf{KV}}_\ell^{\mathrm{Tgt,Ctx}} = (\bar{\mathbf{K}}_\ell^{\mathrm{Tgt,Ctx}}, \bar{\mathbf{V}}_\ell^{\mathrm{Tgt,Ctx}})$. It is defined as

$$\overline{\mathbf{KV}}_\ell^{\mathrm{Tgt,Ctx}} = \begin{cases} \widehat{\mathbf{KV}}_\ell^{\mathrm{Tgt,Ctx}}, & \ell \in \mathcal{I}^{\mathrm{Tgt}}, \\ \mathbf{KV}_\ell^{\mathrm{Tgt,Ctx}}, & \ell \notin \mathcal{I}^{\mathrm{Tgt}}. \end{cases}$$

That is, at the target translation layers, we inject the translated context $\mathbf{KV}$ cache, while at the remaining layers, we use the context $\mathbf{KV}$ cache computed by the native replay process of the target model.

Full-attention replay computes, for each context token position $u \in \{1, \ldots, N_{\mathrm{Ctx}}\}$,

$$\mathbf{o}_{\ell,u}^{\mathrm{Full}} = \mathrm{Attn}\left(\mathbf{Q}_{\ell,u}^{\mathrm{Tgt}}, \bar{\mathbf{K}}_{\ell,1:u}^{\mathrm{Tgt,Ctx}}, \bar{\mathbf{V}}_{\ell,1:u}^{\mathrm{Tgt,Ctx}}\right).$$

Thus, each context query attends to the entire causal context $\{1, \ldots, u\}$.

In contrast, Top-S Source-Guided Sparse Attention uses $\pi(\ell)$ to denote the source guidance layer corresponding to target layer ℓ. Here, $\pi(\ell)$ is a source layer map used to guide the sparse attention

pattern of target layer ℓ. Let the attention score used as context replay guidance at source layer $\pi(\ell)$ be $\mathbf{A}^{\mathrm{Src}}_{\pi(\ell)} \in \mathbb{R}^{N_{\mathrm{Ctx}} \times N_{\mathrm{Ctx}}}$. We omit the head index, and when necessary, interpret this as an attention map averaged over heads. For each context query position u, we define the S_{Sparse} context key positions most strongly attended to by the source attention as

$$\mathcal{N}_\ell(u) = \text{Top-S}\left(\mathbf{A}^{\mathrm{Src}}_{\pi(\ell),u,1:u}, S_{\mathrm{Sparse}}\right).$$

Then, the attention in target replay attends only to the selected source-guided sparse context token set, rather than to the entire context:

$$\mathbf{o}^{\mathrm{Sparse}}_{\ell,u} = \mathrm{Attn}\left(\mathbf{Q}^{\mathrm{Tgt}}_{\ell,u}, \bar{\mathbf{K}}^{\mathrm{Tgt,Ctx}}_{\ell,\mathcal{N}_\ell(u)}, \bar{\mathbf{V}}^{\mathrm{Tgt,Ctx}}_{\ell,\mathcal{N}_\ell(u)}\right).$$

In other words, the full-attention reference set $\{1, \ldots, u\}$ is replaced by the source-guided Top-S context position set $\mathcal{N}_\ell(u)$.

When $S_{\mathrm{Sparse}} \ll N_{\mathrm{Ctx}}$, the attention cost of each target layer is reduced from $\mathcal{O}(N^2_{\mathrm{Ctx}}d)$ to $\mathcal{O}(N_{\mathrm{Ctx}}S_{\mathrm{Sparse}}d)$. Accordingly, the overall replay cost is reduced approximately as

$$\mathcal{O}\big(L^{\mathrm{Tgt}}N^2_{\mathrm{Ctx}}d\big) \quad \longrightarrow \quad \mathcal{O}\big(L^{\mathrm{Tgt}}N_{\mathrm{Ctx}}S_{\mathrm{Sparse}}d\big).$$

In implementation, for stability, some bottom layers can be kept as full-attention layers, while source-guided sparse attention is applied only to the remaining layers. Let L_{Full} denote the number of bottom full-attention layers. Then, the attention reference set is defined as

$$\mathcal{N}_\ell(u) = \begin{cases} \{1, \ldots, u\}, & \ell < L_{\mathrm{Full}}, \\ \text{Top-S}\left(\mathbf{A}^{\mathrm{Src}}_{\pi(\ell),u,1:u}, S_{\mathrm{Sparse}}\right), & \ell \geq L_{\mathrm{Full}}. \end{cases}$$

Overall, the cache replay procedure in the post-translation phase can be summarized as follows. First, the translated target context $\mathbf{KV}$ cache is injected into the target translation layers, while the remaining layers are reconstructed through the target model's native computation. Standard full-attention replay attends to all context tokens and therefore incurs a quadratic cost in N_{Ctx}. In contrast, the proposed Top-S Source-Guided Sparse Attention reduces the replay cost to nearly linear in the context length by attending only to the S_{Sparse} context tokens selected by the source attention pattern. The target model then performs prompt prefill conditioned on the target-side context cache obtained through replay and finally generates the completion.

D Benchmark Prompts

This section summarizes the prompt format used for each benchmark. The main goal of our evaluation is to measure whether a target model can use a long input through a translated cache, instead of directly prefilling the full context. Therefore, each prompt is designed to clearly separate long information from the short query. Long and difficult input, such as a passage, abstract, article, or multi-document evidence, is placed in the context region. The short question, answer cue, or choice instruction required by the target model is placed in the prompt region. In the boxes below, the red context region denotes the part subject to cache translation, while the blue prompt and completion regions denote the part where the target model receives a short query and generates the answer.

The benchmarks are divided into two groups: closed-set QA and extractive QA. Closed-set QA benchmarks have a fixed answer space, and the generated answer is matched to the valid answer set to compute accuracy. Extractive QA benchmarks require the model to generate a short answer grounded in the passage, and the output is evaluated by token-level F1 against gold answers.

D.1 Closed-set QA benchmarks

We use BoolQ, PubMedQA, and MMLU-Redux as closed-set QA benchmarks. Each dataset is represented with three regions: context, prompt, and completion. The context contains the long input information, the prompt contains the short question or choice instruction, and the completion contains the generated answer expected to match one of the valid candidates. The model is evaluated by accuracy after answer matching.

Text 1. BoolQ prompt with sample

Dataset. BoolQ is a binary QA benchmark where the model reads a passage and answers with either yes or no.

Context – Translated Cache Region

```
Read the passage and answer the question with yes or no only.
Passage: ⟨Passage⟩
```

Prompt – Target-Side Query Region

```
Question: ⟨Question⟩
Answer:
```

Completion – Target Output Region

```
yes   or   no
```

Sample.

Context – Translated Cache Region

```
Read the passage and answer the question with yes or no only.
Passage: Windows Movie Maker is a discontinued video editing
software by Microsoft. It is a part of Windows Essentials software
suite and offers the ability to create and edit videos as well as
to publish them on OneDrive, Facebook, Vimeo, YouTube, and Flickr.
```

Prompt – Target-Side Query Region

```
Question: is windows movie maker part of windows essentials
Answer:
```

Completion – Target Output Region

```
yes
```

Text 2. PubMedQA prompt with sample

Dataset. PubMedQA is a biomedical QA benchmark where the model reads an abstract and answers with yes, no, or maybe.

Context – Translated Cache Region

```
Read the abstract and answer the biomedical research question with
yes, no, or maybe only.
Abstract: ⟨Abstract⟩
```

Prompt – Target-Side Query Region

```
Question: ⟨Question⟩
Answer:
```

Completion – Target Output Region

```
yes or no or maybe
```

Sample.

Context – Translated Cache Region

```
Read the abstract and answer the biomedical research question with
yes, no, or maybe only.

Abstract: Halofantrine is a newly developed antimalarial drug
used for the treatment of Plasmodium falciparum malaria. The
introduction of this drug has been delayed because of its possible
side effects, and due to insufficient studies on adverse reactions
in humans. ... Halofantrine has mild to moderate pathological
effects on cochlea histology, and can be considered an ototoxic
drug.
```

Prompt – Target-Side Query Region

```
Question: Is halofantrine ototoxic?
Answer:
```

Completion – Target Output Region

```
yes
```

Text 3. MMLU-Redux prompt with sample

Dataset. MMLU-Redux is a multiple-choice QA benchmark with subject-specific questions. The model generates one answer label from four candidates.

Context – Translated Cache Region

```
Subject: ⟨Subject⟩
Question: ⟨Question⟩
```

Prompt – Target-Side Query Region

```
Choices:
(A) ⟨Choice 1⟩
(B) ⟨Choice 2⟩
(C) ⟨Choice 3⟩
(D) ⟨Choice 4⟩
Answer:
```

Completion – Target Output Region

```
(A)  or  (B)  or  (C)  or  (D)
```

Sample.

