KV-LINGO: LEARNING KV-CACHE TRANSLATORS WITH DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models represent context with a key-value (KV) cache. Caches are model-specific: for the same text, models with different architectures or weights produce incompatible representations. This makes it costly to switch models over a shared context: although the context has already been processed by one model, the incoming model must process it again to build its own cache. We introduce KV-Lingo, a method for translating the KV cache of a source model into one that can be read by a target model. KV-Lingo consists of a collection of linear maps, typically one per layer of the target model, that are applied independently on all tokens' key and value representations. We train these maps using distillation, minimising the divergence between the target model's predictions from its native cache and those from the translated cache. We consider several model pairs spanning multiple sizes and architectures, training one translator per pair on a generic text corpus. The resulting translators preserve strong downstream performance in both small-to-large and large-to-small transfers. Since a switch then costs a linear map and a single decoding step instead of a prefill, replacing re-prefill with cache translation reduces the time to first token after a model switch by 9.6× already on a 64-token prompt for Qwen models on an Apple M3 Ultra, and by up to 29× at 32k context length on an H100. These gains make KV-Lingo particularly useful for dynamic model routing: a context can be processed by one model and handed off to another only when needed, without re-prefilling the shared prefix. We finally show that KV-Lingo can be used for seamless model switching, staying close to re-prefill across repeated switches in our multi-turn evaluations.

1 INTRODUCTION

Autoregressive transformers (Vaswani et al., 2017) generate one token at a time, attending over all previous tokens. The key-value (KV) cache stores the past keys and values so that each decode step costs $\mathcal{O}(n)$ rather than $\mathcal{O}(n^2)$ in the context length n; at long context, it dominates the memory cost of serving (Kwon et al., 2023; Hooper et al., 2024). The cache is tied to the model that built it: its shape follows the architecture and its coefficients the parameters, so handing it to another model, even one of matching shape, severely degrades generation (Liu et al., 2024a).

In many settings, several models run over one context: routers and cascades send harder queries to a stronger model (Chen et al., 2024; Ong et al., 2025), token-level routers defer selected tokens to a larger one (Shen et al., 2024; Fu et al., 2025), and multi-agent pipelines share a working context between specialists (Wu et al., 2024). In these cases, the KV-cache incompatibility across models has a cost: systems must either maintain a cache for every model that may be called, or compute the target model's cache when it is first invoked. This means re-encoding the entire prefix, duplicating much of the computation spent processing the same context, at a cost that grows with the context length.

In order to fix this incompatibility, we propose training *cache translators*, which transform the cache of a source model into a cache readable by a target model. The target model can then start generating right away, without paying the prefilling cost. We call our approach KV-Lingo. Its translators are simple and cheap to apply: each target layer uses separate token-wise linear maps for keys and values, reading from a small set of source layers. We train these maps with distillation, which yields strong downstream performance despite this restricted architecture (Figure 1). The fact that translators as

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.

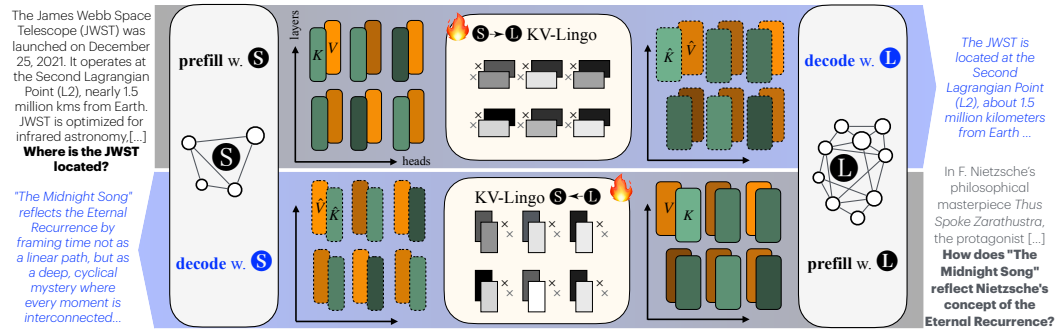


Figure 1: **KV-Lingo translates a KV cache between two frozen models**, so a conversation can switch model without re-prefilling its context. Top: the small model S prefills the prompt, and the large model L decodes the answer from the translated cache $(\widehat{K}, \widehat{V})$, dashed. Bottom: the same from L to S, with a second translator. Each translator (flame, the only trained part) is a set of linear maps, one per target layer for keys and one for values, applied token by token, so its cost grows linearly with the prefix length where re-prefilling grows quadratically.

simple as linear maps perform so well is remarkable; we show in Appendix B that the KV caches of different models typically have a very similar geometry, and are approximately linearly related, which empirically supports the above observation. Our contributions are as follows:

KV-Lingo, a simple translator model and training procedure (§3): a per-target-layer linear map, split across keys and values, fit in closed form by minimising the MSE, and then refined by self-distillation against the target's own next-token distribution. A central finding is that cache reconstruction is not enough: translators that closely match target activations can generate poorly, whereas self-distillation turns the same simple linear family into effective drop-in caches. We justify each architectural choice empirically: linearity, self-distillation, head mixing, pre- vs post-RoPE. We also adapt KV-Lingo so that it can translate caches across models using different tokenizers.

A systematic evaluation of KV-Lingo (§4) spanning multiple model pairs, from the families Qwen3 (Yang et al., 2025a), Gemma-3 (Team et al., 2025), Gemma-4 (Team et al., 2026) and Mistral-v0.3 (Jiang et al., 2023), scales ranging from 0.6B to a 30B mixture of experts, evaluated on a range of tasks from language modelling and multi-turn chat, to long-context understanding. On most tasks, the translated cache performs as well as or better than the smaller model. We also show that KV-Lingo translators are robust to several model switches in multi-turn settings.

An efficiency study (Figure 5): model switching with KV-Lingo costs a linear translation and a single decoding step of the target instead of a full prefill. We quantify the speed-up this yields on an Apple M3 Ultra and on H100 GPUs, reporting between $4.8\times$ and $29\times$ faster time to first token (TTFT) depending on model and prefill length.

The closest prior work, like ours, avoids prefilling the shared context with the target model. Chen et al. (2026) introduce a per-head nonlinear adapter, while Horton et al. (2024) jointly train an auxiliary model and per-layer linear maps. Concurrent works instead use closed-form linear fits between source and target caches (Heo et al., 2026; Qu et al., 2026; Li et al., 2026); this corresponds to the first stage of our training procedure (§3.2). Appendix A discusses the broader literature.

2 Problem Setting

Notation. We call *context* the tokens initially supplied to the models, and *prefix* the full sequence preceding the model switch: the context, possibly followed by a continuation generated by a model. Let $\mathcal{V}$ be the vocabulary and $x_{1:n} \in \mathcal{V}^n$ the prefix. A model with N layers, H key/value heads and head dimension d_h computes, at each layer, the keys and values of every prefix token. Collecting them layerwise, the *key-value (KV) cache* is the collection $\mathcal{C} = (K^\ell, V^\ell)_{\ell=1}^N$, with $K^\ell, V^\ell \in \mathbb{R}^{d \times n}$ and $d = H \cdot d_h$, with one column per prefix token and the H heads stacked along the d-dimensional axis. RoPE rotates keys by their token's position (Su et al., 2024), and some models also normalise

keys per head beforehand; we consider translating both pre-RoPE and post-RoPE keys. A model fills the cache in a single *prefill* pass over the prefix, then *decodes* a continuation one token at a time: each step reads the whole cache and appends the new token's keys and values to the cache.

An LLM models the probability of a sequence as $p(y_{1:m} \mid x_{1:n}) = \prod_{t=1}^{m} p(y_t \mid x_{1:n}, y_{<t})$. Conditioning on the prefix $x_{1:n}$ only requires the cache $\mathcal{C}(x_{1:n-1})$ of the past tokens together with the current token x_n: a forward pass on x_n attends to the cache and produces the next-token distribution. We therefore write $p\big(y_{1:m} \mid \mathcal{C}(x_{1:n-1}),\, x_n\big) = \prod_{t=1}^{m} p\big(y_t \mid \mathcal{C}(x_{1:n-1}),\, x_n,\, y_{<t}\big)$.

2.1 Cache translation

We consider two models: a *source*, which produces the cache of a prefix $x_{1:n}$ it may have partly generated, and a *target*, which must continue generating. We write $\mathcal{C}_{\mathrm{src}}(x_{1:n})$ and $\mathcal{C}_{\mathrm{tgt}}(x_{1:n})$ for the caches they produce on the same prefix.

Decoding from the target's cache, $p_{\mathrm{tgt}}\big(\cdot \mid \mathcal{C}_{\mathrm{tgt}}(x_{1:n-1}),\, x_n\big)$, is *native* decoding. A *translator* is a map T from caches in the source's format to caches in the target's format such that

$$p_{\mathrm{tgt}}\big(\cdot \mid T(\mathcal{C}_{\mathrm{src}}(x_{1:n-1})),\, x_n\big) \approx p_{\mathrm{tgt}}\big(\cdot \mid \mathcal{C}_{\mathrm{tgt}}(x_{1:n-1}),\, x_n\big), \tag{1}$$

as distributions over continuations $y_{1:m}$, for prefixes drawn from the contexts the translator is deployed on. In typical deployment, x_n is the last token of the generation prompt and carries no content.

Two remarks shape the problem. First, T need not reproduce $\mathcal{C}_{\mathrm{tgt}}$ entrywise, only the target's next-token distributions; cache reconstruction error and continuation quality can in fact disagree sharply (§4). Second, translation pays off only if applying T costs less than re-prefilling; since a 100-token cache of Qwen3-4B already holds 7.4 million values, this forces structured sparsity on T (§3.1).

We study the *post-hoc* setting where source and target are frozen; only T is trained, so no access to either model's pretraining pipeline or resources is needed. Training the two models jointly for compatible caches is an interesting direction, outside the scope of this paper.

2.2 Deployment scenarios

Model switching. We focus on settings involving two models of different capability: a *large* model L (better but slower) and a *small* model S (less capable but faster). Currently, when switching models, the target re-prefills the whole prefix before decoding its first token (Liu et al., 2024a; Geng et al., 2026). KV-lingo bypasses this. The two directions of transfer serve different purposes:

- In $S \to L$ transfer, S processes the prefix cheaply and hands it off to L only when the stronger model is needed. Translation reduces the latency of this hand-off by avoiding the L prefill, at the cost of a possible degradation relative to L decoding from its native cache.
- In $L \to S$ transfer, the representation computed by L can be handed off to the cheaper model S, allowing subsequent generation to benefit from the wealth in latent representation of L's processing while decoding with S. This is useful, for example, in agentic pipelines where a large model processes a shared context or produces a plan before delegating subsequent work to smaller models. Another usecase is privacy, where S and L models differ in data-access policies. A large model can process a non-sensitive context, whose translated cache is handed to a restricted model that subsequently receives private queries. This avoids exposing the private input to the large model while avoiding a second prefill of the shared context. In realistic scenarios where S runs on-device, this strategy would be assessed by comparing the gain in performance and the added communication cost to transfer the cache, against the Watts cost of prefilling with the small model.

Prefilling for a mixture of experts. In memory-bound inference, such as on-device generation, a mixture-of-experts (MoE) model (Shazeer et al., 2017) decodes about as fast as a dense model of its active size, reading only the experts each token is routed to, but prefills more slowly, as the tokens of a prompt together reach most of its experts. A dense S can prefill the prompt, and an MoE L decode from its translated cache (§4.6). The MoE, having more capacity, then decodes with better quality.

Remark 1. In a memory-bound case, a smaller model S could be used to prefill the context once and keep it as its own, smaller cache, directly used by L for decoding, by merging the translators with queries. This method, which requires post-RoPE and sparse translators, in only tested as a proof-of-concept (Appendix Z)

3 KV-LINGO: LINEAR TRANSLATORS TRAINED WITH SELF-DISTILLATION

3.1 LINEAR TRANSLATOR FAMILY

Given a prefix $x_{1:n}$, let $\mathcal{C}_{\mathrm{src}}(x_{1:n}) = (K^{\ell}_{\mathrm{src}}, V^{\ell}_{\mathrm{src}})_{\ell=1}^{N_{\mathrm{src}}}$ be the source cache, where $K^{\ell}_{\mathrm{src},i}, V^{\ell}_{\mathrm{src},i} \in \mathbb{R}^{d_{\mathrm{src}}}$ stack every head of token i. KV-Lingo translates the cache of target layer ℓ from a small subset $\mathcal{E}_\ell$ of source layers, simply $\{\ell\}$ when both models have the same number of layers:

$$\widehat{K}^{\ell}_i \;=\; \sum_{\ell' \in \mathcal{E}_\ell} T_K^{\ell' \to \ell}\, K^{\ell'}_{\mathrm{src},i}, \qquad \widehat{V}^{\ell}_i \;=\; \sum_{\ell' \in \mathcal{E}_\ell} T_V^{\ell' \to \ell}\, V^{\ell'}_{\mathrm{src},i}, \qquad i = 1, \dots, n-1, \tag{2}$$

with $T_K^{\ell' \to \ell}$, $T_V^{\ell' \to \ell} \in \mathbb{R}^{\,d_{\mathrm{tgt}} \times d_{\mathrm{src}}}$ (the token-wise structure is motivated in Appendix B). We write $T_K^{\ell} \in \mathbb{R}^{\,d_{\mathrm{tgt}} \times |\mathcal{E}_\ell|\, d_{\mathrm{src}}}$ for their concatenation over $\ell' \in \mathcal{E}_\ell$, and likewise T_V^{ℓ}. The keys and values of token n come instead from a forward pass of the target on the last prefix token, attending to the translated cache. We write $T(\mathcal{C}_{\mathrm{src}}(x_{1:n}))$ for the resulting cache. Sharing the token index i across both sides presumes the two models have the same tokenizer; §4.4 treats a pair that does not, where the correspondence is built between token *groups* covering the same characters. Equation 2 leaves several choices open: whether heads mix, where in the target's stack the map acts, and how to set $\mathcal{E}_\ell$.

The power of linearity. We choose linear maps first for speed. Translating a prefix is, like prefilling, compute-bound, so its wall-clock cost is set by the translator's FLOPs, and a linear map translates faster than an MLP while nearly matching its performance (Figures 2(a) and 3). Linearity brings two further benefits: the first training stage has a closed-form solution (§3.2), and choosing the source layers from data costs only ridge solves (greedy match, below).

Keys and values apart. T_K^{ℓ} acts only on keys, and T_V^{ℓ} only on values: this halves the parameter count against a joint map over $[K; V]$ at negligible cost on performance (Appendix W).

Head mixing. We propose two variants of KV-Lingo: a *head-wise* version where target head h reads only source head h, so $T_K^{\ell' \to \ell}$ and $T_V^{\ell' \to \ell}$ are block-diagonal; and a *head-mixing* one which leaves the matrices unconstrained. Head-mixing stores $2\,|\mathcal{E}_\ell|\, d_{\mathrm{tgt}} d_{\mathrm{src}}$ parameters per target layer; head-wise has H times fewer, making it faster but less expressive.

Rotation and normalisation. Translators can act *post-RoPE* (on rotated keys), *post-norm* (after key normalisation, before RoPE) or *pre-norm* (before both); in the last two cases the translated keys are normalised (pre-norm only) and rotated before being stored. Post-norm is worse than pre-norm on long context (Table 27). Pre-norm and post-RoPE match on short prefixes ($\lesssim$ 2k tokens), but pre-norm is slightly better on longer ones (Table 28) and extrapolates better beyond training lengths (Figure 35). We therefore use pre-norm in most experiments, although post-RoPE translation has a significant advantage in the memory-bound setting of Appendix Z.

Source-layer selection. We call a pair *shape-preserving* when its two caches have the same number of layers, where each target layer reads its counterpart, $\mathcal{E}_\ell = \{\ell\}$, and *cross-layer* when depth differs, so that an assignment $\mathcal{E}_\ell$ must be chosen. Every target layer then reads the same number ν of source layers. *Relative-depth match* takes the band of ν consecutive source layers centred on the relative-depth counterpart $c(\ell) \coloneqq 1 + (\ell - 1)(N_{\mathrm{src}} - 1)/(N_{\mathrm{tgt}} - 1)$ of target layer $\ell \in \{1, \dots, N_{\mathrm{tgt}}\}$. The window is shifted at the boundaries, so $|\mathcal{E}_\ell| = \nu$ for every target layer. Figure 17 illustrates this method. *Greedy match* instead picks $\mathcal{E}_\ell$ from data, separately for each target layer, by forward selection over source layers on the residual of a ridge fit to that layer, and the R^2 *assignment* keeps the ν source layers whose single-layer fits score best. A larger ν trades speed for quality (Appendix H).

3.2 TRAINING THE TRANSLATOR

Training data are texts split into a prefix $x_{1:n}$, whose cache is translated, and a continuation $y_{1:m}$. Only the matrices $\{T_K^{\ell}, T_V^{\ell}\}_\ell$ are trained, in two stages. For ease of exposition we consider $\mathcal{E}_\ell = \{\ell\}$, the extension to more involved assignments is straightforward.

Stage 1: closed-form initialisation on the prefixes. Each map is initialised to the minimiser of the cache-reconstruction Frobenius error

$$T_K^{\ell} \;=\; \arg\min_{M}\; \mathbb{E}_{x_{1:n}} \left\| M K^{\ell}_{\mathrm{src}}(x_{1:n}) - K^{\ell}_{\mathrm{tgt}}(x_{1:n}) \right\|_F^2, \tag{3}$$

and likewise for T_V^ℓ. It is solved in closed form on the fitting corpus, without gradient steps: the normal equations need only the second moments $\mathbb{E}[K_{\text{src}} K_{\text{src}}^\top]$ and $\mathbb{E}[K_{\text{src}} K_{\text{tgt}}^\top]$ per layer, and are solved through a pseudo-inverse for numerical stability.

Stage 2: self-distillation on the continuation. As stated in equation 1, the ultimate goal of translation is to match the distribution of continuations one would obtain with the native cache. This directly translates into a distillation objective:

$$\mathcal{L}(T) \;=\; \mathbb{E}_{(x_{1:n},\, y_{1:m})}\, \frac{1}{m} \sum_{j=1}^{m} \mathrm{KL}\Big(p_{\text{tgt}}\big(\cdot \mid \mathcal{C}_{\text{tgt}}(x_{1:n}),\, y_{<j}\big) \;\Big\|\; p_{\text{tgt}}\big(\cdot \mid T(\mathcal{C}_{\text{src}}(x_{1:n})),\, y_{<j}\big)\Big), \quad (4)$$

the divergence between the target's predictions from its *own* cache and from the translated one. We minimise $\mathcal{L}$ using Adam, starting from the stage-1 initialisation. As usual in distillation, the supervision comes from the target; the examples only fix the positions at which the two distributions are compared. As an ablation, we replace it with a cross-entropy loss, which we find more brittle at long context (Appendix U). Heo et al. (2026) and Qu et al. (2026) use a Frobenius loss similar to equation 3 to train their translators, with architectures different from KV-Lingo's and without stage 2 (Appendix A). In our experiments, we find that stage 2 training yields vastly superior performance. In Appendix O we find a reweighted norm in equation 3 which empirically gives better initialisations for stage 2, but these gains vanish when training stage 2 long enough.

4 Experiments

4.1 Setup and baselines

We train one translator per model pair on a single multi-turn chat dataset, Nemotron-SFT-Instruction-Following-Chat-v2 (NVIDIA et al., 2025), disjoint from every evaluation benchmark we consider below. Each sample is a (prefix, continuation) pair split at the start of an assistant turn, where a switch would happen, with reasoning-on and reasoning-off samples alternating; prefixes range from $n = 10$ to 16k tokens and continuations are capped at $m = 2048$. The *same* translator serves every task and language we evaluate. Appendix C details the training mixture, Appendix E the training procedure.

Evaluation datasets. We cover six abilities: *language modelling*, by decoding on held-out training conversations; *knowledge and reasoning*, with MMLU-Pro (Wang et al., 2024), ARC-Challenge (Clark et al., 2018) and GSM8K (Cobbe et al., 2021); *instruction following*, with IFEval (Zhou et al., 2023); *context-grounded QA*, with RepLiQA (Monteiro et al., 2024), whose fictional documents rule out answers from parametric memory, CoQA (Reddy et al., 2019) for the multi-turn case and, in 11 languages, XQuAD (Artetxe et al., 2020); *multi-turn chat*, judged on MT-Bench-101 (Bai et al., 2024); and *long context*, with RULER (Hsieh et al., 2024), LongMemEval (Wu et al., 2025a) and LongBench-v2 (Bai et al., 2025). Judged benchmarks use a local Qwen3.6-27B judge, and several are run both reasoning-on and reasoning-off. Table 4 gives the prefix length range for these tasks, from a few hundred tokens to 32k.

Baselines. We compare KV-Lingo to three baselines.

Head-wise linear closed-form translator (Heo et al., 2026): the head-wise stage-1 solution of equation 3 used on its own, without stage 2.

Head-wise MLP translator: the architecture of Chen et al. (2026), a head-wise two-layer MLP, trained with our recipe (Appendix E): stage 1 with Adam, as no closed form exists, and stage 2 by self-distillation, which we find stronger than the cross-entropy they use (Appendix U).

DroidSpeak (Liu et al., 2024a): the source cache is reused unchanged wherever the two models agree closely enough, and re-prefilled at the layers where they diverge most. It requires same-shape caches, and its cost grows with the fraction of layers it recomputes. Its results are in Appendix M.

We compare only against methods whose translator costs less than the re-prefill it replaces (§2.1). This excludes cache-fusion methods such as Cache-to-Cache (Fu et al., 2026) and the transformer adapters of Dery et al. (2026), where translating is no cheaper than prefilling.

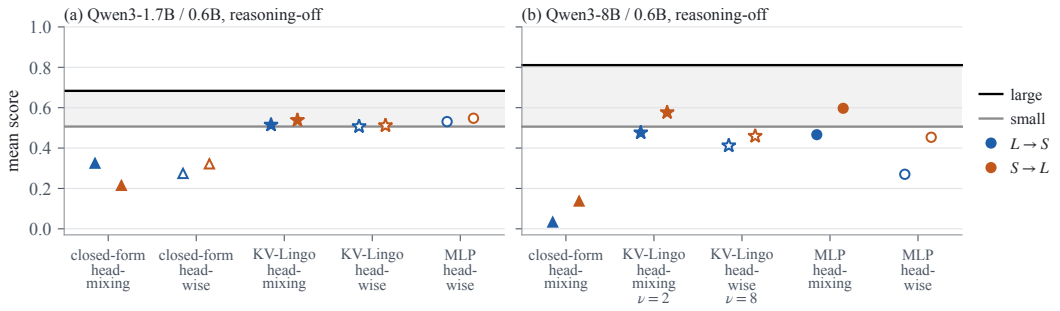


Figure 2: **Cache translation scores**, reasoning-off, mean of MMLU-Pro, ARC-C, RULER, LongMemEval, CoQA, IFEval, XQuAD, RepLiQA and MT-Bench-101. *(a) Shape-preserving*, Qwen3-0.6B/1.7B: a distilled head-wise linear translator suffices. *(b) Cross-layer*, Qwen3-0.6B/8B; head-mixing KV-Lingo at $\nu = 2$ uses the R^2 assignment. $\nu = 1$ for MLPs.

