CROSS-MODEL KV CACHE CONSTRUCTION IN LLM FAMILIES: RESIDUAL-STREAM MAPPING WITHOUT MATCHED HEAD GEOMETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

Passing context between language models usually means serialising it as text and prefilling it again. We present the residual-stream map (RSM), a method for building one model's key-value (KV) cache from another model's internal states that is head-geometry-independent within tokenizer-compatible model families. A per-layer ridge map carries the sender's pre-normalisation residual states into the receiver's residual space, a small corrector trained on generic text adjusts them, and the receiver's own frozen normalisation, key/value projections and rotary encoding build the cache, so its head count and head dimension are always the receiver's. We evaluate every directed pair among finetunes and sizes of Qwen and Llama models from 0.6B to 8B parameters. Transfer between Llama-3.2-1B (head dimension 64) and Llama-3.2-3B or Llama-3.1-8B (head dimension 128), which KV-space mappers cannot express, runs unchanged: into the 1B model it scores 39.36 and 38.02 F1 on HotpotQA against 40.28 for its own text prefill (97.4% and 93.5% retention). Between finetunes of one model, transfer retains a median 102.5% (Llama) and 90.4% (Qwen2.5) of the F1 gain of text prefill, and across sizes it reaches or comes within 7 F1 of the weaker model's own text reading in 10 of 12 directions. Building the cache is 3.1 to 5.3 times faster than prefilling 1024 tokens. On the pairs where training-free KV-transfer methods also run, RSM matches them between finetunes (40.47 against 42.41 F1 for the best mapper on Llama) and leads them by 10.31 to 31.34 F1 across sizes.

1 INTRODUCTION

Multi-model pipelines pass context between models constantly. The usual mechanism is text: the sender emits tokens and the receiver prefills them. This repeats a quadratic prefill and discards the representation the sender has already built. Building the receiver's KV cache directly from the sender's states avoids both, but two models in one family can differ in depth, width, number of KV heads and head dimension. Llama-3.2-1B, for example, uses head dimension 64 while Llama-3.2-3B and Llama-3.1-8B use 128.

We present the residual-stream map (RSM, Figure 1), a cache construction method that is head-geometry-independent within tokenizer-compatible model families. RSM maps the sender's residual state, before the attention normalisation, into the receiver's residual space, and then runs the receiver's own frozen normalisation, key and value projections and rotary encoding on the mapped state. The receiver's weights therefore fix the head count and head dimension of the cache, and the only quantity the map relates is the residual width of the two models. A small corrector, trained on generic text with the receiver's behaviour as the target, adjusts the mapped states before projection.

We evaluate RSM on every directed pair within two finetune families and two size ladders, on HotpotQA and 2WikiMultihopQA. The four directions between Llama-3.2-1B and its larger siblings connect different head dimensions and run with the method unchanged. Transfer between finetunes is close to text prefill, and transfer across sizes usually reaches the weaker model's own reading. Two map-only ablations, on pairs that differ in family, size relation and corrector teacher, measure what the ridge map achieves alone and what the corrector adds. Finally, we reimplement three training-

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.

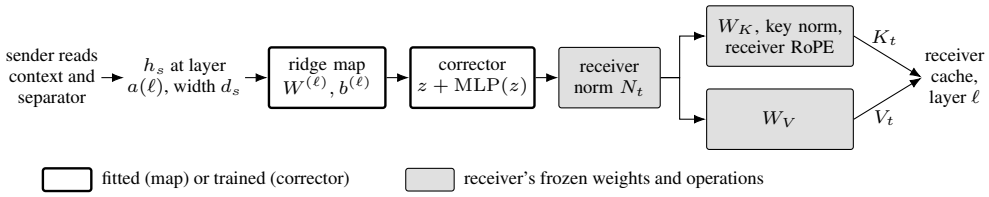


Figure 1: Residual-stream map (RSM). For every receiver layer ℓ and every cached token, the sender's pre-normalisation residual state at the aligned layer $a(\ell)$ is mapped into the receiver's residual space and corrected, and the receiver's own normalisation, key and value projections, key normalisation and rotary positions produce the keys and values. The map relates residual widths only, so the receiver's head count n_{kv}^t and head dimension d_h^t are used whatever the sender's.

free cache-transfer methods, set out which pairs each can run, and compare fidelity and build time pair by pair.

Contributions.

- RSM, a cache construction method that is head-geometry-independent within tokenizer-compatible model families, and runs across different head dimensions, KV head counts, residual widths and depths (Section 3).
- Complete directed evaluation over finetunes and sizes in two families, including transfer across head dimensions 64 and 128, with map-only ablations on two contrasting pairs (Section 5).
- Reimplementations of three published cache-transfer methods, an account of which pairs each can run, and a pairwise comparison of fidelity and build time on the pairs they share with RSM (Section 6).

2 Related Work

KV cache reuse within one model. Several systems avoid recomputing a cache. HCache (Gao et al., 2025) stores intermediate activations and restores the cache through the model's own projections, and our construction is the cross-model form of that step. CacheBlend (Yao et al., 2025) reuses caches of retrieved chunks and recomputes a subset of tokens to repair cross-chunk attention. Compression methods reduce cache size by projection, quantisation or coding (Liu et al., 2024; Chang et al., 2025; Staniszewski & Łańcucki, 2026); they are complementary to cross-model construction.