Context – Translated Cache Region

```
Subject: abstract algebra
Question: Find all c in Z_3 such that Z_3[x]/(x^3 + cx^2 + 1) is a field.
```

Prompt – Target-Side Query Region

```
Choices:
(A) 0
(B) 2
(C) 1
(D) 3
Answer:
```

Completion – Target Output Region

```
(B)
```

D.2 Extractive QA benchmarks

The extractive QA benchmarks include SQuAD-v1.1 and NewsQA. Both datasets use the same context–prompt–completion structure. A passage is placed in the context region, while a short question is placed in the prompt region. The model generates a short answer, and the output is evaluated by token-level F1 against the gold answers.

Under this abstraction, SQuAD-v1.1 and NewsQA evaluate whether the translated context cache preserves the evidence needed for short-answer extraction. Although the model produces the answer autoregressively, the task is not treated as open-ended generation; instead, it is evaluated as extractive QA.

Text 4. Unified extractive QA prompt

Datasets. SQuAD-v1.1 and NewsQA use the same short-answer extractive QA prompt format. HotpotQA-E in the long-context CAG case study follows the same format with a multi-document context.

Context – Translated Cache Region

```
Read the passage and answer the question briefly. Use a short
phrase from the passage when possible.
Passage: ⟨Passage / article / multi-document evidence⟩
```

Prompt – Target-Side Query Region

```
Question: ⟨Question⟩
Answer:
```

Completion – Target Output Region

```
⟨Short answer⟩
```

Text 5. SQuAD-v1.1 sample

Dataset. SQuAD-v1.1 is an extractive QA benchmark where the model reads a single Wikipedia passage and generates a short answer grounded in the passage.

Context – Translated Cache Region

```
Read the passage and answer the question briefly. Use a short
phrase from the passage when possible.

Passage: Architecturally, the school has a Catholic character.
Atop the Main Building's gold dome is a golden statue of the
Virgin Mary. Immediately in front of the Main Building and facing
it, is a copper statue of Christ with arms upraised with the
legend "Venite Ad Me Omnes". Next to the Main Building is the
Basilica of the Sacred Heart. Immediately behind the basilica is
the Grotto, a Marian place of prayer and reflection. ...
```

Prompt – Target-Side Query Region

```
Question: To whom did the Virgin Mary allegedly appear in 1858 in
Lourdes France?
Answer:
```

Completion – Target Output Region

```
Saint Bernadette Soubirous
```

Text 6. NewsQA sample

Dataset. NewsQA is an extractive QA benchmark where the model reads a news article and generates a short answer grounded in the article.

Context – Translated Cache Region

```
Read the passage and answer the question briefly. Use a short
phrase from the passage when possible.

Passage: City officials said the new transportation plan will
expand bus routes, add protected bike lanes, and increase
late-night service. The first phase of construction is expected to
begin next spring and will focus on downtown corridors. Supporters
argue that the plan will reduce congestion, while critics are
concerned about construction delays and costs.
```

Prompt – Target-Side Query Region

```
Question: When is the first phase of construction expected to
begin?
Answer:
```

Completion – Target Output Region

```
next spring
```

E Additional Results

E.1 Translation Taxonomy

We organize cache translation using a three-level **translation taxonomy** based on the relationship between the source and target models. We use **homogeneous translation** when the source and target correspond to the same model configuration, and therefore share the same model architecture and native KV-cache format. We use **intra-family heterogeneous translation** when the source and target are distinct model configurations within the same model family, such as different model scales whose depth, hidden dimension, attention configuration, or KV-cache shape may differ. Finally, we use **cross-family heterogeneous translation** when the source and target belong to different model families.

Accordingly, heterogeneous translation is an umbrella term that includes both intra-family and cross-family heterogeneous translations. For brevity, the main paper uses *heterogeneous translation* when discussing its primary evaluation across Llama-3.2, Gemma-3, and Qwen-3, while this appendix uses the full translation taxonomy when distinguishing the additional experiments. For example, Qwen2.5 (7B)→Qwen2.5 (0.5B) and GPT-2 (XL)→GPT-2 (Large) are intra-family heterogeneous translations, whereas translations between Llama-3.2, Gemma-3, and Qwen-3 are cross-family heterogeneous translations.

E.2 Implementation of Compared Methods

For C2C-Project[1], Interlat[2], and KVComm[3], we use their official open-source implementations. We integrate these implementations into a common experimental interface for training and evaluation parameters and, where applicable, multi-agent cache management. Since no official open-source implementation of LSC was available, we implemented LSC based on the corresponding paper.

For the multi-agent case study, we use the official MALLM implementation[4] and integrate our cache translation and cache-management modes into its discussion pipeline. Our integrated implementation used for the experiments is available online.[5]

E.3 Translator Training Configuration

All trainable translator-based methods are trained on OpenWebText using the same common optimization setup. We use sequences of 128 tokens, with a 64-token context region followed by a 64-token suffix used for prompt-side language-model supervision; method-specific auxiliary objectives may additionally supervise the translated context or communicated representation. Unless otherwise specified, translators are trained for 500 optimizer steps with a per-micro-batch size of 4 and gradient accumulation over 4 micro-batches, corresponding to an effective batch size of 16 per model stream.

[1] https://github.com/thu-nics/C2C
[2] https://github.com/XiaoDu-flying/Interlat
[3] https://github.com/Zephyroam/KVComm
[4] https://github.com/Multi-Agent-LLMs/mallm
[5] https://anonymous.4open.science/r/MoT-D7E3

Table 5: Homogeneous and intra-family heterogeneous Qwen2.5 translation results. The unshaded Native row denotes the target model without translation; shaded rows denote translation methods. Darker green indicates higher recovery relative to Native.

Translation Category	Method	Cosine Sim	Closed-set QA (Accuracy)				Extractive QA (F1)			Peak Mem. (GiB)
			BoolQ	PubMedQA	MMLU-Redux	Avg	SQuAD	NewsQA	Avg	
Homogeneous 0.5B → 0.5B	Native	N/A	66.0	56.0	35.0	52.0	0.56	0.34	0.45	1.92
	LSC	0.01	6.0	20.0	23.0	16.0	0.06	0.01	0.04	4.09
	C2C-Project	0.02	39.0	22.0	23.0	28.0	0.04	0.01	0.03	1.92
	Interlat	0.98	43.0	42.0	19.0	35.0	0.29	0.25	0.27	1.95
	KVComm	0.98	**62.0**	46.0	27.0	45.0	0.05	0.04	0.04	**1.91**
	MoT	**1.00**	60.0	**52.0**	**35.0**	**49.0**	**0.54**	**0.30**	**0.42**	3.29
Intra-Family Heterogeneous 7B → 0.5B	Native	N/A	66.0	56.0	35.0	52.0	0.56	0.34	0.45	15.25
	LSC	0.01	22.0	9.0	14.0	15.0	0.05	0.01	0.03	18.21
	C2C-Project	0.02	47.0	18.0	24.0	30.0	0.05	0.01	0.03	**15.25**
	Interlat	0.98	41.0	44.0	20.0	35.0	0.29	0.25	0.27	15.27
	MoT	**1.00**	**65.0**	**53.0**	**35.0**	**51.0**	**0.53**	**0.32**	**0.43**	16.62

We use AdamW with learning rate 10^{-4}, weight decay 0.01, 50 warmup steps followed by cosine decay, gradient clipping at 1.0, an OpenWebText shuffle buffer of 50K examples, random seed 42, and `bfloat16` training. These shared hyperparameters are held fixed across the trainable methods. For the baselines, method-specific settings follow the corresponding official implementations where available, or the original paper when no official implementation is available. MoT-specific hyperparameters are selected through the ablations reported in Section E.6.

Compute Worker. All experiments were conducted on a compute worker equipped with two NVIDIA H100 80GB GPUs. Each experiment used only a single H100 GPU; the two GPUs were not combined for any experiment. The same GPU configuration was used across all experiments. Runtime and GPU memory measurements were collected with each experiment running as the sole experiment process on the GPU.

