WHICH KEY–VALUE CACHE COMPRESSIONS CAN CONTINUED PRETRAINING REPAIR?

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrofitting pretrained transformers with smaller key–value (KV) caches can reduce memory use and accelerate decoding with limited additional training. Next-token prediction incentivizes the use of context, but whether continued pretraining restores retrieval after cache compression is unclear. We compare head merging, joint low-rank compression of keys and values, and token pooling at matched cache sizes and training budgets on `Qwen3` from 0.6B to 8B and on `Falcon3-1B`, and extend the merge to five model families spanning 0.6B–14B parameters. Each conversion is evaluated against an unconverted control trained with the same recipe. Recovery in language-modeling loss and commonsense performance does not reliably predict multi-key retrieval, which requires distinguishing a target key from many similar entries. At 4K tokens and half the cache, a low-rank latent retains 97.9% of the control's multi-key retrieval at 4B, but head merging only 34.0%, despite similar commonsense performance. At the same size, preserving the retrieval heads identified in one forward pass also improves recovery at the training length, at a smaller cache saving, but the benefit is gone at 16K tokens. Continued pretraining also fails on a conversion that removes nothing. Rotating value heads without adjusting their output projections permits an exact inverse correction, yet continued pretraining lowers the loss without restoring retrieval. From 0.6B to 8B, a closed-form local re-alignment recovers at least 93.6% of the control's multi-key retrieval without continued pretraining. These results motivate three principles for KV-cache retrofitting: evaluate multi-key retrieval explicitly, preserve retrieval-relevant structure, and correct known misalignments locally before continued pretraining.

1 INTRODUCTION

As context length grows, the key–value (KV) cache can dominate the memory cost of long-context inference. Retrofitting reduces this cost by converting a pretrained model to use a smaller cache, then continuing pretraining to recover performance (Ainslie et al., 2023; Nawrot et al., 2024; Meng et al., 2025; Ji et al., 2025). Success is usually measured by language-modeling loss and commonsense benchmarks. These measures leave a central question open. **Which KV-cache compressions can continued pretraining repair, and what determines whether retrieval recovers?**

Next-token prediction incentivizes the use of context, but its average loss can hide failures on the few tokens that require a precise lookup (Appendix B.1), and short commonsense questions give little context to search. Associative recall tests this ability directly by asking for the value of a key among distractors, and it separates attention from its cheaper alternatives (Arora et al., 2024; Jelassi et al., 2024). We use RULER's needle tasks to separate retrieving an isolated entry from multi-key retrieval among many similar entries (Hsieh et al., 2024).

We compare three conversions that compress different dimensions of the cache (Figure 1a). Head merging follows grouped-query attention uptraining (Ainslie et al., 2023; Jin et al., 2025), the joint low-rank latent follows multi-head latent attention (MLA) (DeepSeek-AI, 2024; Meng et al., 2025; Ji et al., 2025), and token pooling simplifies the compressed attention of DeepSeek-V4 (DeepSeek-AI, 2026). Head merging spans five model families from 0.6B to 14B parameters, and the comparisons across conversions use `Qwen3` from 0.6B to 8B and `Falcon3-1B`. Every conversion gets the same continued-pretraining budget and is compared with an unconverted control trained with the same

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.

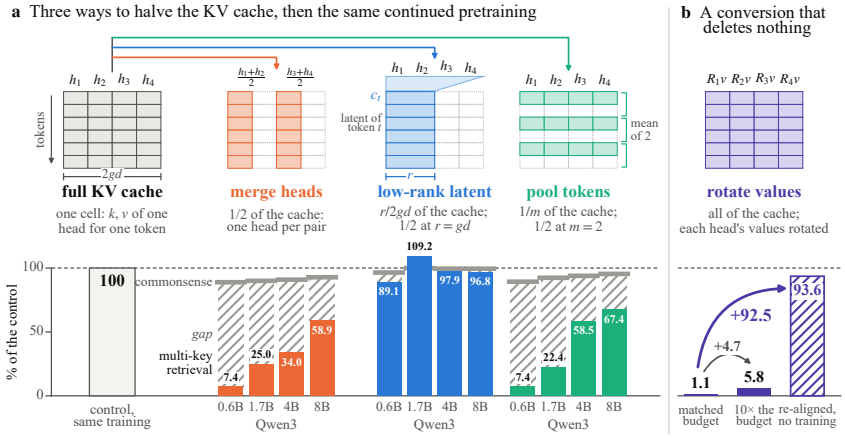

Figure 1: What continued pretraining repairs after a KV-cache conversion. (a) Three conversions that halve a layer's cache (tokens as rows, KV heads as columns, retained parts in color). Below each, multi-key retrieval at 4K tokens (bars) and the commonsense average (gray) after matched continued pretraining, in % of an equally trained control. At 0.6B the latent is shown at a quarter of the cache, since at half it would be exact. (b) A value rotation on `Qwen3-0.6B` deletes nothing and has an exact repair (Proposition 2). Continued pretraining does not find it even with ten times the tokens (arrows: gains over the matched budget), while a closed-form local re-alignment recovers 93.6% without training. Table 5 gives both at every size from 0.6B to 8B.

recipe, which separates the effect of the conversion from that of continued pretraining itself. Cache sizes are matched as well, except that the 0.6B latent keeps a quarter of the cache, since at half it would be exact.

Recovery on conventional benchmarks hides large differences in retrieval. At half the cache, `Qwen3-4B` retains 97.9% of the control's multi-key retrieval with the latent but only 34.0% with head merging, although both stay close to the control on commonsense benchmarks. Across families, token pooling loses about as much as head merging below 4B parameters and less from 4B on, though still far more than the latent. Merged models still retrieve a single needle but fail to tell similar keys apart (§5.2). Within `Qwen3`, the drop shrinks with model size and grows with context length (§5), and the 7B and 8B bases of other families recover far less than `Qwen3-8B` (§6.2). None of the head-merging or latent-retrofitting studies we surveyed reports multi-key retrieval at a controlled context length (Appendix D). Among released conversions, MHA2MLA's `SmolLM-1.7B` checkpoints keep less than half of their reference models' multi-key retrieval while staying close on the reported benchmarks, whereas the 7B and 8B releases of MHA2MLA and TransMLA keep most of it (§5.3).

The differences follow from what each conversion keeps. Head merging keeps the mean of each pair of heads and discards their difference, and token pooling averages consecutive cache entries. At the same cache size, a joint latent can represent any head merge and approximates the original key and value projections at least as closely (Proposition 1). Leaving unmerged the retrieval heads that one forward pass identifies (Wu et al., 2025) saves less of the cache but keeps nearly all multi-key retrieval on `Qwen3-4B` at the training length, against about half for as many randomly chosen heads (§6.2). It also helps `Qwen3-8B` and `Llama-2-7B` but not `Qwen3-0.6B`, and the benefit is gone at 16K tokens (Appendix C.10). The latent's retention does not come with faster decoding. It reconstructs every cached key at each decoding step and decodes more slowly than the full cache, while the MLA form, which folds that reconstruction into the query projection, decodes faster but keeps little retrieval at our budget on `Llama-3.1-8B` (§6.4).

More continued pretraining does not close the gap. A tenfold budget lifts the head merge on `Qwen3-0.6B` only to 18.6% of its control (§5.5). Because merging discards information, this alone cannot separate irreversible loss from a failure of training to restore retrieval. We therefore rotate each value head by a random orthogonal matrix and leave its output projection unchanged (Figure 1b), which keeps all information and admits an exact correction (Proposition 2). Continued pretraining lowers the loss but leaves multi-key retrieval at 1.1% of the control on `Qwen3-0.6B`, and neither ten

times the tokens nor models up to 8B lift it above 8.5%. Matching each attention module's outputs to those of the original module, as in LoLCATs (Zhang et al., 2025), recovers most retrieval, and a closed-form local solve recovers at least 93.6% at every size from 0.6B to 8B without continued pretraining (§7). These results motivate three principles for KV-cache retrofitting: evaluate multi-key retrieval explicitly, preserve retrieval-relevant structure, and correct known misalignments locally before continued pretraining (§8).

2 Related work

KV-cache retrofitting. Retrofitting modifies a pretrained model's attention and adapts it with limited additional training. Head merging converts pretrained models to multi-query or grouped-query attention (Shazeer, 2019; Ainslie et al., 2023), and Align-heads first rotates the heads within each group toward each other (Jin et al., 2025). Latent conversions jointly compress keys and values as in multi-head latent attention (DeepSeek-AI, 2024), initialized from an SVD of the projections in MHA2MLA (Ji et al., 2025) or a PCA of calibration activations in TransMLA (Meng et al., 2025). Dynamic Memory Compression learns to merge consecutive cache entries (Nawrot et al., 2024), whereas our token pooling takes a fixed mean. Other retrofits select tokens (DeepSeek-AI, 2025; Gao et al., 2025b; Lu et al., 2025; Lai et al., 2026) or replace attention with linear or state-space modules, often through local distillation (Zhang et al., 2025; Wang et al., 2024; Bick et al., 2024; 2025; Goldstein et al., 2025). Inference-time eviction and quantization leave the weights unchanged and are outside our scope (Zhang et al., 2023; Xiao et al., 2024; Liu et al., 2024; Zandieh et al., 2026).

The head-merging and latent-conversion studies report perplexity, commonsense benchmarks and sometimes LongBench, but not multi-key retrieval. Several selection and attention-replacement retrofits report RULER or single-needle tests (Appendix D). Perplexity and single-needle retrieval can miss broader retrieval failures (Gao et al., 2025a), while brief continued pretraining on a balanced data mixture can extend retrieval to longer contexts (Fu et al., 2024). We study retrieval recovery across head merging, low-rank compression and token pooling at matched cache sizes and training budgets.

Pruning and recovery training. Structured pruning is a related setting in which the loss can recover while capabilities do not (Gromov et al., 2025; Jaiswal et al., 2024). Sheared-LLaMA and Minitron recover performance through data-mixture adaptation and distillation (Xia et al., 2024; Muralidharan et al., 2024). Pruning can damage stored facts before in-context learning (Jin et al., 2024), whereas head merging costs retrieval from context more than stored knowledge (§5.1).

Retrieval heads. A small set of attention heads copies information from context, and disabling them disrupts retrieval (Wu et al., 2025). Retrieval heads learned from synthetic and from real long-context data largely overlap (Zhao et al., 2025). DuoAttention, RazorAttention and HeadKV use head importance to allocate cached tokens (Xiao et al., 2025; Tang et al., 2025; Fu et al., 2025), and Ada-KV and PyramidKV adapt token budgets to inputs or layers (Feng et al., 2025; Cai et al., 2025). These methods keep every KV head and act at inference, whereas we use retrieval scores at conversion time to choose which KV heads stay unmerged (§6.2).

3 Cache conversions and their properties

Attention and its cache. A causal attention layer has h query heads, g KV heads, head dimension d and layer input $x_t \in \mathbb{R}^D$. Query head i computes $q_t^{(i)} = W_Q^{(i)} x_t$ and reads KV head $j(i)$, which caches $k_t^{(j)} = W_K^{(j)} x_t$ and $v_t^{(j)} = W_V^{(j)} x_t$:

$$o_t = \sum_{i=1}^{h} W_O^{(i)} \sum_{s \le t} a_{ts}^{(i)} \, v_s^{(j(i))}, \qquad a_{ts}^{(i)} = \operatorname*{softmax}_{s \le t}\Big(\langle \mathcal{R}_t \, q_t^{(i)}, \, \mathcal{R}_s \, k_s^{(j(i))} \rangle / \sqrt{d} \Big), \tag{1}$$

where $W_O^{(i)} \in \mathbb{R}^{D \times d}$ and $\mathcal{R}_t$ is the rotary position embedding (Su et al., 2024). Multi-head attention has $g = h$, and grouped-query attention shares each KV head among h/g query heads (Ainslie et al., 2023). The cache stores $2gd$ numbers per token and layer.

Three cache conversions. The three conversions reduce the KV heads, the joint KV feature dimension and the cached tokens (Figure 1a and Table 4). *Head merging* averages adjacent pairs of key projections and, separately, of value projections, so both query groups read the shared head and the cache halves (Ainslie et al., 2023). A *joint latent* caches $c_t = W_D x_t \in \mathbb{R}^r$ and reconstructs the keys and values of all heads as $[\widetilde{k}_t; \widetilde{v}_t] = W_U c_t$, keeping a fraction $r/2gd$ of the cache (DeepSeek-AI, 2024). We initialize $W_U W_D$ with the rank-r truncated SVD of $W = [W_K; W_V] \in \mathbb{R}^{2gd \times D}$ (Ji et al., 2025). *Token pooling* averages the values and rotated keys of non-overlapping m-token blocks. Each query attends to the completed blocks and to its own block's causal running mean, with no selection, sink or local window, so the cache holds about m times fewer entries.

Our latent keeps the original rotary embedding and, like Palu (Chang et al., 2025), reconstructs every cached key before rotating it at each decoding step. Our implementation of this reconstruction decodes more slowly than the full cache (§6.4). The released conversions of §5.3 instead change the rotary embedding so that the non-rotary key reconstruction folds into the query projection (Meng et al., 2025; Ji et al., 2025). We call this variant *the MLA form*. Proposition 1 relates head merging to the latent at equal cache size.

Proposition 1 (Latent compression subsumes head merging)**.** *Let g be even, and let Π map each adjacent pair of head blocks (B_1, B_2) of $W = [W_K; W_V]$ to $((B_1 + B_2)/2, (B_1 + B_2)/2)$, for keys and values separately; Π is an orthogonal projection of rank gd. (i) $\Pi W = W_U W_D$ for some $W_D \in \mathbb{R}^{gd \times D}$ and $W_U \in \mathbb{R}^{2gd \times gd}$, so a latent layer of dimension gd that applies a shared per-head key normalization and the rotary embedding after W_U, with the query and output projections unchanged, reproduces the merged layer exactly. (ii) With A_r the rank-r truncated SVD of A, $\|W - W_{gd}\|_F \le \|W - \Pi W\|_F$, and $\|WX - (WX)_{gd}\|_F \le \|WX - \Pi WX\|_F$ for any inputs $X \in \mathbb{R}^{D \times n}$.*

Part (i) is the grouped-query case of TransMLA's expressiveness result (Meng et al., 2025). The first bound of part (ii) holds for the weight SVD that we use. The second holds for a decomposition fitted to the outputs WX, such as TransMLA's PCA, and does not carry over to $W_{gd}X$. Neither bounds the attention output or retrieval. A merged pair shares one set of parameters, so its two key and value projections stay identical during continued pretraining (proofs in Appendix A). To separate information loss from a failure to learn, Proposition 2 builds a conversion that removes no information but has an exact repair. Our probe leaves the output projections unchanged, and §7 asks whether continued pretraining restores retrieval.

Proposition 2 (Value rotations are invertible and exactly correctable)**.** *Replace every $W_V^{(j)}$ by $R_j W_V^{(j)}$ with $R_j \in O(d)$. Every cached value remains an invertible image of the original, and replacing every $W_O^{(i)}$ by $W_O^{(i)} R_{j(i)}^\top$ restores the layer's original function for every input.*

4 Experimental setup

Models and training. We use the `Qwen3` base models at 0.6B, 1.7B, 4B and 8B, and run the head merge experiments also at 14B (Yang et al., 2025a). All five use grouped-query attention with 8 KV heads of dimension 128. Head merging is also tested on the multi-head `Llama-2-7B` and `SmolLM2-1.7B` and on the grouped-query `Falcon3-1B`, `Llama-3.1-8B` and `Mistral-7B` (Touvron et al., 2023; Ben Allal et al., 2025; Falcon-LLM Team, 2024; Grattafiori et al., 2024; Jiang et al., 2023). The hybrid `Qwen3.5` models at 0.8B, 2B and 4B, with softmax attention in a quarter of their layers, carry it to a hybrid architecture (Qwen Team, 2026; Yang et al., 2025b). The latent runs on `Qwen3` from 0.6B to 8B and `Falcon3-1B`, token pooling also on `SmolLM2-1.7B` and `Llama-2-7B`.

The *matched budget* is 2300 steps on FineWeb-Edu (Penedo et al., 2024) at sequence length 4096 and global batch 16 (1.5×10^8 tokens). We use AdamW (Loshchilov & Hutter, 2019) at learning rate 1e-4 unless stated, with 50 warm-up steps and cosine decay over the final 20%. Longer runs go up to ten times this budget. The *control* is the unconverted base trained with the identical recipe. The main text reports most results as a percentage of this control, so that the recipe's own cost is not charged to the conversion (Appendix F), and Appendix F.1 compares each control with the original pretrained checkpoint.

Table 1: Head merging (8 to 4 KV heads) on `Qwen3` after the matched budget. Loss rows: held-out loss above the original model's, in %. Task rows: % of the matched control (multi-key retrieval: MK2 at 4K, $n = 100$). Absolute values: Appendix Table 23.

	Qwen3 base				
	0.6B	1.7B	4B	8B	14B
Held-out loss above the original model (%)					
control	+1.6	+1.5	+4.4	+3.6	+5.0
head merge	+7.1	+6.6	+8.4	+7.7	+9.2
Head merge, % of the matched control					
OpenBookQA	90.9	93.3	93.2	92.0	101.1
MMLU	62.2	72.7	80.0	81.1	84.1
multi-key retrieval	**7.4**	**25.0**	**34.0**	**58.9**	**81.8**

Evaluation. We measure held-out loss on unseen FineWeb-Edu documents and run the conversion papers' benchmarks zero-shot, without a chat template, in lm-evaluation-harness (Gao et al., 2024) (Appendix I). The primary metric, multi-key retrieval (MK2), is RULER's `niah_multikey_2` at 4K tokens, scored by our own implementation. It uses 100 prompts per run (standard error 0.04–0.05 mid-range, a 95% interval of at most ± 9.8 accuracy points) and 400 where stated (± 4.9). The harness scores the needle tasks of Table 2 and the longer contexts (Appendix C.10). Tables 4 and 8 re-score our checkpoints and the released ones on one suite, the conversion papers' eight commonsense tasks and MK2 on 400 prompts. Figure 1 combines that suite's commonsense average with each run's own retrieval. Prompt sets and scorers shift some retention estimates, so each comparison uses one scorer and one reference (Appendix H).

5 Recovery after continued pretraining

5.1 Loss recovery overstates retrieval recovery

Merging the KV heads of `Qwen3` from 8 to 4 costs multi-key retrieval more than the conversion papers' benchmarks at every size after the matched budget (Table 1). Continued pretraining closes most of a loss gap that starts at +10.43 nats at 0.6B, OpenBookQA ends within a tenth of the control, and MMLU keeps less but rises with size. Multi-key retrieval keeps the smallest share, from 7.4% at 0.6B to 81.8% at 14B (76.4% at $n = 400$), so the drop shrinks with size without closing.

The gap extends beyond `Qwen3`, to the conversion papers' own base. The `Llama-2-7B` merge keeps 92.5% of the control's commonsense average but 5.8% of its multi-key retrieval, and the `SmolLM2-1.7B` and `Falcon3-1B` merges keep 0.0% and 7.5% (Appendix Tables 8 and 15). The hybrid `Qwen3.5` is the exception. Its merge keeps 93.3–98.6% of the control from 0.8B to 4B, measured against controls that themselves lose about a quarter of the base's retrieval under our recipe (Appendix Table 30). We think the hybrid architecture explains this, since the merge leaves untouched the linear-attention layers, which also carry information from the context.

5.2 Similar keys and long contexts expose the retrieval drop

Single-needle success does not imply multi-key retrieval (Table 2). The 0.6B merge keeps 84.3% of the control on multi-key 1, with three distractor needles in ordinary text. It keeps only 4.1% on multi-key 2, where the haystack is all distractor needles, and 2.7% on multi-key 3, whose keys are random strings. The 4B and 8B merges fall in the same order. Token pooling on `Qwen3-0.6B` loses even single needles, keeping 44.6% after pooling pairs of tokens and 11.4% after pooling blocks of four (Appendix Table 32).

Longer contexts widen the gap (Table 2). At 16K tokens no merged `Qwen3` model keeps more than 4.1% of its control's multi-key retrieval, while the half-cache latent of `Qwen3-4B` keeps 104.3% and the quarter-cache latent of `Qwen3-0.6B` 18.3%. On the recall tasks of HELMET (Yen et al., 2025) at 16K tokens, the `Qwen3-4B` merge keeps 5.5% of its control's score and the latent 92.6%. On

Table 2: RULER's needle tasks in lm-evaluation-harness after the matched budget, in % of the matched control. Single averages the three single-needle tasks, multi-key 1 to 3 are defined in §5.2, and dashes lie beyond `Llama-2-7B`'s 4K training context.

		4K tokens				multi-key 2	
model	cache kept	single	multi-key 1	multi-key 2	multi-key 3	8K	16K
Head merge: each pair of KV heads averaged, 8 to 4 (32 to 16 on Llama-2-7B)							
Qwen3-0.6B	1/2	93.7	84.3	4.1	2.7	3.0	0.0
Qwen3-4B	1/2	92.8	76.9	40.9	11.5	8.2	0.0
Qwen3-8B	1/2	100.0	90.9	56.9	8.1	20.6	0.0
Qwen3-14B	1/2	99.8	90.0	88.9	79.4	66.7	4.1
Llama-2-7B	1/2	92.8	37.9	9.6	22.4	–	–
Joint low-rank latent: keys and values from one latent of the given rank							
Qwen3-0.6B, rank 512	1/4	99.3	107.3	89.5	61.6	59.7	18.3
Qwen3-4B, rank 1024	1/2	100.0	101.5	97.0	96.4	98.0	104.3

LongBench (Bai et al., 2024), which some conversion papers report, the 8B merge keeps 39.9% of its control on the retrieval tasks but 75.8% on summarization (Appendix C.10).

5.3 Re-evaluating released conversions

MHA2MLA's `SmolLM-1.7B` conversions keep at least 95.1% of their full-cache twins' commonsense average but less than half of their multi-key retrieval under either scoring (Appendix Table 8). The larger releases keep retrieval. MHA2MLA's most compressed `Llama-2-7B` keeps 85.9%, and TransMLA's releases match their `Llama-3-8B` parent (Ji et al., 2025; Meng et al., 2025). The `SmolLM` loss is not the latent's alone, since MHA2MLA also removes the rotary embedding from most dimensions, which by itself keeps 70.0% of the control on `Qwen3-0.6B` (Appendix C.1). Minitron's prune-and-heal releases, which remove no attention head within the layers they keep, keep multi-key retrieval, and Sheared-LLaMA's, which remove heads, lose much of it. The two recipes also differ in data, budget and teacher (Appendix E; Sreenivas et al., 2024; Xia et al., 2024).