Training-free KV transfer between models. DroidSpeak (Liu et al., 2026) reuses a sender's cache for most layers and recomputes a contiguous block selected by offline profiling, which assumes an identical architecture. Heo et al. (2026) fit, for each target layer and head, a ridge map from the keys and values of all source KV heads in the top-k source layers, with rotary positions removed before mapping and re-applied after it. CacheBridge (Qu et al., 2026) restricts each target head to a matched source head and weights the fit by attention sensitivity. Both mappers act on KV heads and require matched KV head count and head dimension. These are the three methods we compare against in Section 6.

Regenerating keys and values with the receiver's weights. Semantic Cache Distillation (Ma et al., 2026) transfers caches between a producer and consumers that share one architecture and differ in weights. At a few transition layers it sends the pre-attention normalised inputs, and the consumer regenerates keys and values with its own projections. The architecture is identical on both sides, so the transmitted states already have the consumer's width. RSM applies the same regeneration at every layer between models whose residual widths, depths, KV head counts and head dimensions differ, and bridges them with a map in residual space.

Learned cache transport. Concurrent work by Li et al. (2026) trains a transport module that translates a source model's KV states into a representation the target model decodes from directly,

across scale, attention configuration and family. The module produces the target's cache representation itself. RSM instead maps residual streams and delegates KV construction to the frozen receiver, whose own normalisation, projections and rotary encoding produce every key and value; the learned part is a small residual corrector in front of those projections. ICaRus (Woo et al., 2026) addresses a related serving problem by fine-tuning specialised models so that they all read one identical cache produced by a frozen shared encoder.

Representation alignment. Model stitching shows that a simple map between the layers of two networks can often connect them (Lenc & Vedaldi, 2015; Bansal et al., 2021), and representation similarity measures compare networks directly (Kornblith et al., 2019). Relative representations make latent spaces comparable across models (Moschella et al., 2023). Chen et al. (2025) show that affine maps between the residual streams of language models of different sizes transfer learned features between them. We rely on the same observation, that residual spaces of related models are close to linearly related, and measure it by downstream behaviour after cache construction.

3 METHOD

Let $h_{s,p}^{(\ell_s)} \in \mathbb{R}^{d_s}$ be the sender's pre-normalisation residual state at token position p and layer ℓ_s. For each receiver layer ℓ_t we apply a ridge map and a residual corrector,

$$z_{t,p}^{(\ell_t)} = W^{(\ell_t)} h_{s,p}^{(a(\ell_t))} + b^{(\ell_t)}, \qquad \hat{h}_{t,p}^{(\ell_t)} = z_{t,p}^{(\ell_t)} + c_2^{(\ell_t)} + B_2^{(\ell_t)} \,\mathrm{GELU}(B_1^{(\ell_t)} z_{t,p}^{(\ell_t)} + c_1^{(\ell_t)}), \quad (1)$$

where $a(\ell_t) = \mathrm{round}[\ell_t(L_s - 1)/(L_t - 1)]$ aligns layers by proportional depth for models with L_s and L_t layers. With receiver width d_t and corrector width r, $W \in \mathbb{R}^{d_t \times d_s}$, $b, z, \hat{h}, c_2 \in \mathbb{R}^{d_t}$, $B_1 \in \mathbb{R}^{r \times d_t}$, $c_1 \in \mathbb{R}^r$ and $B_2 \in \mathbb{R}^{d_t \times r}$. The receiver then builds its cache with its own frozen weights,

$$u_{t,p}^{(\ell_t)} = N_t^{(\ell_t)}(\hat{h}_{t,p}^{(\ell_t)}), \quad K_{t,p}^{(\ell_t)} = \mathrm{RoPE}_p\Big(N_{K,t}^{(\ell_t)}(W_K^{(\ell_t)} u_{t,p}^{(\ell_t)})\Big), \quad V_{t,p}^{(\ell_t)} = W_V^{(\ell_t)} u_{t,p}^{(\ell_t)}, \quad (2)$$

where N_t is the receiver's pre-attention normalisation, $N_{K,t}$ its key normalisation (the identity for receivers without one), and W_K, W_V its key and value projections. Because these belong to the receiver, K and V have the receiver's head count and head dimension whatever the sender's, and the map relates residual widths only. The residual state precedes rotary encoding, so positions are applied once, by the receiver.

Ridge map. For each directed pair and layer, we run both models on 250 to 1000 unlabeled WikiText-103 paragraphs (Appendix A), collect aligned residual states at every token, and fit W and b by ridge regression on centred states, with the intercept recovered from the means. The coefficient is 1, applied to the unnormalised Gram matrix, so the fit is close to least squares. When sender and receiver widths are equal the penalty is $\|W - I\|_F^2$; otherwise it is $\|W\|_F^2$. This stage uses no labels, teacher, or gradient updates.

Behavioural corrector. After freezing the maps, we train one corrector per receiver layer with width $r = 256$ and a zero-initialised output projection, so training starts from the ridge map alone. Training makes one pass over 520 distinct C4 documents, one mechanically generated task per document (span reconstruction, anchor-sentence retrieval, sentence ordering, or constrained extraction), with AdamW at learning rate 2×10^{-5}, weight decay 0.01, 32 warmup steps and cosine decay. For answer tokens $y_{1:m}$ the loss is

$$\mathcal{L} = \frac{1}{m} \sum_{j=1}^{m} \Big[\mathrm{KL}(p_T^j \| p_R^j) - 0.1 \log p_R^j(y_j)\Big] + 0.01 \sum_{\ell_t \in \mathcal{C}} \frac{1}{T d_t} \sum_{p=1}^{T} \|r_{\ell_t,p}\|_2^2, \quad (3)$$

where p_T^j and p_R^j are the teacher's and the constructed receiver's next-token distributions, $\mathcal{C}$ is the set of corrected layers (all receiver layers), T is the number of cached tokens, and $r_{\ell_t,p} = \hat{h}_{t,p} - z_{t,p}$ is the corrector output. The teacher is the receiver reading the native context for pairs of equal size, and the larger model for pairs of different size. The tasks are rendered with the same short-answer instruction template used at evaluation; their content comes only from C4, and the task builder rejects any document that mentions a benchmark name. Hyperparameters were set on MuSiQue, outside the evaluation datasets.