4.2 Shape-preserving cache translation

We first evaluate cache translation in the setting where the source and the target have the same cache shape. This is the case for two Qwen3 model pairs, both with $H = 8$ and $d_h = 128$: the 0.6B/1.7B pair at $N = 28$ layers and the 4B/8B pair at $N = 36$. Figure 2(a) reports the score of KV-Lingo translators and baselines when translating for the 0.6B/1.7B pair, averaged over nine benchmarks, and evaluated reasoning-off. We observe that distilled translators significantly outperform closed-form ones, and that our linear KV-Lingo translators match the score of the head-wise MLP of Chen et al. (2026), reaching the level of the small model in both directions. Reasoning-on, the score of $S \to L$ translation rises from the level of the small model to close to that of the large model (Figure 13). Moreover, in $L \to S$, when the task is easy enough that source and target have similar scores, translation performs at the level of the large model (Figure 10), which makes KV-Lingo translators suited to the agentic hand-off setting described in §2.2. Replacing the head-wise MLP with a head-mixing one does not change performance (Figure 12). Head-wise translators suffice here because the heads of these released pairs are aligned one to one, which is specific to Qwen3's training recipe (Appendix G). In addition, the left plot of Figure 3 shows that KV-Lingo translators are significantly more efficient than re-prefilling for long prefixes, and faster than MLPs, while almost preserving the performance of the target model on these shape-preserving pairs. Appendix F reports a more complete set of evaluations.

The impact of reasoning. We evaluate our KV-Lingo translators both reasoning-off and reasoning-on. In $L \to S$, adding reasoning barely changes the score gap between KV-Lingo and the target model, while in $S \to L$ reasoning can significantly increase the score of translation (Figure 31). Hence, the score drop caused by handing a slightly degraded cache to a large model can be mitigated by making the model reason. Moreover, the performance of closed-form translators is significantly degraded reasoning-on, contrary to KV-Lingo translators (Figure 13).

Ablations. We test two other evaluation frameworks: (i) only translating the context, while the prompt is prefilled by the target model, showing in Appendix T that KV-Lingo translators are robust to that change of framework (Table 29), and (ii) translating the prefix and the reasoning trace of the large model before handing it to the small model, which significantly boosts the performance of the small model (Table 30). We also demonstrate the robustness of KV-Lingo translators to out-of-distribution languages, showing in Figure 37 that performance is preserved across all the languages of XQuAD (Artetxe et al., 2020). Finally, we test cache translation on a model pair pretrained from scratch with the architectures of Qwen3-0.6B and 1.7B, showing that the performance of KV-Lingo is preserved in that setting (Appendix V).

4.3 Cross-layer cache translation

Qwen3 pairs. We translate between Qwen3-0.6B/8B and Qwen3-1.7B/4B (28 and 36 layers) in both directions, with the relative-depth, greedy and R^2 assignments of §3.1. On the 0.6B/8B pair, reasoning-off and averaged over nine benchmarks (Figure 2(b)), head-mixing KV-Lingo with 2

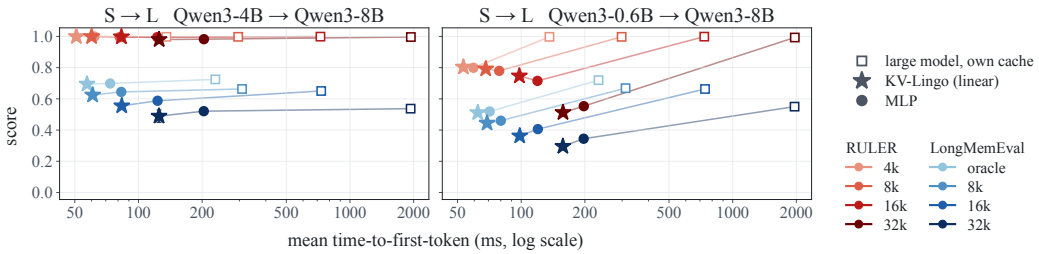


Figure 3: **Score against time to first token, long context,** $S \to L$**.** Each colour is one long-context evaluation task. Open squares: Qwen3-8B decoding from its own cache, timed as one full prefill. Filled markers: Qwen3-8B reading a translated cache, timed as the translation of an already computed source cache plus one decoding step (Appendix Y). *Left*: shape-preserving pair. *Right*: cross-layer pair at $\nu = 2$. Throughput is the same for all methods.

source layers and the R^2 assignment is almost on par with a head-mixing MLP in both directions (0.476 against 0.466 in $L \to S$, 0.577 against 0.597 in $S \to L$), while translating faster (Figure 3, right). Both stay just below the small model's 0.506 in $L \to S$ and clear it by 0.07–0.09 in $S \to L$, whereas the head-wise MLP falls well below it in $L \to S$ (0.270) and stays below it in $S \to L$ (0.454). Both MLPs read $\nu = 1$ source layer per target layer, with the greedy assignment for the head-mixing MLP and the relative-depth one for the head-wise MLP. Mixing heads and adding source layers both improve the scores (Figure 20), notably long-context retrieval (Figure 23), at a cost proportional to ν (Figure 22). The three assignments perform similarly (Figures 18 and 19). On the unbalanced 0.6B/8B pair, translation can land well above the small model: reasoning-on, $S \to L$, on LongMemEval, the 8B decoding from a translated 0.6B cache scores 0.531 (R^2 assignment) against the 0.6B's own 0.383 on the same items. Appendix H provides additional evaluations.

Gemma pairs. We also test two Gemma pairs that differ in depth and in the number of key-value heads, with head dimension 256 throughout. Gemma-3-4B-it and Gemma-3-12B-it (Team et al., 2025) have 34 and 48 layers, one global attention layer every 5 local ones, and 4 and 8 K/V heads, so that no head-wise method applies to either pair, and KV-Lingo translators are rectangular matrices throughout. Gemma-4-E2B-it and Gemma-4-E4B-it (Team et al., 2026) have 35 and 42 layers, 4:1 and 5:1 local/global ratios, and 1 and 2 K/V heads; their last 20 and 18 layers reuse the caches of earlier layers. Source layers are always mapped to target layers of the same attention type (local or global), with $\nu = 1$ for Gemma-4 and $\nu \in \{1, 2, 4\}$ for Gemma-3. On the benchmark means of Figures 24 and 25, KV-Lingo outperforms the small model in $L \to S$ (for Gemma-3, with greedy match or at $\nu = 4$) but in $S \to L$ it only reaches the small model's level (within one standard error, for Gemma-3 at $\nu = 4$) rather than the large model's; Tables 14, 15 and 16 give the per-benchmark scores.

4.4 Cross-tokenizer cache translation

Everything so far assumed the two models segment the prefix identically. We drop that assumption with the model pair Qwen3-8B ↔ Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), where the source and the target use different tokenizers, so that their representations of a given prefix can have a different number of tokens. The models also have different depths (36 layers for Qwen3-8B, against 32 layers for Mistral-7B): we use the relative-depth assignment to determine the source-target layer mapping, with 3 source layers per target layer. We describe in Appendix K our procedure to apply KV-Lingo to that setting. Table 17 reports both directions on seven benchmarks; the average scores are in Figure 27. Even with a change of tokenizer, KV-Lingo translators outperform the small model on two benchmarks in $L \to S$ and on four in $S \to L$.

4.5 MLA cache translation

We test KV-Lingo with multi-head latent attention (MLA), applying it as-is to *latent cache translation* instead of KV-cache translation. As a proof of concept, and since existing MLA pairs that share a tokenizer have 100B–1T parameters (Kimi Team, 2025b;a; DeepSeek-AI, 2024b;a), we convert the

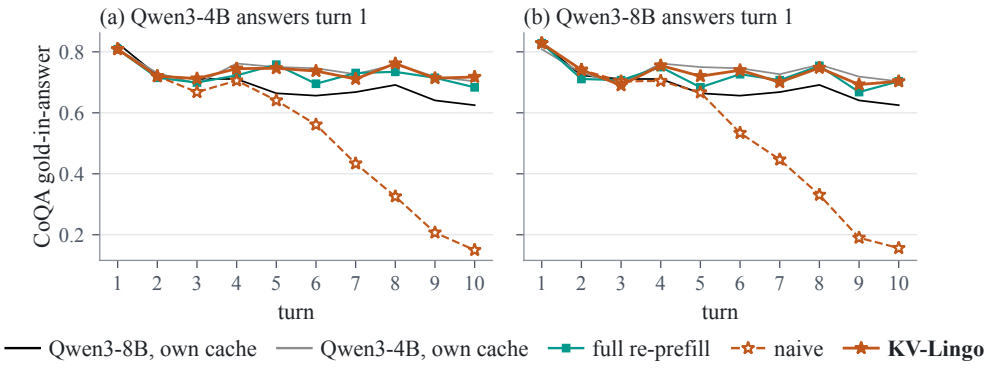


Figure 4: **Multi-turn model switching on CoQA.** Qwen3-4B ↔ Qwen3-8B alternate over 10 CoQA turns, reasoning off; panels differ in which model answers turn 1. Both strategies use the same translators. The naive strategy collapses; KV-Lingo stays level with full re-prefill through turn 10. Translated conditions: mean over three seeds; error bars: standard error over evaluation items (Appendix E.1).

GQA models Qwen3-4B and Qwen3-8B into MLA models of similar architecture, with the conversion recipe CARE (Zhou et al., 2026) (see Appendix L for the experimental protocol). KV-Lingo works off-the-shelf in that setting: the source and target latent caches have the same shape: 36 layers $\times$ 1024 latent dimensions, and each source layer ℓ is mapped to target layer ℓ, in parallel over tokens, with one translator per layer ℓ, separately for the latent vectors c_{KV} and the decoupled RoPE vectors k_R: $\hat{c}^{\mathrm{tgt},\ell}_{KV,i} = T^{\ell}_{KV}\, c^{\mathrm{src},\ell}_{KV,i}$ and $\hat{k}^{\mathrm{tgt},\ell}_{R,i} = T^{\ell}_{R}\, k^{\mathrm{src},\ell}_{R,i}$, for $i = 1, \ldots, n-1$. Figure 28 and Table 18 report the results. Averaged over six benchmarks, KV-Lingo lands about 0.04 below both native models (0.588 and 0.587 against 0.629 and 0.633): latent caches translate as-is, at some cost in quality.

4.6 Dense-to-MoE cache translation

We test the scenario of §2.2 by prefilling with a dense Qwen3-4B, translating its cache, and generating with Qwen3-30B-A3B (Yang et al., 2025a), an MoE that activates 3.3B of its 30.5B parameters per token. Each of the MoE's 48 layers reads $\nu = 3$ of the dense model's 36, assigned by relative depth. This scenario is of particular interest when serving at batch size 1, for instance on device. The added capacity of the MoE is costly at prefill time since it requires fetching most or even all the experts' weights in parallel. Concretely, on an Apple M3 Ultra (`bf16`, batch size 1), the MoE decodes as fast as the dense model but takes 152 ms to prefill a 64-token prompt, against 68 ms, and translation adds only 1.8 ms (see Figure 5). KV-Lingo thus acts as a prefill accelerator for the MoE. Figure 29 reports quality per task category (full results in Table 22). On the two judged short-context benchmarks (RepLiQA and MT-Bench-101), the MoE decoding from the translated cache comes within about 0.01 of its own cache, with reasoning on or off. On the six other short-context benchmarks it lands between the two models with reasoning on, and below the dense model with reasoning off. Translation fails on long context, mostly on RULER; on the dense cross-layer pairs, more source layers per target layer buy back part of this recall (§4.3). In that case, non-linearity does not help: an MLP in place of the linear map reaches a higher KL loss (Table 23).

4.7 Multi-turn cache translation

The experiments of §4.2 and §4.3 involve a single model switch. For multiple switches, we propose to use KV-Lingo with *alternated translation and generation*: each model keeps its own cache of the prefix, mixing native and translated segments (Figure 32). At a switch, only the tokens generated since the previous switch are translated. We compare this method with a naive baseline which, at every switch, translates the full prefix: the context and earlier generations are then translated back and forth, once per switch, and the cost of each switch grows with the prefix length.

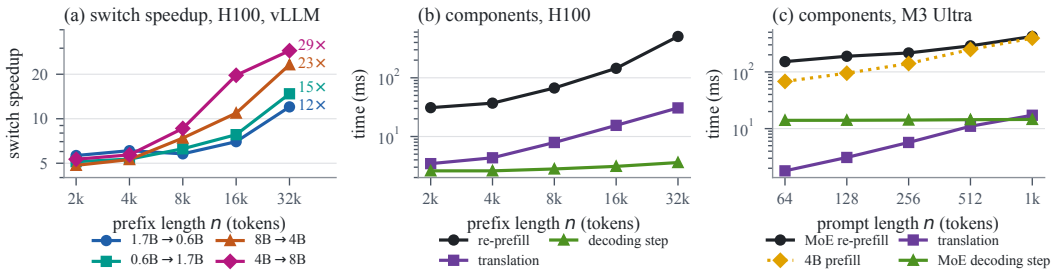


Figure 5: **Switch cost of KV-lingo.** (a) Speedup of the switch on an H100, the target served by vLLM: its time to first token after re-prefilling, divided by that after translation plus one decoding step. (b) The terms of that ratio for the switch into Qwen3-1.7B. (c) The same terms on an Apple M3 Ultra (MLX), from Qwen3-4B into Qwen3-30B-A3B, plus the 4B's own prefill.

On CoQA (Reddy et al., 2019), QuAC (Choi et al., 2018) and MultiChallenge (Sirdeshmukh et al., 2025), the naive method holds for a few turns and then drops sharply, while alternated translation and generation stays close to re-prefilling across all turns (within 0.05 on QuAC) (Figures 4, 33 and 34).

4.8 Computational gains of cache translation

At a switch, re-prefilling is a forward pass of the target on the n prefix tokens. Translation replaces it with two much cheaper operations. The translator is a token-wise linear map: on our shape-preserving Qwen3 pairs it costs 7.5 to 92 times fewer FLOPs per token than applying the target's own weight matrices, and it computes no quadratic attention. The target then runs a single decoding step over the current token, whose cost depends on n only mildly, through reading the cache. The time to first token thus goes from $t_{\text{prefill}}(n)$ to $t_T(n) + t_{\text{step}}(n)$.

Figure 5 times both re-prefilling, translation and one decoding step; Appendix Y details the protocol and reports every timing. On an H100 with the target served by vLLM (Kwon et al., 2023), the switch reaches the first token 4.8 to 5.6× sooner already at 2k tokens, and 12 to 29× sooner at 32k (Figure 5a). Panel (b) shows why the gap widens with n: re-prefill grows quadratically, translation linearly, and the decoding step, which only reads the cache, stays at 2.6 to 3.6 ms. At long context, translation is thus most of the switch's cost. Since it depends only on the cache width, which all four Qwen3 models share, while the prefill grows with the target's size, larger targets gain more: 12.0× into Qwen3-0.6B against 28.9× into Qwen3-8B at 32k. On a device that decodes one request at a time, the gain does not even need long contexts, since an MoE's prefill must load most of its experts. On an Apple M3 Ultra with MLX (Hannun et al., 2023), Qwen3-30B-A3B prefills a 64-token prompt in 152 ms, whereas translating the cache of Qwen3-4B and one decoding step of the MoE take 15.8 ms, so the MoE answers 9.6× sooner (Figure 5c, §4.6). After the switch, the target decodes at its native speed, since the translated cache has the target's own shape.

5 Conclusion

KV-Lingo shows that KV caches can be translated between frozen language models using simple token-wise linear maps trained with self-distillation. The translated caches retain strong downstream performance while avoiding the target model's re-prefill, substantially reducing the latency of model switching. Cache translation therefore provides a simple primitive for efficient dynamic routing and hand-off between language models.

AI use statement

In this work, we used generative AI tools to draft sections of the paper and to aid and polish its writing, for retrieval and discovery of related work, and for research execution: cleaning and reformatting the training and evaluation data, implementing the method and its evaluation and plotting code, interpreting experimental results, and designing the experiments that followed. We have not used generative AI tools to generate synthetic data sets, to develop the conceptual framework, or to propose

the hypotheses the paper tests: the cache-translation problem, the translator family and the finding that linear maps suffice are the authors'. Proving mathematical claims and qualitative or thematic data analysis are not applicable to this work. We have reviewed all AI-assisted work. The authors read and revised every AI-drafted passage; every bibliography entry was copied by an author from the published record rather than generated, and every AI-suggested reference was checked against the cited paper. AI-written code was checked through its outputs rather than line by line: native and translated caches are scored by the same harness on the same items, the three-seed runs are compared against their evaluation noise (Appendix E.1), and every reported number is read from logged runs (W&B and JSON records) rather than produced by an AI tool. Separately, an LLM (Qwen3.6-27B) serves as the judge in several evaluations (§4.1), as a measurement instrument, not an assistant. We take responsibility for the final content of this work, including text, claims or artefacts produced with the aid of generative AI.

Reproducibility statement

Every model we translate between is a public checkpoint, except the from-scratch and MLA pairs, whose construction we describe, and every dataset we train or evaluate on is public. The translator family and its two training stages are defined in §3.1 and §3.2; the greedy-match and R^2 layer assignments are specified in Appendix H, and the token correspondence for the cross-tokenizer pair in Appendix K. Appendix C describes how the training mixture is built from Nemotron-SFT-Instruction-Following-Chat-v2, down to the chat-template rendering of a sample and the placement of reasoning traces. Appendix E lists the exact checkpoints, the hardware and numerical precision, the hyperparameters of both training stages and of the MLP baselines, and the learning rate selected for every translator (Table 5). Appendix D gives, for every benchmark, the prefix lengths, number of items, reasoning mode, generation budget and metric, together with the decoding settings, the judge models and the judging rubrics, and Appendix E.1 the seeds and the definition of every error bar. The pretraining recipe of the from-scratch pair is in Appendix V, the MLA conversion in Appendix L, and the measurement protocol behind every reported speed-up in Appendix Y.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4623–4637, 2020.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7421–7454, 2024.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=M2UzLRoqic.

Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring linear features across language models with model stitching. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Qvvy0X63Fv.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=cSimKw5p6R. Featured Certification.

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

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac : Question answering in context. *CoRR*, abs/1808.07036, 2018. URL http://arxiv.org/abs/1808.07036.

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, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Adrián Csiszárik, Péter Kőrösi-Szabó, Akos Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. *Advances in Neural Information Processing Systems*, 34:5656–5668, 2021.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a. URL https://arxiv.org/abs/2405.04434. DeepSeek-V2-Lite is described in Appendix B.

DeepSeek-AI. DeepSeek-V2-Lite. https://huggingface.co/deepseek-ai/DeepSeek-V2-Lite, 2024b. Model card, Hugging Face.

Lucio M Dery, Zohar Yahav, Henry Prior, Qixuan Feng, Jiajun Shen, and Arthur Szlam. Latent space communication via kv cache alignment. *arXiv preprint arXiv:2601.06123*, 2026.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers, 2026. URL https://arxiv.org/abs/2505.01618.

Sabri Eyuboglu, Ryan Saul Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Ruoyu Liu, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0k5w8O0SNg.

Anastasiia Filippova, David Grangier, Marco Cuturi, and Joao Monteiro. Stochastic kv routing: Enabling adaptive depth-wise cache sharing. *arXiv preprint arXiv:2604.22782*, 2026.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. *Advances in Neural Information Processing Systems*, 38:124108–124145, 2025.

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

Yoav Gelberg, Yam Eitan, Michael Bronstein, Yarin Gal, and Haggai Maron. Training transformers for kv cache compressibility. *arXiv preprint arXiv:2605.05971*, 2026.

Yingsheng Geng, Yuchong Gao, Weihong Wu, Guyue Liu, and Jiang Liu. Relaycaching: Accelerating llm collaboration via decoding kv cache reuse, 2026. URL https://arxiv.org/abs/2603.13289.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.

Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. MLX: Efficient and flexible machine learning on apple silicon, 2023. URL https://github.com/ml-explore.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Maxwell Horton, Qingqing Cao, Chenfan Sun, Yanzi Jin, Sachin Mehta, Mohammad Rastegari, and Moin Nabi. Kv prediction for improved time to first token. *arXiv preprint arXiv:2410.08391*, 2024.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=BH8TYy0r6u.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=JFygzwx8SJ.

Junhyuck Kim, Jongho Park, Jaewoong Cho, and Dimitris Papailiopoulos. Lexico: Extreme KV cache compression via sparse coding over universal dictionaries. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=Yh9vxlxnjA.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025a. URL https://arxiv.org/abs/2507.20534.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025b. URL https://arxiv.org/abs/2510.26692.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Andrew Lee, Melanie Weber, Fernanda Viégas, and Martin Wattenberg. Shared global and local geometry of language model embeddings. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=aJDykpJAYF.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

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

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. Matryoshkakv: Adaptive kv compression via trainable orthogonal projection. In *International Conference on Learning Representations*, volume 2025, pp. 86669–86690, 2025.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, et al. Droidspeak: Kv cache sharing for cross-llm communication and multi-llm serving. *arXiv preprint arXiv:2411.02820*, 2024a.

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 *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=L057s2Rq8O.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Joao Monteiro, Pierre-Andre Noel, Étienne Marcotte, Sai Rajeswar, Valentina Zantedeschi, David Vazquez, Nicolas Chapados, Christopher Pal, and Perouz Taslakian. RepliQA: A question-answering dataset for benchmarking LLMs on unseen reference content. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=4diKTLmg2y.

João Monteiro, Michal Klein, Pierre Ablin, and Marco Cuturi. Nectar: Neural estimation of cached-token attention via regression. *arXiv preprint arXiv:2605.09778*, 2026.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SrC-nwieGJ.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=tDRYrAkOB7.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KJNcAkY8tY4.