5.4 MLP conversions cost far less retrieval

Replacing SiLU with ReLU in every MLP (Mirzadeh et al., 2024) starts from a larger loss gap than the head merge at 0.6B, 4B and 8B but keeps far more multi-key retrieval (Appendix Table 10 and Figure 3). At 8B the two keep about the same MMLU, while the ReLU swap keeps 98.9% of multi-key retrieval against the merge's 58.9%. How much retrieval returns therefore depends more on which module a conversion changes than on how far it moves the model.

5.5 Training ten times longer recovers only part of the retrieval

On one schedule run to ten times the matched budget, the 0.6B merge ends at 18.6% of a control trained the same way (Appendix Table 11). `Qwen3-4B` gains more but ends at about half its control, with runs that end at one, four and ten times the budget keeping 34.0, 34.4 and 53.3%. Its checkpoints before the learning-rate decay scatter by tens of points, so only the ends of schedules are compared (Appendix Figure 5). Ten times our budget is at most a quarter of what TransMLA trains on (Appendix Table 28; Meng et al., 2025), so these results bound recovery only over the tested range. Section 7 asks whether it also fails when nothing is removed.

6 What determines recovery

6.1 Compression dimension

The three conversions shrink the cache along three different dimensions, head merging the number of KV heads, the joint latent the feature dimension and token pooling the number of cached tokens. The joint latent keeps far more multi-key retrieval than merging or fixed pooling at a matched cache,

Table 3: Multi-key retrieval after the matched budget, in % of the matched control ($n = 100$). (a) Half the cache, across `Qwen3` sizes. (b) Cache reductions on `Qwen3-0.6B`. †Exact by construction: the 0.6B hidden state has 1024 dimensions, so a rank-1024 latent reproduces the key and value projections exactly. Appendix Table 15 gives the full grid.

(a) Half the cache, across sizes

	`Qwen3` size			
conversion	0.6B	1.7B	4B	8B
latent, rank 1024	100.0†	109.2	97.9	96.8
head merge, mean of a pair	7.4	25.0	34.0	58.9
token pooling, mean of 2	7.4	22.4	58.5	67.4

(b) Cache reduction, `Qwen3-0.6B`

conversion	reduction	%
latent, rank 1024	2×	100.0†
latent, rank 512	4×	89.1
latent, rank 256	8×	28.6
latent, rank 128	16×	1.1
head merge	2×	7.4
head prune	2×	9.6
token pooling, 2 tokens	2×	7.4
token pooling, 4 tokens	4×	1.1

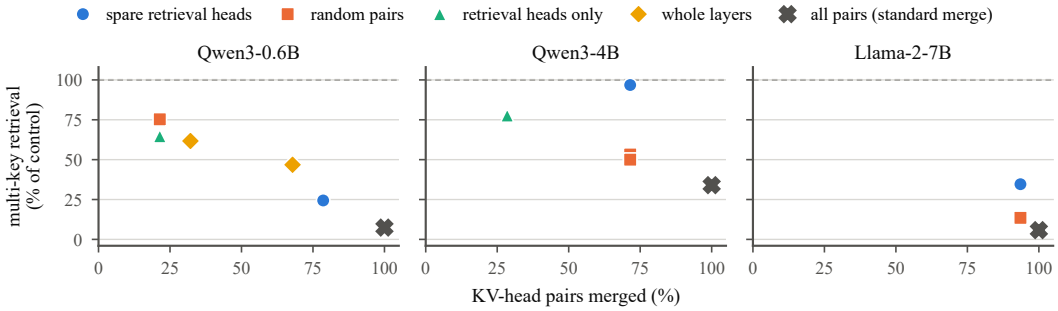

Figure 2: Which KV-head pairs to merge. For each base after the matched budget, multi-key retrieval in % of the matched control (dashed line) against the share of KV-head pairs merged. Retrieval heads come from one forward pass of the base (§6.2), and random pairs match the count of the retrieval-head condition at the same share. Sparing the retrieval heads keeps nearly all retrieval on `Qwen3-4B` and more than random pairs on `Llama-2-7B`, the conversion papers' base, while on `Qwen3-0.6B` the choice of pairs matters little (Appendix Table 16).

at every size (Table 3). At half the cache it keeps 109.2%, 97.9% and 96.8% of the control at 1.7B, 4B and 8B, and on `Qwen3-4B` a rank-512 latent still keeps 83.0% at a quarter (Appendix Table 15). Pooling pairs of tokens roughly matches the head merge at 0.6B and 1.7B (7.4% against 7.4% and 22.4% against 25.0%). From 4B on it keeps more (58.5% against 34.0% and 67.4% against 58.9%), though far less than the latent. Across families pooling matches or beats the merge, and the `Falcon3-1B` latent keeps 102.5% (Appendix Table 15).

Proposition 1 gives the latent the better start, since at a matched cache it can represent any head merge, but it does not guarantee retrieval. Below rank 512 the latent's retention falls steeply, to 1.1% of the control at rank 128 on `Qwen3-0.6B` (Appendix C.5).

6.2 Preserving retrieval heads

We first identify the heads that retrieval relies on. Following Wu et al. (2025), one teacher-forced forward pass of the base over needle prompts scores each attention head by how often its strongest attention falls on the needle token being copied. We call the heads scoring at least 0.1 *retrieval heads*. On `Qwen3-4B`, sparing the retrieval heads, that is, merging only the 103 of 144 pairs without them, keeps 96.8% of the control, where as many random pairs keep 53.2% and 50.0% on two draws (Figure 2). The spared merge keeps 64% of the cache rather than half.

Sparing the retrieval heads also beats as many random pairs on `Qwen3-8B` (89.5% against 63.2%) and on `Llama-2-7B` (34.6% against 13.5%). Disabling the retrieval heads of the `Qwen3-4B` base removes its multi-key retrieval (a drop of 96.0 points, against 0.0 for as many random heads).

Table 4: What each conversion buys at inference (whole model at the shape of `Qwen3-4B`, one GH200). Cache: share kept per token and layer, and million tokens fitting in the memory that holds 0.6 million of the full cache. Relative to the full cache: decoding FLOPs, prefill of one 16K-token prompt, decoding at 16K tokens and batch 8. Recovery: MK2 and commonsense average in % of the matched control ($n = 400$), on the model marked where it differs (Appendix C.12).

	cache		relative to the full cache (×)			% of control ↑	
conversion	kept (%)	fits (M tok.)	FLOPs ↓	prefill ↑	decode ↑	MK2	commonsense
full cache	100	0.6	1.0	1.00	1.00	100.0	100.0
head merge	50	1.1	1.0	1.01	1.37	31.5	90.9
pooled tokens, m=4 †	25	2.3	0.6	1.20	1.63	0.6	83.5
our latent, r=1024	50	1.1	73.0	1.00	0.27	97.1	99.5
MLA form, 512+64 ‡	28	2.0	2.8	0.85	1.41	1.3	95.6

Recovery is measured on Qwen3-4B except †Qwen3-0.6B and ‡Llama-3.1-8B.

Choosing which heads to merge buys little at 0.6B, where disabling as many random heads already removes most of the retrieval. The number of retrieval heads predicted the `Qwen3-14B` merge but not the merges of `Llama-3.1-8B` and `Mistral-7B`, which keep 8.6% and 7.0% (Appendix C.6). Model families thus tolerate head merging very differently, and the number of retrieval heads predicts recovery within `Qwen3` but not across families.

6.3 Lookup data and transfer

Replacing 12.4% of the training tokens with synthetic registry documents, a lookup format disjoint from the evaluation's, raises the 4B merge from 34.0% of its control on web text to 92.0% of a control trained on that mixture. The 0.6B and 1.7B merges gain as well (Appendix Table 12). The gain falls off with distance from the trained task. On random-string keys the 4B merge rises only from 3.0% to 20.0% accuracy, against 94.0% for the web-text control, and on variable tracking it gains little.

6.4 Memory savings and decoding cost

Table 4 reports how many tokens each cache fits, how fast each model prefills and decodes, and how much retrieval it keeps. Measured on the whole model at the shape of `Qwen3-4B`, at 16K tokens and batch 8, the merge decodes 1.37 times as fast as the full cache, pooled tokens 1.63 times and the MLA form 1.41 times. Our implementation of the latent decodes at 0.27 times the full cache's speed, since it rebuilds the keys of every cached token at each step, which takes 73.0 times the decoding FLOPs.

The MLA form folds this reconstruction into the query projection. Yet converted from `Llama-3.1-8B` with TransMLA's converter and trained for the matched budget, it keeps 1.3% of the control's multi-key retrieval, while TransMLA's `Llama-3-8B` releases, trained on 30B to 60B tokens, keep it (§5.3). The two differ in base and budget, so they do not show an inherent trade-off between speed and retrieval.

7 Continued pretraining does not find an exact repair

The merge stays below its control (§5.5) because it deleted what retrieval needs or because the training loss does not lead back to it. The value-rotation probe of Proposition 2 separates the two, since it deletes nothing and has an exact inverse. On `Qwen3-0.6B`, continued pretraining closes most of the gap between its initial held-out loss of 18.19 nats and the original model's 2.67. Yet multi-key retrieval stays below a tenth of the control at every size from 0.6B to 8B (Table 5), and at 5.8% on `Qwen3-0.6B` after ten times the budget.

Of the training targets we tried on `Qwen3-0.6B`, each for 1000 steps on the same data (Appendix Table 18), only a local one brings retrieval back. Cross-entropy under AdamW or Muon (Jordan et al., 2024), KL divergence to the original model's outputs and matching the residual stream after every layer keep at most 1.1% of the control. Matching each attention module's output to the original's on the original inputs, the target of LoLCATs (Zhang et al., 2025), keeps 75.3%. A closed-form ridge

Table 5: The value-rotation probe of Proposition 2. Multi-key retrieval in % of the matched control after continued pretraining or after one closed-form ridge fit of the output projections. Gradient: per head, the cosine at the rotated initialization between the negative cross-entropy gradient on its output projection and its exact repair, as the mean over heads and the share of heads where it is positive (random directions: about 0 and 50%).

	multi-key (% of control)		gradient cosine to the repair	
base	continued pretraining	closed-form solve	mean ($\times 10^{-4}$)	heads > 0 (%)
Qwen3-0.6B	1.1	93.6	−2.0	48.4
Qwen3-1.7B	1.3	113.2	0.3	49.3
Qwen3-4B	8.5	102.1	−0.8	49.0
Qwen3-8B	8.4	103.2	1.3	51.9
SmolLM2-1.7B	0.0	106.5	−0.4	49.1

fit of the output projections from 8 sequences restores 93.6% on `Qwen3-0.6B`, 113.2, 102.1 and 103.2% at 1.7B, 4B and 8B, and 106.5% on `SmolLM2-1.7B` (Table 5).

At the rotated initialization the cross-entropy gradient on the output projections points toward the exact repair on about half of the heads (48.4% to 51.9%), as a random direction would, on every base we measured (Table 5). Training does not find the direction later, since after the matched budget the output projections sit at cosine 0.01 to the repair and 0.99 to where they started (Appendix C.7). The head merge poses a different obstacle. Aligning each value pair before merging, as Align-heads does (Jin et al., 2025), moves the merge only from 7.4 to 8.5% at 0.6B and from 34.0 to 39.4% at 4B (Appendix C.7). The merge deletes what the two heads do not share. A local repair therefore fixes this controlled misalignment but is no general remedy for what a compression deletes.

8 Practical implications and limitations

Evaluation and design. Evaluate multi-key retrieval against a matched control, beside the loss, the commonsense benchmarks and single needles, and at longer contexts and in formats absent from any targeted training data (§5.1, §5.2, §6.3). On the `Qwen3` models we tested, compress the feature dimension rather than heads or tokens and keep the rotary embedding (§6.1, §5.3), but measure the decoding cost, since our latent decodes at 0.27 times the full cache's speed (§6.4). The MLA form, which folds the key reconstruction into the query projection and runs on kernels built for it, decodes 1.41 times as fast as the full cache, although at our budget it keeps little retrieval (§6.4). Kernels built for a conversion are what turn its smaller cache into faster decoding. If heads must be merged, spare the retrieval heads, after checking that disabling them removes more retrieval than disabling as many random heads (§6.2). For a known misalignment, try matching each module's output on the original inputs, or a closed-form correction, before continued pretraining (§7).

Limitations. The latent and fixed pooling are compared with the merge on `Qwen3` from 0.6B to 8B and on `Falcon3-1B`, pooling also on `SmolLM2-1.7B` and `Llama-2-7B`, at one training budget, and learned pooling and larger budgets may differ. The merge's retrieval drop shrinks with model size (§5.1). Our largest budget is at most a quarter of TransMLA's (Appendix Table 28), and one recipe, tuned on `Qwen3`, serves most bases, which lowers some controls, most of them outside `Qwen3` (Appendix F.1). Our implementation and the harness agree on every ordering we report but differ in absolute scores (Appendix H). Our own MLA-form conversion is trained only from `Llama-3.1-8B`, and the main-text speeds use the shape of `Qwen3-4B`. The local repair is tested only on an exactly invertible probe.

Conclusion. After a KV-cache conversion, recovery of the loss and the commonsense benchmarks does not predict recovery of multi-key retrieval. At the budgets we tested, a conversion that compresses the feature dimension or spares the retrieval heads keeps more multi-key retrieval than ten times the web-text tokens buy back. The value-rotation probe shows that an exact repair can exist without continued pretraining finding it, which argues for a local alignment when the source of a mismatch is known.

REPRODUCIBILITY STATEMENT

All base models and released converted models evaluated in this work are publicly available checkpoints (§4 and §5.3). We describe the conversions in §3 and Appendix F, and detail the continued-pretraining recipe and evaluation settings in §4 and Appendices F and H. We evaluate the released checkpoints in §5.3 using the same harness as our own conversions. Proofs of both propositions appear in Appendix A. Upon publication, we will release the conversion, continued-pretraining, and evaluation code, together with our evaluation outputs.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901. Association for Computational Linguistics, 2023.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Learning Representations*, 2024.

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

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Agustín Piqueres Lajarín, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Ben Burtenshaw, Clémentine Fourrier, Haojun Zhao, Hugo Larcher, Mathieu Morlon, Cyril Zakka, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Conference on Language Modeling*, 2025.

Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. Transformers to SSMs: Distilling quadratic knowledge to subquadratic models. In *Advances in Neural Information Processing Systems*, 2024.

Aviv Bick, Tobias Katsch, Nimit Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. *arXiv preprint arXiv:2502.14458*, 2025.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Conference on Language Modeling*, 2025.

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: KV-cache compression with low-rank projection. In *International Conference on Learning Representations*, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936. Association for Computational Linguistics, 2019.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Falcon-LLM Team. The Falcon 3 family of open models. https://huggingface.co/blog/falcon3, December 2024.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems*, 2025.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128K context. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. In *International Conference on Learning Representations*, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness. https://zenodo.org/records/12608602, July 2024.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7376–7399. Association for Computational Linguistics, 2025a.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Self-distilled attention gating for efficient long-context prefilling. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

Daniel Goldstein, Eric Alcaide, Janna Lu, and Eugene Cheah. RADLADS: Rapid attention distillation to linear attention decoders at scale. In *Conference on Language Modeling*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers. In *International Conference on Learning Representations*, 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling*, 2024.

Xiang Hu, Xinyu Wei, Hao Gu, Minshen Zhang, Tian Liang, Huayang Li, Lei Zhu, Yan Wang, Sirui Han, Yushi Bai, Kewei Tu, Haitao Mi, and Leo Liang. Hierarchical sparse attention done right: Toward infinite context modeling. *arXiv preprint arXiv:2607.02980*, 2026.

Ajay Jaiswal, Zhe Gan, Xianzhi Du, Bowen Zhang, Zhangyang Wang, and Yinfei Yang. Compressing LLMs: The truth is rarely pure and never simple. In *International Conference on Learning Representations*, 2024.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

Tao Ji, Bin Guo, Yuanbin Wu, Qipeng Guo, Lixing Shen, Zhan Chen, Xipeng Qiu, Qi Zhang, and Tao Gui. Towards economical inference: Enabling DeepSeek's multi-head latent attention in any transformer-based LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33313–33328, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Qingyun Jin, Xiaohui Song, Feng Zhou, and Zengchang Qin. Align attention heads before merging them: An effective way for converting MHA to GQA. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 8804–8816, 2025.

Tian Jin, Nolan Clement, Xin Dong, Vaishnavh Nagarajan, Michael Carbin, Jonathan Ragan-Kelley, and Gintare Karolina Dziugaite. The cost of scaling down large language models: Reducing model size affects memory before in-context learning. In *International Conference on Learning Representations*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. `https://kellerjordan.github.io/posts/muon/`, 2024.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, Jinkai Hu, Jiayao Li, Rui Gao, Zekun Li, Songquan Zhu, Jingkai Zhou, and Pengyu Zhao. MiniMax sparse attention. *arXiv preprint arXiv:2606.13392*, 2026.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *International Conference on Machine Learning*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Fanxu Meng, Pingzhi Tang, Zengwei Yao, Xing Sun, and Muhan Zhang. TransMLA: Migrating GQA models to MLA with full DeepSeek compatibility and speedup. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. ReLU strikes back: Exploiting activation sparsity in large language models. In *International Conference on Learning Representations*, 2024.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In *Advances in Neural Information Processing Systems*, 2024.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37396–37412. PMLR, 2024.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, pp. 30811–30849, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, 2026.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392. Association for Computational Linguistics, 2016.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, Chenhan Yu, Wei-Chun Chen, Hayley Ross, Oluwatobi Olabiyi, Ashwath Aithal, Oleksii Kuchaiev, Daniel Korzekwa, Pavlo Molchanov, Mostofa Patwary, Mohammad Shoeybi, Jan Kautz, and Bryan Catanzaro. LLM pruning and distillation in practice: The Minitron approach. *arXiv preprint arXiv:2408.11796*, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. RazorAttention: Efficient KV cache compression through retrieval heads. In *International Conference on Learning Representations*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The Mamba in the Llama: Distilling and accelerating hybrid models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62432–62457, 2024.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *International Conference on Learning Representations*, 2025.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. In *International Conference on Learning Representations*, 2025.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025b.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context models effectively and thoroughly. In *International Conference on Learning Representations*, 2025.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online vector quantization with near-optimal distortion rate. In *International Conference on Learning Representations*, 2026.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. LoLCATs: On low-rank linearizing of large language models. In *International Conference on Learning Representations*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Xinyu Zhao, Fangcong Yin, and Greg Durrett. Understanding synthetic context extension via retrieval heads. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77885–77910. PMLR, 2025.

GUIDE TO THE APPENDIX

Appendix A proves the two propositions of §3. Appendix B gives the per-channel analysis from which the study started. In that analysis a conversion loses a capability in one of two ways. Either the information its readers need is gone, as after the head merge or token pooling, or the readers can no longer read it, as for the probe of Proposition 2. The later appendices keep the vocabulary of that analysis. *Axis A* is whether the information survives, *axis B* is the teacher-forced alignment of a module's output with the original at initialization, and a condition read from the two falls into the cheap tier, the cost tier or the *wall*, where it stays near zero at every budget we ran. Appendix C gives the full controlled evidence behind §5 and §6. Appendix D surveys what conversion papers evaluate, and Appendix E scores two public prune-and-heal families. Appendix F gives the experimental details, and Appendix G shows that the updates of continued pretraining are not low-rank. Appendix H compares our two builders of MK2 with the standard harness. The remaining appendices hold the short-context benchmarks (Appendix I) and the per-token loss decomposition (Appendix J). The last appendix states how a generative AI tool was used (Appendix K).

A PROOFS

Proposition 1. Order the $2gd$ rows of W by head, $W = [W_K^{(1)}; \dots; W_K^{(g)}; W_V^{(1)}; \dots; W_V^{(g)}]$, in blocks of d rows. For a merged pair of key heads $(2j-1, 2j)$ the merge gives the query heads of both groups the key $\frac{1}{2}\big(k^{(2j-1)} + k^{(2j)}\big)$, that is, it applies $P = \frac{1}{2}\left(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right) \otimes I_d$ to the two stacked keys, and the value side is the same. Π is therefore block diagonal with g copies of P, $g/2$ for the keys and $g/2$ for the values. Since $P = P^\top = P^2$ and $\operatorname{rank} P = d$, Π is an orthogonal projection of rank gd.

(i) Let $B \in \mathbb{R}^{2gd \times gd}$ have orthonormal columns spanning the range of Π, for instance the normalized pair sums $(e_a + e_{a+d})/\sqrt{2}$ within each pair. Then $\Pi = BB^\top$, and $W_U = B$, $W_D = B^\top W$ give $\Pi W = W_U W_D$. The latent layer reconstructs $W_U c_t = \Pi W x_t$, in which both key blocks of a merged pair equal the merged key $\frac{1}{2}\big(k^{(2j-1)} + k^{(2j)}\big)$ and both value blocks the merged value. A per-head key normalization, which normalizes each key block separately with parameters shared across heads, and the rotary embedding, which rotates every key block of position t by the same $\mathcal{R}_t$, therefore give both key blocks the key that the merged layer computes once for the pair. Values carry neither operation, and the query and output projections are unchanged, so the latent layer computes the merged layer's output exactly.

(ii) $\operatorname{rank}(\Pi W) \le \operatorname{rank} \Pi = gd$. By the Eckart–Young–Mirsky theorem (Eckart & Young, 1936), the truncated SVD W_{gd} minimizes $\|W - M\|_F$ over all matrices M of rank at most gd, so $\|W - W_{gd}\|_F \le \|W - \Pi W\|_F$. The same argument applies to WX, since $\operatorname{rank}(\Pi W X) \le gd$ for every X, and gives the bound for its own truncated SVD $(WX)_{gd}$, not for $W_{gd}X$. □

Proposition 2. Every cached value $R_j v_s^{(j)}$ is an invertible image of the original, since $v_s^{(j)} = R_j^\top \big(R_j v_s^{(j)}\big)$. In equation 1 the weights $a_{ts}^{(i)}$ depend on the query and key projections only, so with $W_O^{(i)} \mapsto W_O^{(i)} R_{j(i)}^\top$ the term of query head i becomes $W_O^{(i)} R_{j(i)}^\top \sum_s a_{ts}^{(i)} R_{j(i)} v_s^{(j(i))} = W_O^{(i)} \sum_s a_{ts}^{(i)} v_s^{(j(i))}$, since $R^\top R = I$. The repair thus composes the inverse map into W_O. On teacher-forced inputs the repair is also a solution of a least-squares problem in W_O alone (the unique one when the teacher-forced head outputs have full column rank), which the closed-form refit of §7 solves up to its small ridge term (Appendix F). □

B TWO WAYS A CONVERSION LOSES A CAPABILITY

This section defines the two quantities behind the evidence of §5 and §6, both measured per channel before training. The first is whether a modified module still carries the information its readers need;

the axes, the choice of merged heads and the data of §6.1 and §6.2 concern information that is deleted. The second is whether the readers can still read what the module carries, and §7 concerns information that is left in place but unreadable.