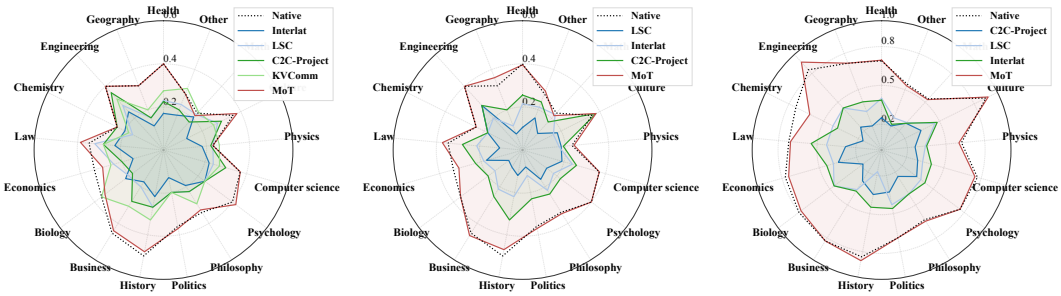
E.4 Homogeneous and Intra-Family Heterogeneous Translations

Table 5 provides supplementary results for homogeneous and intra-family heterogeneous translation within Qwen2.5. We evaluate homogeneous Qwen2.5 (0.5B)→Qwen2.5 (0.5B) translation and intra-family heterogeneous Qwen2.5 (7B)→Qwen2.5 (0.5B) translation. The latter changes model depth, hidden dimension, attention configuration, and KV-cache shape while retaining the same model family.

MoT remains stable across both translations. For homogeneous translation, MoT achieves an average accuracy of 49.0% and an average F1 of 0.42, while for intra-family heterogeneous translation it achieves an average accuracy of 51.0% and an average F1 of 0.43. The strong performance under Qwen2.5 (7B)→Qwen2.5 (0.5B) indicates that MoT can handle substantial cache-space mismatch even within a model family.

Among the baselines, C2C-Project performs poorly on extractive QA, indicating that a projection-only mapping is insufficient to reconstruct the target cache. Interlat partially preserves closed-set QA performance but remains limited on extractive QA, while LSC also degrades substantially. KVComm is evaluated only in the homogeneous setting because its cache-sharing mechanism assumes homogeneous model compatibility.

Figures 16 further breaks down the MMLU-Redux results by 17 categories. MoT remains consistently close to the target-model reference across a broad range of categories, indicating that its performance does not depend on isolated category-specific gains. Across the three levels of the translation taxonomy, the task becomes progressively more challenging from homogeneous to intra-family heterogeneous and cross-family heterogeneous translation. Nevertheless, MoT consistently recovers performance close to Native across all three cases, whereas the other methods degrade as the translation difficulty increases. Consequently, the performance gap between MoT and the competing methods widens with increasing heterogeneity. In settings where translated-cache performance exceeds Native, one possible explanation is a regularization-like effect of cache translation, consistent with prior observations that selective KV communication can emphasize task-relevant information (Shi et al., 2026).



(a) Homogeneous: Qwen2.5 (0.5B)→Qwen2.5 (0.5B). (b) Intra-family heterogeneous: Qwen2.5 (7B)→Qwen2.5 (0.5B). (c) Cross-family heterogeneous: Llama-3.2 (3B)→Qwen-3 (4B).

Figure 16: MMLU-Redux category-wise results for 3 translation categories.

Table 6: Homogeneous translation: `gpt2-large` → `gpt2-large`. The unshaded Native row denotes the target model without translation; shaded rows denote translation methods. Darker green indicates higher recovery relative to Native.

Method	Cosine Sim	Closed-set QA (Accuracy)				Extractive QA (F1)			Peak Mem. (GiB)
		BoolQ	PubMedQA	MMLU-Redux	Avg	SQuAD	NewsQA	Avg	
Native	N/A	51.0	36.0	17.0	35.0	0.50	0.29	0.40	7.06
LSC	0.07	25.0	2.0	**19.0**	15.0	0.06	0.01	0.04	12.67
C2C-Project	0.12	37.0	7.0	9.0	18.0	0.13	0.02	0.07	7.09
Interlat	0.63	**65.0**	38.0	15.0	39.0	0.16	**0.04**	0.10	7.18
KVComm	0.82	17.0	2.0	4.0	8.0	0.02	0.00	0.01	**6.80**
MoT	**0.89**	62.0	**45.0**	18.0	**42.0**	**0.42**	0.01	**0.22**	13.54

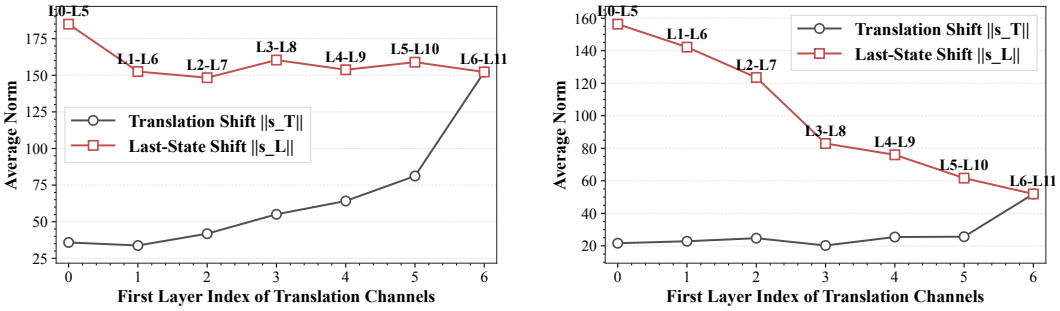
Table 6 reports the homogeneous translation result for `gpt2-large`→`gpt2-large`. Among the translation methods, MoT achieves the strongest overall performance, obtaining the highest average accuracy of 42.0% and the highest average F1 of 0.22. It also achieves the highest cosine similarity of 0.89, indicating that the translated cache is better aligned with the target cache space than those produced by the baselines. In closed-set QA, MoT obtains the best PubMedQA score and remains competitive on BoolQ and MMLU-Redux, while in extractive QA it substantially improves SQuAD over the other translation methods. Although Interlat achieves the best BoolQ score, its extractive QA performance remains limited, suggesting that last-hidden-state communication does not sufficiently preserve the token-level evidence needed for answer extraction. Similarly, C2C-Project, KVComm, and LSC show limited F1, indicating that projection-only mapping, selective KV sharing, or global shared-space alignment is insufficient for robust cache reuse under this homogeneous translation. Overall, MoT provides the most balanced behavior across closed-set and extractive QA.

Table 7 further evaluates intra-family heterogeneous translation, `gpt2-xl`→`gpt2-large`, where the source and target models belong to the same GPT-2 family but differ in depth and representation size. For this more challenging translation, MoT again achieves the highest cache similarity of 0.89, the best average accuracy of 42.0%, and the best average F1 of 0.22 among the translation methods. Compared with the baselines, MoT improves both closed-set QA and extractive QA: it obtains the best BoolQ, PubMedQA, MMLU-Redux, SQuAD, and average scores, while matching the best NewsQA score. These results indicate that MoT remains effective even when a larger source cache must be mapped into a smaller and structurally different target cache space. We omit KVComm for this heterogeneous translation because it assumes homogeneous KV cache compatibility and does not support heterogeneous source and target architectures.

Across both GPT-2 translations, MoT consistently provides stronger cache alignment and more robust downstream performance than the competing translation methods. The gains are especially clear on extractive QA, where accurate preservation of context information is critical. MoT introduces additional peak memory due to the use of multiple translators, but this is a marginal overhead relative to the benefit of improved cache alignment and stable downstream performance. Together with the Qwen2.5 results above, these experiments provide additional evidence that MoT remains effective

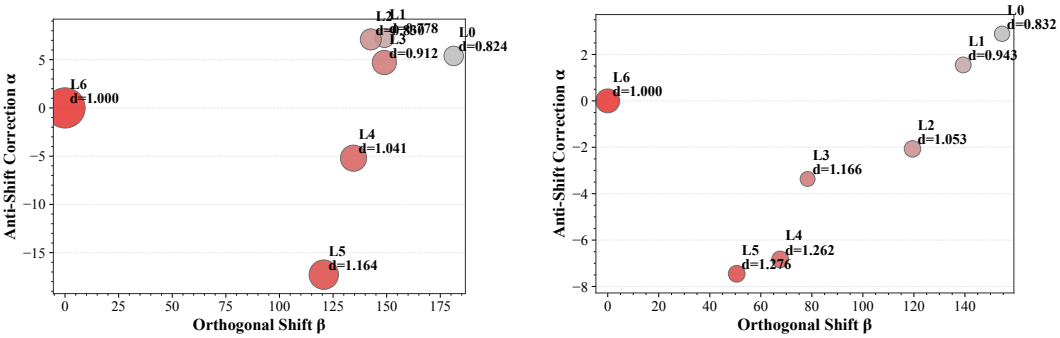
Table 7: Intra-family heterogeneous translation: `gpt2-xl` → `gpt2-large`.