NVIDIA, :, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Kondratenko, Alexander Bukharin, Alexandre Milesi, Ali Taghibakhshi, Alisa Liu, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amir Klein, Amit Zuker, Amnon Geifman, Amy Shen, Anahita Bhiwandiwalla, Andrew Tao, Ann Guan, Anubhav Mandarwal, Arham Mehta, Ashwath Aithal, Ashwin Poojary, Asif Ahamed, Asma Kuriparambil Thekkumpate, Ayush Dattagupta, Banghua Zhu, Bardiya Sadeghi, Barnaby Simkin, Ben Lanir, Benedikt Schifferer, Besmira Nushi, Bilal Kartal, Bita Darvish Rouhani, Boris Ginsburg, Brandon Norick, Brandon Soubasis, Branislav Kisacanin, Brian Yu, Bryan Catanzaro, Carlo del Mundo, Chantal Hwang, Charles Wang, Cheng-Ping Hsieh, Chenghao Zhang, Chenhan Yu, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christopher Parisien, Collin Neale, Damon Mosk-Aoyama, Dan Su, Dane Corneil, Daniel Afrimi, Daniel Rohrer, Daniel Serebrenik, Daria Gitman, Daria Levy, Darko Stosic, David Mosallanezhad, Deepak Narayanan, Dhruv Nathawani, Dima Rekesh, Dina Yared, Divyanshu Kakwani, Dong Ahn, Duncan Riach, Dusan Stosic, Edgar Minasyan, Edward Lin, Eileen Long, Eileen Peters Long, Elena Lantz, Ellie Evans, Elliott Ning, Eric Chung, Eric Harper, Eric Tramel, Erick Galinkin, Erik Pounds, Evan Briones, Evelina Bakhturina, Faisal Ladhak, Fay Wang, Fei Jia, Felipe Soares, Feng Chen, Ferenc Galko, Frankie Siino, Gal Hubara Agam, Ganesh Ajjanagadde, Gantavya Bhatt, Gargi Prasad, George Armstrong, Gerald Shen, Gorkem Batmaz, Grigor Nalbandyan, Haifeng Qian, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Himanshu Soni, Hiren Upadhyay, Huizi Mao, Huy C Nguyen, Huy Q Nguyen, Iain Cunningham, Ido Shahaf, Igor Gitman, Ilya Loshchilov, Ivan Moshkov, Izzy Putterman, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jatin Mitra, Jeffrey Glick, Jenny Chen, Jesse Oliver, Jian Zhang, Jiaqi Zeng, Jie Lou, Jimmy Zhang, Jining Huang, Joey Conway, Joey Guman, John Kamalu, Johnny Greco, Jonathan Cohen, Joseph Jennings, Joyjit Daw, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kai Xu, Kan Zhu, Kari Briski, Katherine Cheung, Katherine Luna, Keshav Santhanam, Kevin Shih, Kezhi Kong, Khushi Bhardwaj, Krishna C. Puvvada, Krzysztof Pawelec, Kumar Anik, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Li Ding, Lucas Liebenwein, Luis Vega, Maanu Grover, Maarten Van Segbroeck, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Manoj Kilaru, Maor Ashkenazi, Marc Romeijn, Mark Cai, Markus Kliegl, Maryam Moosaei, Matvei Novikov, Mehrzad Samadi, Melissa Corpuz, Mengru Wang, Meredith Price, Michael Boone, Michael Evans, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Nabin Mulepati, Natalie Hereth, Nave Assaf, Negar Habibi, Neta Zmora, Netanel Haber, Nicola Sessions, Nidhi Bhatia, Nikhil Jukar, Nikki Pope, Nikolai Ludwig, Nima Tajbakhsh, Nirmal Juluru, Oleksii Hrinchuk, Oleksii Kuchaiev, Olivier Delalleau, Oluwatobi Olabiyi, Omer Ullman Argov, Ouye Xie, Parth Chadha, Pasha Shamis, Pavlo Molchanov, Pawel Morkisz, Peter Dykas, Peter Jin, Pinky Xu, Piotr Januszewski, Pranav Prashant Thombre, Prasoon Varshney, Pritam Gundecha, Qing Miao, Rabeeh Karimi Mahabadi, Ran El-Yaniv, Ran Zilberstein, Rasoul Shafipour, Rich Harang, Rick Izzo, Rima Shahbazyan, Rishabh Garg, Ritika Borkar, Ritu Gala, Riyad Islam, Roger Waleffe, Rohit Watve, Roi Koren, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Ryan Timbrook, Sadegh Mahdavi, Sahil Modi, Samuel Kriman, Sanjay Kariyappa, Sanjeev Satheesh, Saori Kaji, Satish Pasumarthi, Sean Narentharen, Sean Narenthiran, Seonmyeong Bak, Sergey Kashirsky, Seth Poulos, Shahar Mor, Shanmugam Ramasamy, Shantanu Acharya, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shiqing Fan, Shreya Gopal, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Simeng Sun, Smita Ithape, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Stephen Ge, Sugam Dipak Devare, Sumeet Kumar Barua, Suseella Panguluri, Suyog Gupta, Sweta Priyadarshi, Syeda Nahida Akter, Tan Bui, Teodor-Dumitru Ene, Terry Kong, Thanh Do, Tijmen Blankevoort, Tom Balough, Tomer Asida, Tomer Bar Natan, Tugrul Konuk, Twinkle Vashishth, Udi Karpas, Ushnish De, Vahid Noorozi, Vahid Noroozi, Venkat Srinivasan, Venmugil Elango, Vijay Korthikanti, Vitaly Kurin, Vitaly Lavrukhin, Wanli Jiang, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenfei Zhou, Will Jennings, William Zhang, Wojciech Prazuch, Xiaowei Ren, Yashaswi Karnati, Yejin Choi, Yev Meyer, Yi-Fu Wu, Yian Zhang, Ying Lin, Yonatan Geifman, Yonggan Fu, Yoshi Subara, Yoshi Suhara, Yubo Gao, Zach Moshe, Zhen Dong, Zihan Liu, Zijia Chen, and Zijie Yan. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2025. URL https://arxiv.org/abs/2512.20848.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete

Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

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

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

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pp. 606–610. IEEE, 2007.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12974–12990, 2024.

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

Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms, 2025. URL https://arxiv.org/abs/2501.17399.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, 2025.

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

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=y10DM6R2r3.

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

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=pZiyCaVuti.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

You Wu, Haoyi Wu, and Kewei Tu. A systematic study of cross-layer kv sharing for efficient llm inference. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 396–403, 2025b.

Zhoutong Wu, Yuan Zhang, Yiming Dong, Chenheng Zhang, Cong Fang, Kun Yuan, and Zhouchen Lin. Improving model representation and reducing KV cache via skip connections with first value heads. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c. URL https://openreview.net/forum?id=bjV8Y38aFF.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

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

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=oDcAGSXZZP.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024a.

Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. Kvsharer: Efficient inference via layer-wise dissimilar kv cache sharing. *arXiv preprint arXiv:2410.18517*, 2024b.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In *Proceedings of the twentieth European conference on computer systems*, pp. 94–109, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Yi Zhao, Yajuan Peng, Nguyen Cam-Tu, Zuchao Li, Wang Xiaoliang, hai zhao, and Xiaoming Fu. SmallKV: Small model assisted compensation of KV cache compression for efficient LLM inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0BVrpXMr5Y.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Zhongzhu Zhou, Fengxiang Bie, Ziyan Chen, Zhenyu Zhang, Yibo Yang, Junxiong Wang, Ben Athiwaratkun, Xiaoxia Wu, and Shuaiwen Leon Song. CARE: Covariance-aware and rank-enhanced decomposition for enabling multi-head latent attention. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DVurf4kGag.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast KV compaction via attention matching. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=t01SKTj8pP.

A Related work

Cross-model cache transfer. Work on making one cache serve two models spans four regimes. **First,** the cache is handed over as it is, recomputing the layers where two variants diverge (Liu et al., 2024a), sharing the cache-writing layers across siblings that differ only in a LoRA adapter above them (Woo et al., 2026), or transmitting only the layers that carry the most attention (Shi et al., 2026). Reuse of this kind needs the two models to share an architecture, and even then concedes recomputation, restricted model differences, or partial transfer. **Second,** one of the two models is trained for the purpose: Horton et al. (2024) fit per-layer linear maps from a small auxiliary model's cache into a base model's, fine-tuning the auxiliary through the frozen base to cut time to first token. The construction is close to ours; we differ in keeping the source off the shelf, in crossing families and tokenizers, and in fitting at lengths where a prefill is worth avoiding, where their prompts stay well under a thousand tokens. **Third,** a cache is combined with the target's own rather than replacing it: Cache-to-Cache (Fu et al., 2026) fuses the two, which beats either model alone but presumes the target has *already* prefilled the context, so the pair pays two prefills where a switch should pay one. **Fourth,** our setting, both models stay frozen and only a translator between them is fitted. Dery et al. (2026) map every model's cache into one shared latent space, a channel that also carries learned skills between models; each adapter is a transformer as deep as the model it serves, cross-attending over the whole cache, so it reads the prefix much as a prefill would. The instance closest to us is a token-wise two-layer MLP per target layer and key/value group, trained to reconstruct the target's cache and then with cross-entropy on generation (Chen et al., 2026), which we reimplement and match to our translator in parameters and in FLOPs (Appendix V). Its two phases mirror our two stages, but the first is trained rather than solved, the second fits the data rather than the target's own predictions, and the map is non-linear and confined to matching key/value groups where ours is a single matrix per layer and side that mixes heads freely. Neither targets the cost of a switch: one map is as expensive as a prefill, and the other is measured against communicating in text, on prompts of about 180 tokens, where the cost constraint that shapes our translator does not bite.

Concurrent work on cross-model cache transfer. Heo et al. (2026) independently study cross-model KV transfer for prefill reuse within a model family, from the same starting point as ours: the cross-model cache relation is largely linear, a ridge fit on a small calibration set recovers much of it, and stripping RoPE before fitting makes the map position-free. Four things separate the studies. **(i)** Their method is a closed-form fit, corresponding to our stage 1 equation 3, so their results confirm the initialiser is sound and our contribution is what the behavioural stage adds on top. **(ii)** Where a closed-form map degrades, they recover it with a nonlinear MLP; we measure the nonlinear headroom in the fit to be small (Table 1) and locate the missing quality in the objective instead, which is why our translator stays linear. **(iii)** Their pairs all sit inside one family, unequal depth included, where ours also cross a tokenizer boundary (§4.4), latent caches (Appendix L) and a pair pretrained from scratch. Their map is affine, where ours is strictly linear; Appendix B.2 finds a bias term buys nothing. **(iv)** The two studies count the cost differently. They compare their map's latency against the target's transformer body over the prefix, ignoring the forward pass the target must still run over the current token before it can emit anything; we charge that step on both sides and measure end to end (§4.8). Their maps also read $k = 8$ to all 26 source layers, since one layer leaves too much of the fit unexplained. That is paid at every switch: the source's cache must be kept for k layers rather than one, each map is k times wider, and the target's first token waits on k matrix products per layer. Ours reads a single source layer on every shape-preserving pair, and three only at a tokenizer boundary (§4.4).

Two follow-ups stay inside the closed-form regime. Qu et al. (2026) weight the residual by causal attention, match each target head to one source head, and invert the source's rotation before fitting. We sweep the first two: attention weighting cuts the off-the-shelf KL from 0.175 to 0.064 but leaves 0.0195 against 0.0186 after stage 2 (Appendix O), and head matching is our block-diagonal member (§3.1). Their gains are in building the map, not in applying it. Building happens once per pair and can take as long as it needs; applying it happens at every switch, and that is the cost our requirement constrains: cheap enough to beat a re-prefill, without giving up the target's quality. Li et al. (2026) frame the problem as we do but never state what their map is, how it is fitted, or where it acts, so we cite it rather than compare against it.

Representation alignment and stitching. Centered kernel alignment measures how far independently trained networks agree on their representations (Kornblith et al., 2019; Nguyen et al., 2021), and what relates them is often a simple transformation, in some cases a linear one (Li et al., 2015; Huh et al., 2024). Model stitching turns that measurement into an intervention: a learned map, classically linear, joins one network's lower layers to another's upper layers, and the hybrid's accuracy scores their compatibility (Lenc & Vedaldi, 2015; Bansal et al., 2021; Csiszárik et al., 2021). Encoding each point by its similarity to shared anchors avoids fitting the map at all (Moschella et al., 2023). In language models, affine maps between residual streams transfer trained components across scales (Chen et al., 2025), and a linear map on token embeddings carries steering vectors between models of different width (Lee et al., 2025). Our translator is fitted in the same spirit but is not a stitch: its output is not injected once into a forward pass for the later layers to absorb, it is the context the target attends over, at every layer and every decoded position.

Cache eviction, compression, and reuse in a single model. A large literature reduces or reuses the cache of a *single* model. Some methods keep less of it, dropping entries during the prompt or after it (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024; Zhao et al., 2025), or compacting them into a smaller cache, or a small network, that reproduces the model's attention (Kim et al., 2025a; Eyuboglu et al., 2026; Zweiger et al., 2026; Monteiro et al., 2026). Others store them more cheaply, by quantisation (Liu et al., 2024b; Hooper et al., 2024; Yang et al., 2024a) or by learning the compression itself (Nawrot et al., 2024; Lin et al., 2025; Kim et al., 2025b), or share them across layers (Brandon et al., 2024; Yang et al., 2024b; Wu et al., 2025b;c; Filippova et al., 2026). A third group reuses what is already stored, across requests (Kwon et al., 2023; Zheng et al., 2024) or beyond a shared prefix (Gim et al., 2024; Yao et al., 2025; Yang et al., 2025b). The model itself can also be trained so that these methods work better on it (Gelberg et al., 2026). All of this is orthogonal to translating between two *different* models, and composes with it in principle.

B MOTIVATING MEASUREMENTS: LAYER CORRESPONDENCE, LINEARITY AND RANK

Applying a translator T should cost less than the re-prefill it replaces (§2.1), which enforces some structure on the translator map; the simplest candidate is a single matrix per target layer, applied to a well-chosen source layer, in parallel over tokens. This appendix reports the measurements behind that choice: how well that structure is suited to the actual geometry of caches across models. Fix a layer of the source and of the target, and a prefix token i: we denote $K_{\mathrm{src},i} \in \mathbb{R}^{d_{\mathrm{src}}}$ the source's key and $K_{\mathrm{tgt},i} \in \mathbb{R}^{d_{\mathrm{tgt}}}$ the target's key. We ask to what extent a single translator map can predict every $K_{\mathrm{tgt},i}$ from $K_{\mathrm{src},i}$, which source layer to choose for each target layer, and whether the dependency between those vectors is approximately linear. Layer correspondence answers the second question (§B.1); linearity and effective rank answer the other two (§B.2).

B.1 LAYER CORRESPONDENCE ACROSS EVERY PAIR

We measure layer correspondence between the source and the target with linear centred kernel alignment (CKA) (Kornblith et al., 2019), computed between every source and target layer pair. On caches centred over tokens it is $\mathrm{CKA}(X, Y) = \|X^\top Y\|_F^2 / (\|X^\top X\|_F \, \|Y^\top Y\|_F)$. Figure 6 shows three of the per-layer CKA grids in full. Figure 7 summarises all of them, each grid reduced to two numbers: the gap between its diagonal and its off-diagonal mean, and how far the per-row argmax sits from the diagonal.

That summary splits the pairs into three regimes. The released shape-preserving Qwen3 pairs are sharply peaked on the diagonal, exceeding the off-diagonal mean by 0.30–0.34 with the per-row argmax exactly on it, on Base and Instruct checkpoints alike. The pairs that share no lineage or no shape keep about half that contrast: a 0.6B/1.7B pair we pretrained from scratch, sharing no checkpoint and no initialisation, sits at 0.17, the Gemma-3 pair (Team et al., 2025) at 0.15 and an unequal-depth Qwen3 pair at 0.16; only the last two are close to registered, the from-scratch pair's argmax drifting 2.5 layers. The randomly initialised controls, built on the same two architectures, reach 0.03 and 0.06 at offsets of 15.7 and 4.9 layers, which is what an absent correspondence looks like under this measurement.

A correspondence of layers by relative depth therefore appears during training, and is not limited to the Qwen3 architecture and training recipe. It is what licenses the identity layer assignment on the equal-depth pairs and a relative-depth window elsewhere (§3.1).

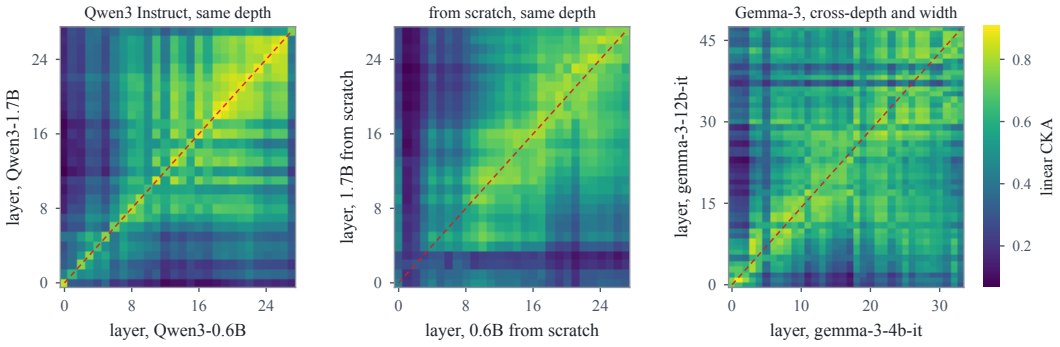


Figure 6: **Layer-to-layer cache correspondence.** Linear CKA between every layer of the large (rows) and small (columns) model, on post-k-norm keys over 16k tokens; shared colour scale, dashed line at equal relative depth. *Left:* pretrained Qwen3 Instruct pair of equal depth. *Middle:* 1.7B/0.6B pair pretrained from scratch, sharing no checkpoint. *Right:* pretrained Gemma-3 pair differing in depth and key/value width. Figure 7 summarises every other pair we measured, including two randomly initialised controls.

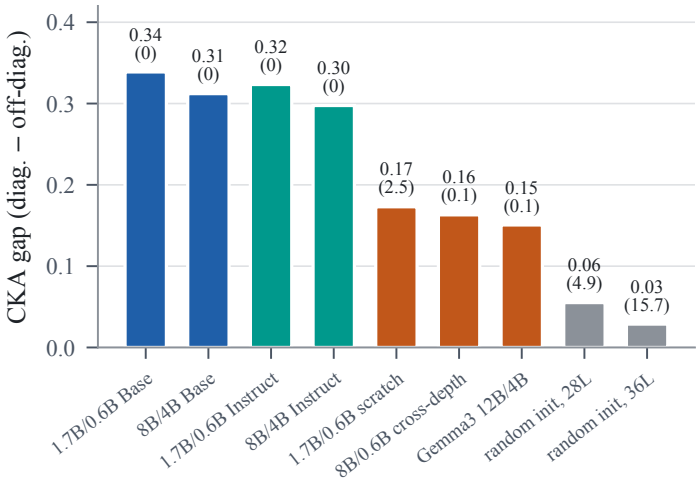


Figure 7: **Diagonal contrast of the per-layer CKA grid, every pair.** Unweighted linear CKA between source layer ℓ and target layer ℓ', summarised as the gap between the diagonal ($\ell = \ell'$) and the off-diagonal mean; the parenthesised number is the mean offset of the per-row argmax from the diagonal, in layers. The released equal-depth Qwen3 pairs, Base and Instruct alike, have a strong on-diagonal peak at offset 0, which licenses the identity assignment. The from-scratch, cross-layer and Gemma-3 pairs keep about half that contrast and, except for the from-scratch pair, sit near the relative-depth diagonal; they are the pairs for which the source layer is selected rather than assumed. Random initialisation gives essentially no structure.

B.2 Cache linearity and rank, per layer

Linear CKA says which layers have similar geometries: two point clouds have a linear CKA close to 1 if they approximately differ by an orthogonal transformation, which is a stronger condition than linear dependency. To measure how linearly related source and target layers are, we fit a no-bias least-squares map $K_{\mathrm{src},i} \mapsto T_K^{\ell} K_{\mathrm{src},i}$ per layer on held-out tokens and compute the fraction R^2_{lin} of the target's cache variance it explains, against the R^2_{nl} of a two-layer MLP on the same data. Table 1 reports both; Table 2 refits the linear map with an intercept, which is how the nonlinear headroom of Table 1 is attributed to curvature rather than to a mean shift; and Table 3 reports how much of its nominal width each cache actually occupies.

The linear map reaches 0.90–0.99 of what the MLP does ($\rho = R^2_{\mathrm{lin}}/R^2_{\mathrm{nl}}$, the MLP adding at most 0.073), so a nonlinear translator should buy little, which §4 confirms downstream. The unexplained residual is substantial: $1 - R^2_{\mathrm{nl}} \approx 0.2$–$0.4$, with values harder than keys and the from-scratch pair harder than the released checkpoints, orderings that recur downstream.

Caches also have low per-layer effective rank (Roy & Vetterli, 2007): 13–177 against $d = 1024$ (Table 3), with a single principal direction carrying up to half the variance in the first layers. Randomly initialised models of the same architecture behave very differently: effective ranks of 365–830, near-isotropic, top direction below 4%. Training therefore concentrates the cache onto a low-dimensional, strongly anisotropic subspace, which can reduce the number of samples needed to estimate the translator accurately along the directions that dominate the cache distribution.

Protocol. For a fixed pair, direction and capture point we forward both models over a shared Nemotron token set (512 documents of 256 tokens, so 131k aligned positions) and collect, for every layer and every prefix token, the aligned pair (x, y) = (source cache, target cache). The split is by *document*, $80/20$ into fit and test, so no scored position shares a document with a fitted one; the MLP's early-stopping set is a further tenth of the fitting tokens. R^2_{lin} is the held-out fit of the no-bias least-squares map of equation 3, with the per-dimension training mean as the $\mathrm{SS}_{\mathrm{tot}}$ baseline. It differs from the deployed closed-form initialisation in its conditioning: the diagnostic adds a small ridge to the normal equations (10^{-3} times the mean diagonal of the source Gram) where the deployed solve truncates the pseudo-inverse instead. Both act on the tail of the source spectrum and neither

Table 1: **Ablation: the impact of translating with an MLP instead of a linear map.** Per-layer linear fit of the target's cache from the source's, averaged over layers, on held-out tokens ($L \to S$; pre-normalisation capture point, which leaves values unchanged; $d = 1024$ per side). R^2_{lin} is the no-bias least-squares fit, the estimator of the closed-form translator; R^2_{nl} a two-layer MLP on the same data; $\rho = R^2_{\text{lin}}/R^2_{\text{nl}}$ the fraction of recoverable signal the linear map already captures. The $S \to L$ direction agrees to within 0.018 on keys and 0.037 on values.

Pair	side	R^2_{lin}	R^2_{nl}	nonlin. extra	ρ
Qwen3 8B/4B	K	0.815	0.825	0.010	0.989
	V	0.693	0.708	0.015	0.980
Qwen3 1.7B/0.6B	K	0.766	0.794	0.028	0.965
	V	0.658	0.697	0.039	0.945
from scratch 1.7B/0.6B	K	0.752	0.805	0.053	0.935
	V	0.651	0.724	0.073	0.903

Table 2: **Ablation: the impact of adding an intercept to the translators.** We compare the per-layer fit of linear translators with and without bias, at the pre-normalisation capture point. The two R^2 columns are layer averages on held-out tokens. On both Qwen3 pairs the bias is worth at most 0.001 of R^2 at any single layer, an order of magnitude below what the MLP finds (Table 1). The from-scratch pair has more room for an offset: its $S \to L$ keys gain 0.012 on average.

			R^2, layer average	
Pair	Dir.	side	no bias	affine
Qwen3 8B/4B	$L \to S$	K	0.815	0.815
		V	0.693	0.693
	$S \to L$	K	0.808	0.808
		V	0.676	0.676
Qwen3 1.7B/0.6B	$L \to S$	K	0.766	0.766
		V	0.658	0.658
	$S \to L$	K	0.753	0.754
		V	0.623	0.623
from scratch 1.7B/0.6B	$L \to S$	K	0.752	0.755
		V	0.651	0.651
	$S \to L$	K	0.757	0.769
		V	0.674	0.675

Table 3: Effective rank of the per-layer cache covariance (of $d = 1024$), at the first layer, mid-stack and last layer, with the variance fraction carried by the top direction. Trained caches are low-rank and anisotropic; randomly initialised models of the same architecture are neither, which is the null for Figure 6.