Setup. Write a pretrained decoder as a residual stream $h_t^{(\ell)}$ updated by attention and MLP modules, $h_t^{(\ell+\frac{1}{2})} = h_t^{(\ell)} + W_\ell^O\, a_\ell(h_{\le t}^{(\ell)})$ and $h_t^{(\ell+1)} = h_t^{(\ell+\frac{1}{2})} + M_\ell(h_t^{(\ell+\frac{1}{2})})$, where the attention read a_ℓ mixes value vectors $v_{\ell,s} = W_\ell^V h_s^{(\ell)}$ of earlier positions with query-key weights. A *retrofit* replaces one module f by $\tilde{f}$ and then continues training every parameter on ordinary language-modelling data for a budget B of tokens. Every module has *consumers*, the parameters that read its output: for a_ℓ the output projection W_ℓ^O, and through the residual stream the MLP and every later projection. They are frozen at the moment of surgery and were trained on the inputs the original module produced, and the structure they carry, bought by the base model's pretraining, is the whole reason to retrofit rather than to train from scratch. The cost of a conversion is the cost of re-establishing agreement between the modified producer and its unmodified consumers.

A consumer reads its input through *channels*: for W_ℓ^O the channels are the query heads' reads, one block of columns per head; for the residual-stream consumers they are the coordinates of the stream. Everything below is stated per channel and then aggregated.

Definition 1 (Readability)**.** *The* readability c_h *of channel* h *is the teacher-forced cosine between what the consumer receives through that channel after and before the modification — for an attention head,* $\cos\big(W'^{O}_{[:,h]} a'_h,\ W^O_{[:,h]} a_h\big)$*, evaluated on the original model's own inputs and averaged over positions. It is a property of the initialization alone: no training enters it. A channel is* unreadable *when* $c_h \approx 0$*, and the readability profile of a modification is the distribution of* c_h *over its channels, not the mean.*

Definition 2 (Absent information)**.** *The information of a channel is* absent *if the quantity the consumer relies on cannot be recovered from what the modified module emits at that position* and the architecture keeps it that way*: two KV heads averaged into one and tied through training, a block pooled into a summary. Information that a modification merely discards at initialization while the capacity to carry it remains is* ***not*** *absent — a rank-*512 *truncation of the stacked KV map removes energy the base was using, reads* $R^2 = 0.84$ *on the value reconstruction, and still recovers 89.1% of control, the same 90.2% as unconstrained training from the same initialization (§C.5, Table 15). Absence is measured by reconstructing the original per-position keys and values from the modified module's output, with the caveat that reconstruction is the right functional for the value path and not for the key path.*

Claim. Let B be a continual-pretraining budget and let the endpoint be a behaviour that depends on reading individual tokens out of the context. We claim five things about how the fraction of control recovered at budget B depends on the readability profile. Appendix B.2 gives the series behind them (Table 7 and Figure 4).

(1) The price. Among modifications that leave every channel readable, the recovered fraction is monotone and convex in readability, steepening as readability falls: on `Qwen3-0.6B` a uniform rotation of every value head gives 90.4/75.5/39.4/5.3% of control at $c = 0.87/0.71/0.45/0.21$, and a noise perturbation calibrated to the same readabilities lands within a few points of each rotation (Table 7, Figure 4a). The shape repeats on a second base and the slope does not: `SmolLM2-1.7B` gives 91.9/71.0/43.5% at $c = 0.985/0.94/0.87$, about four times steeper in $1 - c$, with its 50% crossing between 0.94 and 0.87 where `Qwen3`'s lies between 0.71 and 0.45 (Figure 4a and §C.9).

(2) Budget lifts the curve between the endpoints. More tokens raise the recovered fraction at every readability above zero and raise it most where the curve is steepest: at three times the budget, against a control trained through the same schedule, the rotation at 0.21 goes from 5.3% to 37.5% and the matched noise condition from 3.2% to 20.0%, while the drift-matched noise condition at 0.45 goes from 44.7% to 77.5% (Table 7).

(3) Two degenerate endpoints. Exactly two conditions are flat in budget. A channel at $c_h = 0$ exactly: the fully rotated condition recovers 1.1% at the matched budget, 0.0% at three times it and no more at three times the learning rate. And information absent in the sense of Definition 2: the `Qwen3-0.6B` head merge sits at 7.4% and stays at about a fifth of a control trained through the same schedule from four to ten times the budget (Table 14; on `Qwen3-4B` a ten-times schedule ends at 53.3%, Table 11). The mechanism at the first endpoint is measured rather than assumed (§C.7): under continual pretraining neither end of an unreadable channel moves. The consumer's columns keep

norm ratio 1.00 and direction cosine 0.99 to their initialization, and the producer's value projection stays at cosine 0.99 to its own. The loss gradient on a channel that carries no signal points no closer to its repair than a random direction does (Table 5), so the network routes around the channel instead of repairing it, although the repair is a single linear map that exists in parameter space.

(4) Dependence on the profile. The recovered fraction depends on the distribution of c_h and not on its average. At the same module mean 0.86, the level of the initialization series that leaves 13.8% of heads unreadable with the rest at 1.00 recovers a median 44.6% of control over three data slices, while a uniform rotation that leaves every head at 0.866 recovers a median 71.1%. Nor is the cost additive over channels: losing 13.8% of heads costs far more than its share, and the lost fraction traces a convex curve of its own — rotated, dropped and pruned heads at 13.8/45.1, $12.5/33.5$ and 50.0% of heads recover 43.6/8.5, 51.1/34.0 and 9.6% of control, three different surgeries on one curve within about ten points (Table 7, Figure 4b).

(5) The boundary is a contour of the budget. Between a steeply priced channel and an unreadable one there is no line that belongs to the surgery. The operational cut this paper uses, $c_h < 0.2$, is where the matched budget stops buying; at three times the budget it sits lower, as (2) shows directly. A diagnostic measured before training should therefore return a verdict conditioned on the budget the practitioner intends to spend, together with the two endpoint tests, rather than a class label.

Which of these a modification meets is a property of its *initialization* and not of its architecture. The sharpest case is a single surgery under two initializations: an exactly invertible rotation of every value head, with the inverse folded into the output projection, reproduces the original model bit for bit and costs nothing; without the fold, the same architecture over the same information recovers 1.1% and does not improve with budget or learning rate. A closed-form refit of the consumer moves readability on a fixed surgery in the same way, and no initialization restores a channel whose information is absent.

B.1 Why cross-entropy cannot see it

The loss is unreliable in two opposite directions. First, the language-modelling loss on web text barely contains the behaviour a retrofit can lose. Positions whose bigram already occurred at least 256 tokens earlier, the ones an induction read could copy, are 8.1% of positions and 4.7% of the loss mass on our held-out set (Appendix J), so removing precise long-range reading can move the aggregate by a few percent at most. The residual loss gap of the conditions that lose retrieval is also uniform over positions rather than concentrated in a tail. The head merge's per-token excess loss is +0.212 nats on novel tokens, +0.090 on near copies and +0.122 on far copies, and every trained condition shows the same flat profile (Appendix J). Natural-text copying survives the merge and keyed retrieval of a needle among distractors does not, and since the objective does not contain the second, a gate on the loss cannot see it disappear. Second, a modification that perturbs every position starts with a catastrophic loss and recovers it over the first few hundred steps whether or not it will recover the read (Figure 3). The activation swap starts +11.52 nats above dense and reaches 80.9% of control; the head merge starts +10.43 nats above and reaches 7.4%; the rotation starts +15.29 nats above, recovers most of its loss, and reaches 1.1% (Table 6; Table 13 gives every condition). The loss measures the aggregate distance from the training distribution, and it shows neither whether the information the consumers need is still there nor whether they can reach it.

Table 6: Loss and retrieval of the `Qwen3-0.6B` runs of this subsection and of Figure 3: held-out loss above the original model before and after training, and multi-key retrieval after it, in % of the matched control ($n = 100$). Every run trains for the matched budget except the second head merge, which trains for ten times it and is compared with a control trained through the same schedule (Table 14).

	loss above the original (nats)		multi-key
conversion	before training	after training	(% of control)
activation swap (ReLU)	+11.52	+0.10	80.9
head merge	+10.43	+0.19	7.4
head merge, ten times the budget	+10.43	+0.08	18.6
value rotation (Proposition 2)	+15.29	+0.77	1.1

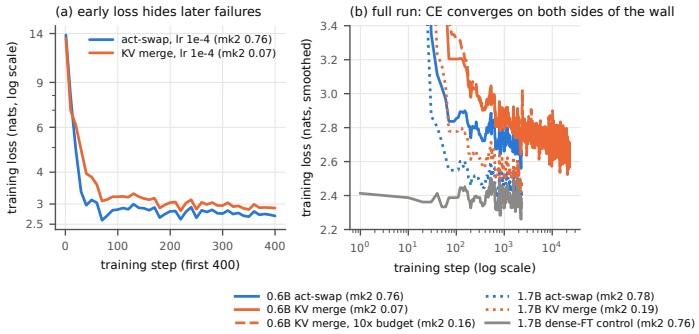

Figure 3: Cross-entropy cannot order the outcomes. (a) The first 400 steps of the activation swap (blue) and the head merge (orange), which start an order of magnitude above the dense loss and converge. (b) Full runs on a log step axis, lightly smoothed, with each run's endpoint MK2 in the legend. Of the three `Qwen3-0.6B` runs, the head merge trained for ten times the budget (orange, dashed) ends with the lowest loss but recovers little retrieval, while the activation swap (blue, solid) ends with a higher loss and recovers most of it. Table 6 gives the held-out loss and the retrieval of these runs.

B.2 How recovery depends on readability

The claims above say that what a conversion recovers depends on how readable each channel is, and that the mean over the channels does not capture it. Testing this needs conditions that change readability in controlled steps while the information stays intact, and conditions that lose the same channels in other ways. We built five series of such conditions on `Qwen3-0.6B`, trained each with the matched budget and scored it as multi-key retrieval in % of the matched control. The *initialization series* rotates every value head and folds the exact inverse into the output projection for a nested share of the heads (none, half, 85% or all), so the remaining heads stay unreadable. The *uniform rotation* turns every value head by the same angle, so every head is partly readable and none is lost. The *lost heads* series deletes the heads that the initialization series leaves unreadable, by overwriting each with its partner, and adds a prune of half the heads. The *K/V noise* series adds Gaussian noise calibrated to the same module alignments, and the *merge refit* raises the head merge's alignment in closed form while its information stays lost. Table 7 lists every condition with its alignment at initialization, its share of unreadable heads and its outcome, and Figure 4 plots them.

The initialization series. The rotation's cost is set by its initialization, and the mean alignment does not predict it. Composing the output projection with the transposes of the head rotations reproduces the original model bit for bit, so the rotation condition's 1.1% measures one initialization of that architecture and not the architecture. We therefore trained the same surgery from four initializations in which a nested subset of the 28×8 head rotations is undone exactly in W^O and the rest are left in place: none ($\alpha = 0$, the published condition), half, 85%, and all ($\alpha = 1$, the exact inverse, verified on the model rather than trained). A partial rotation R^α is unavailable, since about half the Haar-random matrices have no real square root and a linear interpolation passes through singular matrices, which would destroy information and confound the series. We predicted before training a monotone series within 15 points of the line that reads the recovered fraction as $100\times$ the module alignment. The series is monotone, at 1.1, 8.5 and 43.6% of control for $\alpha = 0$, 0.5 and 0.85 (Table 7), and far from its line. The 85% level sits at module alignment 0.86, the activation swap's value, and recovers 43.6% where the swap recovers 80.9% and the line predicts 86.0; the 50% level at alignment 0.55 recovers 8.5% against a predicted 54.8. Both misses exceed the 17-point mark that had separated lost-information conditions from intact ones, with the per-head KV reconstruction R^2 at 1.00 throughout, so the refutation condition set before training was met. The merge run the same way, with a half-strength closed-form refit between the published merge and the full refit, stays flat as predicted (7.4, 4.3 and 5.3% at alignments 0.48, 0.70 and 0.75), so a better initialization buys nothing once the information is gone. Training the fully rotated condition three times longer, against a matched three-times control (0.80), leaves it at 0.00, 0.0% of that control (predicted at most 15%), while the drift-matched noise condition at alignment 0.45 went from 44.7% to 77.5% with the same extra tokens.

Table 7: Five series of conditions on Qwen3-0.6B after the matched budget (lr 1e-4): MK2 at $n = 100$ and its % of the matched control (italic row). $\cos_{\mathrm{mod}}$: mean module alignment at init (the line predicts $100\cos_{\mathrm{mod}}$%); heads < 0.2: query heads with per-head cosine below 0.2. $3\times$ budget: % of a control trained through the same three-times schedule, for the conditions also run at three times the budget.

condition	$\cos_{\mathrm{mod}}$	heads < 0.2 (%)	MK2	% of control	$3\times$ budget
matched control	*1.00*	–	*0.94*	*100.0*	*100.0*
initialization series (one fixed rotation)					
rotation probe ($\alpha = 0$)	0.00	99.8	0.01	1.1	0.0
rotation probe, repaired on 50% of heads ($\alpha = 0.5$)	0.55	45.1	0.08	8.5	–
rotation probe, repaired on 85% of heads ($\alpha = 0.85$)	0.86	13.8	0.41	43.6	–
rotation probe, repaired on every head ($\alpha = 1$)	1.00	0.0	0.88	93.6	–
uniform rotation (no head lost)					
uniform rotation, $\theta = 30^\circ$	0.88	0.0	0.85	90.4	–
uniform rotation, $\theta = 45^\circ$	0.73	0.0	0.71	75.5	–
uniform rotation, $\theta = 63^\circ$	0.48	0.0	0.37	39.4	–
uniform rotation, $\theta = 78^\circ$	0.23	40.8	0.05	5.3	37.5
lost heads (tied partner copies)					
heads unrepaired at 85% deleted	0.86	12.5	0.48	51.1	–
heads unrepaired at 50% deleted	0.62	33.5	0.32	34.0	–
head prune	0.49	50.0	0.09	9.6	–
K/V noise					
K/V noise, small	0.94	0.0	0.88	93.6	–
K/V noise, set for cos ≈ 0.71	0.70	0.2	0.80	85.1	–
K/V noise, drift-matched	0.45	10.3	0.42	44.7	77.5
K/V noise, set for cos ≈ 0.21	0.22	64.5	0.03	3.2	20.0
merge refit (information lost)					
head merge	0.48	9.8	0.07	7.4	–
head merge + half-strength refit	0.70	10.0	0.04	4.3	–
head merge + closed-form refit	0.75	25.9	0.05	5.3	–

Same mean alignment, two distributions. At the same mean alignment, a rotation spread evenly over every head recovers about twice what the 85% level of the initialization series recovers, so the line reads the distribution of per-head alignment rather than its mean. The levels of the initialization series are bimodal per head, with a fraction $1 - \alpha$ of the heads at cosine ≈ 0 and the rest at 1, while the swap and the noise conditions the line was fitted on perturb every channel mildly. To vary the mean alignment with no head lost, we rotate every value head of every layer by the same angle θ in every 2-plane of a random per-head basis, so that the per-head cosine is $\cos\theta$ at every token of every head ($\theta = 30^\circ, 45^\circ, 63^\circ, 78^\circ$, i.e. $0.866, 0.707, 0.454, 0.208$). The rotation is exactly invertible, with per-head reconstruction R^2 0.9999 and no head below 0.2 in the per-head measurement. We predicted before training that each rotation would land within 15 points of its line (87, 71, 45 and 21%), that the 30° rotation would keep at least 70% of control at the mean alignment where the 85% level of the initialization series recovered 43.6%, and that the 63° rotation would land within 15 points of the noise condition at the same alignment. The four rotations recover 90.4, 75.5, 39.4 and 5.3% of control (98.2, 81.0, 39.8 and 5.3% at $n = 400$). The three rotations down to alignment 0.45 sit on the line, and a diffuse re-basing costs about what diffuse noise of the same alignment costs (39.4 against 44.7% at 0.45).

The 78° rotation falls below its line, and the extra budget shows that this floor is a price. At alignment 0.21 it recovers 5.3%, below its line by more than the tolerance and within a few points of the fully rotated condition, and the noise family calibrated to the same alignment (scale 1.20) lands beside it at 3.2%. With three times the tokens the rotation reaches 37.5% and the noise 20.0% of the matched three-times control, where the fully rotated condition reaches 0.0%. Per channel, the recovered fraction is a convex function of alignment that steepens toward zero, and the wall sits at zero signal rather than at a finite alignment.

Replicates on disjoint slices of the training stream measure the variance on the priced side. The 30° rotation reads 0.62 / 0.67 / 0.85 over three runs (median 0.67, 65.8–90.2% of control) and the 63° rotation 0.00 / 0.08 / 0.37 (median 0.08), so at 0.45 a uniform re-basing is bistable and the line's tolerance there is one run's worth of noise. On the same slices the 85% level reads 0.41 / 0.42 / 0.46 and the merge 0.02 / 0.03 / 0.07. Every 30° run stays above every run of the 85% level, so the

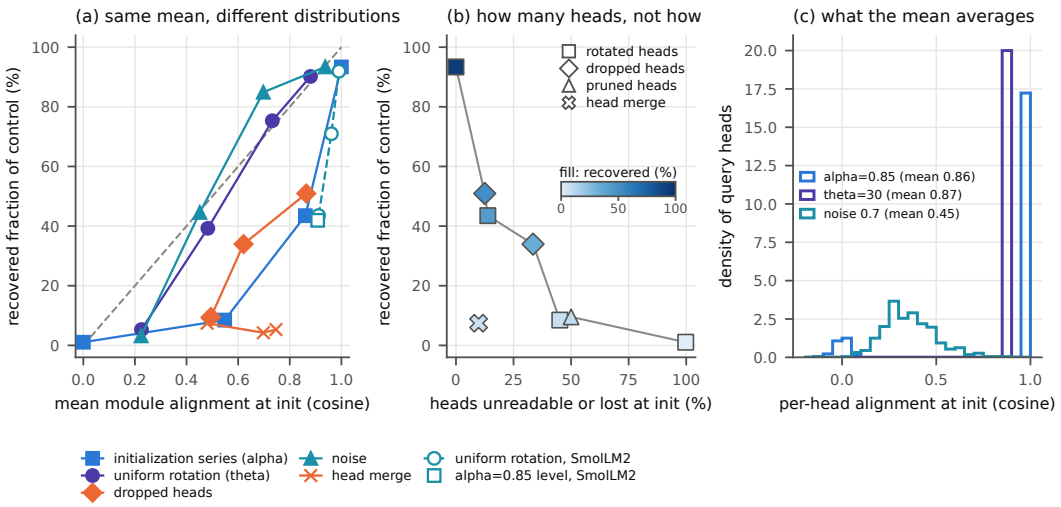

Figure 4: How recovery depends on readability on `Qwen3-0.6B`, after the matched budget (Table 7). (a) Recovered fraction against the mean module alignment at initialization for the five series, with the dashed identity line. The same uniform rotation on `SmolLM2-1.7B`, against its own control (hollow), has the same shape at about four times the slope (§C.9). At the same mean alignment the initialization series and the uniform rotation recover very different fractions, so the mean does not set the cost. (b) The same conditions against the share of heads that are unreadable (cosine below 0.2) or lost at initialization, each marker filled by its recovered fraction. Rotated, dropped and pruned heads fall on one convex curve, so the cost depends on how many heads are lost and not on how they are lost. The head merge (cross) sits far below that curve, because it deletes information rather than leaving heads unreadable (Definition 2). (c) The per-head alignments behind two equal means. The 85% level of the initialization series is bimodal, with 13.8% of the heads near zero and the rest at one, and recovers 43.6%. The 30° rotation puts every head at $\cos 30°$ and recovers 90.4%. The noise condition spreads its heads around a lower mean.

ordering of the pair survives with a worst-case margin of 65.8 against 48.8, smaller than the 46 points of the first run.

Measured against the fraction of heads lost, rotated, dropped and pruned heads lie on one curve, and the module alignment does not predict their cost. The converse series drops exactly the heads the two partial levels left rotated (12.5% and 33.5% of the heads), each overwritten by its bit-exact partner and kept tied through training, so the same channels are lost by destruction instead of by re-basing, from a module alignment *no lower* than the rotated levels' (0.86 and 0.62 against 0.86 and 0.55). We predicted the first within 15 points of its rotated twin's 43.6% and the second within 12 of 18.4%, the rotation family interpolated at that lost fraction, with the full prune of §5.5 (9.6% at half the heads) as the family's third point. The two dropped-head conditions recover 51.1% and 34.0%. The first lands within its tolerance of the rotated twin, from a module alignment (0.86) at which the line reads 86.4%, so the same channels cost the same whether they are lost by destruction or by re-basing. The second lands between the two rotated levels and above the log-linear rule it was predicted from (18.4%), by more than the tolerance at $n = 100$ and inside it at $n = 400$ (27.0%). Against the fraction of heads lost, the rotated levels (43.6 and 8.5% at 13.8 and 45.1% of the heads), the dropped-head conditions (51.1 and 34.0% at 12.5 and 33.5%) and the full prune (9.6% at 50%) lie on one convex curve to within about ten points, a shape the log-linear interpolation behind the predictions did not have (Figure 4b).

The noise family, extended to the alignments of the uniform-rotation series ($\cos \approx 0.71$ and 0.21) with noise scales 0.40 and 1.20 chosen by calibration before training, recovers 85.1% and 3.2% of control against the uniform rotations' 75.5% and 5.3% at the same alignments.