Method	Cosine Sim	Closed-set QA (Accuracy)				Extractive QA (F1)			Peak Mem. (GiB)
		BoolQ	PubMedQA	MMLU-Redux	Avg	SQuAD	NewsQA	Avg	
Native	N/A	51.0	36.0	17.0	35.0	0.50	0.29	0.40	10.18
LSC	0.10	35.0	4.0	17.0	19.0	0.07	0.00	0.03	18.67
C2C-Project	0.11	45.0	14.0	13.0	24.0	0.13	**0.02**	0.08	**10.21**
Interlat	0.58	55.0	36.0	18.0	36.0	0.12	0.02	0.07	10.36
MoT	**0.89**	**61.0**	**46.0**	**19.0**	**42.0**	**0.43**	0.02	**0.22**	16.66



(a) Single Backbone & Prompt LM Loss. (b) MoT & Context Correction Loss.

Figure 17: Shift improvement.



(a) Single Backbone & Prompt LM Loss. (b) MoT & Context Correction Loss.

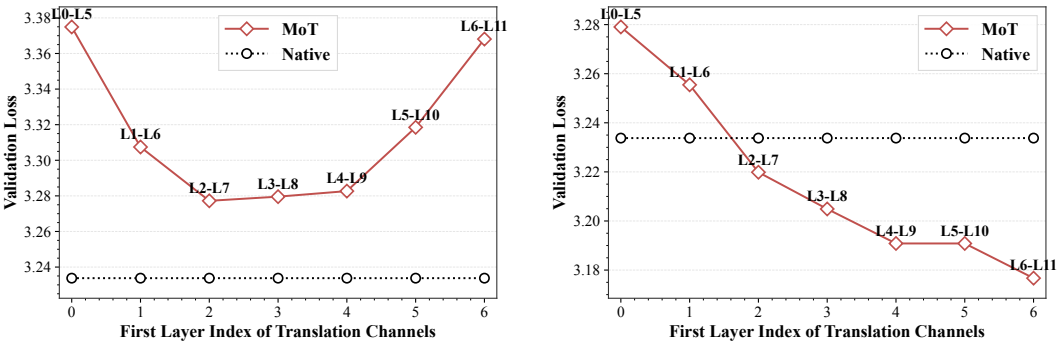
Figure 18: Correction improvement from layer L0 to L6.

in both homogeneous and intra-family heterogeneous translations, complementing the cross-family heterogeneous translations evaluated in Section 4.2.

E.5 Root Cause Analysis

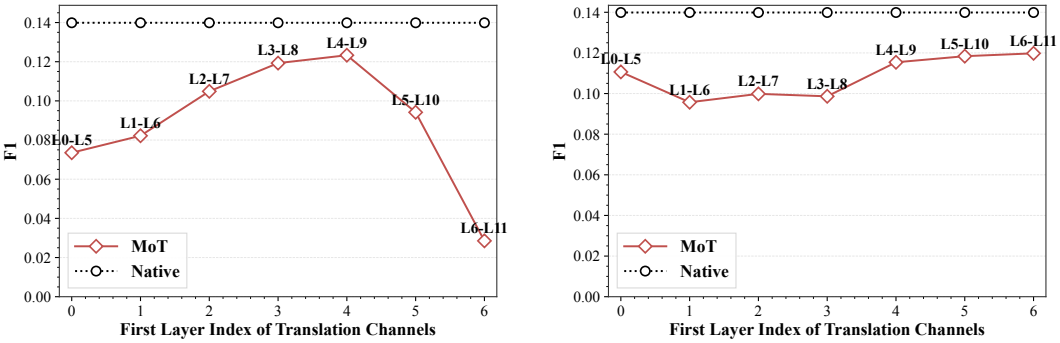
Shifts and Correction. Figure 17 and Figure 18 reorganize the shift and correction results shown separately in Figure 4, Figure 10, Figure 5, and Figure 11. These appendix figures place the before-and-after results side by side for easier comparison between the setting before applying the proposed method, Single Backbone & Prompt LM Loss, and the setting after applying it, MoT & Context Correction Loss.

Similarly, Figure 19 and Figure 20 show the improvements in validation loss and F1 score. The U-shaped loss curve changes substantially into a sharply decreasing trend, and the F1 curve also becomes flat across the entire layer range without large variation, rather than exhibiting an inverted U-shape. This can likewise be interpreted as a consequence of reducing both the initial translation shift $\|\mathbf{s}_T\|$ and the last-state shift $\|\mathbf{s}_L\|$.



(a) Single Backbone & Prompt LM Loss. (b) MoT & Context Correction Loss.

Figure 19: Validation loss improvement.



(a) Single Backbone & Prompt LM Loss. (b) MoT & Context Correction Loss.

Figure 20: F1 score improvement.

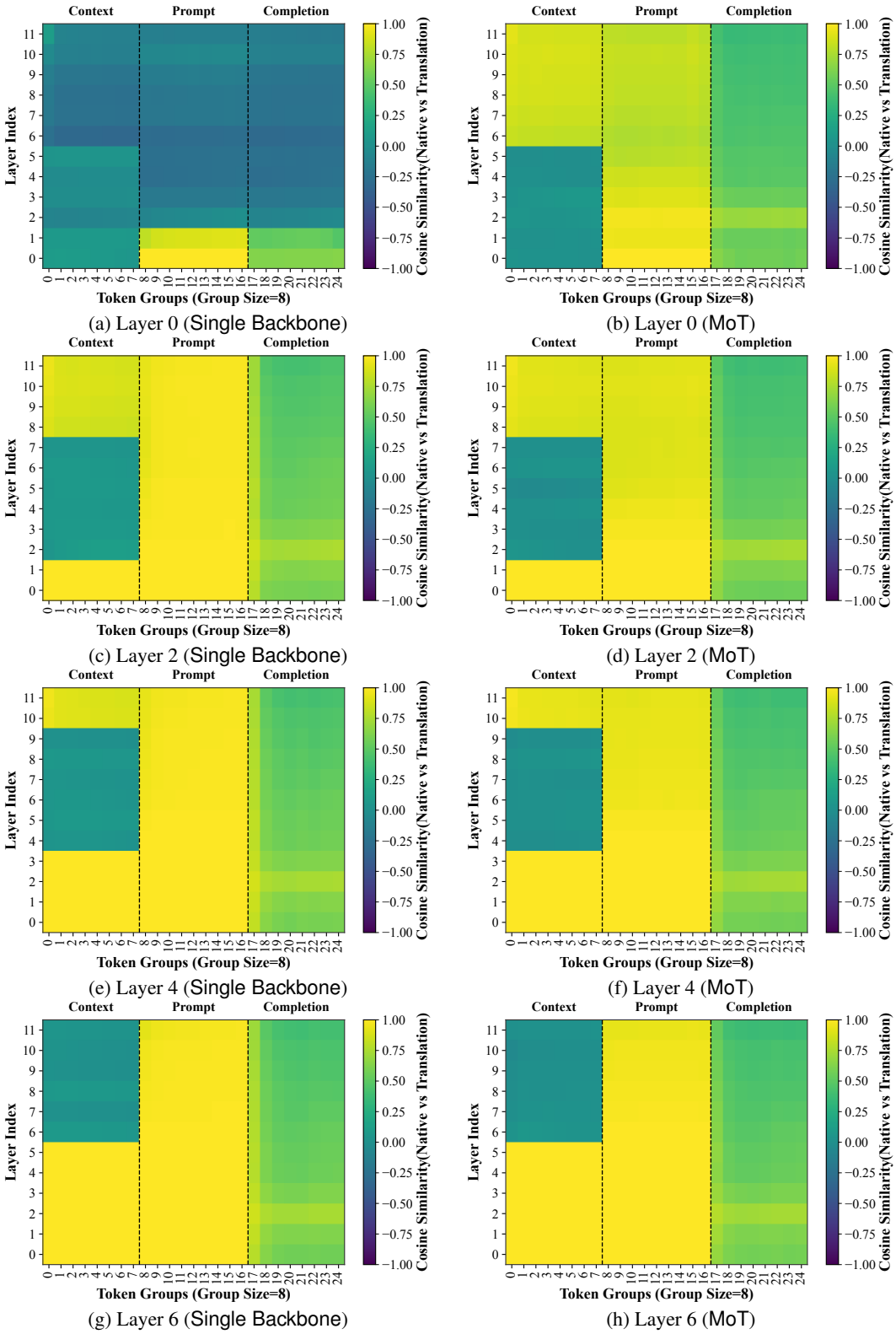
Layer-wise Translated Error Propagation and Correction. In Figure 6, we examined the KV cache similarity heatmap between Native and MoT when translation starts at layer 3, corresponding to the layer range [3,8]. This analysis confirmed the phenomena of error propagation and correction at the KV level. Figure 21 provides an overall heatmap overview comparing the trends in KV similarity before and after applying MoT, while sliding the translation start layer over 0, 2, 4, and 6.