Model	effective rank (first $\to$ mid $\to$ last)	top-1 variance fraction (first $\to$ last)
Qwen3-1.7B, trained	$59 \to 15 \to 177$	$0.31 \to 0.10$
Qwen3-1.7B, *random init*	$365 \to 583 \to 558$	$0.04 \to 0.01$
Qwen3-8B, trained	$24 \to 48 \to 135$	$0.52 \to 0.10$
Qwen3-8B, *random init*	$401 \to 821 \to 830$	$0.04 \to 0.00$
from-scratch 1.7B, trained	$13 \to 113 \to 157$	$0.51 \to 0.10$

disturbs the leading directions, but they are different regularisers, so R^2_{lin} is the fit quality of that estimator rather than of the shipped matrix exactly. R^2_{nl} is an MLP on the same split, early-stopped on validation with the linear solution as its initial best checkpoint, so $R^2_{\text{nl}} \geq R^2_{\text{lin}}$ by construction and is a *lower* bound on what is recoverable from x. The variance splits into linear-explained R^2_{lin}, nonlinear-extra $R^2_{\text{nl}} - R^2_{\text{lin}}$, and irreducible $1 - R^2_{\text{nl}}$.

What the per-layer curves show. The layer averages of Table 1 hide a shape common to all three pairs, though not the monotone one might expect. R^2 peaks early (at layer 0 or 1 on both Qwen3 pairs, at layer 4 on the from-scratch pair, which rises before it turns over), then drifts down and falls off sharply at the last layer, which is the worst layer in nine of the twelve pair $\times$ side $\times$ direction cells (8B/4B values run 0.828 at layer 1 to 0.515 at layer 35). How much of that is trend rather than the final-layer cliff depends on the pair: Spearman(layer, R^2) is null on 1.7B/0.6B (-0.04 to -0.26, $p \geq 0.19$), moderate on 8B/4B (-0.34 to -0.42, $p \leq 0.04$) and strong on the from-scratch pair (-0.51 to -0.79, $p \leq 0.005$). Values sit below keys at every layer of both Qwen3 pairs and in both directions; the from-scratch pair is the exception, with values above keys at layer 0 in $L \to S$ (by 0.068) and at three early layers in $S \to L$. The nonlinear-extra term is small but not uniform: it concentrates in the first layers on every pair, peaking at 0.074 against a layer average of 0.015 for 8B/4B values and at 0.175 against 0.073 for from-scratch values, so what nonlinearity there is to buy sits at the bottom of the stack. That is also where the bias term of Table 2 lives on the from-scratch pair: its largest single-layer gains are at layers 2 and 5, inside the same early band, so on that pair part of the early nonlinear headroom is an offset rather than curvature. Both Qwen3 pairs show no such band at any layer.

Depth $\times$ position. Refitting within position buckets on long documents (PG-19 (Rae et al., 2019), $256 \to 16$k) gives, for Qwen3 8B/4B pre-normalisation keys, layer-averaged R^2 of 0.806, 0.815, 0.814, 0.816 over buckets $[0, 256)$, $[256, 1\text{k})$, $[1\text{k}, 4\text{k})$, $[4\text{k}, 16\text{k})$: flat to slightly rising, with the paired sign test across layers null everywhere ($p = 0.19$–0.87). The depth slope, however, steepens with position: Spearman(layer, R^2) moves from -0.29 in the first bucket to -0.62 in the last. Splitting the stack in thirds, the position effect is -0.031 (early), -0.010 (mid), $+0.032$ (late) for values. Long context helps the first two thirds and hurts the last third, and the largest single-layer losses are all deep and in values. That deep loss is irreducible: at 16k the MLP adds 0.000 over the linear fit. Finally, the isolated low-CKA "dead" layers that appear in pre-normalisation keys at short context (layers 13, 15–18 for 8B/4B) largely dissolve past $\sim$1k tokens (their R^2 dip shrinks from $+0.102$ to $+0.026$ and their CKA dip from $+0.351$ to $+0.067$), confirming by an independent route that they are a pre-normalisation scaling artefact rather than a genuine cross-model misalignment.

C TRAINING DATA AND TRAINING RECIPE

Figure 8 summarises how a KV-Lingo translator is applied and trained; this section details the data it is trained on.

The training stream. We train all our translators using the same general-knowledge multi-turn conversation dataset: Nemotron-SFT-Instruction-Following-Chat-v2 (NVIDIA et al., 2025). In our experiments, the training dataset is disjoint from every downstream benchmark: our translators are universal for text data, in the sense that we train one translator per model pair once and for all, and this translator generalises to several domains and languages.

Our training samples all take the form of a (prefix, continuation) pair. The prefix is the beginning of a conversation between a human and an assistant, containing one or several turns, and ending at the start of an assistant turn (after the last boilerplate token of the template), just before the first token of the reply. We alternate between reasoning-on and reasoning-off samples, structured as follows. For *reasoning-on* samples, the reasoning trace is part of the continuation. For *reasoning-off* samples, the continuation contains no reasoning trace, and an empty reasoning marker closes the prefix instead.

Prefix lengths. To train the translator on prefixes of varied and controlled lengths, we build a modified training dataset from Nemotron-SFT-Instruction-Following-Chat-v2, whose conversations are too short on their own to train long-context translators. It is made of:

- 50% natural samples drawn from Nemotron-SFT-Instruction-Following-Chat-v2, half of which have a reasoning-off continuation and the other half a reasoning-on one (reasoning traces are removed from the prefix). The prefix length of these samples ranges from 10 to 10k tokens, and is approximately uniformly distributed over that range.
- 50% samples obtained by concatenating natural samples, from which one question and its answer are removed at random. The question is placed at the end of the prefix, together with the boilerplate tokens that open the assistant turn, while the answer to the question is placed in the continuation. The concatenations are built so as to have prefix lengths between 12k and 16k tokens.

Figure 9 shows an estimation of the resulting distribution of prefix lengths. Continuation lengths are capped at 2048 tokens, with an average length of 750 tokens reasoning-off and 1k tokens reasoning-on.

How a sample is built. A training record is a multi-turn conversation. We pick one assistant turn t at random, render the conversation with the target family's chat template up to and including the header that opens turn t, and drop every later turn. That render is the prefix; turn t's own answer is the continuation. The prefix therefore always ends inside the template's boilerplate rather than at a content token, which is exactly where a served conversation sits when the model is about to speak, and where a switch of model would happen.

Where the reasoning trace goes. Whether a sample reasons is a property of the record, not a setting: the loader detects a `<think>` block in turn t and renders the sample accordingly. If turn t reasons, the prefix ends at the bare assistant header and the continuation opens with the real `<think>` block, so the trace is scored. If it does not, Qwen3's template emits an empty `<think>\n\n</think>\n\n` placeholder, and we keep that placeholder at the end of the *prefix*, where inference puts it, prefilled by whoever holds the cache, rather than at the start of the continuation, where it would be scored as four fully predictable tokens. Independently of t, the Qwen3 template strips the reasoning block from every assistant turn before the last, so earlier turns appear as plain chat in the prefix. In the packed half of the mixture above the geometry is the same, with the evaluated conversation's final user query lifted to the very end of the prefix and its answer kept as the continuation.

Template tokens. The boilerplate of the three families that appear in the paper is as follows; in Qwen3 and Gemma a newline follows each delimiter and each generation prompt, and in Mistral a single space follows the opening delimiter.

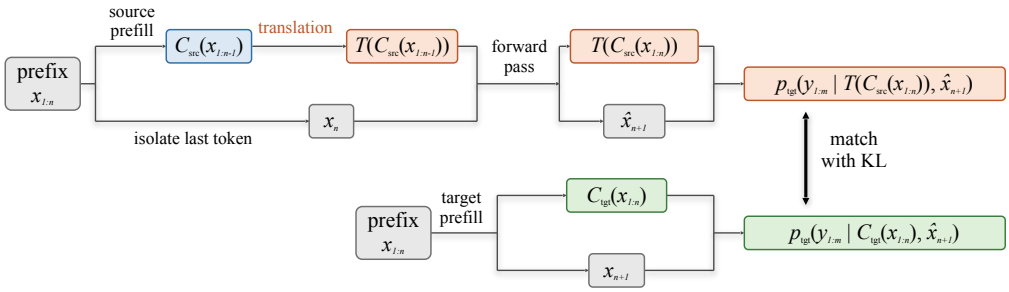


Figure 8: **Architecture and training of KV-Lingo translators.** Both rows start from the same prefix $x_{1:n}$ and end at a distribution over the same continuation $y_{1:m}$; Stage 2 of §3.2 asks the two to agree. Top: the source prefills $x_{1:n-1}$, the translator T maps that cache into the target's format, and one forward pass of the target on the held-back token x_n completes the cache and emits $\hat{x}_{n+1}$. Bottom: the target prefills the whole prefix itself. The cache is the only thing that differs between the two rows. Source and target are frozen; the KL is backpropagated through the target into T, which carries all the trainable parameters.

- **Qwen3.** Turns are wrapped in `<|im_start|>{role} ... <|im_end|>`, with roles `user` / `assistant`; the generation prompt is `<|im_start|>assistant`, followed on a reasoning-off sample by the placeholder `<think>\n\n</think>\n\n`.
- **Gemma.** Turns are wrapped in `<start_of_turn>{role} ... <end_of_turn>`, with roles `user` / `model`, and the generation prompt is `<start_of_turn>model`. There is no thinking placeholder.
- **Mistral.** User turns are wrapped in `[INST] ... [/INST]`, and `[/INST]` is itself the generation prompt. There is no thinking placeholder.

Only Qwen3 exposes a thinking placeholder; Gemma's template has no such flag and does not strip reasoning from earlier turns, and it refuses two consecutive turns in the same role. For the cross-tokenizer pair of §4.4, each model renders the conversation with its *own* template and only content tokens are put in correspondence: `<|im_start|>assistant` and `[/INST]` have no shared referent across the two families.

A rendered sample. Below is the shape of one reasoning-off Qwen3 sample. The control tokens are verbatim; the conversation content is abridged and stands in for a real record.

Prefix (translated by T, never decoded):

```
<|im_start|>user
Give me three uses for baking soda.<|im_end|>
<|im_start|>assistant
Cleaning, deodorising, and leavening.<|im_end|>
<|im_start|>user
Which of those needs an acid?<|im_end|>
<|im_start|>assistant
<think>

</think>
```

Continuation (decoded by the target, scored against the target's own distribution):

```
Leavening: it needs an acid to release CO2.<|im_end|>
```

On a reasoning-on sample the prefix stops one line earlier, after `<|im_start|>assistant\n`, and the continuation begins with the record's own `<think>` block.

D EVALUATION DATA

Table 4 lists the evaluation suite: the ability each benchmark probes, the translated prefix length, the number of scored items, the reasoning mode and generation budget, and the metric. MMLU-Pro and ARC-Challenge are scored by generate-and-parse, and MT-Bench-101 on one randomly chosen turn per conversation, with the earlier turns as context. Each translator is evaluated on a subset of this evaluation suite.

Decoding and judging. Every model decodes greedily, in both reasoning modes, up to the generation budget of Table 4. The judge, run locally, is Qwen3.6-27B. It also decodes greedily, with reasoning off, and sees the answer with any reasoning trace removed. MT-Bench-101 uses the benchmark's own per-task prompts and 1–10 scale (Bai et al., 2024); since we judge a single turn per conversation, we report the mean over conversations of that turn's rating, not the benchmark's minimum over turns. LongMemEval uses the benchmark's own answer-check templates (Wu et al., 2025a), with the closing *Answer yes or no only* replaced by a request for a bracketed yes-or-no verdict, so that a malformed reply cannot be read as a wrong answer. RepLiQA answers are judged against the reference answer on the answerable questions, under one of two rubrics: binary correctness, or the fraction of the reference's key facts that the answer conveys, the judge first splitting the reference into those facts. The shape-preserving Qwen3 results report the fact coverage, and the cross-tokenizer results the binary correctness.

Table 4: The evaluation suite. *Reasoning* is the mode the reported numbers are scored in, with the generation budget (max new tokens) per mode in parentheses; teacher-forced rows need none. *Prefix* is the translated prefix length in tokens: a range or ladder where the benchmark fixes it, a cap where it does not. The language-modelling row is scored on held-out conversations from the training mixture.

Benchmark	Ability	Prefix	n_{samples}	Reasoning (budget)	Metric
Nemotron chat	LM	10–16k	held-out	off/on (−)	cont. PPL, KL
MMLU-Pro	knowledge MC	$\leq$ 3k	1,400	off/on (1k/8k)	accuracy (macro)
ARC-Challenge	science MC	$\leq$ 3k	1,172	off/on (1k/8k)	accuracy
GSM8K	arithmetic	$\leq$ 1k	256	on (2k)	EM flexible
IFEval	constraints	$\leq$ 2k	541	off (1k)	prompt/instr. acc.
CoQA	grounded QA	$\leq$ 1k	256	off (128)	F1, gold-in-ans.
QuAC	dialogue QA	0.7–1.1k	256 (8 turns)	off (128)	F1
RepLiQA	unseen-doc QA	$\leq$ 3k	512 (358a)	off/on (512/2k)	judge correct.
XQuAD	QA, 11 langs	$\leq$ 2k	2,816	off (128)	token-F1, DiD
MT-Bench-101	multi-turn chat	103 (p50)	256	off/on (2k)	judge, 1–10
MultiChallenge	multi-turn instr.	1–5k	273	off (2k)	judge pass rate
RULER (NIAH)	NIAH retrieval	4/8/16/32k	400/len.	off/on (2k)	recall
LongMemEval	session memory	oracle, 8–32k	389 common	off/on (2k)	judge accuracy
LongBench-v2	long documents	$\leq$ 32k	116/395	off/on (512/4k)	accuracy

Relevant model pairs for each dataset. LongBench-v2 is not reported on Qwen3-0.6B↔1.7B (either reasoning mode) nor on Gemma-4-E2B↔E4B reasoning-off: the larger model of the pair is near or below the 25% chance floor (0.319/0.250 and 0.208 respectively), so no converted-vs-native delta is readable there. RULER reasoning-off on Qwen3-4B↔8B is at ceiling at 4/8/16k (native recall 0.998–1.000); only the 32k rung separates conditions on that pair.

E TRAINING PROTOCOL

Models. We use the following released Hugging Face checkpoints:

- `Qwen/Qwen3-0.6B`, `Qwen/Qwen3-1.7B`, `Qwen/Qwen3-4B` and `Qwen/Qwen3-8B` (Yang et al., 2025a);
- `Qwen/Qwen3-0.6B-Base` and `Qwen/Qwen3-1.7B-Base`, in Appendix V only;
- `google/gemma-3-4b-it` and `google/gemma-3-12b-it` (Team et al., 2025);
- `google/gemma-4-E2B-it` and `google/gemma-4-E4B-it` (Team et al., 2026);
- `mistralai/Mistral-7B-Instruct-v0.3` (Jiang et al., 2023).

The four Qwen3 models are the hybrid-reasoning releases. The MoE pair of Appendix N adds Qwen3-30B-A3B, the MLA models of Appendix L are converted by us from Qwen3-4B and Qwen3-8B as described there, and the from-scratch pair of Appendix V is pretrained by us with the recipe given there.

Hardware and precision. Every translator is trained on NVIDIA H100 80GB GPUs. Source and target stay frozen in `bf16` throughout; the translator's parameters are held in `fp32`. Most translators are trained on a single GPU, accumulating gradients over the 8 conversations of a step; the cross-layer head-mixing translators with $\nu \geq 2$ and all head-wise cross-layer translators use 8-way data parallelism instead, one conversation per GPU, at the same effective batch of 8.

The training procedure. Both stages of the translator fitting procedure (§3.2) read the same stream of (prefix, continuation) pairs, one conversation at a time: prefix lengths span three orders of magnitude, so samples are never padded together into a batch.

Stage 1. The second-moment statistics of equation 3 are accumulated in `fp64` over 400 conversations of the training stream (≈2.8M prefix positions), and the per-layer normal equations are then solved once, offline, by symmetric eigendecomposition, with eigenvalues below a relative tolerance of 10^{-8} treated as zero. Note that during this phase, the continuations are not used. The resulting maps are both the initialisation of stage 2 and the closed-form baseline we carry through every table.

Stage 2. Starting from those maps, we optimise equation 4 for 5000 steps using AdamW with an effective batch of 8 conversations, cosine schedule with 5% warmup, no weight decay, gradient clipping at 1.0. Prefixes are capped at $16{,}384$ tokens and continuations at 2048 (Appendix C). For the Base pairs of Appendix V, every sample is instead a 256-token prefix followed by a 256-token continuation. The learning rate is the only tuned hyperparameter. For each (pair, direction, translator), we train at five learning rates spaced half a decade apart, centred on the best rate of an earlier run, with one seed, and extend the window whenever the best rate sits on its edge, so that every selected learning rate is interior to the rates tried; over all cells the rates tried span $3{\times}10^{-7}$ to $3{\times}10^{-2}$. We select the rate with the lowest final validation KL on 64 held-out conversations drawn from the same dataset and disjoint from training, and train the other seeds at that rate. Table 5 lists the selected rates.

MLP translators. The MLP baselines read $\nu = 1$ source layer per target layer and replace each linear block by a two-layer MLP $z \mapsto W_2\,\mathrm{GELU}(W_1 z + b_1) + b_2$, with no skip connection. The head-wise MLP of Chen et al. (2026) has one such MLP per target layer, KV head and side (keys or values), $128 \to 2048 \to 128$ on Qwen3, and scales its output by a learned sigmoid gate per layer and head, initialised at $\sigma(4) \approx 0.98$. The head-mixing MLP of the cross-layer pairs has one gated MLP per target layer and side, spanning all heads, $1024 \to 2048 \to 1024$, which matches the head-wise MLP in parameters and FLOPs; the one on Qwen3-4B/8B is ungated and $1024 \to 4096 \to 1024$. With no closed-form solution available, stage 1 trains the MLP from random initialisation with AdamW on the cache-reconstruction error of equation 3, for 5000 steps with the batch and schedule of stage 2; stage 2 is then the recipe above. For the head-wise MLP, the stage-1 learning rate is swept over $\{10^{-3}, 3{\times}10^{-3}, 10^{-2}, 3{\times}10^{-2}\}$ (extended when the best sits on an edge) and selected on held-out reconstruction error, and the stage-2 one over five half-decade values around the best of an earlier short-context run. The head-mixing MLPs are not swept: the cross-layer ones reuse the head-wise MLP's learning rates for the same cell, and the Qwen3-4B/8B one uses 10^{-3} in stage 1

Table 5: **Selected stage-2 learning rates** of the linear translators, by pair, direction and layer assignment. On the shape-preserving pairs every target layer reads its counterpart, so the two $\nu = 1$ columns share one rate; on the cross-layer pairs ν is the number of source layers per target layer. *Depth*: relative-depth match. *Selected*: the R^2 assignment on Qwen3, which at $\nu = 1$ coincides with greedy match up to the ridge penalty, and greedy match on Gemma. The head-mixing columns at $\nu \geq 2$ use the R^2 assignment and the head-wise columns greedy match, as in Appendix H. The head-mixing rates at $\nu \geq 2$ were selected at seed 56, all others at seed 42. The MLP rates are given in the text, and those of the MoE pair in Appendix N.

		head-mixing				head-wise			
Pair	Direction	ν=1, depth	ν=1, selected	ν=2	ν=4	ν=1	ν=2	ν=4	ν=8
Qwen3-0.6B/1.7B	$L \to S$	3×10^{-5}				10^{-4}			
Qwen3-0.6B/1.7B	$S \to L$	10^{-4}				10^{-3}			
Qwen3-4B/8B	$L \to S$	3×10^{-5}				10^{-4}			
Qwen3-4B/8B	$S \to L$	3×10^{-5}				3×10^{-4}			
Qwen3-0.6B/8B	$L \to S$	10^{-4}	3×10^{-4}	10^{-4}	3×10^{-5}	3×10^{-3}	10^{-3}	3×10^{-4}	3×10^{-4}
Qwen3-0.6B/8B	$S \to L$	3×10^{-5}	3×10^{-5}	10^{-5}	3×10^{-6}	3×10^{-4}	10^{-4}	10^{-4}	10^{-5}
Qwen3-1.7B/4B	$L \to S$	3×10^{-4}	3×10^{-4}	10^{-4}	10^{-4}	3×10^{-3}	10^{-3}	10^{-3}	3×10^{-4}
Qwen3-1.7B/4B	$S \to L$	10^{-5}	10^{-5}	3×10^{-6}	3×10^{-6}	10^{-4}	10^{-4}	3×10^{-5}	10^{-5}
Gemma-3-4B/12B	both	3×10^{-5}	3×10^{-5}						
Gemma-4-E2B/E4B	both	10^{-4}	10^{-4}						

and the best of $\{3\times10^{-6}, 10^{-5}, 3\times10^{-5}\}$ in stage 2. Head-wise MLPs on the shape-preserving pairs are trained with three seeds, every other MLP with one.

Figure 9 shows the prefix lengths of the long-context training mixture of Appendix C.

Appendix D gives the evaluation protocol: decoding, judging and scoring. Appendix E.1 defines the error bars and the seeds.

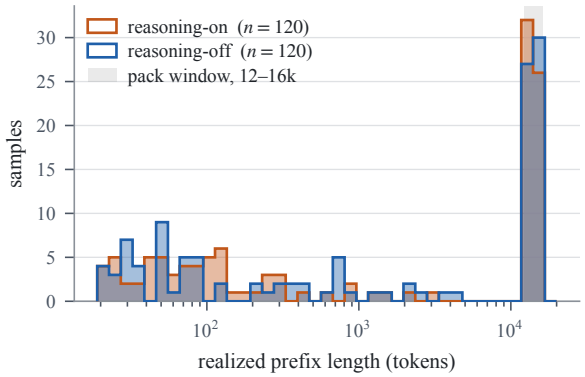


Figure 9: **Realised prefix length of the long-context instruct training mixture.** Histogram over 240 probe samples, split by reasoning status, on log-spaced bins. Both statuses span the short unpacked prompts and the packed window alike, so a downstream reasoning-on/off comparison is not also a comparison of prefix lengths.

E.1 Seeds and error bars

Error bars. Every $\pm$ in the tables and every error bar in the figures is the standard error of the benchmark score over its evaluation items: for a score that averages the per-item scores $s_1, \ldots, s_n$, it is their sample standard deviation divided by $\sqrt{n}$. It measures how far the score would move on a fresh draw of evaluation items, with the translator held fixed. A score averaged over disjoint item sets (MMLU-Pro over its 14 categories, RULER over its rungs, a mean over benchmarks) combines the standard errors of its parts as $\sqrt{\sum_j w_j^2 \, \mathrm{se}_j^2}$. LongMemEval asks the same questions at every rung, so its mean over rungs is clustered by question instead. A three-seed mean averages each item over the

three seeds before taking the standard error, and a judge rating rescaled to $[0, 1]$ rescales its standard error with it. The from-scratch and released Base pairs stored only aggregate accuracies, so theirs is the binomial $\sqrt{p(1-p)/n}$; perplexities and validation KLs carry none.