4 EXPERIMENTAL SETUP

We evaluate on the validation splits of HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020) with $n = 200$ items, seed 2, a 1024-token context budget, hard TF-IDF distractors, greedy decoding, and token-level F1. Models are Qwen2.5-1.5B with its Instruct and Coder finetunes, three finetunes of Llama-3.2-1B, Qwen3 at 0.6B, 1.7B and 4B, and Llama at 1B, 3B and 8B (Appendix Table 4); Llama-3.1-8B is loaded in 4-bit precision to fit one 12 GB GPU. Every directed pair within each group is evaluated, 30 pairs in all (24 without Qwen2.5-Math, Section 5.3), each with one fitted map and one trained corrector.

The sender's residual states cover the context and the separator that precedes the question (Appendix A), in both fitting and evaluation, so the receiver gets a cache for exactly the tokens it would have prefilled. As a regression test, a receiver whose own residual states pass through the same construction with an identity map agrees with its own text prefill within 0.94 F1 on every receiver and both datasets (Appendix Table 5). We report raw F1 together with *retention*,

$$\text{retention} = \frac{F_1(\text{RSM}) - F_1(\text{no context})}{F_1(\text{text}) - F_1(\text{no context})}, \tag{4}$$

the fraction of the F1 gain from reading the context that survives transfer. All three terms come from the same evaluation run; for one receiver the text and no-context references vary across runs by at most 1.22 F1. Retention can exceed 100% when transfer beats the receiver's own text prefill. Every printed retention is computed from the two-decimal F1 values printed with it. Paired 95% intervals use 2,000 bootstrap resamples of identical items.

5 RESULTS

5.1 TRANSFER BETWEEN FINETUNES

Figure 2a shows retention for every directed pair among three finetunes of Qwen2.5-1.5B and three of Llama-3.2-1B, on both datasets; raw F1 is in Appendix B. Transfer between Llama finetunes has a median retention of 102.5%, with all 12 cells at or above 90%; for example, MATH→Inst. scores 40.47 F1 on HotpotQA against 39.51 for the receiver's text prefill (102.8%). Transfer between the Base, Instruct and Coder finetunes of Qwen2.5 has a median of 90.4% (6 of 12 at or above 90%) and a wider spread. Its lowest cells have the Coder model as sender: Coder→Base scores 43.60 against 52.14 on HotpotQA (79.3%) and 20.23 against 25.69 on 2WikiMultihopQA (73.0%). Including Qwen2.5-Math, which cannot use the context even as text (Section 5.3), the Qwen2.5 median over 24 cells is 79.5%.

5.2 TRANSFER ACROSS SIZES

Across head dimensions. Four directions, into and out of Llama-3.2-1B, connect head dimension 64 with head dimension 128. RSM runs on them unchanged, because the receiver's projections produce keys and values of its own shape. Downward transfer into the 1B model scores 39.36 (from 3B) and 38.02 F1 (from 8B) on HotpotQA against 40.28 for its own text prefill, 97.4% and 93.5% retention, and 18.23 and 13.35 against 18.21 on 2WikiMultihopQA, 100.1% and 68.4%. Upward transfer from the 1B model scores 47.45 (into 3B) and 46.79 F1 (into 8B) on HotpotQA and 24.50 and 20.60 on 2WikiMultihopQA, above the 1B model's own 40.28 and 18.21 (68.0% and 63.6% of the larger receivers' HotpotQA gain).

Against both models' readings. A small sender can only pass on what it encoded, so Figure 2b places each transfer against both models' own text readings. In 10 of 12 directions, transfer reaches or comes within 7 F1 of the weaker model's reading. Upward transfer from Llama-3.2-1B and from Qwen3-0.6B into 1.7B (49.43 F1 against the sender's 31.87, 89.1% retention) scores above the sender's own reading, so the larger receiver adds to what the sender provides. Downward transfer lands close to the receiver's own reading, and above it on Qwen3 1.7B→0.6B (37.56 against 31.87, 121.0%). The two directions between Qwen3-1.7B and Qwen3-4B score 33.74 and 39.78 F1 against the 1.7B model's 54.90 (44.2% and 69.8%), and the directions between Qwen3-0.6B and Qwen3-4B score 27.71 and 25.76 (34.8% and 77.4%). Qwen3-4B has 36 layers against 28 in the smaller