C ADDITIONAL CONTROLLED EVIDENCE

This appendix gives the full evidence that Sections 5 and 6 summarize, with every condition that supports each result and, where one was written before training, the prediction it was tested against.

C.1 ADDITIONAL COMPARISONS SUPPORTING THE MAIN TEXT

Tables 8, 10, 11 and 12 give the released-checkpoint, MLP, training-budget and lookup-data comparisons that Sections 5 and 6 summarize. Table 9 breaks the commonsense average of Table 8 down into its eight tasks.

Table 8: Released conversions and our conversions: the conversion papers' commonsense average (MMLU, ARC-E/C, PIQA, HellaSwag, OpenBookQA, WinoGrande, BoolQ (Clark et al., 2019)) and multi-key retrieval (MK2 at 4K, $n = 400$; 2K for SmolLM), in % of each block's reference. A parent is not a matched control, so a release can exceed it. The last column scores MK2 with lm-evaluation-harness (Appendix H), whose easier version reads mid-range retention higher.

		% of reference		
checkpoint	setting	commonsense	MK2	MK2, lm-eval
MHA2MLA releases (reference: the full-cache twin released with them)				
MHA2MLA, Llama-2-7B	$d_{kv} = 64$	99.1	116.6	115.8
	$d_{kv} = 32$	98.3	89.8	110.3
	$d_{kv} = 16$	96.9	85.9	97.6
MHA2MLA, SmolLM-1.7B	$d_{kv} = 32$	97.6	22.8	37.6
	$d_{kv} = 16$	97.1	23.3	48.4
	$d_{kv} = 8$	95.1	25.7	49.5
TransMLA releases (reference: the parent model)				
TransMLA, Llama-3-8B	8K	100.8	102.9	100.0
	32K	102.0	104.5	100.0
Our conversions after the matched budget (reference: the matched control)				
head merge, Llama-2-7B	all pairs	92.5	8.9	9.6
	retrieval heads spared	92.6	42.2	38.5
head merge, Qwen3-4B	all pairs	90.9	31.5	40.9
	retrieval heads spared	94.2	96.8	87.9
	as many random pairs	94.6	44.5	59.1
head merge, Qwen3-8B	all pairs	92.8	57.7	56.9
our latent, Qwen3-4B	rank 1024	99.5	97.1	97.0
our latent, Qwen3-0.6B	rank 512	96.5	83.7	89.5
probe of Prop. 2, Qwen3-0.6B	module-local repair	99.7	84.9	101.2

Partial rotary embedding alone. MHA2MLA keeps the rotary embedding on a few frequency pairs of each head and makes the rest position-free, so its released checkpoints change the positional encoding as well as the cache. To measure that change alone, we kept the rotary embedding of `Qwen3-0.6B` on the 16 highest-frequency of the 64 frequency pairs of every head, made the other pairs position-free with the weights unchanged, and trained the model through the matched budget. Multi-key retrieval reads 0% before training and 64% after it, 67.9% of the matched control (70.0% at $n = 400$), with a held-out loss of 2.730 nats against the original model's 2.661.

Table 9: Accuracy (%) of every checkpoint of Table 8 on the eight tasks behind its commonsense average, zero-shot in lm-evaluation-harness, with acc_norm for ARC-E, ARC-C, PIQA, HellaSwag (HS) and OpenBookQA (OBQA) and acc for MMLU, WinoGrande (WG) and BoolQ, as the average (avg.) uses them. Italic rows are the references of Table 8.

model	checkpoint	MMLU	ARC-E	ARC-C	PIQA	HS	OBQA	WG	BoolQ	avg.
MHA2MLA releases (reference: the full-cache twin released with them)										
Llama-2-7B	*full-cache twin*	*43.3*	*73.3*	*48.5*	*77.8*	*74.8*	*45.0*	*68.8*	*75.9*	*63.4*
	MHA2MLA, $d_{kv} = 64$	42.1	72.5	46.5	77.4	74.3	44.4	69.9	75.9	62.9
	MHA2MLA, $d_{kv} = 32$	39.1	73.1	46.3	77.8	73.6	44.4	69.5	75.1	62.4
	MHA2MLA, $d_{kv} = 16$	35.2	72.9	45.6	77.7	72.3	45.4	68.7	73.9	61.5
SmolLM-1.7B	*full-cache twin*	*27.5*	*72.8*	*45.6*	*76.0*	*65.4*	*41.4*	*60.1*	*66.4*	*56.9*
	MHA2MLA, $d_{kv} = 32$	25.9	71.3	42.9	76.1	63.9	41.2	59.8	63.2	55.6
	MHA2MLA, $d_{kv} = 16$	25.6	70.8	42.8	75.8	63.3	41.0	59.9	63.0	55.3
	MHA2MLA, $d_{kv} = 8$	24.2	69.4	41.6	74.9	61.5	39.0	58.6	63.5	54.1
TransMLA releases (reference: the parent model)										
Llama-3-8B	*parent*	*62.3*	*77.6*	*53.9*	*80.4*	*79.3*	*44.8*	*73.5*	*82.1*	*69.2*
	TransMLA, 8K	68.5	79.9	53.8	79.5	77.2	42.4	72.9	83.7	69.7
	TransMLA, 32K	69.6	79.5	53.1	80.3	78.0	44.8	74.3	85.2	70.6
Our conversions after the matched budget (reference: the matched control)										
Llama-2-7B	*matched control*	*35.5*	*70.8*	*42.7*	*76.3*	*71.3*	*39.8*	*68.2*	*75.7*	*60.0*
	head merge, all pairs	24.8	68.1	40.6	73.6	66.3	39.6	62.9	68.3	55.5
	head merge, retrieval heads spared	26.4	66.9	38.7	75.2	66.7	38.8	62.6	69.4	55.6
Qwen3-4B	*matched control*	*65.6*	*79.3*	*55.2*	*76.6*	*69.4*	*41.8*	*66.3*	*80.8*	*66.9*
	head merge, all pairs	52.5	74.4	48.6	75.0	64.2	40.2	60.1	71.4	60.8
	head merge, retrieval heads spared	58.2	75.6	51.2	74.7	66.5	41.6	62.7	73.3	63.0
	head merge, as many random pairs	59.2	75.6	50.6	75.4	66.2	40.8	63.2	75.0	63.3
	our latent, rank 1024	65.0	79.5	54.5	76.8	69.3	41.6	66.8	79.0	66.6
Qwen3-8B	*matched control*	*70.6*	*81.6*	*59.6*	*79.2*	*75.7*	*43.4*	*71.7*	*81.7*	*70.4*
	head merge, all pairs	57.2	77.9	53.6	76.9	70.9	42.6	67.6	76.1	65.4
Qwen3-0.6B	*matched control*	*44.6*	*65.0*	*38.1*	*69.2*	*51.2*	*35.0*	*58.6*	*65.0*	*53.3*
	our latent, rank 512	42.6	61.3	34.7	68.4	49.1	34.4	55.7	65.5	51.5
	probe of Prop. 2, module-local repair	43.7	65.1	36.4	68.7	50.9	35.0	57.9	67.6	53.2

Table 10: Head merge and MLP conversions on `Qwen3` after the matched budget. The initial loss gap is the held-out loss before training minus the original model's; MMLU and multi-key retrieval (MK2 at 4K) are percentages of the matched control. The ReLU swap replaces SiLU in every MLP, and 2:4 sparsity deletes half of every MLP projection.

		initial loss	% of the matched control	
base	conversion	gap (nats)	MMLU	multi-key
Qwen3-0.6B	head merge	+10.43	62.2	7.4
	ReLU swap	+11.52	89.3	80.9
	MLP 2:4 sparsity	–	81.0	75.5
Qwen3-4B	head merge	+5.64	80.0	34.0
	ReLU swap	+9.87	79.1	100.0
Qwen3-8B	head merge	+3.58	81.1	58.9
	ReLU swap	**+9.54**	**81.2**	**98.9**

Table 11: The head merge's MK2 at 4K as a percentage of a control trained through the same schedule (1.5×10^8 tokens per multiple of the matched budget). Top: runs trained to exactly that budget, the decay of the learning rate included. Bottom: checkpoints of one run to ten times the budget, read before its learning rate decays. –: no run of that length.

	head merge, % of control	
budget	`Qwen3-0.6B`	`Qwen3-4B`
schedule ends at this budget		
1×	7.4	34.0
4×	–	34.4
10×	18.6	53.3
checkpoints of the 10× run		
1×	5.4	16.2
2×	7.1	41.1
4×	23.4	68.5
7×	21.8	47.8

Table 12: The head merge with lookup data. Left: multi-key retrieval after the matched budget, in % of a control trained on the same data, with 12.4% of the tokens replaced by registry documents. Right: accuracy of the `Qwen3-4B` conditions on the standard harness's RULER tasks; Appendix Table 17 adds the 0.6B transfer rows.

	multi-key (% of control)			Qwen3-4B, accuracy (%)		
condition	0.6B	1.7B	4B	multi-key	random-string keys	variable tracking
control, web text	–	–	–	97.5	94.0	99.8
control, lookup data	–	–	–	100.0	95.5	99.5
head merge, web text	7.4	25.0	34.0	44.5	3.0	15.1
head merge, lookup data	**45.9**	**74.2**	**92.0**	**96.0**	**20.0**	**20.5**

C.2 MODELS, TRAINING AND THE FULL RESULTS TABLES

Most conditions start from `Qwen3-0.6B-Base` (Yang et al., 2025a) (28 layers, 16 query heads, 8 KV heads, head dimension 128), and Table 13 adds replications on `Qwen3-1.7B-Base`, `Falcon3-1B-Base` and `SmolLM2-1.7B`. Retrofit training is plain next-token continued pretraining on FineWeb-Edu (Penedo et al., 2024) at sequence length 4096, global batch 16, 2300 steps (1.5×10^8 tokens), AdamW (Loshchilov & Hutter, 2019), 50 warm-up steps, cosine decay over the last 20% of steps (the matched budget). The functional metric is RULER (Hsieh et al., 2024) `niah_multikey_2` at 4K context (MK2; our implementation, cross-checked against lm-evaluation-harness in Appendix H): one needle keyed by a random word among distractor needles, exact match, 100 prompts, seed 42, standard error 0.04–0.05 in the middle of the range, so single-cell gaps below 0.1 are noise. The headline 0.6B cells were re-measured at $n = 400$ (standard error at most 0.025; Tables 34 and 31) with the ordering unchanged. Held-out cross-entropy (CE) is measured on FineWeb-Edu documents disjoint from training, against the original dense model. Table 13 compares every condition to the dense-FT control of its base, trained through the identical recipe at 1e-4, not to the zero-shot model (details: Appendix F).

Table 13: Continued-pretraining conditions after the matched budget: MK2 at 4K ($n = 100$) and its % of the base's matched control at lr 1e-4 (italic row). A: information at init; B: alignment at init; ΔCE: held-out loss above the original model. *0.6B signature reused.

		at init			after training		
condition	lr	A: info.	B: align.	ΔCE (nats)	ΔCE (%)	MK2	% of ref.
Qwen3-0.6B							
matched control	*1e-4*	*intact*	*1.00*	*+0.00*	*+1.6*	*0.94*	*100.0*
matched control	7.3e-6	intact	1.00	+0.00	-0.1	0.83	88.3
matched control	3e-4	intact	1.00	+0.00	+9.1	0.23	24.5
activation swap	7.3e-6	intact	0.86	+11.52	+14.4	0.13	13.8
activation swap	1e-4	intact	0.86	+11.52	+3.8	0.76	80.9
activation swap	3e-4	intact	0.86	+11.52	+10.0	0.25	26.6
activation swap + closed-form refit	1e-4	intact	0.94	+0.71	+4.5	0.81	86.2
K/V noise, small	1e-4	intact	0.94	+0.19	+1.8	0.88	93.6
K/V noise, drift-matched	1e-4	intact	0.45	+10.83	+4.9	0.42	44.7
head merge	7.3e-6	lost (heads)	0.48	+10.43	+49.6	0.00	0.0
head merge	1e-4	lost (heads)	0.48	+10.43	+7.1	0.07	7.4
head merge	3e-4	lost (heads)	0.48	+10.43	+11.6	0.07	7.4
head merge	1e-3	lost (heads)	0.48	+10.43	+21.2	0.02	2.1
head merge + closed-form refit	1e-4	lost (heads)	0.75	+9.52	+9.2	0.05	5.3
head merge, retrieval-head layers	1e-4	lost (heads)	0.84	+0.95	+2.9	0.58	61.7
head merge, other layers	1e-4	lost (heads)	0.64	+6.48	+5.5	0.44	46.8
rotation probe	1e-4	intact	0.00	+15.29	+29.1	0.01	1.1
rotation probe	3e-4	intact	0.00	+15.29	+19.5	0.01	1.1
rotation probe, repaired on every head	1e-4	intact	1.00	+0.00	+1.6	0.88	93.6
Qwen3-1.7B							
matched control	*1e-4*	*intact*	*1.00*	*+0.00*	*+1.5*	*0.76*	*100.0*
activation swap	1e-4	intact	0.86*	+11.52	+3.8	0.78	102.6
head merge	1e-4	lost (heads)	0.51	+6.83	+6.6	0.19	25.0
Falcon3-1B							
matched control	*1e-4*	*intact*	*1.00*	*+0.00*	*+0.2*	*0.80*	*100.0*
activation swap	1e-4	intact	0.69	+10.07	+7.6	0.27	33.8
head merge 4→2	1e-4	lost (heads)	0.69	+3.49	+6.0	0.06	7.5
SmolLM2-1.7B							
matched control	*1e-4*	*intact*	*1.00*	*+0.00*	*+0.3*	*0.62*	*100.0*
activation swap	1e-4	intact	0.82	+8.40	+12.7	0.45	72.6
head merge 32→16	1e-4	lost (heads)	0.53	+11.71	+19.4	0.00	0.0

Every number in Table 13 comes from one version of our RULER prompt builder, on which the dense zero-shot model reads 0.89.

C.3 THE ACTIVATION SWAP AND THE HEAD MERGE ON QWEN3-0.6B, AND THE CHOICE OF LEARNING RATE

This subsection gives the details of the two conversions that anchor the `Qwen3-0.6B` results. The activation swap is the MLP conversion of §5.4, whose information stays intact, and the head merge

is the KV-cache conversion of §5.1, which deletes information. Both are trained with the matched-budget recipe of Appendix C.2. The last paragraph explains how that recipe's learning rate, 1e-4, was chosen, since the conditions of this paper are trained at it and compared with a control trained at the same rate.

Activation swap. Replacing SiLU by ReLU in every MLP keeps A trivially and sits at alignment 0.86 (MLP output). At init it is catastrophic (loss +11.52 nats above dense, zero-shot MK2 0.00). After the matched budget at learning rate 1e-4 it reaches 0.76 against the control's 0.94, that is 80.9% of control with CE +3.8% above dense. This is most of the way back, and the residual gap of 0.18, four standard errors on our harness, closes on the standard one (Appendix H).

Head merge. Averaging each adjacent pair of KV heads in weight space ($8 \to 4$, the mean-pooling conversion of GQA uptraining (Ainslie et al., 2023)) keeps one key and one value per token but loses the two heads' separate content. Its alignment is 0.48 and its initial loss +10.43 nats above dense, while the layer-0 identity probe is intact (0.81 versus 0.83). After the matched budget at the same rate the loss gap has closed to +7.1% but retrieval is 0.07, 7.4% of control. The residual loss gap is real (+0.19 nats at the matched budget, +0.08 at ten times it). CE recovers most of the way, and retrieval almost none.

Choosing the learning rate. A retrofit verdict is the best endpoint over a learning-rate grid, judged against the control trained at that rate, because a single rate can mislead in both directions. At the lowest rate of the grid (7.3e-6) both conditions look like walls, the swap at 0.13 and the merge at 0.00. At 1e-4 the swap reaches 0.76 and the merge 0.07. Higher rates do not lift the merge, which stays at 0.07 at 3e-4 with a worse loss (+0.31 nats) and falls to 0.02 at 1e-3. The swap falls to 0.25 at 3e-4, but the control trained at that rate falls with it, to 0.23 (on `Falcon3` from 0.80 to 0.63), so the drop belongs to the recipe and the swap keeps 108.7% of its same-rate control. The control reads 0.83 at 7.3e-6 and 0.94 at 1e-4. At $n = 400$ its difference from the zero-shot model is -0.07, so continued pretraining at 1e-4 costs some retrieval at 0.6B, as it does at 1.7B (-0.08 at $n = 100$).

C.4 The head merge under ten times the budget

Ten times the matched budget leaves the `Qwen3-0.6B` head merge at about a fifth of its control, flat from four times the budget on (Figure 5, Table 14). One run at 1e-4 over 1.5×10^9 tokens, evaluated at 1, 2, 4, 7 and 10 times the matched budget, reads 0.04/0.05/0.15/0.17/0.16, so ten times the tokens lift it only to about a fifth of the control (Table 28 lists the budgets of published conversions). The ratio needs a control trained through the same schedule, because the dense control is not flat along it. Trained through the identical 23,000 steps, the control reads 0.74, 0.70, 0.64, 0.78 and 0.86 at the same five points, falling while the learning rate is held at its peak and recovering over the cosine decay, so the drag of §C.3 comes from the schedule. Against that control the merge keeps 5.4, 7.1, 23.4, 21.8 and 18.6%, inside the bound set before training for four to ten times the budget (at most 30%, with at most 10 points of movement). The same schedule on `Qwen3-4B`, with its own control, ends at 53.3% of that control (52.7% at $n = 400$), above the bound of at most 50% set before training, where runs that end at the matched budget and at four times it keep 34.0 and 34.4% (Table 11). Before its learning rate decays, the larger model's checkpoints scatter between 0.0 and 70.8% of a control that stays at or above 84% (Figure 5). The checkpoint at eight times the budget reads 0.0% on a second read at $n = 400$ too, and again after a fresh conversion of the raw checkpoint, whose weights lie on the path between the checkpoints before and after it. Its answers are well-formed numbers with corrupted digits, so at that checkpoint the model still finds the needle but loses exact copying, and it regains it within the next 2,300 steps. Its held-out loss equals that of the checkpoint before it, which reads 47.8%, so within one run the loss does not register the retrieval it loses (§5.1). Two runs of the same recipe scatter as widely at the same step (Appendix C.9), so across budgets we compare the ends of schedules only.

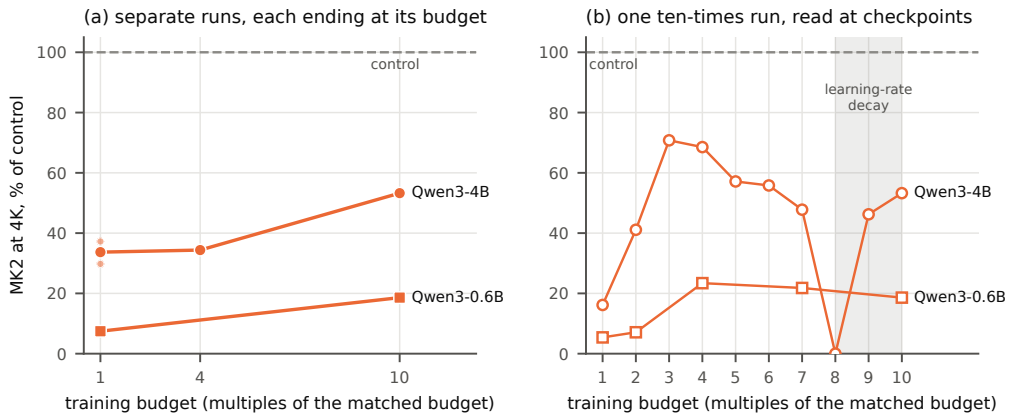

Figure 5: The head merge's MK2 at 4K against the training budget, as a percentage of a control trained through the same schedule, on `Qwen3-0.6B` (squares) and `Qwen3-4B` (circles). (a) Separate runs, each ending with its own learning-rate decay after one, four or ten times the matched budget. At one times the budget `Qwen3-4B` has three seeds, drawn small around their mean. (b) The checkpoints of the single ten-times run, whose learning rate stays at its peak until the decay over the last fifth (shaded), so its checkpoints before the decay are not endpoints. Its last checkpoint is the ten-times run of panel (a). Its eight-times checkpoint on `Qwen3-4B` reads zero on two reads and after a fresh conversion of the raw checkpoint, a transient loss of exact digit copying that the next checkpoint repairs (Appendix C.4).

Table 14: The head merge through one 23,000-step run at lr 1e-4 and the matched control through the same schedule (italic row), MK2 at 4K at intermediate checkpoints (§5.5); the last row is the merge in % of the control at the same checkpoint.

	budget (multiple of the matched budget)				
	1×	2×	4×	7×	10×
tokens (B)	0.15	0.30	0.60	1.06	1.51
matched control, MK2	*0.74*	*0.70*	*0.64*	*0.78*	*0.86*
head merge, MK2	0.04	0.05	0.15	0.17	0.16
head merge, % of control	5.4	7.1	23.4	21.8	18.6

C.5 Comparing the three compression dimensions, and what latent rank costs

In Table 15 pooled tokens keep the mean of every m consecutive keys and values. At $n = 400$ the head merge keeps 9.8% and pooled pairs of tokens 8.9% at half the cache, and at a quarter of the cache pooled blocks of four keep 0.6% against the rank-512 latent's 83.7%, so heads and tokens stay within a point of each other at both sample sizes.