Seeds. On the two shape-preserving Qwen3 pairs, 0.6B/1.7B and 4B/8B, every trained translator is trained with seeds 42, 43 and 44 and reported as their mean. Every other translator is reported at a single seed: seed 42 for the families that were also trained at three seeds (the cross-layer pairs at $\nu = 1$, Gemma-3 and Gemma-4), and the only seed for the wider maps and the MLP baselines, which were trained once. The rest of this subsection shows what the other two seeds change.

Table 6 compares the two sources of variation benchmark by benchmark. On the shape-preserving pairs the spread over seeds is below the error bar of a single seed on every benchmark but XQuAD, whose token-F1 over eleven languages has a small standard error. On the cross-layer pairs it is as large as the error bar or larger on MMLU-Pro, ARC-C, CoQA, XQuAD, RepLiQA and RULER, and on Gemma-3 on all of those but RULER: there, a single-seed number carries training noise that its $\pm$ does not show. In 54 of the 513 three-seed cells behind the table, seed 42 lies more than two of its own standard errors from the three-seed mean.

Table 6: **Seed spread against evaluation noise.** For each family trained at three seeds, *seed sd* is the standard deviation of a cell's three per-seed scores and *eval se* the standard error over evaluation items of one seed's score, each the median over the family's cells (translators, directions and reasoning modes) on that benchmark. RULER pools its per-rung cells, LongMemEval is the mean over its four rungs, and judge ratings are rescaled to $[0, 1]$. —: not evaluated for that family.

	shape-preserving Qwen3		cross-layer Qwen3		Gemma-3		Gemma-4	
Benchmark	seed sd	eval se	seed sd	eval se	seed sd	eval se	seed sd	eval se
MMLU-Pro	0.010	0.012	0.016	0.012	0.016	0.013	0.011	0.013
ARC-C	0.006	0.009	0.014	0.012	0.023	0.012	0.005	0.010
GSM8K	0.012	0.026	0.021	0.026	0.012	0.022	—	—
IFEval	0.008	0.020	0.012	0.021	0.010	0.021	—	—
CoQA	0.013	0.025	0.036	0.024	0.055	0.027	—	—
XQuAD	0.014	0.007	0.013	0.006	0.024	0.008	0.035	0.010
RepLiQA	0.009	0.013	0.017	0.018	0.031	0.018	—	—
MT-Bench-101	0.010	0.021	0.014	0.025	0.009	0.018	0.005	0.017
RULER	0.004	0.009	0.021	0.017	0.009	0.014	0.004	0.007
LongMemEval	0.011	0.020	0.013	0.019	—	—	—	—
LongBench-v2	0.021	0.045	0.030	0.042	—	—	0.016	0.022

Table 7 follows the families that the paper reports at seed 42 through one benchmark-set mean per translator. Seed 42 lies within two standard errors of the three-seed mean on 11 of the 16 rows. The largest departure comes from a collapsed seed: on 0.6B/8B in $L \to S$ with the greedy assignment, seed 42 scores 0.018 on XQuAD against 0.190 and 0.235 for the other two seeds, and 0.208 on MMLU-Pro against 0.337 and 0.314, which puts its four-benchmark mean at 0.360 against a three-seed mean of 0.408. Collapses of this kind, where one seed loses a third or more of its score on one benchmark, occur five times, all on the cross-layer pairs and all reasoning-off. The other three hit seed 43: on the same cell with the depth assignment (0.308 on ARC-C against 0.650 and 0.676), and on 1.7B/4B in $S \to L$ with the depth assignment (0.296 on CoQA against 0.581 and 0.616, and 0.197 on XQuAD against 0.387 and 0.392).

Table 7: **Seed** 42 **against the three-seed mean, for the families reported at seed** 42**.** One benchmark-set mean per translator, reasoning-off: the four-benchmark mean of the cross-layer figures (MMLU-Pro, ARC-C, RULER, LongMemEval), the six-benchmark mean of Figure 25, and the mean over the eleven rows of Table 14, with judge ratings rescaled to $[0, 1]$. *Seed 42* is the score the paper reports, $\pm$ its standard error over evaluation items; *3-seed mean* is $\pm$ the seed standard deviation.

Pair	Direction	Assignment	seed 42	3-seed mean	seed 43	seed 44
Qwen3-0.6B/8B	$L \to S$	depth	$0.371_{\pm 0.007}$	$0.351_{\pm 0.061}$	0.282	0.400
Qwen3-0.6B/8B	$L \to S$	greedy	$0.360_{\pm 0.007}$	$0.408_{\pm 0.042}$	0.439	0.424
Qwen3-0.6B/8B	$S \to L$	depth	$0.408_{\pm 0.007}$	$0.410_{\pm 0.003}$	0.411	0.413
Qwen3-0.6B/8B	$S \to L$	greedy	$0.435_{\pm 0.007}$	$0.441_{\pm 0.005}$	0.442	0.446
Qwen3-1.7B/4B	$L \to S$	depth	$0.458_{\pm 0.007}$	$0.461_{\pm 0.013}$	0.476	0.451
Qwen3-1.7B/4B	$L \to S$	greedy	$0.478_{\pm 0.007}$	$0.492_{\pm 0.013}$	0.504	0.493
Qwen3-1.7B/4B	$S \to L$	depth	$0.536_{\pm 0.007}$	$0.530_{\pm 0.007}$	0.522	0.531
Qwen3-1.7B/4B	$S \to L$	greedy	$0.518_{\pm 0.007}$	$0.519_{\pm 0.007}$	0.513	0.526
Gemma-4-E2B/E4B	$L \to S$	depth	$0.668_{\pm 0.006}$	$0.678_{\pm 0.009}$	0.680	0.685
Gemma-4-E2B/E4B	$L \to S$	greedy	$0.665_{\pm 0.006}$	$0.675_{\pm 0.009}$	0.677	0.684
Gemma-4-E2B/E4B	$S \to L$	depth	$0.657_{\pm 0.006}$	$0.655_{\pm 0.003}$	0.651	0.655
Gemma-4-E2B/E4B	$S \to L$	greedy	$0.664_{\pm 0.006}$	$0.658_{\pm 0.007}$	0.650	0.659
Gemma-3-4B/12B	$L \to S$	depth	$0.712_{\pm 0.005}$	$0.725_{\pm 0.012}$	0.733	0.730
Gemma-3-4B/12B	$L \to S$	greedy	$0.747_{\pm 0.005}$	$0.744_{\pm 0.007}$	0.736	0.749
Gemma-3-4B/12B	$S \to L$	depth	$0.692_{\pm 0.005}$	$0.685_{\pm 0.007}$	0.683	0.679
Gemma-3-4B/12B	$S \to L$	greedy	$0.706_{\pm 0.005}$	$0.694_{\pm 0.012}$	0.694	0.683

F QWEN3-0.6B/1.7B AND 4B/8B, PRETRAINED

This section presents the full set of evaluations realised with the pretrained model pairs Qwen3-0.6B/1.7B and 4B/8B. Tables 8 and 9 report both pairs over the whole evaluation suite, one table per translation direction: the two native anchors against the closed-form fit, the three trained translators of §4.1 and the head-wise MLP adapter of Chen et al. (2026). Both are scored reasoning-off throughout. Figures 10 and 11 draw the same measurement, one figure per reasoning mode. Figure 12 includes a head-mixing MLP as a baseline, in $S \to L$ with the model pair Qwen3-4B/8B, showing that it performs on par with the head-wise MLP baseline.

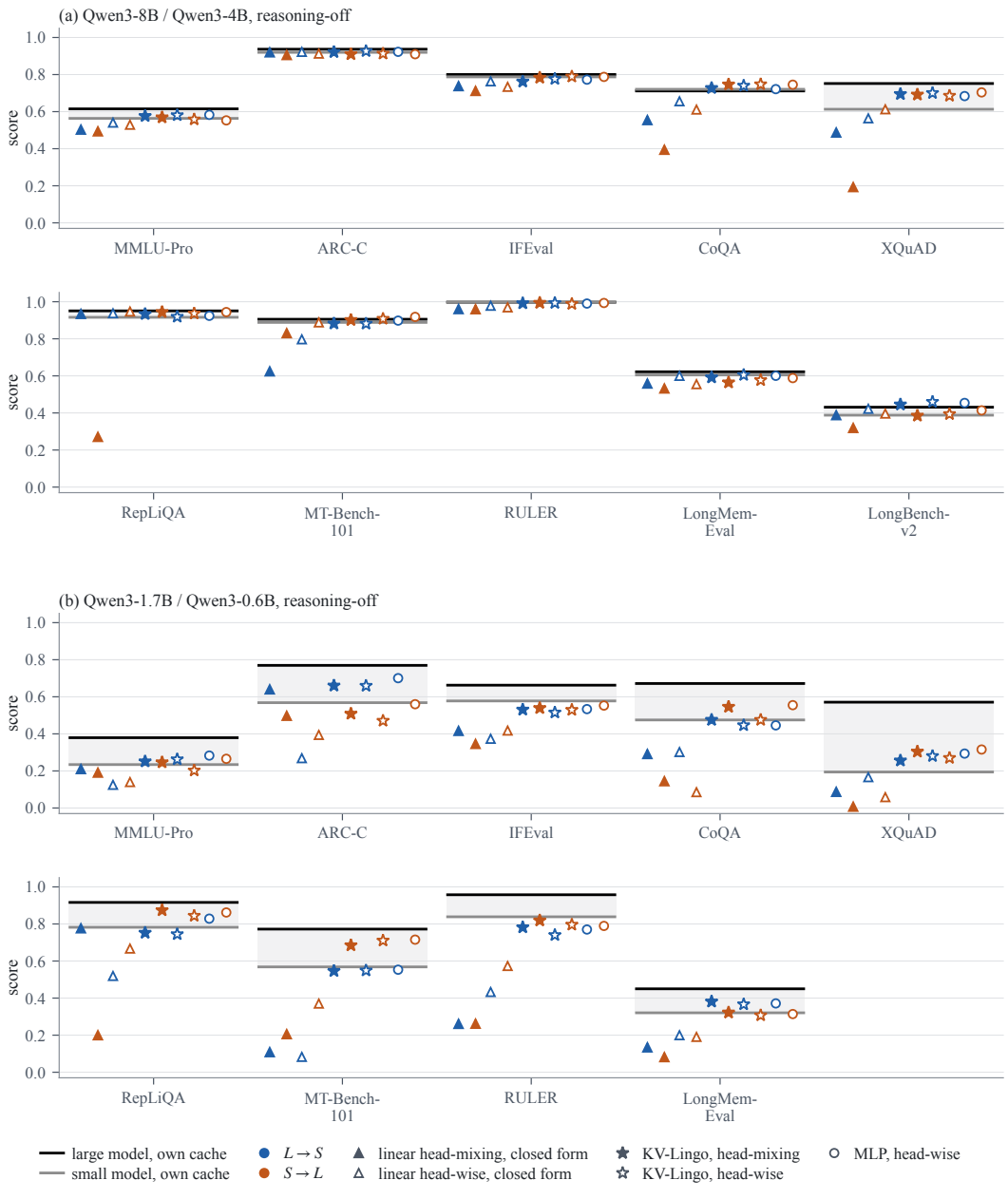


Figure 10: **Shape-preserving Qwen3 pairs, reasoning-off, both directions.** Trained translators are the mean over three seeds; error bars: standard error over evaluation items (Appendix E.1).

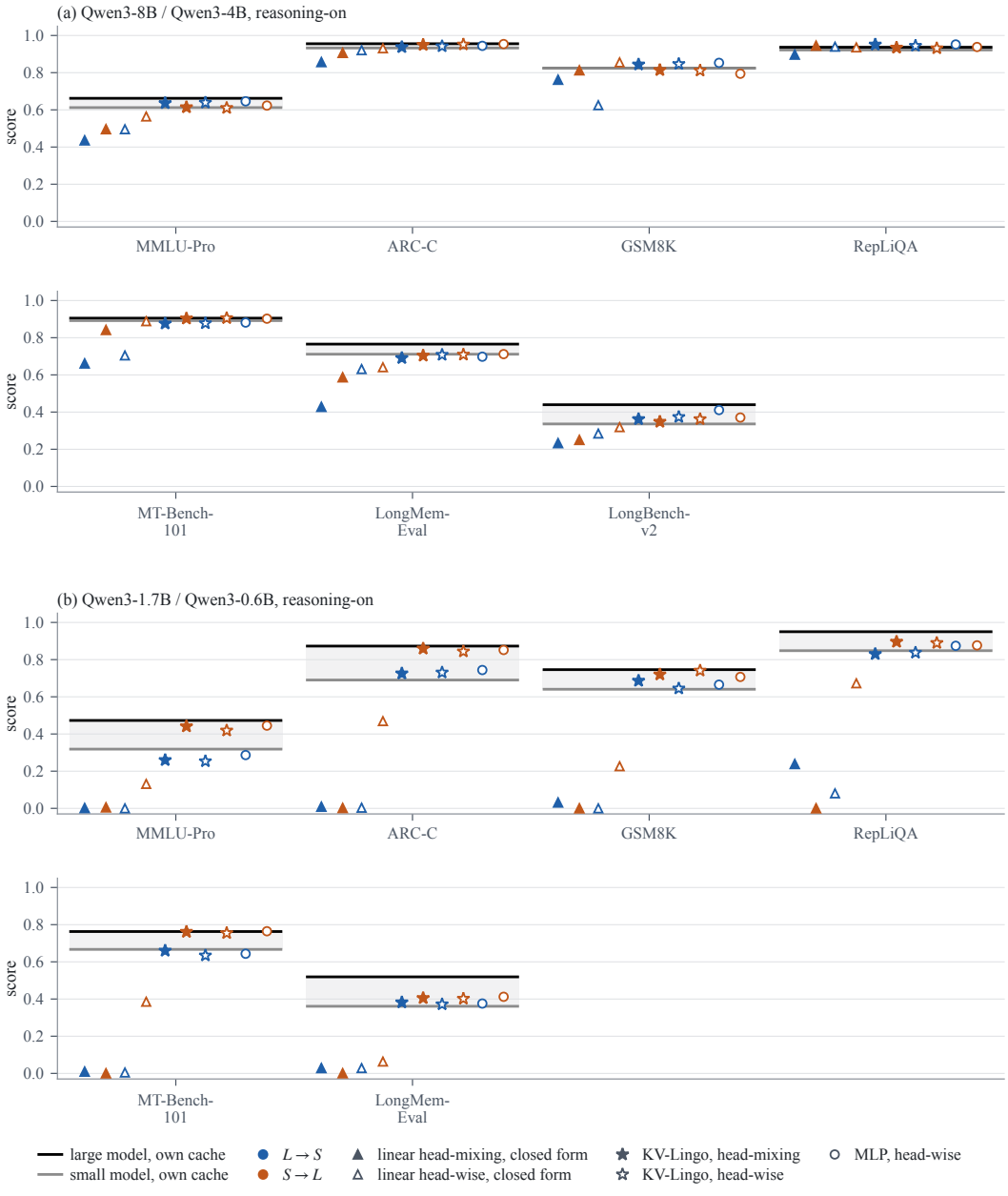


Figure 11: **Shape-preserving Qwen3 pairs, reasoning-on, both directions.** Trained translators are the mean over three seeds; error bars: standard error over evaluation items (Appendix E.1).

Table 8: **Large to small evaluation suite, Qwen3 pairs with the same shape, reasoning-off.** The large model prefills, the small one decodes from the translated cache. Trained translators are the mean over three seeds; the closed-form fit is a single solve. ±: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model. **Bold**: the best translator on that row. The head-wise MLP adapter of Chen et al. (2026) is shown at the width that matches our map in parameters and FLOPs (hidden dimension is $4\times$ the input dimension). Pre-normalisation capture throughout. Metrics follow Table 4: accuracy for MMLU-Pro (macro over 14 categories), ARC-C and LongBench-v2 (untruncated subset), prompt-level strict accuracy for IFEval, a 1–10 judge score for MT-Bench-101, token-F1 for CoQA and for XQuAD (pooled at equal n over its eleven languages), graded fact coverage on the answerable split for RepLiQA, recall averaged over 4k–32k for RULER, and judged accuracy averaged over the four dilution rungs for LongMemEval. H-M means head-mixing. The mixed loss is an ablation, presented in Appendix U. —: not run.

		native		translated				
	Benchmark	small	large	closed form	KV-Lingo H-M, KL	KV-Lingo H-M, mixed loss	KV-Lingo head-wise	MLP head-wise
Qwen3-0.6B/1.7B	MMLU-Pro	$0.234_{\pm 0.011}$	$0.379_{\pm 0.013}$	$0.210_{\pm 0.011}$	$0.251_{\pm 0.010}$	$0.279_{\pm 0.010}$	$0.262_{\pm 0.010}$	$\mathbf{0.283}_{\pm 0.010}$
	ARC-C	$0.568_{\pm 0.014}$	$0.770_{\pm 0.012}$	$0.640_{\pm 0.014}$	$0.660_{\pm 0.012}$	$0.694_{\pm 0.012}$	$0.659_{\pm 0.012}$	$\mathbf{0.701}_{\pm 0.012}$
	IFEval	$0.580_{\pm 0.021}$	$0.662_{\pm 0.020}$	$0.416_{\pm 0.021}$	$0.530_{\pm 0.019}$	$0.476_{\pm 0.019}$	$0.515_{\pm 0.019}$	$\mathbf{0.534}_{\pm 0.020}$
	CoQA	$0.475_{\pm 0.026}$	$0.672_{\pm 0.026}$	$0.291_{\pm 0.022}$	$0.476_{\pm 0.024}$	$\mathbf{0.516}_{\pm 0.023}$	$0.446_{\pm 0.023}$	$0.446_{\pm 0.024}$
	RepLiQA	$0.782_{\pm 0.019}$	$0.916_{\pm 0.012}$	$0.776_{\pm 0.020}$	$0.752_{\pm 0.017}$	$\mathbf{0.841}_{\pm 0.014}$	$0.745_{\pm 0.017}$	$0.829_{\pm 0.015}$
	XQuAD	$0.194_{\pm 0.004}$	$0.571_{\pm 0.008}$	$0.087_{\pm 0.004}$	$0.256_{\pm 0.006}$	$\mathbf{0.362}_{\pm 0.007}$	$0.280_{\pm 0.006}$	$0.294_{\pm 0.006}$
	MT-Bench-101	$6.11_{\pm 0.26}$	$7.95_{\pm 0.22}$	$1.97_{\pm 0.16}$	$5.92_{\pm 0.25}$	$5.44_{\pm 0.25}$	$5.94_{\pm 0.26}$	$\mathbf{5.98}_{\pm 0.28}$
	RULER	$0.839_{\pm 0.006}$	$0.958_{\pm 0.004}$	$0.261_{\pm 0.008}$	$0.782_{\pm 0.005}$	$\mathbf{0.799}_{\pm 0.004}$	$0.740_{\pm 0.005}$	$0.770_{\pm 0.005}$
	LongMemEval	$0.331_{\pm 0.019}$	$0.470_{\pm 0.020}$	$0.134_{\pm 0.011}$	$\mathbf{0.403}_{\pm 0.020}$	$0.402_{\pm 0.020}$	$0.380_{\pm 0.019}$	$0.385_{\pm 0.020}$
	LongBench-v2	$0.276_{\pm 0.042}$	$0.319_{\pm 0.043}$	$\mathbf{0.284}_{\pm 0.042}$	$0.261_{\pm 0.030}$	$0.270_{\pm 0.035}$	$0.250_{\pm 0.035}$	—
Qwen3-4B/8B	MMLU-Pro	$0.564_{\pm 0.013}$	$0.615_{\pm 0.013}$	$0.503_{\pm 0.013}$	$0.576_{\pm 0.012}$	$0.572_{\pm 0.012}$	$0.580_{\pm 0.012}$	$\mathbf{0.583}_{\pm 0.012}$
	ARC-C	$0.920_{\pm 0.008}$	$0.937_{\pm 0.007}$	$0.920_{\pm 0.008}$	$0.921_{\pm 0.007}$	$0.914_{\pm 0.007}$	$\mathbf{0.927}_{\pm 0.007}$	$0.923_{\pm 0.007}$
	IFEval	$0.786_{\pm 0.018}$	$0.802_{\pm 0.017}$	$0.738_{\pm 0.019}$	$0.762_{\pm 0.016}$	$0.754_{\pm 0.016}$	$\mathbf{0.776}_{\pm 0.016}$	$0.773_{\pm 0.016}$
	CoQA	$0.721_{\pm 0.024}$	$0.712_{\pm 0.023}$	$0.554_{\pm 0.024}$	$0.727_{\pm 0.023}$	$0.725_{\pm 0.023}$	$\mathbf{0.740}_{\pm 0.023}$	$0.721_{\pm 0.023}$
	RepLiQA	$0.916_{\pm 0.012}$	$0.951_{\pm 0.009}$	$0.934_{\pm 0.011}$	$0.934_{\pm 0.009}$	$\mathbf{0.944}_{\pm 0.009}$	$0.919_{\pm 0.011}$	$0.924_{\pm 0.010}$
	XQuAD	$0.613_{\pm 0.007}$	$0.752_{\pm 0.006}$	$0.487_{\pm 0.007}$	$0.695_{\pm 0.007}$	$0.699_{\pm 0.006}$	$\mathbf{0.700}_{\pm 0.007}$	$0.684_{\pm 0.007}$
	MT-Bench-101	$9.00_{\pm 0.19}$	$9.15_{\pm 0.16}$	$6.62_{\pm 0.26}$	$8.94_{\pm 0.15}$	$8.77_{\pm 0.15}$	$8.93_{\pm 0.15}$	$\mathbf{9.08}_{\pm 0.13}$
	RULER	$0.998_{\pm 0.001}$	$0.998_{\pm 0.001}$	$0.960_{\pm 0.004}$	$0.991_{\pm 0.002}$	$0.992_{\pm 0.002}$	$\mathbf{0.993}_{\pm 0.001}$	$0.990_{\pm 0.002}$
	LongMemEval	$0.633_{\pm 0.019}$	$0.643_{\pm 0.020}$	$0.583_{\pm 0.020}$	$0.621_{\pm 0.019}$	$0.615_{\pm 0.019}$	$\mathbf{0.632}_{\pm 0.019}$	$0.627_{\pm 0.020}$
	LongBench-v2	$0.388_{\pm 0.045}$	$0.431_{\pm 0.046}$	$0.388_{\pm 0.045}$	$0.445_{\pm 0.044}$	$0.448_{\pm 0.043}$	$\mathbf{0.460}_{\pm 0.044}$	$0.454_{\pm 0.045}$

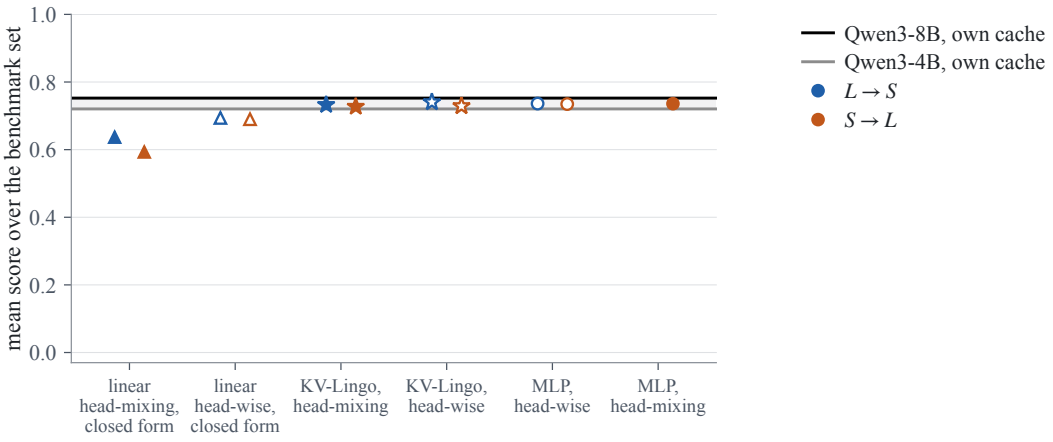


Figure 12: Qwen3-4B/8B, reasoning-off, average scores of MMLU-Pro, ARC-C, IFEval, CoQA, XQuAD, MT-Bench-101, RULER, LongMemEval, LongBench-v2. The head-mixing MLP, computed only in the small to large setting, performs on par with the head-wise MLP. The head-mixing MLP is a single seed (seed 42), the other trained translators the mean over three seeds; error bars: standard error over evaluation items (Appendix E.1).