The most noticeable change appears between Figure 6(a) and Figure 6(b): under Single Backbone, severe error propagates across all regions, whereas MoT substantially alleviates this propagation. In the context-token region above the translation window in Figure 6(c–f), the corresponding regions in (d) and (f) become slightly brighter than those in (c) and (e), which indirectly indicates improved correction. In particular, when focusing on token group 0, both Single Backbone and MoT show a tendency for correction to become stronger near the top layers.

Figure 22(a) shows the inverted U-shaped similarity trend that appears especially in the context region, whereas Figure 22(b) shows that the error propagation effect is almost eliminated under MoT. This can be attributed to the large reduction in $\|\mathbf{s}_T\|$ observed earlier. However, at the final translation-layer index 6, correction-deficit error still remains, which is consistent with the analysis in Proposition 2.3 that the correction deficit increases to 1 at the final layer. Note that this does not mean that correction itself has failed.

E.6 Ablation

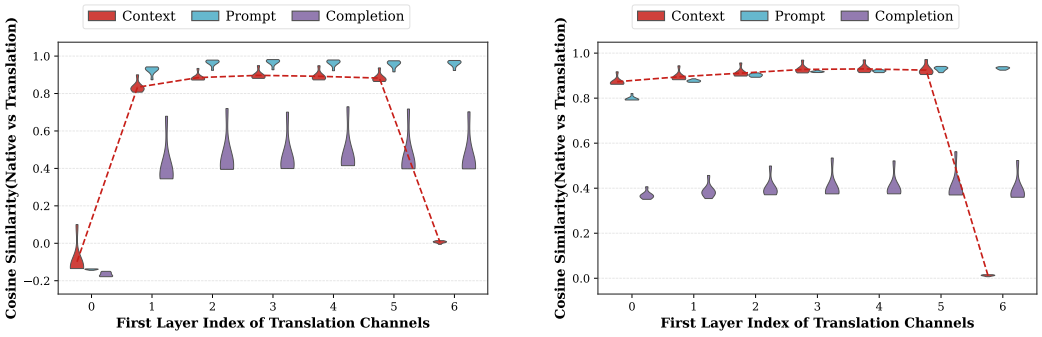
Top-K and Number of Translators. Table 8 compares F1 and GPU Peak Memory under different routing and translator-count settings. The (1,1) row corresponds to Single Backbone and is included only as a reference point, while the remaining rows are MoT configurations used for hyperparameter selection. Across the MoT configurations, the best F1 is obtained either with (1,2) or (2,4) depending on the translation, indicating that increasing Top-K and the number of translators does not consistently yield a clear performance advantage. In contrast, GPU Peak Memory increases mainly with the number



(a) Layer 0 (Single Backbone)
(b) Layer 0 (MoT)
(c) Layer 2 (Single Backbone)
(d) Layer 2 (MoT)
(e) Layer 4 (Single Backbone)
(f) Layer 4 (MoT)
(g) Layer 6 (Single Backbone)
(h) Layer 6 (MoT)

Figure 21: Error propagation improvement.

of translators, and (1,2) consistently uses the least memory among the MoT configurations. Therefore, we conservatively choose $(\text{Top-}K, N_{\text{Tr}}) = (1, 2)$ as the default configuration.



(a) KV Similarity (Single Backbone)

(b) KV Similarity (MoT)

Figure 22: Overall error propagation improvement.

Translation	(Top-K, N_{Tr})	F1	GPU Peak Memory (GiB)
gpt2→gpt2-medium	(1,1)	0.34	3.678
	(1,2)	0.34	**5.131**
	(1,4)	0.35	7.960
	(2,4)	**0.35**	7.959
gpt2-medium→gpt2	(1,1)	0.12	3.442
	(1,2)	0.13	**4.835**
	(1,4)	0.12	7.526
	(2,4)	**0.14**	7.525
Qwen2.5-0.5B→Qwen2.5-1.5B	(1,1)	0.53	10.280
	(1,2)	**0.54**	**12.828**
	(1,4)	0.53	17.898
	(2,4)	0.52	17.899
Qwen2.5-1.5B→Qwen2.5-0.5B	(1,1)	0.39	10.106
	(1,2)	**0.40**	**12.530**
	(1,4)	0.36	17.321
	(2,4)	0.37	17.322
opt-125m→opt-350m	(1,1)	0.05	3.524
	(1,2)	0.05	**4.975**
	(1,4)	0.05	7.806
	(2,4)	**0.05**	7.806
opt-350m→opt-125m	(1,1)	0.04	3.286
	(1,2)	**0.05**	**4.681**
	(1,4)	0.04	7.371
	(2,4)	0.04	7.371

Table 8: F1 and peak memory for Top-K routing over N_{Tr} translators.

Channel Mapping. To examine the effect of the channel-mapping strategy, we compare Terminal Mapping and Depth-Ratio Mapping. Terminal Mapping concentrates the channels near the final layers of the target model, whereas Depth-Ratio Mapping connects source and target layers that have similar relative depths.

As shown in Figure 23, Depth-Ratio Mapping achieves higher Gen F1 than Terminal Mapping in most settings. In particular, when the Channel Ratio is 1.0, Terminal Mapping suffers a large drop in F1, whereas Depth-Ratio Mapping maintains relatively high F1. This shows that connecting layers with similar depth ratios provides more stable cache translation than concentrating channels in the terminal region.

Therefore, in the remaining experiments, we use Depth-Ratio Mapping as the default channel-mapping strategy.

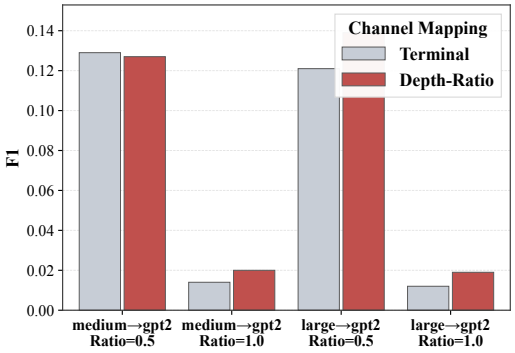


Figure 23: Channel mapping comparison : Terminal vs. Depth-Ratio.

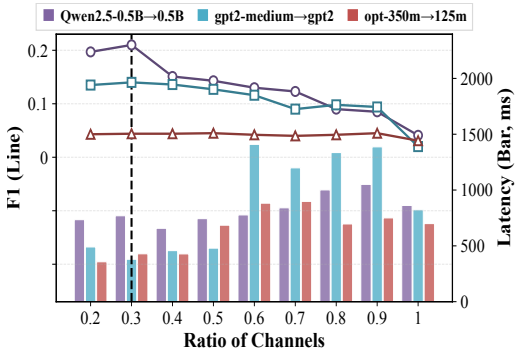


Figure 24: Effect of the translation-channel ratio ($ChannelRatio$).

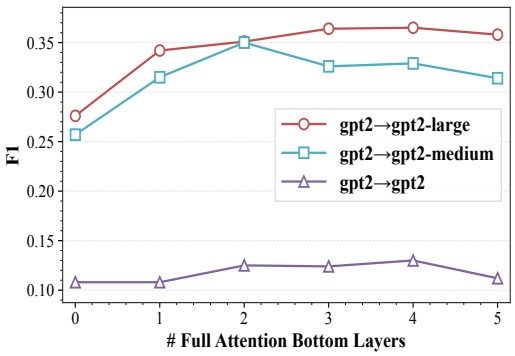


Figure 25: Effect of the number of full attention layers at the bottom.

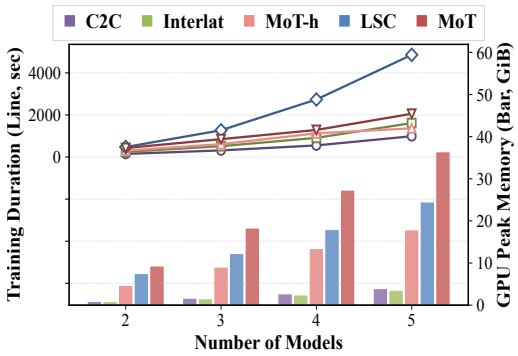
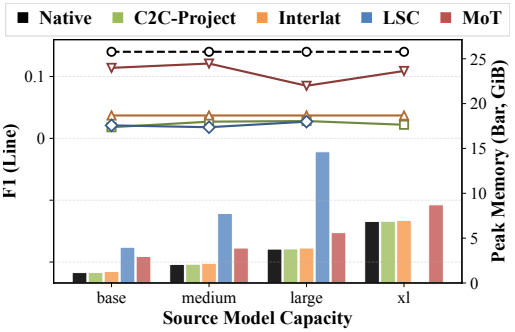


Figure 26: Training scalability of GPT-2 with an increasing number of models.