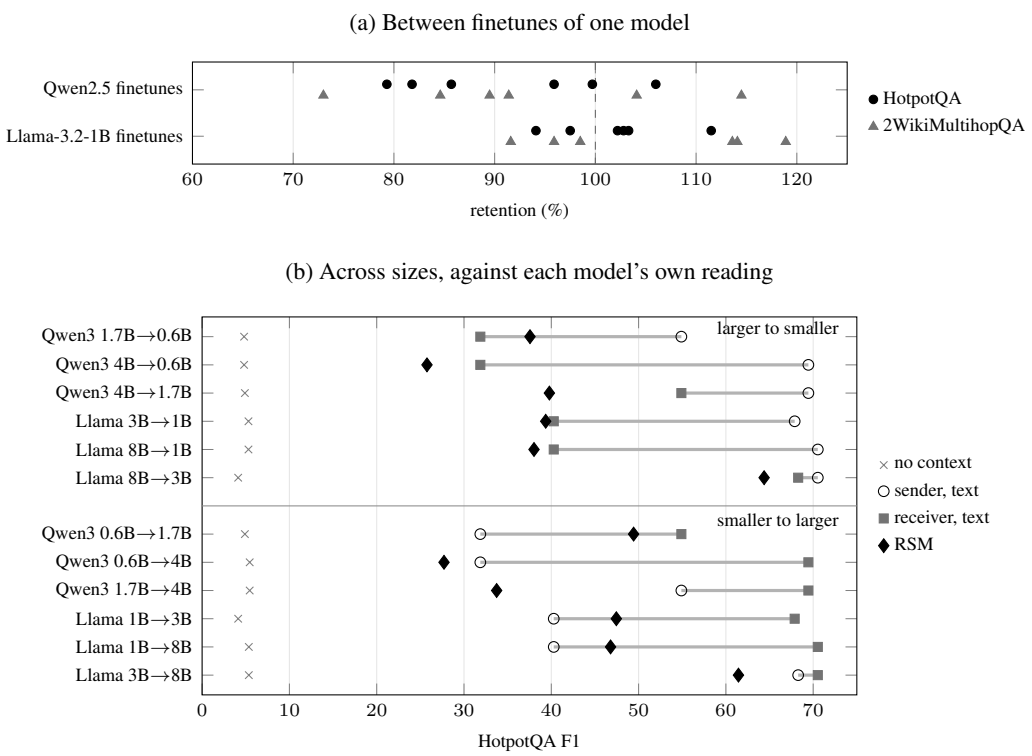


Figure 2: (a) Retention between finetunes, one mark per directed pair and dataset; the dashed line is text prefill. (b) Transfer across sizes on HotpotQA against the text-prefill F1 of both models. A sender's text reading is taken from the run of the reverse direction, in which it is the receiver. The four Llama directions into and out of Llama-3.2-1B connect head dimension 64 with 128 and run with RSM unchanged; KV-space mappers cannot express them. 2WikiMultihopQA values are in Appendix Table 9.

Qwen3 models. Against the receiver alone, the median retention across sizes is 70.1% for Qwen3 and 82.1% for Llama over both datasets (Appendix Table 9).

5.3 A MODEL THAT CANNOT READ ITS CONTEXT

Qwen2.5-Math-1.5B-Instruct scores 4.28 F1 on HotpotQA from its own text prefill, against 1.53 with no context, so it cannot answer these questions even when it reads the text. As a receiver, other senders' caches give it 2.71 to 3.38 F1 (42.9% to 67.3%). As a sender into the three other Qwen2.5 finetunes, it scores 23.85 to 29.58 F1 on HotpotQA (31.3% to 42.8%), against 43.60 to 60.35 F1 (79.3% to 106.0%) for the other senders into the same receivers; on 2WikiMultihopQA the ranges are 30.2% to 47.7% and 73.0% to 114.5% (Appendix Table 7). RSM moves the context that a sender has encoded into a receiver that can use it; the Math model is weak at both ends, and we report it separately from the other summaries.

5.4 THE MAP AND THE CORRECTOR

RSM has two parts, and only the corrector is trained. We evaluate three contrasting pairs on HotpotQA with the ridge map alone, the corrector removed and nothing else changed (Table 1): a Qwen3 pair across sizes, a Llama pair between finetunes, and a Llama pair across sizes and head dimensions. Between finetunes the ridge map carries the transfer by itself: on Llama MATH→Inst. it scores 41.11 F1 (104.7%), above the receiver's own text prefill (39.51), and the corrector changes this by -0.64 $[-1.84, +0.06]$. Across sizes the corrector, taught by the larger model, adds more. On Qwen3 1.7B→0.6B the map alone scores 27.41 F1 (83.5%) and the corrector adds $+10.15$ $[+5.49, +14.67]$, to 37.56 (121.0%), above the receiver's own text prefill. On Llama 3B→1B, across

Table 1: Ridge map alone against map and corrector on HotpotQA, $n = 200$, same items within each pair. Retention (Equation 4) uses the references of the run each arm comes from. Qwen3 and Llama MATH→Inst.: both arms re-run with the released code. Llama 3B→1B: map only from a released-code run (text 39.51), map and corrector from the matrix run (text 40.28).

pair	no context	text	map only	map + corrector	corrector − map
Qwen3 1.7B→0.6B (sizes)	4.81	31.87	27.41 (83.5%)	37.56 (121.0%)	$+10.15$ $[+5.49, +14.67]$
Llama MATH→Inst. (finetunes)	5.31	39.51	41.11 (104.7%)	40.47 (102.8%)	-0.64 $[-1.84, +0.06]$
Llama 3B→1B (head dim 128→64)	5.31	39.51 / 40.28	26.62 (62.3%)	39.36 (97.4%)	$+12.74$ $[+6.42, +18.88]$

Table 2: Which directed pairs each method can run, by group. DroidSpeak needs identical architectures; the two KV mappers need equal KV head count and head dimension. The last column lists the pairs on which all applicable methods were run.

group	pairs	DroidSpeak	closed-form	CacheBridge	RSM	compared
Qwen2.5 finetunes	6	runs	runs	runs	runs	Coder→Inst.
Llama-3.2-1B finetunes	6	runs	runs	runs	runs	MATH→Inst.
Qwen3 across sizes	6	refuses	runs	runs	runs	1.7B→0.6B, 4B→1.7B
Llama 3B↔8B	2	refuses	runs	runs	runs	–
Llama sizes with 1B	4	refuses	refuses	refuses	runs	–

head dimensions 128 and 64, the map alone scores 26.62 F1 (62.3%) and the corrector adds $+12.74$ $[+6.42, +18.88]$, to 39.36 (97.4%).