Table 9: **Small to large evaluation suite, Qwen3 pairs with the same shape, reasoning-off.** The small model prefills, the large one decodes from the translated cache. Trained translators are the mean over three seeds; the closed-form fit is a single solve. ±: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model by more than one standard error. **Bold**: the best translator on that row. The head-wise MLP adapter of Chen et al. (2026) is shown at the width that matches our map in parameters and FLOPs (hidden dimension is 4× the input dimension). Pre-normalisation capture throughout. Metrics follow Table 4: accuracy for MMLU-Pro (macro over 14 categories), ARC-C and LongBench-v2 (untruncated subset), prompt-level strict accuracy for IFEval, a 1–10 judge score for MT-Bench-101, token-F1 for CoQA and for XQuAD (pooled at equal n over its eleven languages), graded fact coverage on the answerable split for RepLiQA, recall averaged over 4k–32k for RULER, and judged accuracy averaged over the four dilution rungs for LongMemEval. H-M means head-mixing. The mixed loss is an ablation, presented in Appendix U. On the 0.6B/1.7B pair the head-wise MLP's MT-Bench-101 cell is the mean of two seeds: the third returns a score outside the judge's 1–10 scale, a judge-parser bug that read "1996" as a rating. —: not run.

		native		translated				
	Benchmark	small	large	closed form	KV-Lingo H-M, KL	KV-Lingo H-M, mixed loss	KV-Lingo head-wise	MLP head-wise
Qwen3-0.6B/1.7B	MMLU-Pro	$0.234_{\pm 0.011}$	$0.379_{\pm 0.013}$	$0.191_{\pm 0.010}$	$\underline{0.246}_{\pm 0.009}$	$0.234_{\pm 0.009}$	$0.202_{\pm 0.008}$	$\underline{\mathbf{0.265}}_{\pm 0.010}$
	ARC-C	$0.568_{\pm 0.014}$	$0.770_{\pm 0.012}$	$0.497_{\pm 0.015}$	$0.509_{\pm 0.012}$	$0.488_{\pm 0.012}$	$0.471_{\pm 0.012}$	$\mathbf{0.560}_{\pm 0.013}$
	IFEval	$0.575_{\pm 0.021}$	$0.664_{\pm 0.020}$	$0.346_{\pm 0.020}$	$0.539_{\pm 0.019}$	$0.502_{\pm 0.019}$	$0.529_{\pm 0.018}$	$\mathbf{0.553}_{\pm 0.020}$
	CoQA	$0.475_{\pm 0.026}$	$0.672_{\pm 0.026}$	$0.144_{\pm 0.018}$	$\underline{0.546}_{\pm 0.024}$	$\underline{\mathbf{0.568}}_{\pm 0.024}$	$0.475_{\pm 0.022}$	$\underline{0.555}_{\pm 0.025}$
	RepLiQA	$0.782_{\pm 0.019}$	$0.916_{\pm 0.012}$	$0.200_{\pm 0.020}$	$\underline{\mathbf{0.874}}_{\pm 0.013}$	$\underline{0.854}_{\pm 0.013}$	$\underline{0.844}_{\pm 0.013}$	$\underline{0.862}_{\pm 0.013}$
	XQuAD	$0.194_{\pm 0.004}$	$0.571_{\pm 0.008}$	$0.007_{\pm 0.001}$	$\underline{0.304}_{\pm 0.006}$	$\underline{\mathbf{0.366}}_{\pm 0.007}$	$\underline{0.270}_{\pm 0.006}$	$\underline{0.316}_{\pm 0.007}$
	MT-Bench-101	$6.11_{\pm 0.26}$	$7.95_{\pm 0.22}$	$2.85_{\pm 0.21}$	$\underline{7.16}_{\pm 0.19}$	$\underline{7.14}_{\pm 0.19}$	$\underline{7.39}_{\pm 0.19}$	$\underline{\mathbf{7.44}}_{\pm 0.20}$
	RULER	$0.839_{\pm 0.006}$	$0.958_{\pm 0.004}$	$0.262_{\pm 0.009}$	$\mathbf{0.819}_{\pm 0.004}$	$0.815_{\pm 0.004}$	$0.796_{\pm 0.004}$	$0.789_{\pm 0.005}$
	LongMemEval	$0.331_{\pm 0.019}$	$0.470_{\pm 0.020}$	$0.087_{\pm 0.010}$	$\mathbf{0.335}_{\pm 0.017}$	$0.333_{\pm 0.018}$	$0.318_{\pm 0.017}$	$0.328_{\pm 0.019}$
	LongBench-v2	$0.276_{\pm 0.042}$	$0.319_{\pm 0.043}$	$0.302_{\pm 0.043}$	$0.296_{\pm 0.029}$	$\underline{\mathbf{0.322}}_{\pm 0.035}$	$0.270_{\pm 0.031}$	—
Qwen3-4B/8B	MMLU-Pro	$0.564_{\pm 0.013}$	$0.615_{\pm 0.013}$	$0.494_{\pm 0.013}$	$\mathbf{0.569}_{\pm 0.012}$	$0.558_{\pm 0.012}$	$0.558_{\pm 0.012}$	$0.553_{\pm 0.012}$
	ARC-C	$0.920_{\pm 0.008}$	$0.937_{\pm 0.007}$	$0.905_{\pm 0.009}$	$0.910_{\pm 0.008}$	$\mathbf{0.914}_{\pm 0.007}$	$0.913_{\pm 0.008}$	$0.910_{\pm 0.008}$
	IFEval	$0.789_{\pm 0.018}$	$0.799_{\pm 0.017}$	$0.712_{\pm 0.019}$	$0.783_{\pm 0.016}$	$0.766_{\pm 0.016}$	$\mathbf{0.789}_{\pm 0.016}$	$0.787_{\pm 0.016}$
	CoQA	$0.721_{\pm 0.024}$	$0.712_{\pm 0.023}$	$0.394_{\pm 0.022}$	$\underline{0.746}_{\pm 0.022}$	$\underline{0.747}_{\pm 0.023}$	$\underline{\mathbf{0.747}}_{\pm 0.022}$	$\underline{0.745}_{\pm 0.022}$
	RepLiQA	$0.916_{\pm 0.012}$	$0.951_{\pm 0.009}$	$0.271_{\pm 0.023}$	$\underline{0.944}_{\pm 0.009}$	$\underline{0.944}_{\pm 0.009}$	$\underline{0.937}_{\pm 0.009}$	$\underline{\mathbf{0.944}}_{\pm 0.009}$
	XQuAD	$0.613_{\pm 0.007}$	$0.752_{\pm 0.006}$	$0.193_{\pm 0.006}$	$\underline{0.692}_{\pm 0.007}$	$\underline{0.677}_{\pm 0.007}$	$\underline{0.686}_{\pm 0.007}$	$\underline{\mathbf{0.704}}_{\pm 0.007}$
	MT-Bench-101	$9.00_{\pm 0.19}$	$9.15_{\pm 0.16}$	$8.47_{\pm 0.21}$	$9.11_{\pm 0.14}$	$\underline{9.18}_{\pm 0.14}$	$\underline{9.17}_{\pm 0.15}$	$\underline{\mathbf{9.26}}_{\pm 0.14}$
	RULER	$0.998_{\pm 0.001}$	$0.998_{\pm 0.001}$	$0.960_{\pm 0.003}$	$\mathbf{0.994}_{\pm 0.002}$	$0.992_{\pm 0.002}$	$0.988_{\pm 0.002}$	$0.993_{\pm 0.002}$
	LongMemEval	$0.633_{\pm 0.019}$	$0.643_{\pm 0.020}$	$0.563_{\pm 0.020}$	$0.591_{\pm 0.019}$	$0.603_{\pm 0.019}$	$0.604_{\pm 0.019}$	$\mathbf{0.613}_{\pm 0.019}$
	LongBench-v2	$0.388_{\pm 0.045}$	$0.431_{\pm 0.046}$	$0.319_{\pm 0.043}$	$0.385_{\pm 0.041}$	$0.399_{\pm 0.043}$	$0.394_{\pm 0.042}$	$\mathbf{0.414}_{\pm 0.044}$

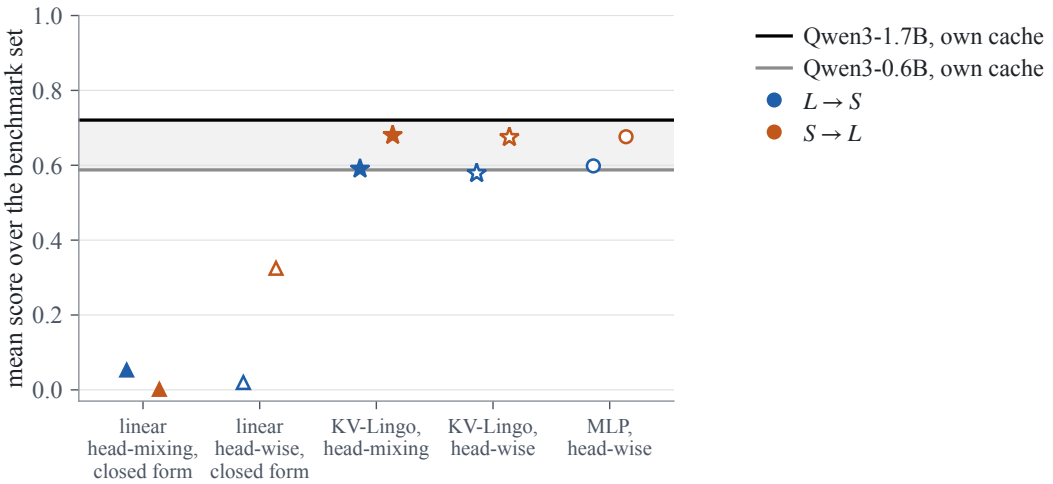


Figure 13: **Shape-preserving small Qwen3 pair, reasoning-on, both directions.** Average score over MMLU-Pro, ARC-C, GSM8K, RepLiQA, MT-Bench-101 and LongMemEval.

G HEAD ALIGNMENT IN RELEASED QWEN3 PAIRS

On the two shape-preserving Qwen3 pairs, head-wise translators match head-mixing ones (§4.2), whereas on the pair pretrained from scratch they fall behind (Table 31). Figure 14 explains the difference: the heads of the released models correspond one to one, while those of the from-scratch models do not.

To measure this without imposing any head structure, we take the closed-form joint map of Appendix W, fitted by equation 3 with no constraint on head mixing. We cut its key-to-key and value-to-value blocks into 8×8 blocks of size $d_h \times d_h$. Entry (i, j) of a grid is the Frobenius norm of the block that sends source head i to target head j, averaged over layers and normalised so that each row sums to one. Chance is therefore $1/8$. We compare the mean diagonal against the same statistic under 10^5 random permutations of the target heads.

On both released pairs, in $S \to L$, every source head sends its largest block to the target head with the same index. The mean diagonal reaches 0.166–0.216, 33–73% above its permutation-null mean. This is the block that a head-wise translator keeps, and the off-diagonal blocks it drops are smaller, each below 0.135. From 8B to 4B the diagonal is stronger still (0.196 on keys and 0.249 on values). The exception is 1.7B $\to$ 0.6B, whose grids sit at chance, so for that direction the closed-form map does not by itself account for the head-wise result.

The from-scratch pair has no such structure: its diagonal is at chance (0.125 on both keys and values), and each grid is nearly uniform. The only structure is by column: a few target heads receive more from every source head. The alignment therefore comes from how the released checkpoints were produced, not from the architecture. A shared initialisation recovers only a small part of it. We initialise the small model by slicing the first heads and channels out of the large one and pretrain it for the same 35B tokens (last column). This brings the identity back, but weakly: the diagonal is 0.130 and 0.133, 4% and 6% above chance.

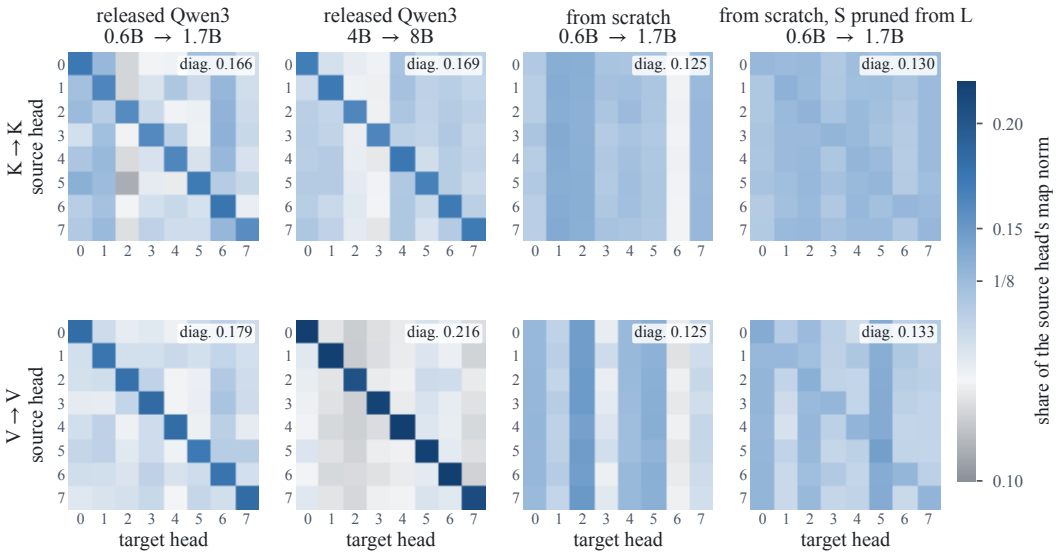


Figure 14: **Per-head structure of the closed-form map, $S \to L$.** Share of each source head's map norm sent to each target head, for key-to-key (top) and value-to-value (bottom) blocks, averaged over layers; rows sum to one, and white is chance ($1/8$). *diag.*: mean of the diagonal. Released Qwen3-0.6B/1.7B and 4B/8B against a 0.6B/1.7B pair we pretrained from scratch (Appendix V), from independent initialisations or with the small model sliced out of the large one.

H QWEN3-0.6B/8B AND QWEN3-1.7B/4B, PRETRAINED

This section complements §4.3 on the two Qwen3 pairs whose depths differ, 28 layers for Qwen3-0.6B and Qwen3-1.7B against 36 for Qwen3-4B and Qwen3-8B, each translated in both directions, which makes four cross-layer cells. It first defines the layer assignments (Figure 17, then greedy match and the R^2 assignment below), then compares them: at one source layer per target layer in Tables 10, 11 and 12, and across the number ν of source layers per target layer in Figures 18 and 19. Figure 20 compares the head-mixing and head-wise translators over ν; the last paragraph reads what a larger ν buys and what it costs. Figures 15 and 16 set every translator evaluated on the two pairs side by side, closed-form, KV-Lingo and MLP, head-mixing and head-wise, on the mean over nine reasoning-off benchmarks.

Greedy match. Greedy match (§3.1) chooses $\mathcal{E}_\ell$ from data, independently for each target layer. Fix a target layer ℓ and a calibration set of n_{cal} tokens; let $Y \in \mathbb{R}^{n_{\text{cal}} \times d_{\text{tgt}}}$ collect its features and $X_s \in \mathbb{R}^{n_{\text{cal}} \times d_{\text{src}}}$ those of source layer s on the same tokens. For a support $S \subseteq \{1, \dots, N_{\text{src}}\}$, let $X_S \coloneqq [\, X_s \,]_{s \in S} \in \mathbb{R}^{n_{\text{cal}} \times |S| d_{\text{src}}}$ be the column-wise concatenation, and define the ridge block map and its relative residual

$$\hat{B}_S \coloneqq \operatorname*{arg\,min}_{B \in \mathbb{R}^{|S| d_{\text{src}} \times d_{\text{tgt}}}} \left\| Y - X_S B \right\|_F^2 + \lambda \|B\|_F^2, \qquad r(S) \coloneqq \frac{\left\| Y - X_S \hat{B}_S \right\|_F^2}{\|Y\|_F^2}, \tag{5}$$

with the convention $r(\varnothing) = 1$. Since $|S| d_{\text{src}}$ may exceed n_{cal}, the penalty $\lambda > 0$ is what guarantees that $\hat{B}_S$ is unique. The support is then grown one source layer at a time, re-solving the full block map at every step: starting from $S^{(0)} = \varnothing$, for $j = 1, \dots, \nu$,

$$s^{(j)} \in \operatorname*{arg\,min}_{s \notin S^{(j-1)}} r\big(S^{(j-1)} \cup \{s\}\big), \qquad S^{(j)} = S^{(j-1)} \cup \big\{s^{(j)}\big\}, \tag{6}$$

and we set $\mathcal{E}_\ell \coloneqq S^{(\nu)}$, so that $|\mathcal{E}_\ell| = \nu$. Each step evaluates every remaining candidate exactly, at a cost of $O(\nu\, N_{\text{src}})$ ridge solves per target layer. At $\nu = 1$ the single step picks the source layer whose single-layer fit has the highest R^2, which is, up to the ridge penalty, the R^2 assignment below.

The R^2 assignment. The R^2 assignment ranks source layers by their single-layer fit alone, without the forward selection. For each target layer ℓ and source layer s, it scores $R^2_\ell(s) \coloneqq 1 - r(\{s\})$, the fraction of the target layer's features explained by the least-squares map from source layer s alone, and sets $\mathcal{E}_\ell$ to the ν source layers with the highest $R^2_\ell(s)$. Keys and values are ranked separately, so a target layer may read different source layers for its keys and for its values. The scores are read off the Gram matrices the closed-form fit already accumulates, so the assignment costs no extra pass over the data. Unlike greedy match, it does not account for redundancy between the selected layers: two adjacent source layers that explain the same part of the target are both kept.

Comparing the assignments. Figure 18 compares the three assignments on Qwen3-8B/0.6B from $\nu = 1$ to $\nu = 8$, and Figure 19 does the same on Qwen3-4B/1.7B. Table 10 compares relative-depth match and the R^2 assignment at $\nu = 1$, both closed-form and self-distilled, on all four cross-layer cells. Table 11 breaks their RULER recall down by context budget in both reasoning modes, and Table 12 gives LongMemEval with reasoning on. Figure 21 draws the reasoning-on mean score of the Qwen3-0.6B/8B pair.

Head-mixing against head-wise translators. Figure 20 sets the head-mixing KV-Lingo translator against its head-wise counterpart on the Qwen3-8B/0.6B pair, at every ν of the sweep.

The number of source layers. Widening the assignment $\mathcal{E}_\ell$ of §3.1 so that every target layer reads ν source layers instead of one costs ν matrices per target layer, to store and to apply at every translation. Figure 22 reads the resulting trade-off on the two long-context evaluations at once: translation gets more expensive as ν grows from 2 to 8, so a cell's markers move right, and what the extra source layers buy is the vertical distance they cover. RULER is where they buy the most, gaining recall at every context budget in both directions; on LongMemEval the same widening moves the score by little. How much cheaper than re-prefilling the widest map still is depends on the context budget: at 32k every translated marker sits far to the left of the decoding model's own prefill. On the shortest

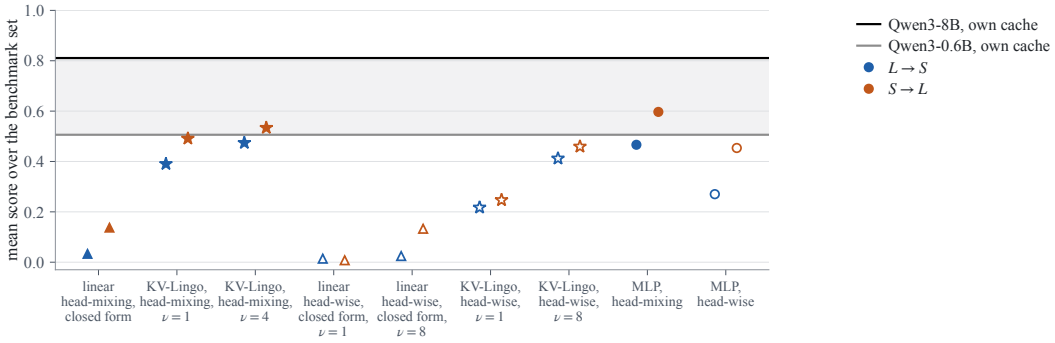


Figure 15: **Every translator on Qwen3-8B/0.6B**, both directions, reasoning-off. Each marker is the mean score over MMLU-Pro, ARC-C, RULER, LongMemEval, CoQA, IFEval, XQuAD, RepLiQA and MT-Bench-101, weighted equally, with RULER's four context budgets averaged first and judge scores rescaled to $[0, 1]$; LongBench-v2 is left out, as it was run in $S \to L$ only. Black and grey lines are Qwen3-8B and Qwen3-0.6B on their own cache. Triangles are closed-form fits, stars KV-Lingo after self-distillation, circles the two MLP translators of Appendix E; filled markers mix heads, open ones are head-wise. The head-mixing linear translators use the R^2 assignment, their closed form at $\nu = 1$; the head-wise linear translators and the head-mixing MLP use greedy match, the head-wise MLP relative depth. Head-mixing KV-Lingo at $\nu = 4$ matches the two MLPs in parameters; head-wise KV-Lingo would need $\nu = 32$, so $\nu = 8$ stands in at a quarter of their count. Head-mixing KV-Lingo at $\nu = 1$ is the mean over three seeds, at $\nu = 4$ seed 56, and every other trained translator seed 42; no error bars.

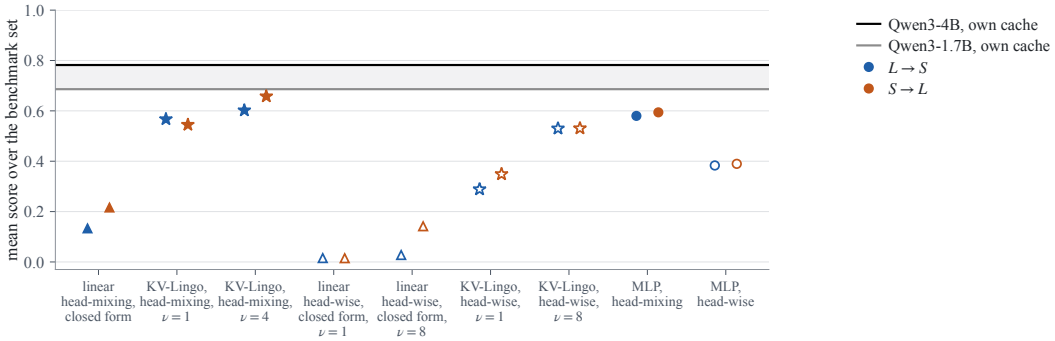


Figure 16: **Every translator on Qwen3-4B/1.7B**, both directions, reasoning-off. Everything as in Figure 15, for the other cross-layer pair.