A real conversion on the axis: the latent bottleneck. The conversion practitioners run, grouped-query attention into multi-head latent attention (Meng et al., 2025; Ji et al., 2025), loses channels through a latent bottleneck of rank r initialized from an SVD, where every lost-channel condition so far loses them by a head operation. We ran it as a rank series at $r = 512, 256, 128$ of a full rank of 1024, with predictions written from measurements taken before training. Each layer's stacked key and value projections are replaced by their best rank-r approximation and re-truncated after every optimizer step, so the trained model is a rank-r latent-attention model in function. The three ranks recover 89.1, 28.6 and 1.1% of control, in the order of their minimum value-reconstruction R^2 before training (0.84, 0.57, 0.34), so half rank is cheap while a quarter of the rank is a wall and an eighth loses retrieval entirely. The frozen rule's threshold of 0.9 on that R^2 was too strict, since the half-rank condition it called a wall lands 19 points above its line. The energy a rank-512 truncation removes is not what keyed retrieval reads, and the family's wall lies between R^2 0.57 and 0.84. Two unconstrained controls, the same SVD initializations trained without the rank constraint, show what the wall is made of. At rank 512 the unconstrained model lands at 90.2% against the constrained 89.1%, so the constraint costs nothing where the read fits. At rank 256 it lands at 54.1%

Table 15: The three compression dimensions of the KV cache on Qwen3 from 0.6B to 8B and on other families: MK2 at 4K in % of the reference named in each block, with the cache reduction against the full cache. †Absolute MK2 (0–1) of three training-free windows.

axis	condition	setting	cache reduction (×)	MK2 (% of reference)
Qwen3-0.6B, matched budget (reference: the matched control)				
feature	our latent	rank 512	4	89.1
		rank 256	8	28.6
		rank 128	16	1.1
head	head merge	8 → 4 KV heads	2	7.4
	head prune	8 → 4 KV heads	2	9.6
token	pooled tokens	m=2	2	7.4
		m=4	4	1.1
Qwen3-0.6B, no training (reference: the dense model)				
token	sliding window, no training	three windows	–	0.01/0.15/0.31†
Qwen3-1.7B, matched budget (reference: the matched control)				
feature	our latent	rank 1024	2	109.2
head	head merge	8 → 4 KV heads	2	25.0
token	pooled tokens	m=2	2	22.4
Qwen3-4B, matched budget (reference: the matched control)				
feature	our latent	rank 1024	2	97.9
		rank 512	4	83.0
head	head merge	8 → 4 KV heads	2	34.0
token	pooled tokens	m=2	2	58.5
Qwen3-8B, matched budget (reference: the matched control)				
feature	our latent	rank 1024	2	96.8
head	head merge	8 → 4 KV heads	2	58.9
token	pooled tokens	m=2	2	67.4
Falcon3-1B, matched budget (reference: the matched control)				
feature	our latent	rank 1024	2	102.5
head	head merge	4 → 2 KV heads	2	7.5
token	pooled tokens	m=2	2	6.2
SmolLM2-1.7B, matched budget (reference: the matched control)				
head	head merge	32 → 16 KV heads	2	0.0
token	pooled tokens	m=2	2	0.0
Llama-2-7B, matched budget (reference: the matched control)				
head	head merge	32 → 16 KV heads	2	5.8
token	pooled tokens	m=2	2	32.7

against 28.6%, so removing the constraint doubles the recovery and still leaves about half of the retrieval missing at the matched budget. A channel is lost when the architecture can no longer carry it. Information that the initialization discarded while the capacity to carry it remains has a price instead, 9.8 points at half rank and 45.9 at a quarter, and Definition 2 is stated in these terms.

C.6 The retrieval circuit, the choice of merged heads and the data in detail

The retrieval circuit is sparse: 8.5% of query heads carry 97.8% of the retrieval mass at 0.6B. Its head count grows with the model (38, 42, 78, 82 and 119 heads from 0.6B to 14B). The retrieval-head score of §6.2, which finds those heads, is causally valid from 4B up: disabling the 78 heads it names on the 4B base costs 96.0 points of retrieval against 0.0 for the same number of randomly chosen heads, and 98.0 against 16.0 at 8B. At 0.6B it does not separate (89.0 against 63.0): at that size zeroing any 38 heads wrecks retrieval at 4K.

After a head merge the retrieval mass moves onto new heads, but the recovered behaviour does not run on them. The 4B and 8B endpoints end at 109.1% and 110.0% of their bases' retrieval mass, with 34.7% and 34.9% of it on 37 and 35 heads that did no retrieval before. The 0.6B endpoint recruits 2 and reaches 52.2%. The matched controls keep their circuits (1 and 7 new heads, Jaccard 0.90 and 0.87 against their bases). We predicted before training that the recruited heads carry the recovered behaviour, and the refutation condition was met on all four endpoints. Disabling the recruited heads costs as much as disabling the same number of random heads (1.0 against 0.0 and 17.0 against 17.0 at the 0.6B and 4B endpoints of an untied merge, which stores each merged head in both slots of its pair and trains the two copies apart, and 2.0 against 20.0 and 5.0 against -3.0 at the 4B and 8B endpoints of the merge itself). Disabling the *retained* heads, those that already carried retrieval in

the base, costs two to five times as much as their own matched random set (73.0 against 16.0, 32.0 against 16.0, 56.0 against 25.0).

Since the recovered retrieval depends no more on the heads the merge recruits than on random heads, what decides recovery is how much of the original retrieval circuit survives the surgery, and this is also why choosing which heads to merge works where it does. On `Qwen3-4B`, merging only the pairs without retrieval heads leaves the circuit intact and keeps 96.8% of control while merging 103 of the 144 candidate pairs, which takes the model's 288 KV groups to 185, a 35.8% reduction. The merged count does not order the conditions. The 41 flagged pairs are a smaller cut (14.2%) at a larger loss (77.7%), and all 144 pairs (50.0%) leave 34.0%. Holding the compression fixed and varying only which pairs are merged, the same number of pairs merged at random keeps 53.2% and 50.0% on two draws (Table 16). The spared groups are not spread evenly across the model, so the converted network carries between 4 and 8 KV groups per layer rather than one head count throughout. At 0.6B, where the retrieval-head score has no causal grip on the base to begin with, the same choice of pairs buys 24.4%, and the count-matched pair puts the choice of pairs inside noise: merging the 24 flagged pairs keeps 64.7% of control against 75.3% for as many random pairs. The size trend of the merge endpoint therefore reflects how much of a larger, more redundant retrieval circuit a fixed merge ratio leaves standing, rather than spare capacity to re-route through.

Table 16: Where to merge (Figure 2): MK2 after the matched budget in % of the matched control, when all pairs of KV heads are merged, only the pairs without retrieval heads or only the pairs with them, or as many pairs at random.

pairs merged	pairs	cache saved (%)	MK2 (% of control)
Qwen3-4B			
all	144	50.0	34.0
sparing the retrieval heads	103	35.8	96.8
random, first draw	103	35.8	53.2
random, second draw	103	35.8	50.0
retrieval heads only	41	14.2	77.7
Llama-2-7B			
all	512	50	5.8
sparing the retrieval heads	479	46.8	34.6
random	479	46.8	13.5
Qwen3-0.6B			
all	–	50	7.4
sparing the retrieval heads	–	–	24.4
random	24	–	75.3
retrieval heads only	24	–	64.7

The variable behind the merge's rise with size (§C.9) is not parameter count. Read across model families, the merge's fraction tracks the size of the base's *retrieval circuit* (the heads the zero-training retrieval-head score puts at or above 0.1) and not the number of parameters. `Llama-2-7B` is the case that separates the two: it carries 35 retrieval heads of 1024 (mass 14.3), a *smaller* circuit than `Qwen3-0.6B`'s 38 heads and 17.7 mass, and at eleven times the parameters it recovers 5.8% of its matched control where `Qwen3-0.6B` recovers 7.4%. Ordered by circuit rather than by size, the eight scored bases read 17/29/35/38/42/78/82/119 heads against 7.5/0.0/5.8/7.4/25.0/34.0/58.9/81.8% of control. The refutation condition written for this reading was not met: no base carries a `Qwen3-8B`-sized circuit and recovers like a 0.6B.

On the families scored before training, the circuit size did not predict the merge. The prediction failed on `Llama-3.1-8B` and, in a re-run at a lower learning rate, on `Mistral-7B`, and could not be scored on `Gemma-2-2B`. `Mistral-7B` carries 62 retrieval heads (zero-shot MK2 85%) and `Llama-3.1-8B` 49 (zero-shot 99%), and from the head count alone each was predicted to keep 15–45% of its matched control after the merge. On `Llama-3.1-8B` the merge reads 5% against a control of 58%, 8.6% of control and below the band, which places the base beside `Llama-2-7B` (5.8%) rather than beside the `Qwen3` sizes with circuits of similar size (25.0% at 42 heads). The recipe itself costs this base retrieval, since the control falls from 99% to 58% over the budget, so the ratio is taken against a degraded control. The gate set before training was the base's zero-shot score alone, so this pair stands as the outcome. Its refutation condition is defined only for circuits of at most 40 or at least 70 heads and does not apply at 49 heads, so we report a failed prediction rather than a refutation. On `Mistral-7B` the same recipe (learning rate 10^{-4}) collapsed the control to 1%,

which leaves the ratio undefined and voids the pair. Re-run at 2×10^{-5} with a new gate that requires the control to keep MK2 at or above 60%, the merge keeps 7.0% of control on `Mistral-7B` (control 71%) and 9.3% on `Llama-3.1-8B` (control 97%). The lower-rate `Llama-3.1-8B` pair checks sensitivity to the recipe and does not replace the first outcome. On `Gemma-2-2B`, gated the same way at 2×10^{-5}, the recipe lowers the control from 76% to 48%, below the minimum set for it, so the comparison with its retrieval heads is uninterpretable there.

Where the merge's cost lives. The merge's cost is concentrated neither in the retrieval-head layers nor in one half of the head. Merging only the 9 retrieval-head layers (pairs kept bit-identical throughout training, Appendix F) leaves retrieval at 61.7% of control (0.58), merging only the other 19 layers leaves 46.8% (0.44), and merging all 28 leaves 7.4%. These drops of 38.3%, 53.2% and 92.6% for 9, 19 and 28 merged layers are not localised in the retrieval-head layers: the drops of the two layer sets add up to about the drop of merging all 28, so at 0.6B exempting the retrieval-head layers is not a cheap route to head merging. Merging only the keys, which loses routing information and keeps the values per head, recovers 43.6% of control, and merging only the values recovers 28.7%, again with the pairs kept bit-identical. On the standard harness the two read 27.9% and 27.9%. Neither half accounts for the tier alone and the two compound, so the routing reading predicted before training (keys at most 25%, values at least 40%) failed. Which half costs more is not settled across harnesses (Appendix H).

What the data asks for. Lookup data changes what the head merge re-grows at a fixed budget, far more than ten times the web text does. Replacing part of the continued-pretraining stream with synthetic registry lookups, a keyed-retrieval format disjoint from the metric's, lifts the 0.6B merge from 0.07 absolute on web text to 0.59 with 5.3% of the tokens replaced, 0.45 at 12.4% and 0.42 at 22.2%. At the highest dose the matched control keeps retrieval at 0.98 but loses common-word extraction, from 0.506 to 0.015, a capability the synthetic data does not exercise and the target metric does not show. At 12.4% of tokens the merge reaches 45.9, 74.2 and 92.0% of the mixed control at 0.6B, 1.7B and 4B, against 7.4, 25.0 and 34.0% on web text alone, while the same data leaves the rotation of §7 where it was (1.1% of control on web text, 1.0% with the lookups). On web text alone the 4B merge keeps 34.0, 34.4 and 53.3% of its control at the ends of schedules of one, four and ten times the budget (Table 11). On random-string keys the lookup data adds nothing at 0.6B, and of the four other RULER tasks only frequent-word extraction gains (20.8% to 77.2% of the control). Table 17 gives the transfer to the other keyed reads. On `Qwen3-0.6B` the control itself is weak on random-string keys, below the level set before training for a test, so that column is descriptive there. On `Qwen3-4B` it passes. Variable tracking on `Qwen3-0.6B` uses our own builder, whose answer budget of 30 tokens truncates the answers of models that prefix each name. The drop of the control trained with lookup data is that truncation, since on the standard harness it reads 0.664 against 0.689 for the web-text control.

Table 17: Whether the lookup-data gain of §6.3 reaches other keyed reads: accuracy (%) at 4K on RULER's multi-key task, the same with random-string keys and values, and variable tracking.

	multi-key		
condition	standard	random keys	variable tracking
Qwen3-0.6B, our builder, accuracy (%)			
matched control	94.0	26.0	90.0
with lookup data	98.0	29.0	42.0
head merge	7.0	0.0	16.0
with lookup data	45.0	1.0	1.0
Qwen3-4B, standard harness, accuracy (%)			
matched control	97.5	94.0	99.8
with lookup data	100.0	95.5	99.5
head merge	44.5	3.0	15.1
with lookup data	96.0	20.0	20.5

C.7 THE MISALIGNED CHANNEL: CONTROLS AND THE MECHANISM

Every conversion above deletes information. A conversion can also lose a capability without deleting anything, by leaving information in place in a basis the frozen layers downstream no longer read. To

measure that case we need a surgery whose exact repair is known in closed form, so that the question of whether training finds the repair can be asked at all, and the value-head rotation is that surgery: each value head is multiplied by a random orthogonal matrix, and folding its inverse into the output projection that reads it restores the model bit for bit. We do not propose it as a conversion. Table 18 compares the training targets and optimizers of §7 with the closed-form refit below, all from the same rotated initialization. It stands for the part of any conversion that re-bases a module without re-aligning its readers, which is the part the successful conversions handle with a local alignment step (Jin et al., 2025; Zhang et al., 2025; Bick et al., 2024).

Table 18: Repairing the misaligned probe on Qwen3-0.6B from the same rotated initialization. Distance covered: the share of the way from that initialization to the exact repair of Proposition 2, measured on the composed map $W_O W_V$ of every rotated head, so that a repair written into the value projections counts too.

training target	optimizer	held-out CE (nats)	MK2 (% of control)	distance covered (%)
References				
original model	–	2.67	–	–
rotated model, before training	–	18.19	–	–
matched control	–	2.70	–	–
Rotated model, trained on the same data and steps				
next-token cross-entropy	AdamW	3.47	0.0	2.3
next-token cross-entropy	Muon	3.64	0.0	2.1
KL to the original model's outputs	AdamW	3.37	1.1	2.8
original residual stream, every layer	AdamW	3.81	0.0	5.0
original attention output, per module	AdamW	2.76	**75.3**	66.6
closed-form refit of W_O, no training	–	–	93.6	98.9

The head merge refuted the prediction that per-token surgery is cheap. It touches weights, loses information and moves its consumers' inputs, and three controls at 1e-4 and the matched budget separate these three candidate causes (Table 13, Figure 6).

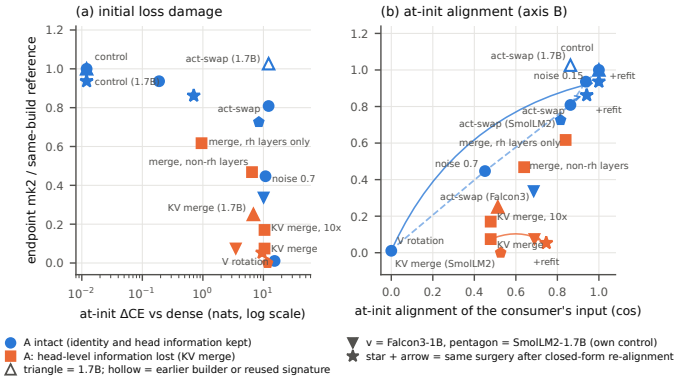

Figure 6: The two axes; y is each condition's endpoint as a fraction of its reference. (a) Against the zero-shot loss gap the conditions are unordered. (b) Against at-init alignment the information-intact conditions (blue) lie on one monotone curve from the rotation to the control, and the head-merge conditions (orange) sit below it. Stars with arrows are the same surgeries after closed-form re-alignment (§C.7).

An exactly invertible rotation (axis B alone). Each value head's projection rows are multiplied by an independent random orthogonal matrix. Weights are touched and no information is lost: composing the output projection with the transposes restores the function exactly (the refit below does so on the real model). The consumer's input is nevertheless orthogonal to what it was trained on (alignment 0.00, loss +15.29 nats above dense). After the matched budget the loss has recovered to +0.77 nats above dense while the attention outputs stay nearly orthogonal to the original model's (cosine 0.17) and retrieval is 0.01 (1.1% of control; 0.01 at 3e-4). Gradient descent did not find the linear undo.

Drift-matched noise (axis A at fixed B). Isotropic Gaussian noise is added to every key and value projection with a scale calibrated so that the at-init alignment and loss match the merge (0.45 and +10.83 nats against 0.48 and +10.43). It lands at 0.42, 44.7% of control and 6 times the merge, with a smaller residual loss (+0.13 against +0.19). At the same aggregate damage, losing the two heads' separate content is what the merge pays for.

The high end of the axis. Noise at 0.15 times the weight scale keeps the consumers' inputs aligned (0.94, loss +0.19 nats above dense) and recovers to 0.88, 93.6% of control, as predicted before training (at least three quarters).

Raising alignment in closed form. On a fixed surgery, raising alignment moves the outcome when the information is intact and does nothing when head-level information is lost, so the two axes separate. Every point on the alignment axis so far is a different surgery, so to move alignment alone we refit, for a fixed surgery, the one projection through which its consumers read the modified module (W^O for the attention surgeries, the down-projection for the swap). The refit is a ridge regression against the original module's teacher-forced output on eight held-out sequences, one regression per layer and no gradient step. What the module emits is a linear image of the same intermediate, so its information is unchanged (Appendix F). The rotation's refit is its exact inverse and lands at the control (alignment 0.00 $\rightarrow$ 1.00, MK2 0.01 $\rightarrow$ 0.88, 93.6% of control). The swap's refit raises its alignment from 0.86 to 0.94 and its endpoint from 80.9% to 86.2% of control, or from 89.3% to 94.4% at $n = 400$, against the $100\, B_{\text{new}} \pm 15$ predicted before training. That shift is within noise for a swap that the standard harness already puts at parity (Appendix H). The merge's refit raises its alignment from 0.48 to 0.75, into the band where the swap and the noise control recover, and leaves retrieval at 0.05 (5.3% of control, from 7.4%; predicted at most 20%). A linear re-read of merged values therefore does not buy down the head-merge tier.

Aligning before merging. Align-heads (Jin et al., 2025) rotates the two heads of a pair into a common basis before it merges them. We applied the orthogonal alignment of each value pair, and on `Llama-2-7B` also of each key pair within its rotary planes, before the 2:1 merge, so that the aligned but unmerged model equals the base, and trained the merge through the matched budget. It keeps 8.5, 39.4 and 9.6% of control on `Qwen3-0.6B`, `Qwen3-4B` and `Llama-2-7B`, 1.1, 5.4 and 3.8 points above the plain merge of the same base. Aligning the pair raises what the two heads share, and the merge still deletes what they do not.

Two practitioner classes. Two further modifications the line was never fitted on were predicted the same way before training: symmetric per-channel int4 quantization of every decoder projection, dequantized and then trained (alignment 0.93, outcome 74.5% of control; predicted at least 85% and within 10 points of the line), and 2:4 magnitude sparsity of every MLP projection with the zero pattern enforced through training by gradient masks, so that the endpoint is a semi-structured-sparse MLP (MLP-output alignment 0.83, outcome 75.5%; predicted within 15 points of the line and at least 50%). The sparsified MLP is priced by its alignment as the activation swap is. The quantized model recovers less than its alignment says, by about the same margin as the RoPE conditions, and it is the one modification here that perturbs every projection of every module at once, which the module-level reading averages away. It is therefore a second class on which the line is an upper bound rather than an estimate.

Why an unreadable channel stays unreadable. The inverse of a rotated head sits in parameter space as one orthogonal matrix per head, folded into the output projection, and continued pretraining does not find it. On the published rotation condition, across all 448 rotated heads, the trained output-projection columns keep their norm (ratio 1.00) and their direction (cosine 0.99 to the initialization, 0.01 to the inverse). Of the heads, 0.0% are shut and 0.0% learn the inverse, with 6.0% merely turning toward it. The producer does not move either: the rotated value projections sit at cosine 0.99 to their own initialization and drift toward their original rows by 0.01. Neither end of a channel read at cosine zero moves, and what the model recovers it recovers through the other channels. This is the mechanical content of the two degenerate endpoints and of the initialization result: the gradient on a channel carrying no signal points no closer to the repair than a random direction, so the repair is free to write in closed form and was not learned at these budgets.

C.8 MERGING WITHOUT TRAINING, UP TO 70B PARAMETERS

Without training, multi-key retrieval falls first, or at the same merged fraction as MMLU, at every size. Figure 7 follows Gromov et al. (2025) in plotting the unhealed model: the head merge is applied to a nested random fraction of the KV pairs and the model is evaluated with no training at all (MMLU 5-shot and MK2 at 4K with $n = 100$; `Llama-2-70B` and `Qwen3-32B` with 8-bit weights). On `Llama-2-7B`, merging a quarter of the pairs takes multi-key retrieval from 64% to 0% and MMLU from 45.5% to 28.0%, near the 25% of chance, and `Llama-2-13B` and `Qwen3-0.6B` have the same shape. From `Qwen3-4B` up, MMLU still stands at a quarter of the pairs (61.7, 66.6 and 73.5% at 4B, 8B and 14B, against 72.8, 76.9 and 80.7% unmerged) while retrieval has fallen to 72, 57 and 66%. Retrieval halves at a merged fraction of 0.50 on all three, and MMLU at 0.50 on `Qwen3-4B` and at 0.75 on `Qwen3-8B` and `Qwen3-14B`.

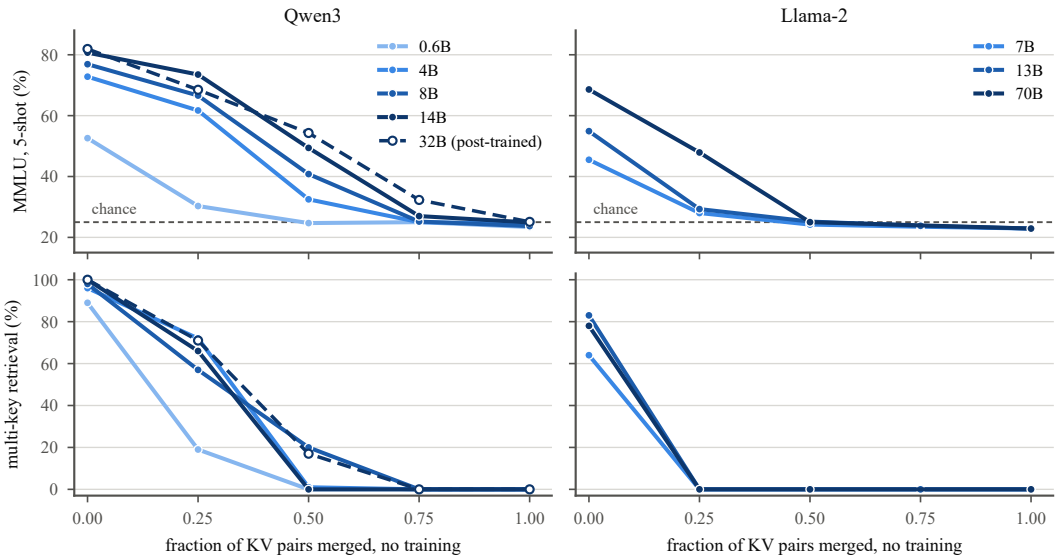

Figure 7: The head merge with no training, in the manner of the unhealed curves of Gromov et al. (2025): MMLU (top, with the random-guess level) and multi-key retrieval (bottom) against the fraction of KV pairs merged, one line per size. Before training, merging part of the pairs costs both, and multi-key retrieval falls first. After continued pretraining MMLU recovers more of the control than multi-key retrieval, by a margin that shrinks with size (Table 23). The dashed line is the post-trained `Qwen3-32B`, the only 32B release, compared with itself unmerged.