5.5 Cost relative to text prefill

Building the cache takes the map, the corrector and the receiver's projections for every layer, and no attention or feed-forward computation over the context. At the 1024-token budget used in the quality experiments this is 3.1 to 5.3 times faster than prefilling the text, on four pairs with 0.6B to 1.7B receivers, and the gap grows with length, to 6.2 to 13.0 times at 4096 tokens (Section 6.2 and Appendix Table 6). The stored map and corrector take 294 to 705 MB per pair. The timings start from the sender's residual states, which in the setting we target the sender has already computed; fitting a map and training its corrector (520 optimiser steps) happen once per pair.

6 Comparison with Existing Methods

We compare against the three training-free methods of Section 2, reimplemented from their descriptions (implementation notes in Appendix D).

6.1 What can be compared

Table 2 shows the scope of each method over the 24 directed pairs that do not involve Qwen2.5-Math. DroidSpeak runs between finetunes of one model. The two KV mappers also run across sizes when KV geometry matches, which holds for Qwen3 and for Llama 3B and 8B. The four pairs involving Llama-3.2-1B across sizes are served by RSM alone. We compare four pairs, two between finetunes and two across sizes. For Qwen3 4B→1.7B, both mappers were fitted from features staged to disk, which on a smaller pair reproduces the in-memory fit exactly. DroidSpeak is reported on the Llama pair (Appendix D).

6.2 Fidelity and speed on the compared pairs

Table 3 compares fidelity and build time pair by pair. Between finetunes of one model, every interval includes zero. At the k they are scored at, both KV mappers take 3 to 4 times longer than RSM to build the cache on the Qwen2.5 pair ($k = 28$). On the Llama pair the closed-form mapper and DroidSpeak are about as fast as RSM and CacheBridge ($k = 16$) is slower. Across sizes, RSM leads the closed-form mapper by 10.31 and 21.59 F1 and CacheBridge by 23.14 and 31.34 F1, with every interval above zero. RSM is also faster than both mappers on 1.7B→0.6B, where the closed-form

Table 3: Pairwise comparison, HotpotQA, $n = 200$. "RSM − method" is the mean per-item difference with a paired 95% bootstrap interval, computed from unrounded scores. Build time is at the 1024-token quality budget, median of 15 trials, sender forward excluded; speedup is over text prefill of the same receiver; all arms were timed in one run. Linear mappers are scored and timed at the best k measured (Qwen3 4B→1.7B was fitted only at $k = 4$); DroidSpeak is scored and timed at its selected width $w = 2$.

pair	method	F1	RSM − method	build (ms)	speedup
Qwen2.5 Coder→Inst.	RSM	52.78		50.7	4.9×
(text 249.9 ms)	closed-form (k=28)	55.19	-2.40 $[-7.92, +2.81]$	206.8	1.2×
	CacheBridge (k=28)	55.18	-2.39 $[-7.01, +1.91]$	155.8	1.6×
Llama MATH→Inst.	RSM	40.47		45.6	5.3×
(text 239.5 ms)	closed-form (k=8)	42.41	-1.94 $[-4.67, +0.55]$	42.4	5.6×
	CacheBridge (k=16)	40.17	$+0.30$ $[-1.57, +2.31]$	57.3	4.2×
	DroidSpeak	43.07	-2.60 $[-5.62, +0.26]$	43.2	5.5×
Qwen3 1.7B→0.6B	RSM	37.56		53.4	3.8×
(text 202.1 ms)	closed-form (k=8)	27.25	$+10.31$ $[+5.20, +15.18]$	261.3	0.8×
	CacheBridge (k=8)	14.43	$+23.14$ $[+17.08, +29.02]$	97.1	2.1×
Qwen3 4B→1.7B	RSM	39.78		95.4	3.1×
(text 295.0 ms)	closed-form (k=4)	18.20	$+21.59$ $[+14.50, +28.61]$	106.1	2.8×
	CacheBridge (k=4)	8.44	$+31.34$ $[+24.85, +37.54]$	52.6	5.6×

mapper at $k = 8$ is slower than text prefill; on 4B→1.7B, fitted only at $k = 4$, CacheBridge is faster than RSM. At $k = 4$ the mappers are faster than RSM on every pair except the closed-form mapper on Qwen3 1.7B→0.6B (Appendix Table 6), so their speed depends on the k needed for quality. The closed-form map for each target layer takes the keys or values

of all source KV heads over k layers, so its input width is $k\, n_{kv} d_h$ and its size and cost grow with the square of the KV width $n_{kv} d_h$: Qwen3 has 8×128 KV dimensions against 2×128 for Qwen2.5-1.5B, which makes its map 16 times larger at the same k. CacheBridge maps each head from one source head and grows linearly with the head count. RSM's cost depends on the residual widths and the receiver's own key and value projections, not on k or the product of KV widths.