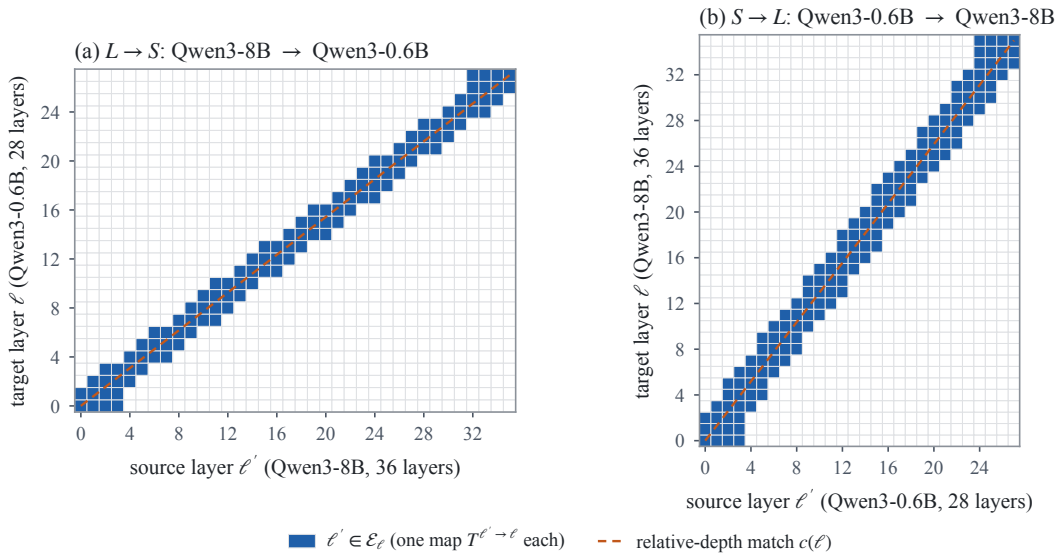


Figure 17: **The relative-depth assignment** $\mathcal{E}_\ell$ of §3.1, drawn for the unequal-depth Qwen3-0.6B/8B pair in both directions at $\nu = 4$ source layers per target layer. A filled cell (ℓ', ℓ) means that target layer ℓ reads source layer ℓ', through its own matrices $T_K^{\ell' \to \ell}$, $T_V^{\ell' \to \ell}$; each row therefore carries exactly ν filled cells, and each panel $2\nu N_{\text{tgt}}$ matrices in total. The dashed line is the exact relative-depth counterpart, $c(\ell) = \ell(N_{\text{src}} - 1)/(N_{\text{tgt}} - 1)$ with layers indexed from 0 as in the plot, on which the window is centred everywhere except at the two ends of the stack, where it shifts inward rather than truncating so that every target layer still reads ν distinct source layers. Going up in depth ($S \to L$, right) the band is steeper than the diagonal and consecutive target layers reuse source layers; going down ($L \to S$, left) it is shallower and consecutive windows slide faster than one layer at a time.

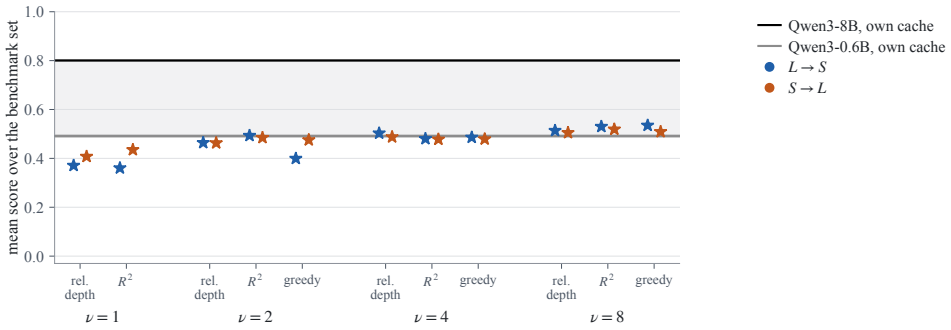


Figure 18: **Layer assignment against ν, Qwen3-8B/0.6B**, both directions, head-mixing KV-Lingo, reasoning-off. Markers and native lines as in Figure 20; each marker is the mean score over MMLU-Pro, ARC-C, RULER and LongMemEval, weighted equally, with RULER's four context budgets averaged first. Within each group of ν, the ticks are relative-depth match, the R^2 assignment (the ν source layers with the highest single-layer R^2, chosen separately for keys and values) and greedy match. At $\nu = 1$ greedy match reduces to the R^2 assignment up to the ridge penalty, so it has no separate translator there. Every point is a single seed, seed 42 at $\nu = 1$ and seed 56 at $\nu \geq 2$; error bars: standard error over evaluation items (Appendix E.1).

Table 10: **Cross-layer cache translation on two Qwen3 pairs**, 28 against 36 layers, both directions. Each target layer reads a single source layer, chosen either by relative depth or by the R^2 of its best single-layer fit; each assignment is reported as its closed-form fit and after self-distillation, the latter at seed 42. $\pm$: standard error over evaluation items (Appendix E.1). The two anchor columns are the same measurement for both directions of a pair, so they are shown even where that direction's translator cells are not. Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. Five further $S \to L$ cells clear the small model on the value, by $+0.009$ to $+0.026$, but by less than their own standard error, and are left unmarked. **Bold**: the best of the four translators on that row. A dash marks a LongBench-v2 cell left out because the decoding model is at the chance floor. Reasoning-off throughout; Table 12 gives reasoning-on LongMemEval and Table 11 the full RULER ladder. MMLU-Pro is the macro-average over its 14 categories and ARC-C the micro-average, both lenient generate-and-parse; RULER is recall averaged over 4k–16k; LongBench-v2 is accuracy on its untruncated subset; LongMemEval is judge accuracy pooled over its oracle, 8k, 16k and 32k rungs.

	native		closed form		self-distilled	
Benchmark	small	large	depth	R^2	depth	R^2
$L \to S$, Qwen3-8B $\to$ Qwen3-0.6B						
MMLU-Pro	$0.234_{\pm 0.011}$	$0.615_{\pm 0.013}$	$0.011_{\pm 0.003}$	$0.104_{\pm 0.008}$	$0.205_{\pm 0.011}$	$\mathbf{0.208}_{\pm 0.011}$
ARC-C	$0.568_{\pm 0.014}$	$0.937_{\pm 0.007}$	$0.000_{\pm 0.000}$	$0.028_{\pm 0.005}$	$\underline{0.650}_{\pm 0.014}$	$\underline{\mathbf{0.659}}_{\pm 0.014}$
RULER	$0.879_{\pm 0.006}$	$0.998_{\pm 0.001}$	$0.000_{\pm 0.000}$	$0.001_{\pm 0.001}$	$0.323_{\pm 0.010}$	$\mathbf{0.327}_{\pm 0.010}$
LongBench-v2	$0.276_{\pm 0.042}$	$0.431_{\pm 0.046}$	—	—	—	—
LongMemEval	$0.331_{\pm 0.019}$	$0.643_{\pm 0.020}$	$0.010_{\pm 0.003}$	$0.017_{\pm 0.004}$	$\underline{\mathbf{0.348}}_{\pm 0.020}$	$0.289_{\pm 0.019}$
$L \to S$, Qwen3-4B $\to$ Qwen3-1.7B						
MMLU-Pro	$0.379_{\pm 0.013}$	$0.564_{\pm 0.013}$	$0.134_{\pm 0.009}$	$0.213_{\pm 0.011}$	$0.283_{\pm 0.011}$	$\mathbf{0.286}_{\pm 0.011}$
ARC-C	$0.770_{\pm 0.012}$	$0.920_{\pm 0.008}$	$0.349_{\pm 0.014}$	$0.580_{\pm 0.014}$	$\underline{\mathbf{0.817}}_{\pm 0.011}$	$\underline{0.811}_{\pm 0.011}$
RULER	$0.980_{\pm 0.004}$	$0.999_{\pm 0.001}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.415_{\pm 0.010}$	$\mathbf{0.460}_{\pm 0.011}$
LongBench-v2	$0.319_{\pm 0.043}$	$0.388_{\pm 0.045}$	—	—	—	—
LongMemEval	$0.470_{\pm 0.020}$	$0.633_{\pm 0.019}$	$0.006_{\pm 0.002}$	$0.095_{\pm 0.010}$	$0.385_{\pm 0.019}$	$\mathbf{0.410}_{\pm 0.019}$
$S \to L$, Qwen3-0.6B $\to$ Qwen3-8B						
MMLU-Pro	$0.234_{\pm 0.011}$	$0.615_{\pm 0.013}$	$0.122_{\pm 0.009}$	$0.149_{\pm 0.010}$	$0.211_{\pm 0.011}$	$\mathbf{0.245}_{\pm 0.011}$
ARC-C	$0.568_{\pm 0.014}$	$0.937_{\pm 0.007}$	$0.282_{\pm 0.013}$	$0.400_{\pm 0.014}$	$0.445_{\pm 0.015}$	$\mathbf{0.522}_{\pm 0.015}$
RULER	$0.879_{\pm 0.006}$	$0.998_{\pm 0.001}$	$0.000_{\pm 0.000}$	$0.002_{\pm 0.001}$	$0.681_{\pm 0.010}$	$\mathbf{0.704}_{\pm 0.009}$
LongBench-v2	$0.276_{\pm 0.042}$	$0.431_{\pm 0.046}$	$0.009_{\pm 0.009}$	$0.181_{\pm 0.036}$	$0.284_{\pm 0.042}$	$\underline{\mathbf{0.362}}_{\pm 0.045}$
LongMemEval	$0.331_{\pm 0.019}$	$0.643_{\pm 0.020}$	$0.092_{\pm 0.010}$	$0.114_{\pm 0.011}$	$\underline{\mathbf{0.361}}_{\pm 0.019}$	$0.340_{\pm 0.020}$
$S \to L$, Qwen3-1.7B $\to$ Qwen3-4B						
MMLU-Pro	$0.379_{\pm 0.013}$	$0.564_{\pm 0.013}$	$0.016_{\pm 0.003}$	$0.226_{\pm 0.011}$	$\mathbf{0.324}_{\pm 0.012}$	$0.297_{\pm 0.012}$
ARC-C	$0.770_{\pm 0.012}$	$0.920_{\pm 0.008}$	$0.012_{\pm 0.003}$	$0.595_{\pm 0.014}$	$\underline{\mathbf{0.794}}_{\pm 0.012}$	$0.717_{\pm 0.013}$
RULER	$0.980_{\pm 0.004}$	$0.999_{\pm 0.001}$	$0.003_{\pm 0.001}$	$0.007_{\pm 0.002}$	$0.645_{\pm 0.009}$	$\mathbf{0.651}_{\pm 0.010}$
LongBench-v2	$0.319_{\pm 0.043}$	$0.388_{\pm 0.045}$	$0.181_{\pm 0.036}$	$0.284_{\pm 0.042}$	$0.328_{\pm 0.044}$	$\mathbf{0.345}_{\pm 0.044}$
LongMemEval	$0.470_{\pm 0.020}$	$0.633_{\pm 0.019}$	$0.145_{\pm 0.012}$	$0.141_{\pm 0.012}$	$0.445_{\pm 0.020}$	$\mathbf{0.465}_{\pm 0.020}$

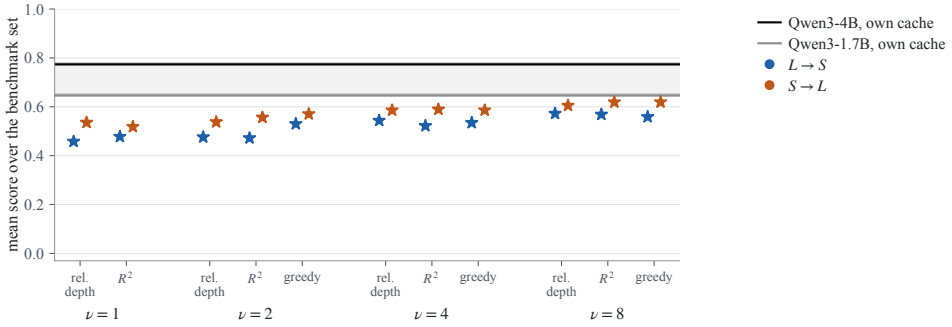


Figure 19: **Layer assignment against ν, Qwen3-4B/1.7B**, both directions, head-mixing KV-Lingo, reasoning-off. Everything as in Figure 18, for the other cross-layer pair.

Table 11: **RULER recall by context length**, four cross-layer Qwen3 cells, both reasoning modes. *small* and *large* are the two models of the pair decoding their own caches; *depth* and R^2 are the two self-distilled translators, at seed 42. $\pm$: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error; no cell of this table qualifies. **Bold**: the better of the two assignments on that row, within each reasoning mode. The translators are fitted at up to 16k tokens, so 32k is extrapolation. The closed-form fits score 0.000–0.013 everywhere in this table and are omitted.

	reasoning-off				reasoning-on			
Context	small	large	depth	R^2	small	large	depth	R^2
$L \to S$, Qwen3-8B → Qwen3-0.6B								
4k	$0.964_{\pm 0.008}$	$0.998_{\pm 0.003}$	$0.361_{\pm 0.017}$	$\mathbf{0.395}_{\pm 0.017}$	$0.819_{\pm 0.013}$	$0.992_{\pm 0.004}$	$\mathbf{0.372}_{\pm 0.018}$	$0.335_{\pm 0.017}$
8k	$0.855_{\pm 0.012}$	$0.998_{\pm 0.003}$	$\mathbf{0.352}_{\pm 0.017}$	$0.333_{\pm 0.016}$	$0.737_{\pm 0.008}$	$0.966_{\pm 0.008}$	$\mathbf{0.348}_{\pm 0.017}$	$0.307_{\pm 0.017}$
16k	$0.819_{\pm 0.012}$	$0.999_{\pm 0.001}$	$\mathbf{0.257}_{\pm 0.017}$	$0.254_{\pm 0.016}$	$0.708_{\pm 0.011}$	$0.983_{\pm 0.006}$	$0.246_{\pm 0.017}$	$\mathbf{0.274}_{\pm 0.017}$
32k	$0.717_{\pm 0.013}$	$0.996_{\pm 0.003}$	$0.150_{\pm 0.015}$	$\mathbf{0.161}_{\pm 0.013}$	$0.637_{\pm 0.014}$	$0.986_{\pm 0.006}$	$\mathbf{0.155}_{\pm 0.015}$	$0.149_{\pm 0.014}$
$L \to S$, Qwen3-4B → Qwen3-1.7B								
4k	$0.994_{\pm 0.004}$	$1.000_{\pm 0.000}$	$0.506_{\pm 0.019}$	$\mathbf{0.511}_{\pm 0.018}$	$0.869_{\pm 0.013}$	$0.960_{\pm 0.009}$	$\mathbf{0.336}_{\pm 0.020}$	$0.336_{\pm 0.020}$
8k	$0.981_{\pm 0.006}$	$1.000_{\pm 0.000}$	$0.425_{\pm 0.019}$	$\mathbf{0.483}_{\pm 0.018}$	$0.865_{\pm 0.015}$	$0.949_{\pm 0.010}$	$0.250_{\pm 0.019}$	$\mathbf{0.359}_{\pm 0.020}$
16k	$0.963_{\pm 0.008}$	$0.998_{\pm 0.003}$	$0.314_{\pm 0.017}$	$\mathbf{0.386}_{\pm 0.018}$	$0.862_{\pm 0.015}$	$0.955_{\pm 0.009}$	$0.204_{\pm 0.017}$	$\mathbf{0.320}_{\pm 0.019}$
32k	$0.891_{\pm 0.013}$	$0.995_{\pm 0.003}$	$0.146_{\pm 0.014}$	$\mathbf{0.237}_{\pm 0.017}$	$0.820_{\pm 0.014}$	$0.944_{\pm 0.010}$	$0.107_{\pm 0.011}$	$\mathbf{0.194}_{\pm 0.015}$
$S \to L$, Qwen3-0.6B → Qwen3-8B								
4k	$0.964_{\pm 0.008}$	$0.998_{\pm 0.003}$	$0.748_{\pm 0.015}$	$\mathbf{0.754}_{\pm 0.014}$	$0.819_{\pm 0.013}$	$0.992_{\pm 0.004}$	$0.419_{\pm 0.021}$	$\mathbf{0.476}_{\pm 0.020}$
8k	$0.855_{\pm 0.012}$	$0.998_{\pm 0.003}$	$0.668_{\pm 0.018}$	$\mathbf{0.723}_{\pm 0.016}$	$0.737_{\pm 0.008}$	$0.966_{\pm 0.008}$	$0.397_{\pm 0.020}$	$\mathbf{0.534}_{\pm 0.019}$
16k	$0.819_{\pm 0.012}$	$0.999_{\pm 0.001}$	$0.627_{\pm 0.018}$	$\mathbf{0.636}_{\pm 0.018}$	$0.708_{\pm 0.011}$	$0.983_{\pm 0.006}$	$0.447_{\pm 0.018}$	$\mathbf{0.493}_{\pm 0.018}$
32k	$0.717_{\pm 0.013}$	$0.996_{\pm 0.003}$	$0.411_{\pm 0.017}$	$\mathbf{0.421}_{\pm 0.017}$	$0.637_{\pm 0.014}$	$0.986_{\pm 0.006}$	$0.312_{\pm 0.018}$	$\mathbf{0.388}_{\pm 0.017}$
$S \to L$, Qwen3-1.7B → Qwen3-4B								
4k	$0.994_{\pm 0.004}$	$1.000_{\pm 0.000}$	$0.697_{\pm 0.015}$	$\mathbf{0.698}_{\pm 0.016}$	$0.869_{\pm 0.013}$	$0.960_{\pm 0.009}$	$0.446_{\pm 0.020}$	$\mathbf{0.457}_{\pm 0.022}$
8k	$0.981_{\pm 0.006}$	$1.000_{\pm 0.000}$	$0.646_{\pm 0.016}$	$\mathbf{0.664}_{\pm 0.017}$	$0.865_{\pm 0.015}$	$0.949_{\pm 0.010}$	$0.366_{\pm 0.020}$	$\mathbf{0.413}_{\pm 0.020}$
16k	$0.963_{\pm 0.008}$	$0.998_{\pm 0.003}$	$\mathbf{0.593}_{\pm 0.017}$	$0.593_{\pm 0.019}$	$0.862_{\pm 0.015}$	$0.955_{\pm 0.009}$	$0.368_{\pm 0.020}$	$\mathbf{0.391}_{\pm 0.020}$
32k	$0.891_{\pm 0.013}$	$0.995_{\pm 0.003}$	$0.378_{\pm 0.016}$	$\mathbf{0.422}_{\pm 0.018}$	$0.820_{\pm 0.014}$	$0.944_{\pm 0.010}$	$0.319_{\pm 0.017}$	$\mathbf{0.346}_{\pm 0.018}$

Table 12: **LongMemEval with reasoning on**, four cross-layer Qwen3 cells. Judge accuracy pooled over the oracle, 8k, 16k and 32k rungs, on the 389 questions common to all four ($n = 1556$ question-rung pairs). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the best of the four translators on that row. Self-distilled columns are seed 42. $\pm$: standard error over evaluation items (Appendix E.1).

	native		closed form		self-distilled	
Cell	small	large	depth	R^2	depth	R^2
Qwen3-8B → Qwen3-0.6B	$0.383_{\pm 0.019}$	$0.763_{\pm 0.017}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$\mathbf{0.373}_{\pm 0.019}$	$0.343_{\pm 0.018}$
Qwen3-4B → Qwen3-1.7B	$0.531_{\pm 0.019}$	$0.723_{\pm 0.017}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$\mathbf{0.453}_{\pm 0.019}$	$0.453_{\pm 0.018}$
Qwen3-0.6B → Qwen3-8B	$0.383_{\pm 0.019}$	$0.763_{\pm 0.017}$	$0.006_{\pm 0.002}$	$0.004_{\pm 0.002}$	$\underline{0.506}_{\pm 0.020}$	$\underline{\mathbf{0.531}}_{\pm 0.019}$
Qwen3-1.7B → Qwen3-4B	$0.531_{\pm 0.019}$	$0.723_{\pm 0.017}$	$0.002_{\pm 0.001}$	$0.003_{\pm 0.001}$	$\underline{0.566}_{\pm 0.019}$	$\underline{\mathbf{0.585}}_{\pm 0.019}$

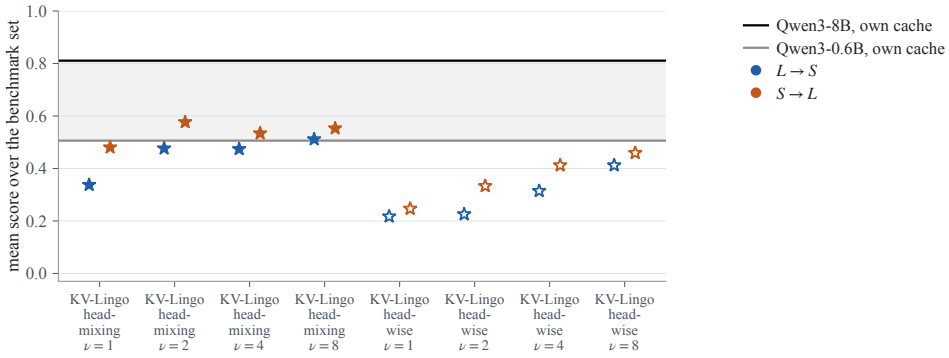


Figure 20: **Head-mixing against head-wise KV-Lingo over ν**, Qwen3-8B/0.6B, both directions, reasoning-off. Each marker is the mean score over the nine benchmarks of Figure 15, weighted equally. Black and grey lines are Qwen3-8B and Qwen3-0.6B on their own cache. Filled stars are the head-mixing translator with the R^2 assignment, open stars the head-wise one with greedy match. Every marker is a single seed: head-mixing at seed 42 for $\nu = 1$ and seed 56 for $\nu \geq 2$, head-wise at seed 42.

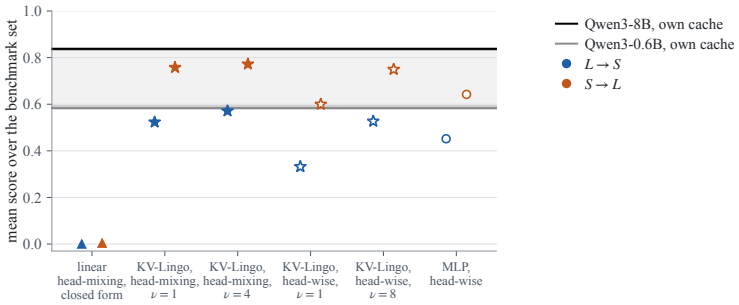


Figure 21: **Cache translation with Qwen3-0.6B/8B, reasoning-on.** Average over MMLU-Pro, ARC-C, GSM8K, RepLiQA, MT-Bench-101 and LongMemEval. Every point is a single seed, seed 42 except the head-mixing translator at $\nu = 4$ (seed 56); error bars: standard error over evaluation items (Appendix E.1).