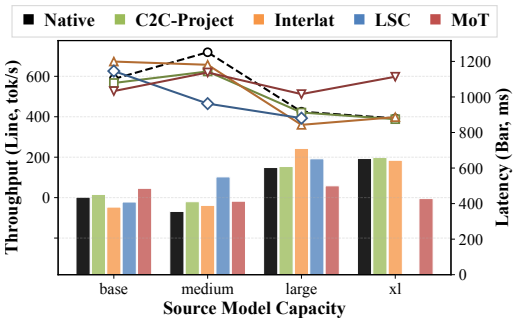
Ratio of Translation Channels. To determine the optimal size of the channel set, we vary the ratio of channels used for actual translation among all Depth-Ratio channel candidates and measure both validation performance and latency. As shown in Figure 24, F1 generally decreases and latency increases as the channel ratio becomes larger. This indicates that using more channels does not necessarily lead to better translation; rather, it may increase the likelihood of including propagation error or correction-deficit error, while also increasing the translation cost.

Although a low channel ratio is advantageous in terms of both accuracy and latency, using too small a ratio can limit the available translation channels and make performance unstable depending on the model and task. Therefore, we use $ChannelRatio = 0.3$ as the default setting, since it maintains high F1 stably across multiple models while keeping the latency increase small.

Number of Full Attention Layers at Bottom. Top-S Source-Guided Sparse Attention reduces the replay cost, but small perturbations in the lowest target layers can substantially disrupt the subsequent representation mapping. Therefore, it is necessary to keep full attention in some bottom layers. To verify this, we vary the number of full attention layers at the bottom and measure F1 in three settings with target models of 12, 24, and 36 layers: (`gpt2→gpt2`, `gpt2→gpt2-medium`, and `gpt2→gpt2-large`). As shown in Figure 25, even as the target model becomes deeper, most of the performance gain occurs when full attention is applied to only a small number of bottom layers, and converting additional bottom layers to full attention provides limited benefit. This is consistent with prior analysis showing that representation-mapping collapse caused by perturbations is most pronounced in the lowest layers (Zheng et al., 2025). Therefore, in this work, we set the number of bottom full-attention layers to $L_{\text{Full}} = 2$ to balance efficiency and accuracy. This choice minimizes the additional cost of full-attention replay while mitigating the loss of lower-layer mapping caused by sparse replay.

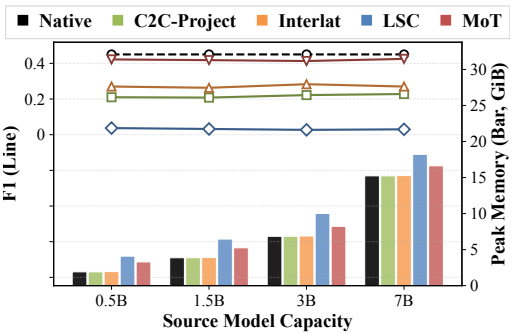


(a) F1 and peak inference memory.

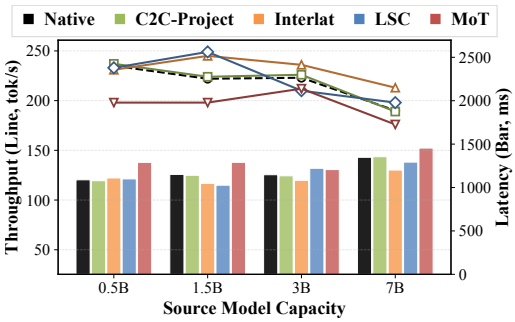


(b) Inference throughput and latency.

Figure 27: Scalability of the GPT-2 model family with increasing source-model capacity.

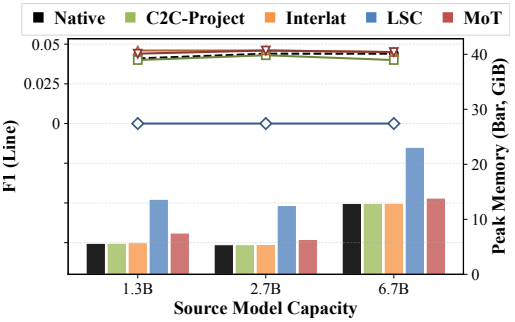


(a) Peak memory and F1.

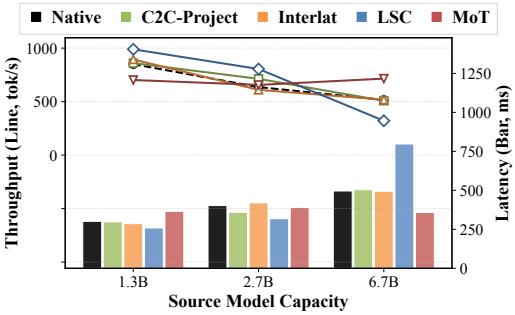


(b) Latency and throughput.

Figure 28: Scalability of the Qwen2.5 model family with increasing source-model capacity.



(a) Peak memory and F1.



(b) Latency and throughput.

Figure 29: Scalability of the OPT model family with increasing source-model capacity.

E.7 SCALABILITY

This appendix provides additional scalability analyses that complement the main benchmark results. We study two axes of scalability. First, we evaluate training scalability as the number of models in the model pool increases. Second, we evaluate capacity scalability by fixing the target model and increasing the capacity of the source model within the same model family. The purpose of these experiments is not to show that larger source models always improve downstream accuracy, but to examine whether translation quality, peak memory, latency, and throughput remain stable as the model pool grows or model heterogeneity increases.

Model Capacity. We next evaluate capacity scalability by fixing the target model and increasing only the source-model capacity. This is a challenging setting because the target cache space remains fixed while the source representation becomes larger and potentially more mismatched. Empirically,

we found this source-scaling regime to be more difficult than the reverse regime where the source model is fixed and the target model capacity increases. Therefore, this experiment directly tests whether each method can translate increasingly larger source caches into the same target cache space without degrading downstream quality.

For the GPT-2 family, we fix the target model to `gpt2` and increase the source model from `gpt2` to `gpt2-xl`. As shown in Figure 27, MoT keeps F1 closest to Native even as the source model capacity increases. C2C-Project remains close to Native in peak memory and latency, but its F1 drops substantially, indicating that simple projection is not sufficient for preserving the information needed by the target model. Interlat and LSC also fail to preserve F1 consistently, and LSC becomes unstable in the largest `gpt2-xl` setting. In contrast, MoT requires additional peak memory due to the translator module, but its F1 degradation is limited and its latency and throughput remain stable in larger-capacity regimes.

Figure 28 shows the same analysis for the Qwen2.5 model family. The target model is fixed to `Qwen2.5-0.5B`, while the source model increases from `Qwen2.5-0.5B` to `Qwen2.5-7B`. Across this range, MoT maintains F1 closest to Native. The F1 of MoT remains stable as source capacity increases, while C2C-Project, Interlat, and especially LSC show much larger degradation. In terms of peak memory, MoT uses additional memory compared with Native, but this overhead remains nearly constant across source capacities. Thus, the relative impact of the translator overhead becomes smaller as the source model grows. Latency and throughput also remain stable, suggesting that the overhead of MoT does not accumulate with source-model capacity.

Figure 29 shows the capacity-scaling behavior for the OPT model family. The target model is fixed to `opt-125m`, and the source model is scaled up to `opt-1.3b`, `opt-2.7b`, and `opt-6.7b`. The `opt-125m` source result is excluded from the figure because it behaves as an outlier. Homogeneous methods such as KVComm are also excluded from this intra-family heterogeneous capacity-scaling comparison. In this setting, MoT again preserves F1 close to Native across source capacities. While the absolute F1 values of the OPT family are lower than those of Qwen2.5, MoT remains the most consistent method in terms of quality preservation. LSC collapses across all OPT capacity settings, whereas C2C-Project and Interlat show less severe degradation but are less consistent than MoT.

Overall, the capacity-scaling experiments show that MoT remains stable when the source model becomes larger while the target model is fixed. C2C-Project can be memory- and latency-efficient, but often fails to preserve downstream F1. Interlat provides stable system-level metrics but is less reliable in quality preservation. LSC suffers from quality collapse in several large-capacity settings, likely due to the difficulty of maintaining a global shared latent space across increasingly heterogeneous model pairs. By contrast, MoT consistently offers the best F1 preservation, with a mostly constant translator-related memory overhead and stable latency-throughput behavior. These results support the use of MoT in settings where a fixed target model must reuse or communicate with caches produced by larger source models.