On Qwen3 1.7B→0.6B the ridge map alone scores 27.41 F1 (83.5%), level with the closed-form mapper's 27.25 (difference $+0.16$ $[-4.09, +4.36]$), so the cross-size lead there comes from the corrector (Section 5.4). The closed-form mapper's score on this pair rises with k, from 23.19 F1 at $k = 4$ to 27.25 at $k = 8$, the largest k fitted; the linear mappers' k is selected on the reported items.

7 Discussion and Limitations

Building the receiver's cache from mapped residual states makes cache transfer head-geometry-independent within tokenizer-compatible model families. The clearest case is Llama-3.2-1B: its caches have head dimension 64, those of Llama-3.2-3B and Llama-3.1-8B have 128, and RSM transfers in all four directions between them with the method unchanged, reaching 39.36 and 38.02 F1 into the 1B model against 40.28 for its own text prefill (97.4% and 93.5%). Between finetunes, transfer is close to text prefill (median 102.5% for Llama and 90.4% for Qwen2.5), and across sizes it usually reaches the weaker model's own reading. Where the key-space methods also run, RSM matches them between finetunes, is more accurate across sizes, and builds the cache in a time set by residual widths rather than by k or KV width.

Between finetunes the ridge map carries the transfer by itself (41.11 F1 on Llama MATH→Inst., 104.7%). Across sizes the corrector, taught by the larger model, adds 10.15 and 12.74 F1 on HotpotQA and brings transfer to or above the receiver's own text prefill (121.0% on Qwen3 1.7B→0.6B and 97.4% on Llama 3B→1B). The two directions between Qwen3-1.7B and Qwen3-4B are the weakest in the matrix (33.74 and 39.78 F1 against 54.90), and proportional layer alignment across 36 and 28 layers is the first candidate to revisit. Our experiments cover models up to 8B parameters on one 12 GB GPU (Llama-3.1-8B in 4-bit precision), 1024-token contexts and two multi-hop QA datasets, with one map and one corrector per pair.

Reproducibility Statement

The supplementary material contains code for map fitting, corrector training and evaluation; the three baseline reimplementations, including the out-of-core fitting used for Qwen3-4B; the timing benchmark; and a script that recomputes from the result files every retention value, median, count and cross-method paired interval in this paper. It includes the exact 520-task corrector training file with its audit record and a script that verifies its provenance, and a check that out-of-core baseline fitting reproduces in-memory fitting. Appendix A lists training and evaluation settings. Evaluation results record model identifiers, seeds, item hashes, context budgets and dataset hashes. Most evaluations were run with code revisions earlier than the released one. We re-ran two pairs with the released code for the map-only ablations and report the re-run values for both throughout. On Qwen3 1.7B→0.6B every arm without the corrector reproduced exactly, and with the corrector 4 of 200 HotpotQA and 6 of 200 2WikiMultihopQA answers changed (38.00 to 37.56 and 14.48 to 15.08 F1); repeating each changed item gave the same answer every time. On Llama MATH→Inst. the no-context arm reproduced exactly, the text-prefill arm changed on 7 and 10 of 200 items (40.73 to 39.51 and 17.95 to 17.78 F1) and the transferred arm on 7 and 7 (41.15 to 40.47 and 16.43 to 16.52 F1).

AI use statement

Generative AI tools were used for research code implementation, experiment orchestration, literature discovery, artifact and citation auditing, and initial manuscript drafting. Experimental numbers were produced by the supplementary code and cross-checked against the stored result artifacts; bibliographic entries were checked against primary sources. The authors take responsibility for the final content, including all AI-assisted text, code, claims, and analyses.

Ethics Statement

This work uses public benchmarks and publicly released weights. Cache transfer moves internal representations between models, so where those representations encode sensitive context the same handling requirements apply as to the source text.

References

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021. arXiv:2106.07682.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: Compressing KV-cache with low-rank projection. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.21118.

Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring linear features across language models with model stitching. *arXiv preprint arXiv:2506.06609*, 2025.

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 (EuroSys)*, 2025.

Taekyung Heo, Rasoul Shafipour, Ritchie Zhao, Maximilian Golub, Mohammad Mahdi Kamani, Ritika Borkar, Makesh Tarun Chandran, Pantea Zardoshti, and Bita Darvish Rouhani. Cross-model KV cache transfer in LLM families: A closed-form linear mapping for prefill reuse. *arXiv preprint arXiv:2608.03893*, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, 2020.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019. arXiv:1905.00414.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Yi Li, Dongming Jiang, Yi Zhao, and Bingzhe Li. A universal context-reuse layer for cross-model KV sharing. *arXiv preprint arXiv:2608.30963*, 2026.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. Cachegen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, 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. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2026.

Qianli Ma, Zhiqing Tang, Hanshuai Cui, Zhi Yao, and Weijia Jia. Semantic cache distillation: Efficient state transfer via reuse and selective patching. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2606.07684.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2209.15430.

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

Konrad Staniszewski and Adrian Łańcucki. KV cache transform coding for compact storage in LLM inference. In *International Conference on Learning Representations (ICLR)*, 2026. OpenReview id aNVKROYpLB; arXiv:2511.01815.

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 *International Conference on Learning Representations (ICLR)*, 2026. OpenReview id qrMo6R7lOS.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025. Best Paper Award; arXiv:2405.16444.

A Implementation details

Prompt and decoding. The receiver gets a fixed instruction to answer from the context with only a short answer span, followed by "Context:" and the context, then "Question:" and the question, rendered with the receiver's chat template; the exact template is in the supplementary code. The separator is the text between the end of the context and the start of the question, and the transferred cache covers everything up to and including it. Decoding is greedy with at most 32 new tokens and no repetition penalty.