The two largest models follow the same order. `Llama-2-70B`, first scored at merged fractions of 0, 0.5 and 1, falls at half of the pairs from 78% to 0% on multi-key retrieval and from 68.6% to 25.0% on MMLU, the level of chance, so on that grid the two halve together, as on `Qwen3-4B`. On finer grids, chosen after these ties, multi-key retrieval halves first on both models: at a merged fraction of 0.4375 on `Qwen3-4B`, one step before MMLU at 0.5, and at 0.25 on `Llama-2-70B`, where a quarter of the pairs takes retrieval from 78% to 0% while MMLU stays at 47.9%, far above chance. Only the post-trained release of `Qwen3-32B` is public, so it is compared with itself unmerged. Its multi-key retrieval halves at a merged fraction of 0.50 and its MMLU at 0.75, and with half of the pairs merged MMLU reads 54.3% against 81.9% unmerged while retrieval reads 17%. With every pair merged no size keeps any multi-key retrieval. The unhealed curves therefore already order the two tasks, and after continued pretraining (§5.1) MMLU recovers more of the control than multi-key retrieval at every size, by a margin that shrinks with size. The 8-bit weights of `Llama-2-70B` were checked on `Llama-2-13B` before any merge, where they move multi-key retrieval by +0 and MMLU by +0.1 points.

C.9 ROBUSTNESS ACROSS SEEDS, MODEL SIZES, MODEL FAMILIES AND TASKS

This subsection tests whether the results of Section 5 hold beyond one run of one model on one task. It repeats the key conditions with other seeds and data windows, at four `Qwen3` sizes, on two other

model families, and on retrieval tasks other than MK2. The 4B head merge at the matched budget is reproducible across seeds and data windows: two seed replicates, each on its own window of the training stream, keep 24.8% and 23.2% of the control at $n = 400$ against the headline run's 31.5%, a range of 8.3 points. Checkpoints taken in the middle of a longer schedule, at the peak learning rate, vary far more: two runs of the identical recipe on the same data order, whose training losses agree to three decimals, end 70.5% of their update apart in weight space and read 38.2% and 66.2% of their controls at the same step, three times the matched budget. The training loss pins these models down while leaving keyed retrieval free to scatter, and only the annealed endpoints are compared across budgets.

The swap stays above the merge at every size of `Qwen3` from 0.6B to 8B, while the height of the wall rises with the size. On `Qwen3-1.7B-Base` at the matched budget and 1e-4 the swap reaches 0.78 against a control of 0.76 and the merge 0.19 (25.0% of control). The same three conditions were run through the identical recipe on `Qwen3-4B-Base` and `Qwen3-8B-Base` (36 layers, 32 query and 8 KV heads; the 8B run gated on a 30-step feasibility check, which it passed), with the measurements read and the predictions written before training. We predicted before training the merge at most at 20% of its control and the swap at $100\,B \pm 20$ from its MLP-output alignment, and a merge above 35% would have made the head-merge tier a small-model artefact. At 4B the control reaches 0.94, the merge 34.0% of it and the swap 100.0% at alignment 0.77. At 8B the control reaches 0.95, the merge 58.9% and the swap 98.9% at 0.79 (Table 19). The merge's fraction rises from 7.4% at 0.6B through 25.0% at 1.7B and 34.0% at 4B to 58.9% at 8B, so the predicted bound of 20% failed at 4B and 8B, and the 35% refutation condition, under which the tier is a small-model artefact, was missed by one point at 4B and met at 8B. The swap sits at parity at 4B, 22 points above its line, outside the predicted ± 20 but inside the ± 25 band whose breach would have refuted the line's transfer across scale, so the line under-predicts the cheap tier as the base grows.

Table 19: The headline contrast on four Qwen3-Base sizes, same recipe and budget: MK2 at 4K ($n = 100$) and its % of the matched control (italic row); B and ΔCE at init, measured before training.

		at init		after training	
base	condition	B: align.	ΔCE (nats)	MK2	% of control
Qwen3-0.6B	*matched control*	*1.00*	*+0.00*	*0.94*	*100.0*
	head merge	0.48	+10.43	0.07	7.4
	activation swap	0.86	+11.52	0.76	80.9
Qwen3-1.7B	*matched control*	*1.00*	*+0.00*	*0.76*	*100.0*
	head merge	0.51	+6.83	0.19	25.0
	activation swap	0.86	+11.52	0.78	102.6
Qwen3-4B	*matched control*	*1.00*	*+0.00*	*0.94*	*100.0*
	head merge	0.54	+5.64	0.32	34.0
	activation swap	0.77	+9.87	0.94	100.0
Qwen3-8B	*matched control*	*1.00*	*+0.00*	*0.95*	*100.0*
	head merge	0.55	+3.58	0.56	58.9
	activation swap	0.79	+9.54	0.94	98.9

Seed variance does not explain the wall or the shortfall of the initialization series, but it is large on the priced side. Training variance was measured on the cells that carry the argument, each retrained twice on disjoint slices of the training stream. The merge reads 0.02 / 0.03 / 0.07 and the 85% level 0.41 / 0.42 / 0.46 (ranges 0.05 and 0.05), and the worst swap run still clears the best merge run by more than 0.4. The swap reads 0.55 / 0.75 / 0.76 (58.4–80.6% of control, range 0.21) and the two uniform rotations 0.62 / 0.67 / 0.85 and 0.00 / 0.08 / 0.37 (§B.2), so the priced side of the line carries a run-to-run spread of ten to forty points at $n = 100$, and the ± 15 tolerance of the predictions is about one such spread. Every single-run fraction on the priced side should be read with that bar, and Table 13 reports the first run.

The slices are not exchangeable, and the dense control moves with them. Retrained on the second and third slices, the control reads 0.65 and 0.76 against 0.94 on the first, which refutes our prediction that it would stay flat within 0.08, so part of the priced side's spread is the recipe's own drag on the data. The drift-matched noise condition at alignment 0.45 reads 0.35 and 0.16 against 0.42, where a run below 0.2 was the predicted sign that the cliff edge is unstable for every map and not only for the rotation. Re-based on its own slice's control, the second-slice merge sits at 3.1%, the 85% level at 64.6%, the swap at 115.4% and the 30° rotation at 103.1% (third slice: 3.9, 60.5, 72.4 and

81.6%). The wall conditions stay flat in absolute terms across slices while the control and the priced conditions move with the data, and the ordering of the decisive pair holds on every slice. The paper therefore states each fraction against the control trained on the same data as the condition.

On a second family, `Falcon3-1B-Base` (zero-shot MK2 0.79), the same three conditions ran at 1e-4 and the matched budget. The control reaches 0.80, and the 2:1 head merge lands at 0.06 (7.5% of control), so the cost tier replicates. The activation swap does not recover, 0.27 (33.8% against 80.9% on `Qwen3-0.6B`), and its loss gap is the larger one, so cross-entropy again orders the conditions the wrong way. A learning-rate grid does not rescue it: 0.10 against a same-rate control of 0.79 at 3e-5 and 0.31 against 0.63 at 3e-4, a best-of-grid 49.2% of control, which is where its residual-stream alignment puts it.

On a third family, `SmolLM2-1.7B` (32 query and 32 KV heads), the measurements before training ran first and the predictions were written from them by a rule fixed before training. The swap sits at alignment 0.82 and was predicted at $100\,B \pm 20$, or $82 \pm 20\%$ of control, and the $32 \to 16$ head merge (the case of Ainslie et al. (2023)) at most at 20%. The control reached 0.62, the swap 0.45 (72.6%, inside the band) and the merge 0.00 (0.0%). Across the three families the swap keeps 80.9, 72.6 and 33.8% of control at module-output alignments 0.86, 0.82 and 0.69. On `SmolLM2` alone the loss orders the two conditions correctly (swap +0.28, merge +0.43 nats).

The decisive pair of §B.2 does not separate on `SmolLM2`, because this base prices the same per-head misalignment far more steeply. Chosen from its measurements before training, the pair is a uniform 30-degree rotation of every value head (per-head cosine 0.87, no head lost, module alignment 0.91) and a rotation with 85% of the heads compensated exactly (14.1% of heads read at cosine zero, module alignment 0.91). They reach 0.27 and 0.26 against the 0.62 control, 43.5 and 41.9%, where the line calibrated on `Qwen3` predicted 91.2% for the uniform condition and the same pair on `Qwen3-0.6B` gives 65.8–90.2% against 43.5–48.8% across replicates. Endpoint losses within 0.08 nats of the control show that the runs are sound. A uniform-rotation series on `SmolLM2`, set before training to locate its cliff, recovers 91.9, 71.0 and 43.5% of control at per-head cosine 0.985, 0.94 and 0.87 (0.57, 0.44 and 0.27), monotone with no head lost, where the `Qwen3-0.6B` series recovers 90.4, 75.5, 39.4 and 5.3% at 0.87, 0.71, 0.45 and 0.21 (§B.2). The shape is the same, monotone and convex in the alignment, but the slope is about four times steeper in $1 - \cos$, with the 50% crossing between 0.94 and 0.87 on `SmolLM2` against between 0.71 and 0.45 on `Qwen3`. At cosine 0.87 the uniform condition on `SmolLM2` is therefore already at the level of the condition with lost heads. The criterion accounts for the existence of the wall and the shape of the price, the slope of the price belongs to the base, and the separation of the two classes at equal mean is demonstrated on `Qwen3`.

The class verdicts are statements about keyed retrieval, the behaviour the criterion is about, and our own implementation of variable tracking is not a clean probe of the channel. On that implementation at 4K (the standard harness's version is in Table 20, where the classes separate), both per-token conditions collapse at 0.6B (swap 0.19, merge 0.16, control 0.90) and recover to about three quarters of control at 1.7B (0.68, 0.66, control 0.92), so the swap's near-parity is specific to keyed retrieval and to scale (Appendix I).

Table 20: The RULER suite (lm-evaluation-harness, 4K, 200 prompts) on Qwen3-0.6B after the matched budget, in % of the matched control, whose accuracy is the italic first row. MV, MQ: multi-value, multi-query; VT: variable tracking; CWE, FWE: common- and frequent-word extraction; QA: SQuAD (Rajpurkar et al., 2016). Rotation probe: Proposition 2.

	single needle			multi-key								
condition	1	2	3	1	2	3	MV	MQ	VT	CWE	FWE	QA
matched control (accuracy)	*1.00*	*1.00*	*1.00*	*0.92*	*0.90*	*0.26*	*0.80*	*0.79*	*0.69*	*0.50*	*0.58*	*0.49*
activation swap	100.0	100.0	92.5	103.3	98.3	100.0	93.9	108.1	90.4	62.0	85.0	105.8
head merge	99.0	99.5	83.0	73.9	5.6	1.9	62.2	69.9	13.0	28.3	61.4	51.8
head prune	98.0	99.5	81.5	57.6	11.7	5.8	67.0	55.3	8.2	24.4	44.4	60.5
rotation probe, repaired on every head	100.0	100.0	100.0	98.4	95.5	94.2	101.1	103.5	97.8	94.8	104.9	103.8
rotation probe, repaired on 85% of heads	100.0	99.5	96.5	90.8	48.0	25.0	65.3	72.2	79.3	28.9	48.4	95.4
heads unrepaired at 85% deleted	100.0	100.0	98.5	87.0	50.3	23.1	110.8	112.4	47.2	24.7	91.6	83.9

To test whether the class verdicts are properties of the retrofit or of one task at one length, we scored the conditions never scored on a second task: the partial levels of the initialization series, the RoPE condition, the three refits, the best KV recipe and the small-noise condition on variable tracking and common-word extraction at 4K, and the six core conditions on MK2 at 8K. The predictions written before training were the lost-channel conditions below 20% of the control's variable-tracking score, the order of the initialization series preserved, and the 8K rank order within one adjacent swap of the 4K one. The wall holds on variable tracking, where the merge with its refit, the prune and the half-refit merge score 0.00, 0.02 and 0.15 against the control's 0.90, and the fully rotated condition at three times the budget scores 0.00. The ordering does not hold. The initialization series inverts on variable tracking (the 85% level 0.15, the 50% level 0.35), and the dense recipe itself moves the task, since the three-times control reads 0.31 on it while keeping 0.80 on MK2. The swap's collapse is the swap's own rather than the intact class's, since the exact inverse trained afterwards keeps 0.84 (93.1%) and the small-noise condition 0.68 (75.1%), while the swap with its refit keeps 0.03. Common-word extraction at 4K sits at the floor for every condition, the controls included, and is uninformative. At 8K the five weight conditions keep their 4K order (control 0.73, swap 0.50, the 85% level 0.17, merge 0.02, rotation 0.00). The RoPE condition at 10^4 falls to 0.00 because that base cannot address 8K positions, a position-contract effect rather than a channel one.

On the standard harness, the RULER suite from a single needle to many similar keys shows which reads need the channel (Table 20; the swap's parity is read on the same harness, as in Appendix H). Every surgery keeps single-needle retrieval (83.0–99.5% of control for the merge), and the lost-channel conditions lose retrieval as the number of similar keys grows. On multi-key 1, 2 and 3 (§5.2) the merge keeps 73.9, 5.6 and 1.9% of control, the prune 57.6, 11.7 and 5.8%, the 85% level 90.8, 48.0 and 25.0% and its dropped twin 87.0, 50.3 and 23.1%, the last two with one profile, head for head. The exact inverse (98.4, 95.5, 94.2%) and the activation swap (103.3, 98.3, 100.0%) stay at parity. What head-level content and an unread head carry is the discrimination of a key from many similar keys, and on the standard harness the wall reaches every read that must find one. On variable tracking the merge keeps 13.0%, the prune 8.2% and the swap 90.4% of a 0.69 control. On the QA task the merge keeps about half (51.8%) and the swap 105.8%. On common-word extraction every lost-channel condition sits near a quarter of control (merge 28.3%, 85% level 28.9%, prune 24.4%) and the swap itself loses a third (62.0%), the one task on which it is not at parity. Frequent-word extraction is the gentlest (merge 61.4%). Our own variable-tracking implementation (Appendix I), on which the initialization series inverted and the swap collapsed, has a control of 0.90 against 0.69 here and is not the standard task. On the standard one the classes separate as they do on keyed retrieval.

C.10 Long context and LongBench

On `Qwen3` the head merge loses keyed retrieval as the context grows, at every size, and the half-cache latent of `Qwen3-4B` keeps it. Table 21 repeats RULER's needle tasks at 8K and 16K tokens and adds the average of the 16 English tasks of LongBench (Bai et al., 2024), the long-context benchmark some conversion papers report, with 40 examples per task and inputs left-truncated to 16K tokens, on the checkpoints of the matched budget and on their bases. On multi-key 2 the merge keeps 3.0, 8.2, 20.6 and 66.7% of its control at 8K tokens from 0.6B to 14B, and 0.0, 0.0, 0.0 and 4.1% at 16K, where the 8B and 14B merges begin to lose single needles too (74 and 73% against their controls' 99 and 100). Sparing the retrieval heads on `Qwen3-4B` holds to 8K (59.2%) and not to 16K (0.0%). The rank-1024 latent of `Qwen3-4B` keeps 98.0 and 104.3% at 8K and 16K, and the rank-512 latent of `Qwen3-0.6B`, at a quarter of the cache, keeps 59.7 and 18.3%. On LongBench the merges keep 52.4, 65.6, 54.1 and 57.5% of their controls' averages from 0.6B to 14B, and the 4B latent 103.6%. Our continued pretraining at 4K tokens costs the controls themselves long-context ability, since the untrained 8B base reads 44.3 on LongBench and its control 30.0. Every percentage here is therefore of a control trained at 4K. On `Llama-3.1-8B` both the head merge and the MLA form of §6.4 lose keyed retrieval at long context, keeping 6.0 and 4.0% of the control's multi-key 2 at 8K and 0.0 and 1.0% at 16K, and 61.6 and 56.6% of the control's LongBench average.

HELMET. On HELMET (Yen et al., 2025), a long-context benchmark we did not build, the `Qwen3-4B` head merge keeps 28.3% and 5.5% of its control's recall score at 8K and 16K tokens, and 66.5% and 49.0% of its retrieval-augmented question answering, while the half-cache latent keeps 89.2, 92.6, 99.5 and 99.9% (Table 22). The predictions written before the run hold: the merge keeps at most 30% of the recall score at both lengths, keeps a larger share of the question-answering

Table 21: Long context after the matched budget, accuracy (%): RULER's single-needle tasks (the mean of the three), multi-key 2 (MK2) and multi-key 3 (MK3) at 8K and 16K tokens, and the LongBench average (16 English tasks, 40 examples each, inputs left-truncated to 16K tokens). The base model is the release without continued pretraining.

	RULER, 8K			RULER, 16K			
condition	single	MK2	MK3	single	MK2	MK3	LongBench
Qwen3-0.6B							
base model, no training	100	69	56	100	61	37	28.1
matched control	100	67	8	96	71	2	21.6
head merge	60	2	0	24	0	0	11.3
our latent, rank 512	98	40	3	82	13	0	17.6
probe, module-local repair	100	63	0	100	18	0	19.8
Qwen3-4B							
base model, no training	100	99	100	100	99	95	45.4
matched control	100	98	59	100	70	24	25.1
head merge	92	8	0	5	0	0	16.5
head merge, retrieval heads spared	98	58	8	47	0	0	16.2
our latent, rank 1024	100	96	33	95	73	8	26.0
Qwen3-8B							
base model, no training	100	98	100	100	100	100	44.3
matched control	100	97	89	99	95	62	30.0
head merge	100	20	0	74	0	0	16.2
Qwen3-14B							
matched control	100	99	93	100	97	70	33.3
head merge	99	66	7	73	4	0	19.1
Llama-3.1-8B							
matched control	100	100	93	100	99	87	25.4
head merge	98	6	3	59	0	0	15.7
MLA form	90	4	1	90	1	1	14.4

score than of recall, and the latent keeps at least 80% of both at 8K. On `Qwen3-8B`, read without a prior prediction, the merge keeps 21.8% and 14.5% of recall and 67.1% and 61.5% of question answering. HELMET sizes its inputs with the Llama-2 tokenizer, so `Qwen3` reads about 6–7K tokens at the 8K setting and 12–15K at 16K, and we run attention through PyTorch's SDPA rather than FlashAttention-2.

Table 22: HELMET (Yen et al., 2025) after the matched budget. Recall: the mean of JSON key-value retrieval and RULER's multi-key 2, multi-key 3 and multi-value tasks. RAG: the mean of four KILT question-answering tasks (Natural Questions, TriviaQA, HotpotQA, PopQA). The control row gives HELMET's own score (0–100); the other rows give % of the control at the same length.

		Recall		RAG	
base	condition	8K	16K	8K	16K
Qwen3-4B	*control, HELMET score*	*81.2*	*54.9*	*52.0*	*48.8*
	head merge, % of control	28.3	5.5	66.5	49.0
	our latent, rank 1024, % of control	89.2	92.6	99.5	99.9
Qwen3-8B	*control, HELMET score*	*94.2*	*82.2*	*59.2*	*53.6*
	head merge, % of control	21.8	14.5	67.1	61.5

C.11 Benchmark scores hide the retrieval drop

The benchmarks that conversion papers report can stay near their reference while multi-key retrieval drops, and this subsection shows it across conversions and across model sizes. Figure 8 places every conversion of Table 8 by its benchmark average and its multi-key retrieval. Our head merges and several released conversions stay within 10% of their reference on the benchmarks yet keep far less retrieval. Figure 9 follows the head merge from 0.6B to 14B. OpenBookQA stays near the control at every size, and multi-key retrieval rises from 7.4% to 81.8%. Table 23 gives the absolute values behind Table 1.

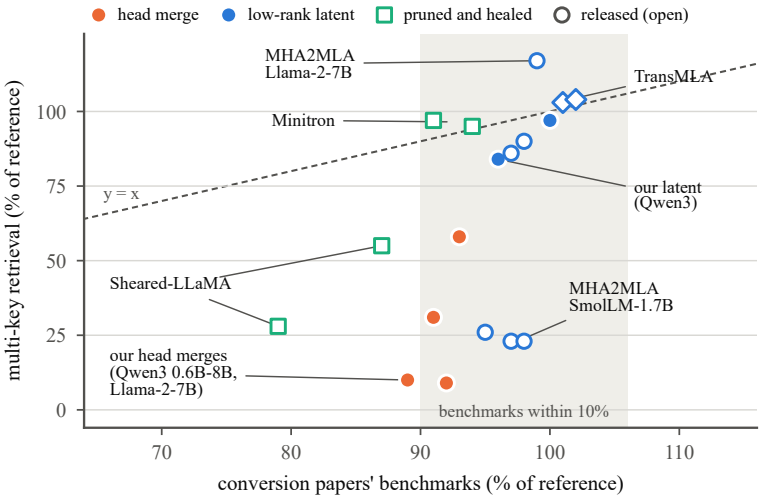

Figure 8: The conversion papers' commonsense average against multi-key retrieval for every conversion of Table 8, both in % of the reference. The reference is the matched control for our conversions (filled markers) and the full-cache twin or the parent for released checkpoints (open markers). The shaded band marks the models within 10% of their reference on the benchmarks. On the dashed line $y = x$ a conversion keeps retrieval and benchmark accuracy in the same proportion, so a point below it loses more retrieval than benchmark accuracy. Our head merges, MHA2MLA's `SmolLM-1.7B` releases and Sheared-LLaMA sit far below the line, most of them inside the band, while our latents, MHA2MLA's `Llama-2-7B`, TransMLA and Minitron lie near or above it.