F Details of Case Studies

F.1 Multi-Agent Reasoning

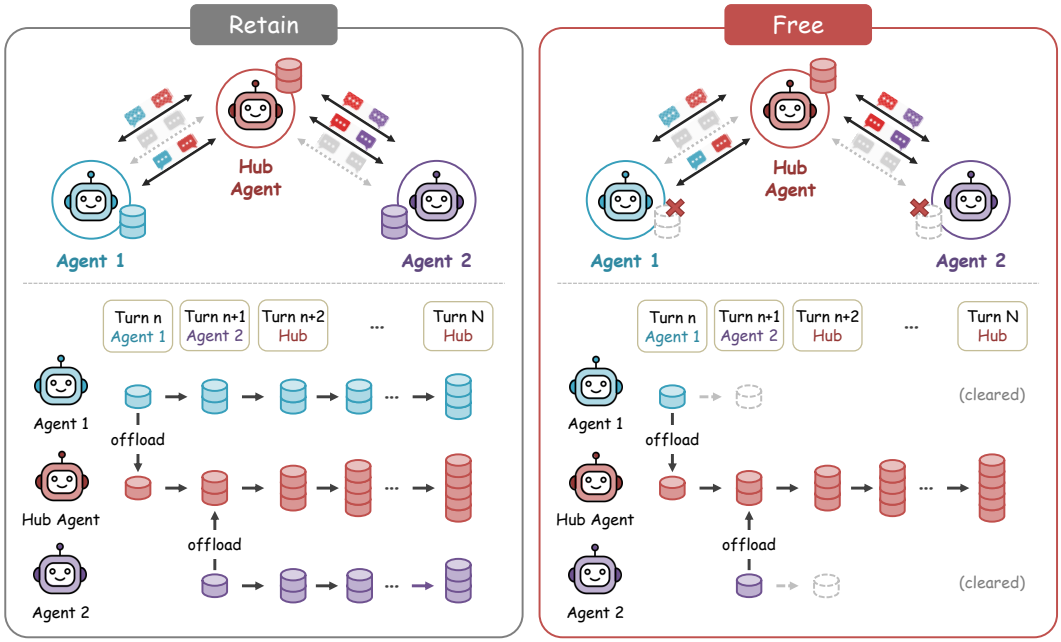


Figure 30: Comparison of Retain and Free cache management strategies in multi-agent reasoning.

We evaluate StrategyQA under two multi-agent settings: heterogeneous 10-agent reasoning with alternating Llama-3.2 (3B) and Qwen-3 (4B) agents, and homogeneous agent scalability with Llama-3.2 (3B)→Llama-3.2 (3B) translation. Both use the same MALLM discussion protocol (Becker et al., 2025) and cache-management strategies described below.

MALLM Configuration. We fix the four major MALLM components across both multi-agent experiments. The Expert persona generator assigns task-specific and complementary roles to the participating agents. The Simple response generator lets an agent propose an initial solution and subsequently agree with or revise the current solution. The Memory discussion paradigm exposes the accumulated discussion history to subsequent agents; in our setting, this history is the state whose KV cache is transferred between agents. Supermajority Consensus is used as the decision protocol, requiring more than 66% agreement on the Yes/No answer.

The multi-agent pipeline and the Expert and Simple prompt templates follow the official MALLM implementation.[6] We integrate cache translation and Retain/Free cache management while otherwise preserving this configuration. We disable MALLM's optional Chain-of-Thought prompting to isolate cache translation and memory-scaling behavior.

Cache Strategies. In addition to the MALLM discussion configuration, we introduce two KV-cache management strategies, Retain and Free, to isolate the effect of cache lifecycle management on agent scalability. Figure 30 conceptually illustrates the two strategies.

In Retain, each agent keeps its resident KV cache after completing its turn. When the accumulated discussion state is transferred to another agent, the already shared prefix can be retained and only the target-missing cache suffix needs to be transferred. However, the resident KV caches of multiple agents remain simultaneously allocated, causing memory usage to grow with the number of agents.

[6] https://github.com/Multi-Agent-LLMs/mallm

Table 9: Training cost and inference efficiency for heterogeneous 10-agent reasoning. KV/1K normalizes KV usage by successfully generated discussion tokens.

	One-Time Training			Inference Efficiency & Memory Breakdown					
Method	**Params.**	**Time (h)**	**Peak (GiB)**	**Model (GiB)**	**Trans. (GiB)**	**KV (GiB)**	**KV/1K (GiB)**	**Succ. Turns**	**Acc.**
C2C-Project	8.405M	0.213	16.67	13.477	0.016	0.242	0.631	3.50	0.00
Interlat	111.747M	0.431	23.78	13.477	0.208	0.826	0.686	12.10	0.65
LSC	3.528B	0.483	46.67	13.477	6.571	0.281	0.634	3.80	0.00
MoT	2.192B	2.350	38.00	13.477	4.082	0.456	**0.235**	**12.95**	**0.70**

In contrast, Free keeps the Hub Agent cache resident but clears a non-hub agent's local KV cache after its state has been offloaded. Subsequent agents reconstruct the required discussion state from the translated cache routed through the Hub Agent. By releasing non-hub resident caches after handoff, Free reduces the number of simultaneously active KV caches and targets memory scalability as the number of agents increases.

Heterogeneous 10-Agent Reasoning. We evaluate 10 agents alternating between Llama-3.2 (3B) and Qwen-3 (4B) on 20 StrategyQA examples, with up to 20 agent turns and 256 generated tokens per turn. C2C-Project, Interlat, and LSC use Retain, whereas MoT uses Free. Table 9 reports the one-time training cost and inference-time memory breakdown; its accuracy and normalized KV results are also visualized in Figure 2.

Raw KV usage is not directly comparable when methods produce different discussion lengths: a failed agent can terminate early and therefore allocate less KV cache. We regard a turn as successful only when the agent produces a valid response choice required by the discussion task—*Yes* or *No* for StrategyQA—and the response passes verification. A turn that fails to produce such a valid choice is treated as failed.

Let $\bar{M}_{\mathrm{KV}}$ be the mean KV usage of a run, $\mathcal{T}_e^+$ the successful turns of example e, and $n_{e,t}$ the number of completion tokens generated at successful turn t. We report token-normalized KV usage as

$$M_{\mathrm{KV/1K}} = \frac{1}{|\mathcal{E}^+|} \sum_{e \in \mathcal{E}^+} \frac{1000\, \bar{M}_{\mathrm{KV}}}{\sum_{t \in \mathcal{T}_e^+} n_{e,t}},$$

where $\mathcal{E}^+$ contains examples with at least one successful turn. The **Succ. Turns** column reports

$$\frac{1}{|\mathcal{E}^+|} \sum_{e \in \mathcal{E}^+} |\mathcal{T}_e^+|.$$

This normalization removes the apparent memory advantage obtained simply by terminating the discussion early.

The resulting efficiency picture is clear. C2C-Project and LSC average only 3.50 and 3.80 successful turns and achieve zero accuracy; their small raw KV values therefore largely reflect short failed discussions. After normalization, MoT has the lowest KV cost by a wide margin: 0.235 GiB/1K tokens versus 0.631–0.686 for the baselines, while also achieving the highest accuracy (0.70) and the largest average number of successful turns (12.95). Thus, MoT combines the strongest reasoning quality with the best normalized KV efficiency in the heterogeneous 10-agent setting.

MoT does incur the largest training time (2.35 h) and substantially more trainable parameters than lightweight baselines, but this is a one-time cost. We release the trained translator checkpoints[7], so checkpoint users can avoid this training stage entirely. Even when training is required, the cost is hours-scale and therefore minor relative to the multi-day timescale of LLM pretraining or finetuning. More importantly, minimizing this one-time cost does not recover the same downstream behavior: none of the compared alternatives matches the accuracy and normalized KV efficiency obtained by MoT with Context Correction Loss.

Interlat is the strongest baseline at 10 agents with 0.65 accuracy, but its normalized KV cost is 0.686 GiB/1K tokens, nearly three times that of MoT. The following homogeneous agent-scalability experiment further shows that its accuracy decreases as the agent count grows, whereas MoT remains substantially more stable.