Table 4: Checkpoints (Hugging Face identifiers) and the short names used in the tables.

short name	checkpoint
Qwen2.5 Base	`Qwen/Qwen2.5-1.5B`
Qwen2.5 Inst.	`Qwen/Qwen2.5-1.5B-Instruct`
Qwen2.5 Coder	`Qwen/Qwen2.5-Coder-1.5B-Instruct`
Qwen2.5 Math	`Qwen/Qwen2.5-Math-1.5B-Instruct`
Llama Inst.	`unsloth/Llama-3.2-1B-Instruct`
Llama Abl.	`mylesgoose/Llama-3.2-1B-Instruct-abliterated`
Llama MATH	`Trelis/Llama-3.2-1B-Instruct-MATH`
Llama 3B	`unsloth/Llama-3.2-3B-Instruct`
Llama 8B	`unsloth/Llama-3.1-8B-Instruct`
Qwen3 0.6B / 1.7B / 4B	`Qwen/Qwen3-0.6B`, `-1.7B`, `-4B`

Map calibration. Each map is fitted on the first WikiText-103 training paragraphs longer than 200 characters, a number that differs by pair: 1000 paragraphs for all 12 Qwen2.5 pairs, Llama Inst.↔MATH, Llama 1B↔3B, Qwen3 0.6B↔1.7B and Qwen3 4B→0.6B; 500 for Llama 8B→1B; 400 for Llama 1B→8B and 3B→8B and for Qwen3 4B→1.7B, 0.6B→4B and 1.7B→4B; 300 for the four Llama pairs involving Abl.; and 250 for Llama 8B→3B. These paragraphs are separate from the corrector corpus. Layers are aligned by proportional depth, and the fitting seed is 5. For eight pairs the ridge coefficient was selected from $\{1, 10, 100\}$ by mean fitting R^2; 1 was selected every time, and the other pairs use 1 directly.

Corrector training. Correctors are pair-specific, one per receiver layer, with width 256, batch accumulation 1, AdamW with learning rate 2×10^{-5} and weight decay 0.01, 32 warmup steps and cosine decay, for 520 optimiser steps (one per document), with seed 7. The ridge coefficient, corrector width and learning rate are the same for every pair and were set on MuSiQue, which is not among the evaluation datasets. Evaluation uses bf16 mapped states. Pairs that do not fit together on the GPU were run with sequential loading and staging to disk. Llama-3.1-8B is loaded with bitsandbytes 4-bit weights and bf16 compute in every pair that includes it: as the sender in 8B→1B and 8B→3B and as the receiver in 1B→8B and 3B→8B. Qwen3 models run with thinking disabled in the chat template.

Timing. Inputs of $\{256, 512, 1024, 2048, 4096\}$ tokens are run in one process in random arm order, with 3 warm-up and 15 timed trials, batch size 1, bf16 weights and PyTorch scaled-dot-product attention for text prefill, with all maps already on one RTX 3060 12 GB, and the median is reported. The sender forward and artifact loading are excluded. The quality experiments use 1024-token contexts.

Table 5: Regression test: F1 when the receiver's own residual states are passed through the same cache construction code with an identity map (self), and when it prefills the text, from the same run. Differences use unrounded scores. Llama-3.2-3B was measured in two runs; both are shown.

	HotpotQA			2WikiMultihopQA		
receiver	self	text	diff.	self	text	diff.
Qwen3-0.6B	31.80	31.87	−0.08	10.94	10.88	+0.07
Qwen3-1.7B	54.93	54.90	+0.03	24.64	25.55	−0.91
Qwen3-4B	69.41	69.45	−0.04	32.83	33.33	−0.50
Llama-3.2-1B	40.36	40.28	+0.08	18.09	18.21	−0.12
Llama-3.2-3B (run 1)	68.28	67.88	+0.40	38.61	39.24	−0.62
Llama-3.2-3B (run 2)	67.34	68.28	−0.94	39.24	38.92	+0.31
Llama-3.1-8B	70.53	70.53	0.00	46.25	46.75	−0.50

B COMPLETE CROSS-FINETUNE MATRICES

Each cell is RSM / receiver text-prefill F1; rows are senders and columns receivers. The last row is the receiver's no-context F1. The Qwen2.5 matrix includes the Math model discussed in Section 5.

Table 6: Cache-build time, ms (speedup over text prefill), from the same run as Table 3, and stored artifact size in decimal MB (RSM: map and corrector). The top block is at 4096 tokens with the linear mappers at their scored k (28, 8 and 8 for the closed-form mapper and 28, 16 and 8 for CacheBridge on the first three pairs; 4 on the last). The middle block times the mappers at $k = 4$. DroidSpeak on the Llama pair exceeded GPU memory at 4096 tokens, so its time is not a valid measurement; DroidSpeak on the Qwen2.5 pair is not reported (Section 6).