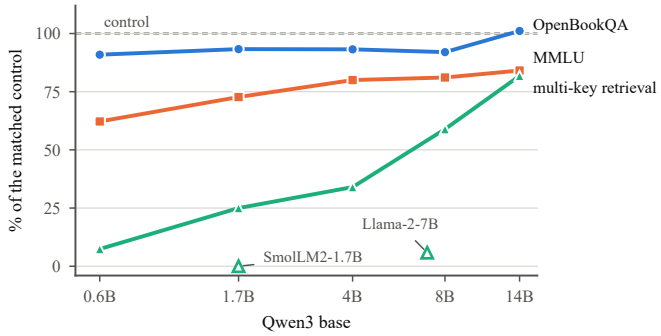

Figure 9: The head merge after the matched budget on five `Qwen3` bases, in % of the matched control on OpenBookQA, MMLU and multi-key retrieval (Table 23), with the `Llama-2-7B` and `SmolLM2-1.7B` merges as open markers. OpenBookQA stays near the control at every size, while multi-key retrieval rises from 7.4% to 81.8% and stays near zero on the other two families.

Table 23: Absolute values behind Table 1: the matched control and the head merge after the matched budget on five sizes of `Qwen3`-Base. Loss in nats above the original model; task scores are accuracies in % (OpenBookQA and MMLU zero-shot in lm-evaluation-harness 0.4.9.2, MK2 at 4K with $n = 100$). OpenBookQA is single-step commonsense, MMLU multi-step knowledge, MK2 keyed multi-key retrieval.

		Qwen3-Base size				
		0.6B	1.7B	4B	8B	14B
loss (nats)	control	+0.04	+0.04	+0.10	+0.08	+0.10
	head merge	+0.19	+0.16	+0.19	+0.17	+0.19
OpenBookQA (%)	control	24.2	30.0	32.4	34.8	35.4
	head merge	22.0	28.0	30.2	32.0	35.8
MMLU (%)	control	44.6	58.6	65.6	70.6	74.5
	head merge	27.8	42.6	52.5	57.3	62.6
MK2 (%)	control	94.0	76.0	94.0	95.0	99.0
	head merge	7.0	19.0	32.0	56.0	81.0

C.12 Decoding speed and memory

The memory of each form follows from its shape. Per token and layer the full cache holds $2gd$ numbers, the merge gd, the latent r and the pooled cache $2gd/m$, so at the matched cache of `Qwen3-8B` the merge and the rank-gd latent keep 50% of the full cache and pooling blocks of four keeps 25%.

Whole-model timing. Table 4 and Figure 10 time the whole model at the shape of `Qwen3-4B`. Each decoding step is captured as one CUDA graph that runs the 36 layers of the form's attention path and the MLP, the final norm, the tied output projection and greedy decoding, and prefill runs one prompt through the whole model and writes the form's cache. The throughput ratios agree with vLLM on the same checkpoints within 2.3% at 11 decoding points from 4K to 64K tokens and batch 1 to 16.

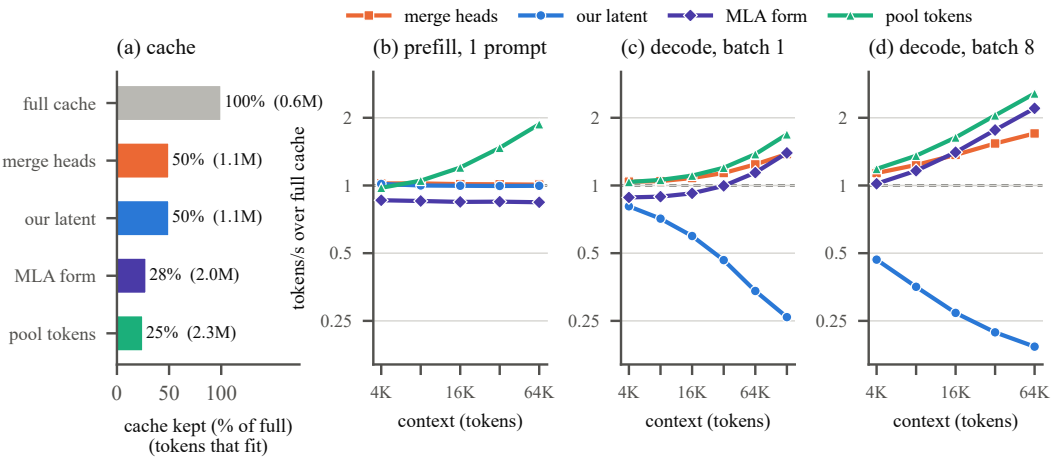

Figure 10: What each conversion keeps of the KV cache and what it buys at inference, at the shape of `Qwen3-4B` (one GH200, the whole model in bf16, the fastest kernel for each form). (a) Cache per token and layer as a share of the full cache, and the tokens that fit where the full cache holds 0.6 million. (b) Prefill throughput of one prompt, over the full cache's. (c, d) Decode throughput at batch 1 and 8, over the full cache's, on a log scale. Pooled tokens and the MLA form gain most at long contexts and large batches, and the head merge up to 1.70 times. Our latent decodes at 0.60 times the full cache's rate at 16K tokens and batch 1 and 0.19 times at 64K and batch 8, because its rotary embedding follows W_U and every cached key is rebuilt at each step. Throughput ratios agree with vLLM on the same checkpoints within 2.3%.

Per-layer timing. At 16K tokens and batch 8, the forms that keep less of the cache decode each layer faster than the full cache, and our latent decodes it slower. Merging every pair runs 1.77 times as fast as the full cache, merging while sparing the retrieval heads 1.30 times, the MLA form 2.02 times and our latent 0.18 times. These measurements time one decoding step of one layer on one NVIDIA GH200 GPU, the attention together with the form's own extra work, as the median of 100 runs of a CUDA graph of 36 layers divided by 36. Every path was first checked against an fp32 reference. Each form uses the fastest kernel we found for it: FlashInfer for the full cache and the merges (the spared merge with one call per head-group size), FlashAttention-3's MLA decode for the MLA form, with FlashMLA and FlashInfer's MLA kernel within 2%, and for our latent a fused Triton key-rebuild kernel with a split-K Triton kernel for the values, whose path is folded into the output projection.

Why our latent is slow. Our latent is bound by compute rather than by memory, because rebuilding every cached key costs $2BTrgd$ floating-point operations per step at batch B and T cached tokens. The straightforward implementation, which also rebuilds the values and applies the rotary embedding unfused, runs at 0.06 times the speed of the full cache, so the fused kernel gains a factor of three and still leaves the form slower than the full cache. A latent that rebuilds its keys at every step can decode

faster than the full cache only if the rank r_k of its keys is below about 330. Each step recomputes $2r_kgd$ floating-point operations per cached token and layer, and once r_k exceeds that rank these take longer, at an achievable 600 TFLOP/s on this GPU, than reading the full cache's $4gd$ bytes at the 3.67 TB/s its decoding reaches. Below about 160 the rebuild would hide under the latent's own read, and the latent would decode as fast as the merge. Lowering the key rank costs retrieval, however. Truncating the keys of our trained `Qwen3-4B` latent without further training keeps 76.1% of its multi-key retrieval at rank 256 and 35.9% at rank 128. With our kernels, rank 128 still decodes at 0.50 times the full cache's throughput at 16K tokens and batch 8, because the rebuild re-reads the rotary tables for each KV group and does not overlap the latent read.

Context and batch. Tables 24 and 25 repeat the per-layer measurement over contexts from 4K to 128K tokens, at batch 1 and at batch 16 (8 at 64K and 4 at 128K, the largest batches at which the full cache fits), for one layer's attention and for the whole attention path with its projections. The merge's gain grows with context and batch toward the ratio of the caches and reaches 1.94 at 32K and batch 16, where pooling blocks of four, the fastest form, reaches 3.60. Our latent is slower than the full cache at every point, by 2.4 to 6.0 times, and the more so the longer the context and the larger the batch, since its extra work grows as $BTrgd$. The MLA form depends on the batch. At batch 1 its one shared latent head leaves the kernel little parallel work, and it overtakes the merge only at 128K. At batch 16 it matches the merge at 4K and overtakes it from 8K.

Table 24: Speed-up over the full cache (×) of one layer's attention with each form's own extra work, at the attention shape of Qwen3-8B on one NVIDIA GH200 GPU, at batch 1 and the largest batch up to 16 at which the full cache fits. Italic rows: the full cache's time for all 36 layers (ms).

		speed-up over the full cache (×)					
conversion	cache (%)	4K	8K	16K	32K	64K	128K
batch 1							
full cache (ms)	*100*	*0.67*	*0.84*	*1.22*	*1.83*	*3.11*	*5.67*
head merge	50	1.13	1.22	1.42	1.48	1.66	1.80
head merge sparing retrieval heads	64	0.68	0.75	0.87	0.98	1.16	1.34
our latent, r=1024, fused key rebuild	50	0.41	0.32	0.26	0.21	0.18	0.17
our latent, r=1024, unfused rebuild	50	0.14	0.10	0.08	0.07	0.06	0.05
MLA form, 512+64	28	0.60	0.67	0.85	1.07	1.45	1.94
pooled tokens, m=4	25	1.20	1.40	1.77	2.14	2.54	3.06
batch 16 (8 at 64K, 4 at 128K)							
full cache (ms)	*100*	*3.30*	*5.83*	*10.93*	*21.09*	*21.10*	*21.11*
head merge	50	1.62	1.76	1.87	1.94	1.95	1.94
head merge sparing retrieval heads	64	1.14	1.29	1.39	1.46	1.48	1.48
our latent, r=1024, fused key rebuild	50	0.19	0.18	0.17	0.17	0.17	0.17
our latent, r=1024, unfused rebuild	50	0.06	0.06	0.05	0.05	0.05	0.05
MLA form, 512+64	28	1.63	2.08	2.46	2.81	2.80	2.81
pooled tokens, m=4	25	2.32	2.85	3.29	3.60	3.61	3.62

Table 25: As Table 24, for the whole attention path of the 36 layers, projections included.

		speed-up over the full cache (×)					
conversion	cache (%)	4K	8K	16K	32K	64K	128K
batch 1							
full cache (ms)	*100*	*2.36*	*2.51*	*2.84*	*3.46*	*4.74*	*7.33*
head merge	50	1.12	1.13	1.19	1.27	1.42	1.59
our latent, r=1024, fused key rebuild	50	0.71	0.59	0.47	0.35	0.26	0.21
our latent, r=1024, unfused rebuild	50	0.36	0.26	0.17	0.12	0.08	0.07
MLA form, 512+64	28	0.79	0.80	0.86	0.98	1.20	1.57
pooled tokens, m=4	25	1.06	1.10	1.20	1.36	1.64	2.09
batch 16 (8 at 64K, 4 at 128K)							
full cache (ms)	*100*	*4.87*	*7.42*	*12.49*	*22.68*	*22.75*	*22.75*
head merge	50	1.39	1.55	1.69	1.82	1.84	1.83
our latent, r=1024, fused key rebuild	50	0.27	0.22	0.19	0.18	0.18	0.18
our latent, r=1024, unfused rebuild	50	0.09	0.07	0.06	0.06	0.06	0.05
MLA form, 512+64	28	1.24	1.59	2.04	2.48	2.47	2.50
pooled tokens, m=4	25	1.62	2.05	2.55	3.05	3.04	3.06

With the MLP half of each layer added, the speed-ups at 32K and batch 16 become 1.62 for the merge, 0.20 for our latent, 2.04 for the MLA form and 2.35 for pooling. Peak memory at the same point follows the cache, with 82 GB for the full cache, 42 for the merge, 43 for our latent, 26 for the MLA form and 23 for pooling. Software: PyTorch 2.11.0.dev20260212, CUDA 12.8, FlashAttention-3 3.0.0, FlashInfer 0.7.0, FlashMLA 1.0.0, Triton 3.6.0, driver 570.148.08.

End to end in vLLM. End to end the gains are smaller than per layer, because at 16K tokens and batch 8 every step reads about as many bytes of weights as of cache. For the same reason the MLA form, whose layers do more work per head, gains no more than the merge at that point. vLLM 0.19.1 decodes 256 tokens after a prefix-cached prompt of 16K tokens at batch 8 with its default attention backends (FlashAttention-3 for grouped-query attention and its MLA backend for the MLA form), `torch.compile` and CUDA graphs, and we report the median time per output token over five repetitions, which agree within 0.5%. vLLM's other backends, FlashMLA for the MLA form and FlashInfer for grouped-query attention, were no faster at 16K tokens and batch 8. Timing does not depend on the weights: the `Qwen3-8B` rows of Table 26 are our trained checkpoints, and the `Llama-3.1-8B` models use random weights of the right shapes, 8 against 4 KV heads and, for the MLA form, the configuration of the conversion trained in §6.4 (a latent of 512 and a rotary part of 64). Table 26 gives the end-to-end measurement over contexts from 4K to 64K and batches of 1, 8 and 16. The MLA form overtakes the merge from 16K tokens at batch 16 and from 32K at batch 8, and at 64K tokens and batch 16, where only the merge and the MLA form fit in memory, it reaches 1.40 times the merge's throughput (0.82 at the least favourable point of the grid). At batch 1 it is slower than even the full cache, and the merge stays ahead of it.

Table 26: End-to-end decoding in vLLM 0.19.1: speed-up in tokens per second over the full cache of the same base (×; above 1 is faster). Italic rows: that full cache's time per output token (ms). Dashes: the full cache does not fit, or Qwen3-8B beyond 32K (not run). At 64K the MLA form runs without its rope scaling.

	batch 1				batch 8				batch 16		
speed-up over the full cache (×)	4K	16K	32K	64K	4K	16K	32K	64K	4K	16K	32K
Llama-3.1-8B, random weights											
full cache (ms)	*5.84*	*6.30*	*6.87*	*8.04*	*6.91*	*10.32*	*14.90*	*24.20*	*8.09*	*14.93*	*24.12*
head merge	1.05	1.09	1.12	1.20	1.12	1.31	1.46	1.63	1.18	1.46	1.61
MLA form, 512+64	0.88	0.90	0.93	0.98	0.99	1.28	1.59	2.02	1.11	1.60	2.01
Qwen3-8B, our trained checkpoints											
full cache (ms)	*6.05*	*6.56*	*7.22*	–	*7.39*	*11.15*	*16.34*	–	*8.59*	*16.34*	–
head merge	1.05	1.08	1.12	–	1.14	1.32	1.49	–	1.20	1.48	–

Pooled tokens also run in vLLM, through a custom attention backend whose cache holds only the pooled entries. At 16K tokens and batch 8 they decode 1.70 times and prefill 1.35 times as fast as the full cache and hold 4.00 times the tokens. Table 27 sets the merge beside them.

Table 27: Decoding in vLLM 0.19.1 on Qwen3-4B: speed-up in tokens per second over the full cache (×; above 1 is faster). Italic row: the full cache's time per output token (ms). Pooled tokens run through a custom attention backend that caches only the pooled entries. At batch 16 the full cache does not fit 64K tokens.

	batch 1				batch 8				batch 16		
speed-up over the full cache (×)	4K	16K	32K	64K	4K	16K	32K	64K	4K	16K	32K
Qwen3-4B											
full cache (ms)	*4.22*	*4.72*	*5.38*	*6.70*	*5.45*	*9.20*	*14.32*	*24.59*	*6.69*	*14.35*	*24.61*
head merge	1.05	1.09	1.15	1.25	1.16	1.39	1.57	1.72	1.25	1.56	1.71
pooled tokens, m=4	1.05	1.13	1.24	1.42	1.23	1.70	2.13	2.66	1.40	2.14	2.66

D WHAT RETROFIT PAPERS EVALUATE

Table 28 supports the evaluation claim of §1. Each row is read from the experiments of the cited paper rather than from its abstract. Long-context retrieval is measured in some conversion families, but not in the regime in which a retrofit is *cheap*. The head-merge and latent conversions, all performed at academic budget on bases of 8B parameters or less, report the commonsense bundle and at most LongBench, and none of them reports keyed multi-key retrieval at a controlled context length. Dynamic Memory Compression, which retrofits a learned merge of consecutive tokens into Llama 2 and trains at 4K tokens, reports MMLU, six commonsense tasks and HumanEval, and no retrieval test either (Nawrot et al., 2024). The other families do measure retrieval. LoLCATs runs a passkey test, which its Alpaca-only linearization fails and passkey data repairs, and Mamba in the Llama runs single-needle retrieval (Zhang et al., 2025; Wang et al., 2024). RADLADS reports RULER at 4K to 16K tokens after converting a 7B model on 350M to 700M tokens, and the selection retrofits SeerAttention and MoBA, the 50B-token conversion of HiLS and the frontier-scale sparse attention of MiniMax report RULER up to 128K tokens (Goldstein et al., 2025; Gao et al., 2025b; Lu et al., 2025; Hu et al., 2026; Lai et al., 2026). LoLCATs' passkey result agrees with ours, since the converted model retrieves only after training on data that asks for it. The gap this paper addresses is therefore specific to the cheap regime, and §5 fills it for five model families from 0.6B to 14B parameters.

Table 28: What conversion papers train on and what they evaluate, read from each paper's own experiments. "Keyed retrieval?" asks whether the paper reports a multi-key retrieval benchmark at controlled context length. [§]LongBench only. [‡]A passkey test, which the Alpaca-only linearization fails and passkey data repairs. [¶]Single-needle retrieval. A dash: not listed.

conversion	bases	recovery data	budget	keyed retrieval?
Head-merge and latent conversions: the classes we study				
MHA→GQA uptraining (Ainslie et al., 2023)	T5 Large / XXL	T5 pretraining mix (C4)	5% of pretrain compute	✗
MHA→GQA, head alignment (Jin et al., 2025)	LLaMA2-7B, Sheared-LLaMA-1.3B	8 commonsense SFT sets	93M–279M tok	✗
GQA→MLA (Meng et al., 2025)	SmolLM-1.7B, LLaMA-2-7B, LLaMA-3-8B	SmolLM corpus	0.3–6B tok (30–60B, LLaMA-3-8B)	✗
MHA→MLA (Ji et al., 2025)	SmolLM 135M–1.7B, Llama-2 7B/13B	SmolLM mixture	0.6–1% of pretrain	partly[§]
Other conversion families				
dense→learned token merging (Nawrot et al., 2024)	Llama-2 7B/13B/70B	Pile, mC4, CC, The Stack, …	24B–168B tok	✗
softmax→linear attention (Zhang et al., 2025)	Llama-3 8B, Mistral-7B	cleaned Alpaca 50K	≈40M tok	passkey[‡]
attention→Mamba hybrid (Wang et al., 2024)	Zephyr-7B, Llama-3 8B-Inst	UltraChat, GenQA, …	≈20B tok	NIAH[¶]
softmax→RWKV variants (Goldstein et al., 2025)	Qwen2.5 7B–72B	—	350M–700M tok	RULER 4K–16K
dense→hierarchical sparse (Hu et al., 2026)	7B	—	50B tok	RULER 8K–128K
Selection over raw tokens, retrofit				
dense→gated sparse prefill (Gao et al., 2025b)	Llama-3.1-8B-Inst	—	≈40 A100-hours	RULER 4K–128K
dense→block selection (Lu et al., 2025)	Llama-3.1-8B	—	100B tok	RULER 128K
dense→blockwise sparse (Lai et al., 2026)	109B MoE	—	400B tok	RULER, HELMET
This paper				
head merge, rotation probe (§5)	Qwen3 0.6–14B, SmolLM2-1.7B, Falcon3-1B, Llama-2-7B	`fineweb-edu`	1.5×10^8–1.5×10^9 tok	RULER MK2, VT, CWE; MMLU, OBQA

[§] Ji et al. (2025) report LongBench, of which only `passage_retrieval_en` is keyed retrieval, and no RULER or NIAH. [¶] Wang et al. (2024) report needle-in-a-haystack at 20× the distillation length, which is *single*-needle retrieval. Table 20 measures the difference between the two behaviours. Every surgery in that table keeps single-needle retrieval, while the head merge keeps 5.6% of the control's multi-key retrieval once the haystack is made of distractor needles.

E PUBLIC PRUNE-AND-HEAL RELEASES

Two public prune-and-heal families, scored against their parents on the suite and with the two scorings of Table 8, split by whether the pruning removes attention heads (Table 29). Minitron's 4B models (Sreenivas et al., 2024), which keep every attention head of the layers they keep, keep 94.7% and 96.7% of their parent's multi-key retrieval, and Sheared-LLaMA's (Xia et al., 2024), which remove heads, keep 27.7% and 55.3%. Under lm-evaluation-harness the Sheared-LLaMA models read 51.9% and 73.2%, more than 10 points higher, and the Minitron models read 99.5% and 100.0%. The two recipes also differ in data, budget and teacher, so the split does not isolate the effect of removing heads.

Table 29: Public prune-and-heal releases against their parents: the commonsense average of Table 8 and multi-key retrieval (MK2 at 4K, $n = 400$) under our builder and under lm-evaluation-harness, in % of the parent. A parent is not a matched control, so a release can exceed it.

		% of the parent model		
release	setting	commonsense	MK2	MK2, lm-eval
Minitron, from Llama-3.1-8B	4B, width	94.3	94.7	99.5
	4B, depth	91.2	96.7	100.0
Sheared-LLaMA, from Llama-2-7B	1.3B	78.8	27.7	51.9
	2.7B	87.3	55.3	73.2

F EXPERIMENTAL DETAILS

Retrofit recipe (per-token conditions and controls). The matched budget is under 10^{-5} of the 36T tokens `Qwen3` was pretrained on and inside the range Align-heads uses for `Llama-2-7B` (93M to 279M tokens). Continued pretraining on FineWeb-Edu at sequence length 4096, local batch 4 (2 at 1.7B) with gradient accumulation to a global batch of 16, 2300 steps (1.5×10^8 tokens), AdamW with default β, weight decay 0, $\epsilon = 10^{-8}$, gradient clipping at 1.0, bf16, 50 warm-up steps, cosine decay over the final 20% of steps to zero. The learning rate is reported per condition in Table 13. The budget-scaling run uses 23,000 steps with 100 warm-up steps and the same decay fraction. Checkpoints every 2,300 steps are converted and evaluated with the same command line. Training uses per-block compilation except for the layer-selective merges, which run eagerly, and no run uses activation checkpointing.