[7] https://anonymous-hf.com/a/tkelnedcist7

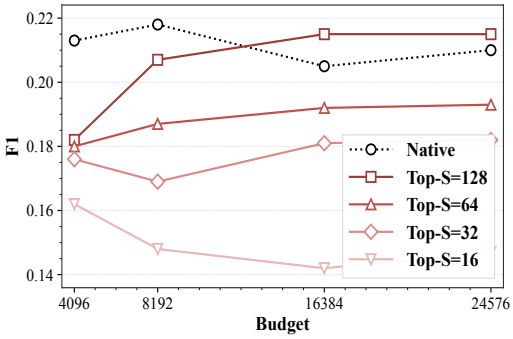


Figure 31: Effect of Top-S for source-guided sparse attention across long-context budgets.

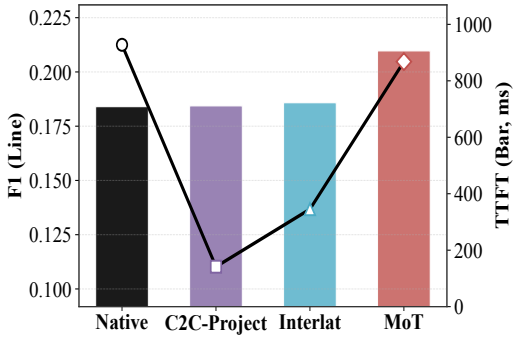


Figure 32: Average F1 and TTFT for long-context CAG.

Homogeneous Agent Scalability. We separately evaluate agent-count scalability using homogeneous Llama-3.2 (3B)→Llama-3.2 (3B) cache translation, increasing the number of agents from 2 to 16 in increments of two. The maximum number of agent turns is twice the number of agents, and each configuration is evaluated on 20 shuffled StrategyQA examples with generation limited to 256 new tokens.

We report the same successful-token-normalized KV memory defined above to account for differences in discussion length caused by early failures. The resulting accuracy and memory trends are shown in Figure 13 and discussed in Section 5.

F.2 Long-Context CAG

Long-Context Specific Ablation. Figure 31 shows the F1 trend of MoT according to the Top-S size used in source-guided sparse attention. Top-S denotes the number of source-side sparse-attention candidates to be referenced during cache reuse, and this analysis is conducted to select the representative MoT configuration used in the subsequent method-level comparison. Overall, Top-$S = 128$ achieves the highest F1. Although it is lower than Native under the 4K budget, it improves rapidly from 8K onward and reaches F1 comparable to or higher than Native at 16K and 24K. This shows that, when a sufficient number of source-side candidates is used, MoT can effectively reuse long-context information from the stored KV cache.

By contrast, performance becomes limited as Top-S decreases. Top-$S = 64$ shows a gradual improvement as the budget increases, but it remains below Top-$S = 128$, while Top-$S = 32$ and Top-$S = 16$ show lower F1 overall. In particular, Top-$S = 16$ does not improve as the budget increases and instead tends to degrade, suggesting that an overly small sparse-attention candidate set is insufficient for preserving or reconstructing useful information from long contexts. Therefore, we use Top-$S = 128$, which shows the most stable F1, as the representative setting for MoT in this paper.

Long-Context CAG Results. Figure 32 summarizes the average F1 and average TTFT over all context budgets. In terms of average F1, Native achieves the highest performance with 0.21, while MoT achieves 0.20, resulting in a gap of only 0.01 F1 points from Native. This means that MoT preserves approximately 96.3% of the F1 of Native, showing that the quality loss in generation is limited even in the CAG setting where the original context is not directly prefilled.

In contrast, the average F1 scores of C2C-Project and Interlat are only 0.11 and 0.14, respectively. MoT achieves approximately 85.8% higher average F1 than C2C-Project and approximately 49.6% higher average F1 than Interlat. This gap suggests that, in the CAG setting, what matters is not merely reusing a cache, but whether the reused cache is preserved in a form that the target model can effectively use during generation.

In terms of average TTFT, MoT shows higher latency than the other CAG methods. This is because MoT performs both cache transformation and context replay. However, given that the average F1 gap

from Native is very small, this overhead can be interpreted not simply as inefficiency, but as the cost of preserving long-context QA quality.

Figure 14 in Section 5 shows that these average results are not dependent on a specific budget, but are also consistent with the trend as the budget increases. In particular, MoT maintains F1 close to Native under large context budgets, supporting that the average performance gap in Figure 32 reflects quality preservation across the overall long-context regime.

Prompt Design.

Text 7. HotpotQA-E sample

Dataset. HotpotQA-E is a long-context multi-hop QA benchmark with multiple Wikipedia-style documents. The model must find the required evidence in the long multi-document context and answer a short question.

Context – Translated Cache Region

```
Read the following context and complete the task.

Context: Title: The Oberoi Group
The Oberoi Group is a hotel company with its head office in Delhi.
The company owns and operates luxury hotels and resorts under the
Oberoi and Trident brands. ...

Title: Oberoi Family
The Oberoi family is an Indian family that is famous for its
involvement in hotels through The Oberoi Group. Members of
the family have been associated with the development of luxury
hospitality in India. ...

Title: Delhi
Delhi, officially the National Capital Territory of Delhi, is
a city and union territory of India containing New Delhi, the
capital of India. ...
```

Prompt – Target-Side Query Region

```
Question or task: The Oberoi family is part of a hotel company
that has a head office in what city?
Answer:
```

Completion – Target Output Region

```
Delhi
```

G LICENSE

We summarize the licenses and terms of use of the datasets, external method implementations, and model checkpoints used in this work.

Benchmarks and training data. BoolQ is released under Creative Commons Share-Alike 3.0. PubMedQA's official repository is released under the MIT license, while the underlying PubMed abstracts may be subject to their original publisher terms. MMLU-Redux is released under CC-BY-4.0. SQuAD-v1.1 and HotpotQA are released under CC BY-SA 4.0. StrategyQA is evaluated using the task definition distributed through BIG-bench, which is released under Apache-2.0. NewsQA is distributed under the official Microsoft Research download terms, and the underlying CNN articles remain subject to CNN's rights. OpenWebText uses an Apache-2.0 dataset script, but the underlying web documents may have heterogeneous upstream terms; we use it only for research training and do not redistribute raw examples.

External methods. The official implementations of C2C-Project (Fu et al., 2026), Interlat (Du et al., 2026), KVComm (Shi et al., 2026), and MALLM (Becker et al., 2025) are released under Apache-2.0. Since no official open-source implementation of LSC (Dery et al., 2026) was available, we implemented it based on the corresponding paper. Implementation details and source repositories are summarized in Appendix E.2.

Models. Qwen2.5 checkpoints used in this work are released under Apache-2.0. GPT-2 family checkpoints are released under the MIT / OpenAI Modified MIT license. DistilGPT2 and Pythia checkpoints are released under Apache-2.0. DialoGPT-small is released under MIT. Llama 3.2 checkpoints used in the multi-agent case study are governed by the Llama 3.2 Community License. OPT checkpoints are governed by the OPT license agreement, which permits non-commercial research use and restricts commercial or production use. For `MathGPT2`, we follow the license of the specific checkpoint used; if `FlameF0X/MathGPT2` is used, it is released under MIT.

Usage. All third-party datasets, model checkpoints, and external method implementations are used only for research, training, and evaluation. We do not redistribute the raw benchmark data, model weights, or third-party source code.

H LIMITATIONS AND BROADER IMPACTS

This work has several limitations. First, as shown in the shift and correction analysis in Section 4.3, MoT reduces the final last-state shift $\|\mathbf{s}_L\|$ mainly by decreasing the translation shift $\|\mathbf{s}_T\|$ and the orthogonal shift component $\beta_{T:L}$. However, the correction-deficit factor $d_{T:L}$ changes only marginally. This suggests that MoT improves the translated trajectory without directly increasing the target model's intrinsic correction ability. Developing objectives or architectures that explicitly reduce correction deficit remains an important future direction.

Second, our channel mapping relies on depth-ratio alignment as a simple and effective default strategy. As discussed in Appendix C.4, this assumption can become weaker when models differ substantially in instruction tuning, training data distribution, or representation organization. In such cases, relative layer depth may not provide a sufficiently reliable mapping between source and target cache spaces. Future work should therefore investigate adaptive channel mapping methods that account for stronger model mismatch.

The broader impact of MoT is primarily tied to efficient reuse of internal model states. By reducing redundant prefilling, memory duplication, and repeated cache storage, MoT can lower computational cost and energy consumption in multi-model and long-context LLM workflows, potentially contributing to more resource- efficient and lower-carbon inference. At the same time, making multi-agent and multi-model systems more scalable may accelerate the deployment of highly automated agentic workflows. While this can improve productivity, it also raises questions about oversight, accountability, and the role of human judgment when increasingly capable agents are used in decision-support settings. These risks are not specific to cache translation, but improved system efficiency can make such deployments easier, and therefore should be considered alongside technical progress.