4096 tokens, scored k	text	RSM	closed-form	CacheBridge	DroidSpeak
Qwen2.5 Coder→Inst.	1948.9	189.3 (10.3×)	351.5 (5.5×)	343.3 (5.7×)	–
Llama MATH→Inst.	2163.8	166.8 (13.0×)	150.3 (14.4×)	204.4 (10.6×)	–
Qwen3 1.7B→0.6B	1950.4	194.8 (10.0×)	688.9 (2.8×)	377.6 (5.2×)	refuses
Qwen3 4B→1.7B	2274.3	368.4 (6.2×)	407.2 (5.6×)	197.4 (11.5×)	refuses
mappers at $k = 4$, 1024 / 4096 tokens					
Qwen2.5 Coder→Inst.			38.5 / 58.5	33.4 / 57.4	
Llama MATH→Inst.			24.9 / 79.5	19.0 / 58.6	
Qwen3 1.7B→0.6B			109.1 / 409.5	53.0 / 184.5	
artifact MB, scored k ($k = 4$)					
Qwen2.5 Coder→Inst.		352.7	411.1 (58.8)	205.6 (29.5)	–
Llama MATH→Inst.		335.9	268.5 (134.3)	67.2 (16.9)	–
Qwen3 1.7B→0.6B		293.9	1879.3 (939.8)	235.2 (117.7)	–
Qwen3 4B→1.7B		705.2	939.8	117.7	–

Table 7: Qwen2.5-1.5B complete directed cross-finetune matrix, $n = 200$.

HotpotQA

	Base	Inst.	Coder	Math
Base	–	60.35/60.49	45.65/54.28	2.71/4.28
Inst.	54.62/52.14	–	52.34/54.28	3.38/4.28
Coder	43.60/52.14	52.78/60.49	–	2.90/4.28
Math	23.85/52.14	29.58/60.49	25.27/54.28	–
floor	10.97	6.48	6.79	1.53

2WikiMultihopQA

	Base	Inst.	Coder	Math
Base	–	30.67/29.60	27.64/29.76	1.80/2.03
Inst.	23.56/25.69	–	33.33/29.76	1.91/2.03
Coder	20.23/25.69	25.57/29.60	–	1.51/2.03
Math	11.74/25.69	15.94/29.60	12.53/29.76	–
floor	5.49	3.50	5.08	0.89

C Complete cross-size results

D Baseline implementation notes

None of the three baselines released code; each was implemented from its method description, and a method refuses a pair rather than being modified when its stated precondition fails. Departures from the source papers, and from our own setup:

- Closed-form and CacheBridge: calibration on 1024-token WikiText/C4 windows instead of FineWeb-Edu, with the source's sequence count, length and stride; the closed-form source reports a ± 5.24 point effect of corpus choice. Our maps are instead fitted on short WikiText-103 paragraphs (Appendix A), so the calibration data are not matched.
- Closed-form: k is capped at 8 on Qwen3 1.7B→0.6B and fitted only at $k = 4$ on Qwen3 4B→1.7B by GPU memory, where the source sweeps k to all layers. Best k is selected on the reported items, as in the source. Near-null directions of the ridge Gram matrix are floored and reported.
- CacheBridge: the fused sufficient-statistics kernel is not implemented; this changes fitting time only.
- DroidSpeak: the profiling grid has 3 widths and 4 start layers rather than the source's full grid, with widths up to 8 layers; w and the start are chosen on a held-out split of 64 items, taking the cheapest setting within one standard error of the best. This rule selected the smallest width, $w = 2$, on 5 of 6 pairs, which favours RSM. On the Qwen2.5 Coder→Inst. pair our implementation starts a recompute block at layer 0 from the sender's embeddings; recomputing all 28 layers then scores 13.92 F1 against 60.49 for text prefill, so we do not report DroidSpeak on that pair.

Table 8: Llama-3.2-1B complete directed cross-finetune matrix, $n = 200$.

HotpotQA

	Abl.	Inst.	MATH
Abl.	–	39.86/40.73	43.15/45.39
Inst.	33.41/30.42	–	46.23/45.39
MATH	31.27/30.42	40.47/39.51	–
floor	4.34	5.31	7.29

2WikiMultihopQA

	Abl.	Inst.	MATH
Abl.	–	20.81/17.95	17.63/15.87
Inst.	14.67/15.24	–	17.57/15.87
MATH	15.03/15.24	16.52/17.78	–
floor	1.36	2.84	3.39

Table 9: All cross-size directions, $n = 200$: our F1, the receiver's text-prefill and no-context F1 from the same run, and retention (Equation 4). Qwen3 1.7B→0.6B uses the released code.

	HotpotQA				2WikiMultihopQA			
direction	RSM	text	none	ret. %	RSM	text	none	ret. %
Qwen3 0.6B→1.7B	49.43	54.90	4.90	89.1	24.37	25.55	3.26	94.7
Qwen3 0.6B→4B	27.71	69.45	5.45	34.8	11.62	33.33	3.55	27.1
Qwen3 1.7B→0.6B	37.56	31.87	4.81	121.0	15.08	10.88	3.19	154.6
Qwen3 1.7B→4B	33.74	69.45	5.45	44.2	13.60	33.33	3.55	33.7
Qwen3 4B→0.6B	25.76	31.87	4.81	77.4	8.56	10.88	3.19	69.8
Qwen3 4B→1.7B	39.78	54.90	4.90	69.8	18.93	25.55	3.26	70.3
Llama 1B→3B	47.45	67.88	4.13	68.0	24.50	39.24	2.48	59.9
Llama 1B→8B	46.79	70.53	5.35	63.6	20.60	46.75	2.01	41.6
Llama 3B→1B	39.36	40.28	5.31	97.4	18.23	18.21	2.84	100.1
Llama 3B→8B	61.44	70.53	5.35	86.1	39.18	46.75	2.01	83.1
Llama 8B→1B	38.02	40.28	5.31	93.5	13.35	18.21	2.84	68.4
Llama 8B→3B	64.38	68.28	4.13	93.9	32.01	38.92	2.48	81.0

- All three are evaluated on HotpotQA only.