Surgeries. The head merge averages the rows of each adjacent pair of key and value projection heads in weight space and halves the KV head count. Every other tensor is copied. The layer-selective merges keep the original head count and store each merged head in both slots of its pair, which is functionally the merged layer. During training, gradient hooks replace the accumulated gradient of each pair by its within-pair mean, so the duplicates stay bit-identical for the whole run (verified on the endpoint checkpoints: maximum within-pair difference 0). The activation swap changes the MLP activation from SiLU to ReLU in the configuration and copies every tensor. The rotation multiplies each value head's projection rows by an independent Haar-random orthogonal matrix and leaves the keys untouched (a key rotation would scramble the RoPE pair structure and would not be undone by a linear re-parameterization). Its exact invertibility through the output projection was verified on a small random model. The noise control adds isotropic Gaussian noise with standard deviation 0.7 times the root-mean-square weight to every key and value projection. The scale was calibrated on CPU to match the merge's teacher-forced drift. All surgeries are applied to the released base checkpoint before training.

Evaluation. MK2 follows the RULER `niah_multikey_2` template: one target needle keyed by a random adjective-noun word with a seven-digit value, embedded among distractor needles with distinct keys, prompt packed to the context length, greedy decoding of 12 tokens, exact string match of the value. The builder draws distractor keys without replacement and packs by exact token count. Held-out cross-entropy uses FineWeb-Edu documents after skipping the first 200,000, packed into 32 (endpoint: 16) sequences of 4096 tokens, teacher-forced, with the original dense model scored on the same sequences. Benchmarks run in lm-evaluation-harness (Gao et al., 2024), zero-shot and

without a chat template on every task, MMLU included. Only the unhealed sweep (Appendix C.8) uses 5-shot MMLU. The size series of Table 23 uses harness version 0.4.9.2 and reports accuracy on every task. The eight-task commonsense average of Table 8 and Figure 1a uses version 0.4.13, with length-normalized accuracy on ARC-Easy, ARC-Challenge, HellaSwag, OpenBookQA and PIQA and accuracy on MMLU, WinoGrande and BoolQ. Each percentage compares a model with its reference under the same harness version and scoring. Multi-key retrieval on the `SmolLM` releases runs at 2K tokens, the end of their context, and at 4K on every other model.

The per-run evaluations that the text quotes use 100 prompts per cell. On the `Qwen3-0.6B` control those prompts read 0.94, against 0.84 over all 400, so a percentage of that control is lower in the text than in Table 8, which scores condition and control on the same 400 prompts. The module-local repair, for example, reads 75.3% in the text and 84.9% in the table. No ordering changes.

Closed-form re-alignment. For a surgery checkpoint, the linear projection that follows the modified module (W^O for the attention surgeries, the down-projection for the activation swap) is replaced by the ridge solution of $\min_W \|AW^\top - Y\|^2 + \lambda\|W\|^2$, where the rows of A are the modified module's pre-projection features (the concatenated head reads, or the gated intermediate $\sigma'(W^g x) \odot W^u x$) computed on the original model's captured module inputs, the rows of Y are the original module's outputs on the same inputs, and $\lambda = 10^{-3}\,\mathrm{tr}(A^\top A)/D$. Eight held-out FineWeb-Edu sequences of 4096 tokens (skipping 300,000 documents, disjoint from the set used for the measurements) are used for the fit, one regression per layer in double precision, and the before/after teacher-forced cosines are reported on two further sequences. For the rotation the solution is the inverse rotation up to numerics (cosine 1.000 after the refit). For the merge it is the best linear read of merged values, which cannot recover the two heads' separate content, and the per-position value reconstruction R^2 of the refit checkpoint is unchanged (0.87). Table 5 scores each refit checkpoint as fitted, before any training, and the refit conditions of Table 13 are then trained with the recipe above.

Update rank. For a trained checkpoint and its initialization (the surgery checkpoint for surgery conditions, the base for the control), the difference of every two-dimensional block weight is decomposed by singular values. We report the relative Frobenius norm, the smallest rank carrying 90% of the squared singular values, and the energy share of the top 64 directions, as medians over the 28 layers per module type.

The hybrid `Qwen3.5` bases. `Qwen3.5-0.8B` and `Qwen3.5-2B` have 24 layers, of which 6 use softmax attention with two KV heads and 18 use Gated DeltaNet, and `Qwen3.5-4B` has 32 layers, of which 8 use softmax attention with four KV heads. The merge averages each adjacent pair of KV heads in the softmax-attention layers and leaves the Gated DeltaNet layers untouched. The control is the unconverted base, and both are trained with the matched budget and learning rate of the size series above, in a Hugging Face harness with the `fla` kernels for the Gated DeltaNet layers, which we checked against the `transformers` reference implementation on one batch (matching loss, and gradients that agree tensor by tensor). Because the harness differs from the one the `Qwen3` size series uses, each merge is compared only with its own control.

The hybrid's merged heads heal in place, and a pure transformer's do not. Two further pairs at 0.8B, specified before any training step, locate the recovery. With only the attention of the six softmax-attention layers trainable (5.5% of the parameters), the merge keeps 88.6% of an identically frozen control. With full training at 2×10^{-5} the control falls only to 89%, from the base's 99, instead of to 74, and the merge still keeps 89.9% of it, so the hybrid's high fraction does not come from a recipe that degrades the control. A merged `Qwen3-0.6B` with only its attention trainable (25.9% of the parameters) keeps 15.5% of its identically frozen control, and merging only every fourth of its layers, the hybrid's share of softmax-attention layers, keeps 53.1%. Low-rank adaptation recovers even less. With rank-64 adapters on the query, key, value and output projections and every other weight frozen, the merged `Qwen3-0.6B` keeps 2.1% of the full-parameter control (3.6% at $n = 400$), against 7.4% with every parameter trained, while the same adapters on the unmerged model keep 88.3% at $n = 100$ and 101.5% at $n = 400$. Which property of the hybrid lets its heads heal is not settled.

Retrieval in the `Qwen3.5-0.8B` base needs both layer types. Zeroing the outputs of token mixers without training (measured by MK2 at 4K and the loss of a fixed paragraph) takes its MK2 from 99 to 0 with the six softmax-attention mixers zeroed and to 0 with six random Gated DeltaNet mixers

zeroed (loss 2.12, 2.87 and 2.60 nats). Because the count-matched control collapses as well, the reading we predicted, that retrieval runs through the softmax layers alone, is not shown. After training, the merged model's six softmax-attention layers, each now with one KV head, still carry its retrieval, which falls from 73 to 0 when they are zeroed, while they carry little of the language model (loss 2.06 to 2.19 nats).

F.1 Continued pretraining against the untrained base

Under our recipe the matched control loses keyed retrieval against its untrained base on most bases, and the loss depends more on the family than on the size (Table 30). It is smallest on the dense Qwen3 bases, Falcon3-1B and SmolLM2-1.7B, and larger on Llama-2-7B, Gemma-2-2B and the hybrid Qwen3.5 bases and, at the rate of 1e-4, on Llama-3.1-8B and Mistral-7B, where multi-key retrieval falls from 85% to 1%. Every retention in the paper is therefore a fraction of the matched control, which carries this loss.

Table 30: Multi-key retrieval (MK2) at 4K tokens with our implementation ($n = 100$ and $n = 400$) and the commonsense average of the suite of Table 8 (lm-evaluation-harness), in %, for each untrained base and its matched control, trained at 1e-4 unless marked. The base rows of Table 21 give 8K and 16K.

	MK2, $n = 100$		MK2, $n = 400$		commonsense avg.	
base model	base	control	base	control	base	control
Qwen3-0.6B	89	94	92	84	–	53.3
Qwen3-1.7B	84	76	–	–	–	–
Qwen3-4B	96	94	–	94	–	66.9
Qwen3-8B	98	95	–	94	–	70.4
Qwen3-14B	100	99	–	99	–	72.9
Qwen3.5-0.8B	99	74	–	72	–	–
at 2×10^{-5}		89		88		–
Qwen3.5-2B	99	72	–	69	–	–
Qwen3.5-4B	100	75	–	78	–	–
Llama-2-7B	64	52	66	48	63.5	60.0
Llama-3.1-8B	99	58	99	–	70.6	–
at 2×10^{-5}		97		98		69.8
Mistral-7B	85	1	–	–	–	–
at 2×10^{-5}		71		–		–
Gemma-2-2B, at 2×10^{-5}	76	48	–	49	–	–
Falcon3-1B	79	80	70	73	–	–
SmolLM2-1.7B	66	62	59	57	–	–

G The recovery is not low-rank

The updates that continued pretraining makes after a surgery are high-rank and are not confined to the modified module. On no matrix of any condition do the top 64 singular directions carry more than 46.8% of the update energy. For the merge at 1e-4, the key projections move by 0.20 in relative Frobenius norm and the MLP matrices by 0.08–0.10, and 90% of the key update's energy needs 295 of 512 singular directions. The swap's update is spread uniformly over attention and MLP, with a relative Frobenius norm of 0.08–0.10 per matrix and a 90%-energy rank of 481–723 of 1024. A rank-8 to rank-64 adapter therefore could not express the solution that full training found. Rank-64 adapters trained on the merged `Qwen3-0.6B` keep 2.1% of the full-parameter control, against 7.4% with every parameter trained (Appendix F). Runs at the lower learning rate moved their weights ten times less (0.8–2.8%), the weight-space counterpart of the learning-rate fragility of §C.3. The method is described in Appendix F, paragraph "Update rank".

H Evaluation harness details

The orderings that carry our claims do not depend on our implementation of the retrieval task. Every number labelled MK2 in the main text comes from our implementation of RULER's `niah_multikey_2`, which the pooled-token conditions need for their patched forwards (prompts

packed to 4K from distractor needles, 12 generated tokens, exact match). The long-context readings at 8K and 16K (Appendix C.10) and the `Qwen3-4B` rows of Table 17 use the harness's RULER tasks. We re-scored every dense checkpoint that carries a claim with the RULER port in lm-evaluation-harness 0.4.13 (Gao et al., 2024), with prompts packed to 4096 tokens from distractor needles as in our builder, 200 of the 500 prompts the task generates, greedy decoding of 128 tokens, string match on the needle value and each base's own tokenizer (Table 31). Absolute scores differ, as the two builders' prompt templates and decoding lengths do (the zero-shot bases read 0.825, 0.915 and 0.880 on the standard harness), so we compare only fractions of the same-family control. The ordering of every condition is identical on the two harnesses, and 12 of 14 fractions agree within 10 points. The 2 that do not are the swap at 3e-4, a ratio of two near-floor scores, and the keys-only merge (43.6% against 27.9%), which is why §C.7 does not rank the two halves of the merge. The activation swap reads higher on the standard harness, 98.3% of control against 89.3% on ours, so the residual gap of §C.3 is a property of the stricter 12-token exact match rather than of the model. The second-family swap's shortfall (28.6% on the standard harness) and every tier reading replicate.

Table 31: Our harness against lm-evaluation-harness's RULER `niah_multikey_2` at 4K on the dense checkpoints: accuracy, and % of the same-family control at the same learning rate (italic rows) on the same harness.

		MK2 accuracy			% of control	
condition	lr	ours, n=100	ours, n=400	lm-eval, n=200	ours	lm-eval
Qwen3-0.6B						
matched control	*1e-4*	*0.94*	*0.84*	*0.90*	*100.0*	*100.0*
base model, no training	–	–	0.92	0.82	–	–
matched control	7.3e-6	0.83	0.89	0.81	–	–
activation swap	1e-4	0.76	0.75	0.88	89.3	98.3
head merge	1e-4	0.07	0.08	0.05	9.8	5.6
K/V noise, drift-matched	1e-4	0.42	0.41	0.39	49.0	43.6
rotation probe	1e-4	0.01	0.00	0.00	0.3	0.0
rotation probe, repaired on every head	1e-4	0.88	0.80	0.85	95.0	95.5
activation swap + closed-form refit	1e-4	0.81	0.80	0.85	94.4	95.5
head merge + closed-form refit	1e-4	0.05	0.04	0.04	5.0	4.5
keys-only merge	1e-4	0.41	0.36	0.25	43.0	27.9
values-only merge	1e-4	0.27	0.27	0.25	32.0	27.9
matched control	*3e-4*	*0.23*	*0.23*	*0.14*	*100.0*	*100.0*
activation swap	3e-4	0.25	0.22	0.16	96.7	114.3
Falcon3-1B						
matched control	*1e-4*	*0.80*	*0.73*	*0.93*	*100.0*	*100.0*
base model, no training	–	–	0.70	0.92	–	–
head merge 4→2	1e-4	0.06	0.04	0.04	6.1	3.8
activation swap	1e-4	0.27	0.24	0.27	32.3	28.6
SmolLM2-1.7B						
matched control	*1e-4*	*0.62*	*0.57*	*0.84*	*100.0*	*100.0*
base model, no training	–	–	0.59	0.88	–	–
head merge 32→16	1e-4	0.00	0.01	0.00	1.3	0.0
activation swap	1e-4	0.45	0.43	0.65	75.1	76.9

The number of prompts does not move the readings either. Scored on $n = 400$ prompts against the control's own score at that size, the 93 percentages of the `Qwen3-0.6B` checkpoints shift by a median of $+1.0$ points from their values at $n = 100$ (from -6.9 to $+17.6$). Of the 3,372 pairs at least 10 points apart at $n = 100$, only 4 change order, three of them involving a control and none an ordering the text states.

I SHORT-CONTEXT BENCHMARKS AND A SECOND RULER TASK

In this harness the pooled-token models decode without a KV cache, so every row of Table 32, its own control included, is decoded without a cache, greedily, stopping at the first newline after content or at 48 tokens (seed 42). Its percentages are therefore of this control, not of the cached control of Table 20, which decodes up to 128 tokens, and its pooled-token rows score the checkpoints of Table 15 in the same job.

Table 33 gives the 0-shot `lm-eval` benchmarks on every checkpoint, and Table 34 the $n = 400$ re-measure of MK2 and the RULER variable-tracking and common-word tasks at 4K. At floor

Table 32: The needle tasks of Table 20 on the token and feature axes, Qwen3-0.6B after the matched budget, every row decoded without a KV cache (lm-evaluation-harness RULER, 4K, 200 prompts): the control's accuracy in the header row, every other cell in % of this control.

	single needle			multi-key			
condition	1	2	3	1	2	3	single mean
control, accuracy (0–1)	*1.00*	*0.99*	*1.00*	*0.90*	*0.86*	*0.36*	*1.00*
Token axis, % of the control							
pooled tokens, m=2	62.0	62.3	9.5	45.0	11.0	0.0	44.6
pooled tokens, m=4	20.0	14.1	0.0	5.0	0.6	0.0	11.4
Feature axis, % of the control							
our latent, rank 256	100.0	100.5	95.0	94.4	31.2	8.2	98.5
our latent, rank 128	98.5	99.0	87.0	56.1	1.2	4.1	94.8

keyed retrieval the merge keeps the common-sense multiple-choice tasks within 89.8–97.1% of the learning-rate-matched control (§5.1) and loses most on LAMBADA (78.5%), a last-word prediction that needs the whole passage, and on ARC-Challenge (83.4%). The rotation, the one condition that loses alignment with information intact, collapses everywhere (LAMBADA 34.7% of control). The zero-shot surgeries have broken decoders, so continued pretraining is doing real repair. The 1.7B control's row is missing because its checkpoint failed to load in the harness, and its RULER cells are in Table 34.

Table 33: 0-shot accuracies on lm-evaluation-harness (LAMBADA also as perplexity) per checkpoint, the matched control first (italic row); the rows without training are the surgeries before any training.

		0-shot accuracy						perplexity
condition	lr	ARC-C	ARC-E	HellaSwag	LAMBADA	PIQA	WinoGrande	LAMBADA ↓
Qwen3-0.6B								
matched control	*1e-4*	*0.35*	*0.69*	*0.40*	*0.49*	*0.70*	*0.59*	*13*
base model, no training	–	0.33	0.66	0.41	0.54	0.70	0.59	10
matched control	7.3e-6	0.34	0.67	0.41	0.54	0.70	0.60	9
activation swap	1e-4	0.33	0.66	0.38	0.46	0.69	0.57	16
activation swap	3e-4	0.26	0.60	0.34	0.35	0.66	0.54	37
activation swap	7.3e-6	0.23	0.52	0.33	0.30	0.61	0.52	49
K/V noise, small	1e-4	0.35	0.69	0.39	0.49	0.70	0.59	13
K/V noise, drift-matched	1e-4	0.32	0.66	0.37	0.43	0.68	0.56	19
head merge	1e-4	0.29	0.64	0.36	0.38	0.68	0.55	25
head merge	3e-4	0.26	0.58	0.33	0.33	0.65	0.53	50
head merge	7.3e-6	0.18	0.40	0.28	0.11	0.59	0.52	781
head merge, 10× budget	1e-4	0.29	0.65	0.37	0.43	0.67	0.55	19
head merge, retrieval-head layers	1e-4	0.32	0.68	0.39	0.46	0.69	0.58	15
head merge, other layers	1e-4	0.31	0.65	0.37	0.43	0.69	0.55	20
rotation probe	1e-4	0.18	0.46	0.28	0.17	0.59	0.51	338
rotation probe	3e-4	0.23	0.52	0.30	0.24	0.62	0.51	119
head merge, no training	–	0.21	0.28	0.26	0.00	0.54	0.51	$> 10^4$
activation swap, no training	–	0.20	0.28	0.26	0.00	0.54	0.50	$> 10^4$
Qwen3-1.7B								
base model, no training	–	0.42	0.74	0.49	0.63	0.76	0.64	6
activation swap	1e-4	0.42	0.76	0.45	0.55	0.73	0.62	8
head merge	1e-4	0.40	0.74	0.43	0.50	0.72	0.57	11

Table 34: MK2 at $n = 400$ (where re-measured) and RULER variable tracking (VT) and common-word extraction (CWE) at 4K, $n = 100$, on our builder: accuracy per checkpoint, the matched control first (italic row).

condition	lr	MK2, n=400	VT	CWE
Qwen3-0.6B				
matched control	*1e-4*	*0.84*	*0.90*	*0.05*
base model, no training	–	0.92	0.80	0.09
matched control	7.3e-6	0.89	0.79	0.07
activation swap	1e-4	0.75	0.19	0.11
K/V noise, drift-matched	1e-4	0.41	0.05	0.05
head merge	1e-4	0.08	0.16	0.03
head merge	3e-4	0.07	0.00	0.01
head merge	1e-3	–	0.00	0.00
head merge, 10× budget	1e-4	0.14	0.23	0.04
head merge, retrieval-head layers	1e-4	0.51	0.69	0.12
head merge, other layers	1e-4	0.42	0.43	0.06
rotation probe	1e-4	0.00	0.00	0.00
Qwen3-1.7B				
matched control	*1e-4*	–	*0.92*	*0.00*
base model, no training	–	–	0.73	0.01
activation swap	1e-4	–	0.68	0.02
head merge	1e-4	–	0.66	0.04

J PER-TOKEN LOSS DECOMPOSITION

Held-out FineWeb-Edu, 32 sequences of 4096 tokens. Each position is classed by the nearest earlier occurrence of its (previous token, token) bigram: *novel* (none), *copy-near* (fewer than 256 positions back) and *copy-far* (at least 256 back). On the original model the classes hold 79.2/12.7/8.1% of positions and 91.2/4.1/4.7% of the loss mass, with per-class loss 3.09/0.86/1.56 nats. Table 35 gives each condition's per-class excess loss over the same-size original model, the share of its total excess loss carried by copy-far positions, and the excess loss in the first and last eighth of the window. The excess loss is proportional to loss mass in every class and flat in position for every trained condition. Only the untrained merge departs from this.

Table 35: Per-token held-out excess loss over the original model, in nats, by token class and by position in the window (§B.1); copy-far share: the fraction of the total excess loss on copy-far positions, against their 4.7% of the loss mass.

			excess loss (nats)				copy-far	by position (nats)	
condition	lr	MK2	mean	novel	copy-near	copy-far	share (%)	first eighth	last eighth
Qwen3-0.6B									
matched control	1e-4	0.94	+0.040	+0.045	+0.022	+0.024	4.7	+0.030	+0.045
matched control	7.3e-6	0.83	-0.004	-0.005	+0.002	+0.001	-2.0	-0.011	-0.002
activation swap	1e-4	0.76	+0.100	+0.114	+0.041	+0.055	4.5	+0.087	+0.097
K/V noise, drift-matched	1e-4	0.42	+0.130	+0.146	+0.066	+0.080	5.0	+0.119	+0.132
head merge, retrieval-head layers	1e-4	0.58	+0.074	+0.081	+0.042	+0.052	5.7	+0.061	+0.081
head merge, other layers	1e-4	0.44	+0.145	+0.165	+0.064	+0.083	4.6	+0.139	+0.154
head merge	1e-4	0.07	+0.189	+0.212	+0.090	+0.122	5.2	+0.180	+0.200
head merge	3e-4	0.07	+0.304	+0.345	+0.132	+0.167	4.5	+0.286	+0.309
head merge	1e-3	0.02	+0.550	+0.627	+0.234	+0.299	4.4	+0.533	+0.543
head merge	7.3e-6	0.00	+1.319	+1.428	+0.871	+0.953	5.9	+1.282	+1.362
head merge, 4× budget	1e-4	0.15	+0.166	+0.184	+0.087	+0.111	5.4	+0.147	+0.177
head merge, 10× budget	1e-4	0.16	+0.084	+0.094	+0.044	+0.056	5.4	+0.071	+0.086
rotation probe	1e-4	0.01	+0.771	+0.869	+0.354	+0.468	4.9	+0.747	+0.763
head merge, no training	–	0.00	+10.432	+10.248	+11.498	+10.549	8.2	+8.511	+10.997
Qwen3-1.7B									
matched control	1e-4	0.76	+0.035	+0.039	+0.016	+0.023	5.3	+0.019	+0.041
activation swap	1e-4	0.78	+0.095	+0.108	+0.037	+0.056	4.8	+0.083	+0.094
head merge	1e-4	0.19	+0.161	+0.180	+0.074	+0.111	5.6	+0.138	+0.170

K AI USE STATEMENT

The authors formulated the research question, wrote the initial manuscript, and directed the research. We used Anthropic's Claude via Claude Code to assist with implementing the code, including the synthetic training and evaluation data generators; formulating the propositions in §3 and drafting their proofs; and interpreting results. Experimental designs were iteratively discussed with the tool, reviewed, and modified under the authors' direction to align with the research objectives. Claude also assisted with polishing the manuscript and checking reference formatting before submission. Claude was not used for translation. The authors made the final decisions on experimental design and scientific claims and take responsibility for all submitted content, including AI-assisted text, code, and analyses.