RULER rung, the $\nu = 8$ marker of the 4B $\rightarrow$ 1.7B direction sits level with the re-prefill, but more than half of its time there is the timing harness's fixed decoding step of about 30 ms, which does not depend on ν (Appendix Y): the translation itself takes about 25 ms, against 49 ms for the 1.7B's prefill. Figure 23 draws the RULER side of this sweep for all four cross-layer cells, rung by rung, and Table 13 gives $\nu \in \{1, 2, 4, 8\}$ on MMLU-Pro, ARC-C, RULER and LongMemEval for the same four cells.

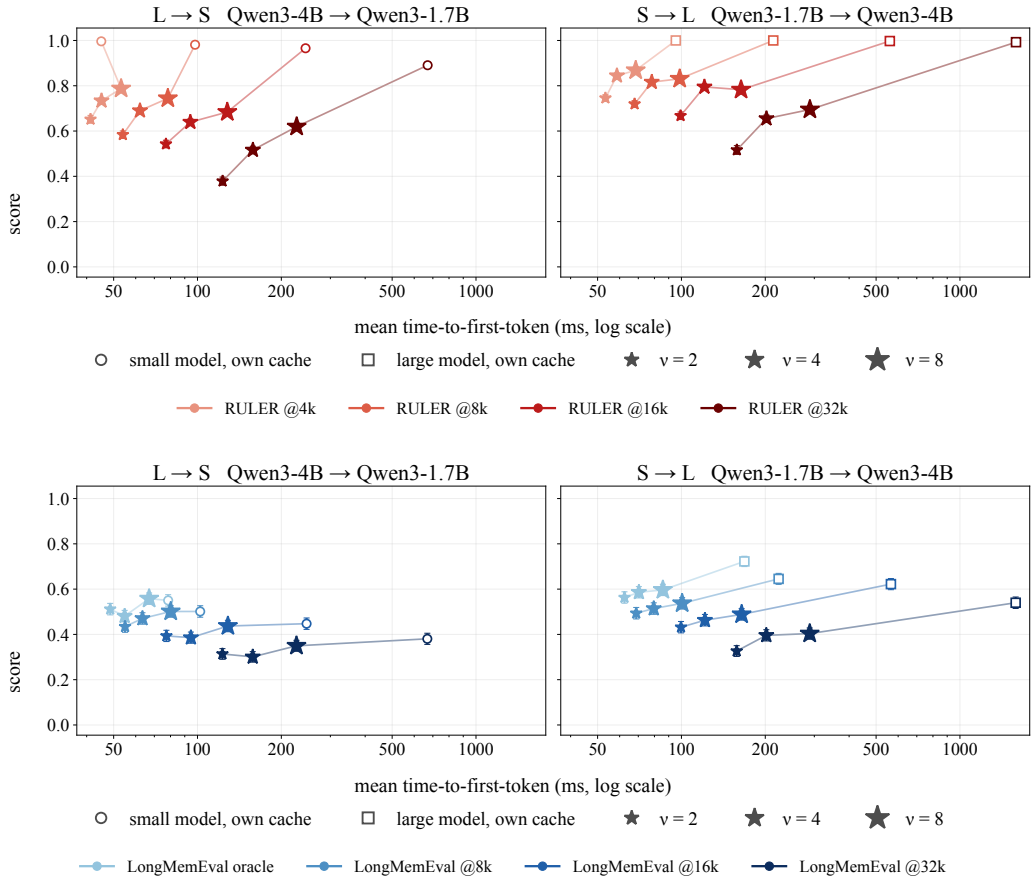


Figure 22: **KV-Lingo score against time to first token, Qwen3-4B/Qwen3-1.7B, R^2 assignment, reasoning-off.** RULER above and LongMemEval below, sharing an x window: x is the mean time to first token over the eval's own prefix lengths, y the eval's own metric, and colour the context budget, light at the shortest rung to dark at 32k. Filled stars are the translated cache at $\nu = 2, 4, 8$ source layers per target layer, growing with ν, each a single seed (seed 56), charged the translation and one decoding step (Appendix Y); open markers are the decoding model on its own cache, the only native baseline each panel carries. Error bars: standard error over evaluation items (Appendix E.1).

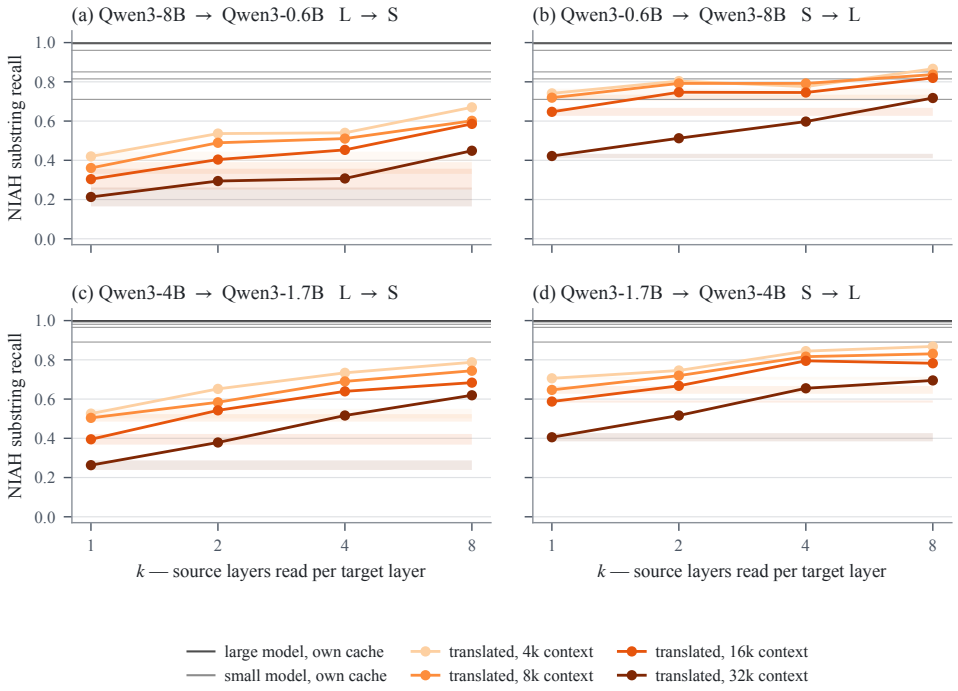


Figure 23: **RULER recall against the number of source layers per target layer** ν, the four cross-layer Qwen3 cells, head-mixing KV-Lingo with the R^2 assignment, reasoning-off. Colour is the context budget, light at 4k to dark at 32k. The black and grey hairlines are the large and the small model on their own cache, one line per budget. Every point is a single seed, seed 42 at $\nu = 1$ and seed 56 at $\nu \in \{2, 4, 8\}$, and the shaded band carries the standard error over evaluation items (Appendix E.1).

Table 13: **Sweeping the number of source layers per target layer** on the four cross-layer Qwen3 cells of Table 10: $\nu \in \{1, 2, 4, 8\}$, the R^2 assignment, self-distilled, reasoning-off. The $\nu = 1$ column is the seed-42 run of Table 10; each wider map is a single seed, seed 56, the only seed trained. $\pm$: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the best ν on that row. MMLU-Pro is the macro-average over its 14 categories and ARC-C the micro-average, both lenient generate-and-parse; RULER is recall averaged over 4k–32k; LongMemEval is judge accuracy pooled over its oracle, 8k, 16k and 32k rungs.

	native		translated, R^2 assignment			
Benchmark	small	large	$\nu = 1$	$\nu = 2$	$\nu = 4$	$\nu = 8$
$L \to S$, Qwen3-8B $\to$ Qwen3-0.6B						
MMLU-Pro	$0.234_{\pm 0.011}$	$0.616_{\pm 0.013}$	$0.208_{\pm 0.011}$	$\underline{\mathbf{0.367}}_{\pm 0.012}$	$\underline{0.274}_{\pm 0.012}$	$\underline{0.320}_{\pm 0.012}$
ARC-C	$0.564_{\pm 0.014}$	$0.939_{\pm 0.007}$	$\underline{0.659}_{\pm 0.014}$	$\underline{\mathbf{0.838}}_{\pm 0.011}$	$\underline{0.826}_{\pm 0.011}$	$\underline{0.832}_{\pm 0.011}$
RULER	$0.834_{\pm 0.006}$	$0.997_{\pm 0.001}$	$0.286_{\pm 0.008}$	$0.431_{\pm 0.009}$	$0.453_{\pm 0.009}$	$\mathbf{0.576}_{\pm 0.008}$
LongMemEval	$0.333_{\pm 0.019}$	$0.652_{\pm 0.019}$	$0.289_{\pm 0.019}$	$\underline{0.337}_{\pm 0.020}$	$\underline{0.368}_{\pm 0.020}$	$\underline{\mathbf{0.392}}_{\pm 0.021}$
$L \to S$, Qwen3-4B $\to$ Qwen3-1.7B						
MMLU-Pro	$0.386_{\pm 0.013}$	$0.546_{\pm 0.013}$	$\mathbf{0.286}_{\pm 0.011}$	$0.211_{\pm 0.011}$	$0.250_{\pm 0.011}$	$0.281_{\pm 0.012}$
ARC-C	$0.772_{\pm 0.012}$	$0.919_{\pm 0.008}$	$\underline{0.811}_{\pm 0.011}$	$0.725_{\pm 0.013}$	$\underline{0.784}_{\pm 0.012}$	$\underline{\mathbf{0.823}}_{\pm 0.011}$
RULER	$0.958_{\pm 0.004}$	$0.998_{\pm 0.001}$	$0.404_{\pm 0.009}$	$0.539_{\pm 0.009}$	$0.645_{\pm 0.007}$	$\mathbf{0.709}_{\pm 0.007}$
LongMemEval	$0.470_{\pm 0.021}$	$0.632_{\pm 0.020}$	$0.410_{\pm 0.019}$	$0.413_{\pm 0.020}$	$0.409_{\pm 0.019}$	$\mathbf{0.461}_{\pm 0.020}$
$S \to L$, Qwen3-0.6B $\to$ Qwen3-8B						
MMLU-Pro	$0.234_{\pm 0.011}$	$0.616_{\pm 0.013}$	$\underline{0.245}_{\pm 0.011}$	$\underline{0.261}_{\pm 0.012}$	$\underline{0.250}_{\pm 0.011}$	$\underline{\mathbf{0.287}}_{\pm 0.012}$
ARC-C	$0.564_{\pm 0.014}$	$0.939_{\pm 0.007}$	$0.522_{\pm 0.015}$	$0.561_{\pm 0.015}$	$0.549_{\pm 0.015}$	$\mathbf{0.562}_{\pm 0.014}$
RULER	$0.834_{\pm 0.006}$	$0.997_{\pm 0.001}$	$0.633_{\pm 0.008}$	$0.714_{\pm 0.007}$	$0.728_{\pm 0.007}$	$\mathbf{0.810}_{\pm 0.006}$
LongMemEval	$0.333_{\pm 0.019}$	$0.650_{\pm 0.020}$	$0.340_{\pm 0.020}$	$\underline{0.404}_{\pm 0.019}$	$\underline{0.385}_{\pm 0.019}$	$\underline{\mathbf{0.416}}_{\pm 0.020}$
$S \to L$, Qwen3-1.7B $\to$ Qwen3-4B						
MMLU-Pro	$0.386_{\pm 0.013}$	$0.546_{\pm 0.013}$	$0.297_{\pm 0.012}$	$0.348_{\pm 0.013}$	$0.346_{\pm 0.013}$	$\underline{\mathbf{0.405}}_{\pm 0.013}$
ARC-C	$0.772_{\pm 0.012}$	$0.919_{\pm 0.008}$	$0.717_{\pm 0.013}$	$0.763_{\pm 0.012}$	$0.746_{\pm 0.013}$	$\mathbf{0.770}_{\pm 0.012}$
RULER	$0.958_{\pm 0.004}$	$0.998_{\pm 0.001}$	$0.594_{\pm 0.009}$	$0.662_{\pm 0.008}$	$0.778_{\pm 0.006}$	$\mathbf{0.794}_{\pm 0.006}$
LongMemEval	$0.470_{\pm 0.021}$	$0.632_{\pm 0.020}$	$0.465_{\pm 0.020}$	$0.454_{\pm 0.020}$	$0.490_{\pm 0.020}$	$\underline{\mathbf{0.506}}_{\pm 0.020}$

I GEMMA-3-4B/12B-IT

Table 14 reports cache translation on the Gemma-3-4B/12B-it pair with one source layer per target layer, Table 15 widens the maps to $\nu \in \{1, 2, 4\}$ source layers, and Figure 24 draws the mean score of both.

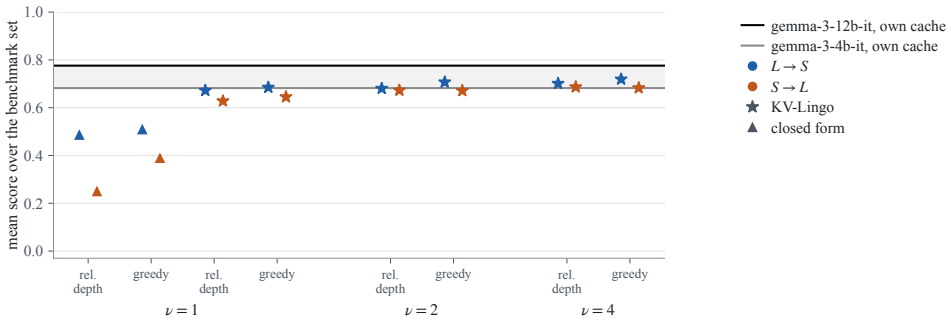


Figure 24: Cache translation with KV-Lingo (head-mixing) with the pair Gemma-3-4B/12B-it, both directions, with $\nu \in \{1, 2, 4\}$ source layers per target layer. Averaged score over MMLU-Pro, ARC-C, LongBench-v2, GSM8K, CoQA, IFEval, XQuAD, RULER at 4k, 8k and 16k, and RepLiQA. MT-Bench-101 is left out of the average; Tables 14 and 15 report it. LongBench-v2 is kept in the average although Table 14 omits it, since its native anchors sit near chance at this pair's 16k budget. The $\nu = 1$ closed-form fits and self-distilled depth map are those of Table 14; every other self-distilled point is a single map from Table 15. Every self-distilled point, the $\nu = 1$ depth map included, is seed 42; error bars: standard error over evaluation items (Appendix E.1).

Table 14: **Cross-layer cache translation on the Gemma-3-4B/12B-it pair**, 34 against 48 layers and a KV width of 1024 against 2048, both directions. One rectangular map covers the stack; each target layer reads a single source layer chosen inside its own attention group, either by relative depth or greedy match, and each assignment is reported as its closed-form fit and after self-distillation, the latter seed 42. ±: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the best of the four translators on that row. MMLU-Pro and ARC-C are lenient generate-and-parse accuracy, GSM8K flexible exact match, CoQA and XQuAD token-F1, IFEval prompt-level strict accuracy, RepLiQA judged accuracy on the answerable questions, MT-Bench-101 a 1–10 judge score, and RULER is recall at each of three prefix lengths. Gemma-3's chat template carries no reasoning switch, so the suite has a single mode rather than the reasoning-on/off pair reported elsewhere. Three points to note. The 4B outscores the 12B on RepLiQA, so on that row the small model is the harder anchor. On the $S \to L$ MT-Bench-101 row both maps print 9.03, but only the depth map clears the 4B's 8.883 by one standard error ($9.033 - 0.148 = 8.884$, against $9.031 - 0.153 = 8.878$). And the two blocks' native anchors come from byte-identical generations that on IFEval alone score differently in the third decimal (0.715 against 0.717 for the 4B, 0.762 against 0.765 for the 12B), because IFEval's language-detection check was unseeded; the block's own pair is the one its rows are read against. LongBench-v2 is omitted: at the 16384-token budget this pair was evaluated at, 94% of its items are truncated and its $L \to S$ native anchor sits half a standard deviation above the four-way chance floor, so no delta against native is readable from it.

	native		closed form		self-distilled	
Benchmark	small	large	depth	greedy	depth	greedy
$L \to S$, Gemma-3-4B/12B-it						
MMLU-Pro	$0.400_{\pm 0.013}$	$0.584_{\pm 0.013}$	$0.315_{\pm 0.012}$	$0.198_{\pm 0.011}$	$\underline{\mathbf{0.429}}_{\pm 0.013}$	$0.391_{\pm 0.013}$
ARC-C	$0.791_{\pm 0.012}$	$0.933_{\pm 0.007}$	$0.665_{\pm 0.014}$	$0.682_{\pm 0.014}$	$\underline{\mathbf{0.830}}_{\pm 0.011}$	$0.773_{\pm 0.012}$
GSM8K	$0.871_{\pm 0.021}$	$0.883_{\pm 0.020}$	$0.738_{\pm 0.028}$	$0.680_{\pm 0.029}$	$0.855_{\pm 0.022}$	$\mathbf{0.871}_{\pm 0.021}$
CoQA	$0.607_{\pm 0.027}$	$0.657_{\pm 0.026}$	$\underline{0.644}_{\pm 0.026}$	$\underline{0.608}_{\pm 0.027}$	$0.541_{\pm 0.027}$	$\underline{\mathbf{0.706}}_{\pm 0.025}$
IFEval	$0.715_{\pm 0.019}$	$0.762_{\pm 0.018}$	$0.396_{\pm 0.021}$	$0.442_{\pm 0.021}$	$0.590_{\pm 0.021}$	$\mathbf{0.610}_{\pm 0.021}$
XQuAD	$0.443_{\pm 0.007}$	$0.629_{\pm 0.006}$	$0.243_{\pm 0.007}$	$0.210_{\pm 0.007}$	$0.428_{\pm 0.007}$	$\underline{\mathbf{0.522}}_{\pm 0.008}$
RepLiQA	$0.908_{\pm 0.015}$	$0.899_{\pm 0.016}$	$0.570_{\pm 0.026}$	$0.804_{\pm 0.021}$	$0.858_{\pm 0.018}$	$\underline{\mathbf{0.916}}_{\pm 0.015}$
MT-Bench-101	$8.88_{\pm 0.16}$	$9.29_{\pm 0.12}$	$8.57_{\pm 0.18}$	$8.47_{\pm 0.17}$	$8.64_{\pm 0.19}$	$\underline{\mathbf{8.96}}_{\pm 0.17}$
RULER, 4k	$0.983_{\pm 0.006}$	$1.000_{\pm 0.000}$	$0.688_{\pm 0.018}$	$0.710_{\pm 0.016}$	$0.955_{\pm 0.009}$	$\mathbf{0.981}_{\pm 0.007}$
RULER, 8k	$0.823_{\pm 0.014}$	$0.964_{\pm 0.008}$	$0.469_{\pm 0.020}$	$0.578_{\pm 0.017}$	$0.812_{\pm 0.015}$	$\underline{\mathbf{0.832}}_{\pm 0.015}$
RULER, 16k	$0.700_{\pm 0.012}$	$0.915_{\pm 0.012}$	$0.341_{\pm 0.019}$	$0.446_{\pm 0.017}$	$0.682_{\pm 0.014}$	$\underline{\mathbf{0.728}}_{\pm 0.014}$
$S \to L$, Gemma-3-4B/12B-it						
MMLU-Pro	$0.400_{\pm 0.013}$	$0.584_{\pm 0.013}$	$0.242_{\pm 0.011}$	$0.256_{\pm 0.011}$	$\mathbf{0.386}_{\pm 0.013}$	$0.380_{\pm 0.013}$
ARC-C	$0.791_{\pm 0.012}$	$0.933_{\pm 0.007}$	$0.484_{\pm 0.015}$	$0.416_{\pm 0.014}$	$\mathbf{0.750}_{\pm 0.013}$	$0.747_{\pm 0.013}$
GSM8K	$0.871_{\pm 0.021}$	$0.883_{\pm 0.020}$	$0.508_{\pm 0.031}$	$0.523_{\pm 0.031}$	$0.855_{\pm 0.022}$	$\mathbf{0.879}_{\pm 0.020}$
CoQA	$0.607_{\pm 0.027}$	$0.657_{\pm 0.026}$	$0.404_{\pm 0.026}$	$0.401_{\pm 0.027}$	$0.518_{\pm 0.028}$	$\mathbf{0.538}_{\pm 0.028}$
IFEval	$0.717_{\pm 0.019}$	$0.765_{\pm 0.018}$	$0.372_{\pm 0.021}$	$0.399_{\pm 0.021}$	$0.604_{\pm 0.021}$	$\mathbf{0.634}_{\pm 0.021}$
XQuAD	$0.443_{\pm 0.007}$	$0.629_{\pm 0.006}$	$0.071_{\pm 0.004}$	$0.086_{\pm 0.004}$	$\mathbf{0.447}_{\pm 0.008}$	$0.427_{\pm 0.008}$
RepLiQA	$0.908_{\pm 0.015}$	$0.899_{\pm 0.016}$	$0.176_{\pm 0.020}$	$0.651_{\pm 0.025}$	$0.821_{\pm 0.020}$	$\mathbf{0.857}_{\pm 0.019}$
MT-Bench-101	$8.88_{\pm 0.16}$	$9.29_{\pm 0.12}$	$8.38_{\pm 0.22}$	$8.41_{\pm 0.21}$	$\underline{\mathbf{9.03}}_{\pm 0.15}$	$9.03_{\pm 0.15}$
RULER, 4k	$0.983_{\pm 0.006}$	$1.000_{\pm 0.000}$	$0.095_{\pm 0.012}$	$0.475_{\pm 0.020}$	$0.945_{\pm 0.011}$	$\mathbf{0.958}_{\pm 0.010}$
RULER, 8k	$0.823_{\pm 0.014}$	$0.964_{\pm 0.008}$	$0.080_{\pm 0.011}$	$0.409_{\pm 0.020}$	$0.733_{\pm 0.014}$	$\mathbf{0.777}_{\pm 0.014}$
RULER, 16k	$0.700_{\pm 0.012}$	$0.915_{\pm 0.012}$	$0.113_{\pm 0.014}$	$0.389_{\pm 0.017}$	$0.661_{\pm 0.013}$	$\mathbf{0.679}_{\pm 0.013}$

Table 15: **Wider maps on the Gemma-3-4B/12B-it pair**: each target layer reads ν source layers inside its own attention group, both directions, self-distilled. *depth* extends the relative-depth assignment of Table 14 to ν neighbouring layers; *greedy* picks the ν source layers by orthogonal matching pursuit on the cache reconstruction, and at $\nu = 1$ is therefore a different assignment from the greedy match of Table 14. Each cell is a single map (seed 42), trained at the learning rate that won its own cell's sweep. $\pm$: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the best of the five maps on that row. Native anchors, metrics and eval sets are those of Table 14. No closed-form fits were evaluated at $\nu > 1$.

	native		depth		greedy		
Benchmark	small	large	$\nu = 2$	$\nu = 4$	$\nu = 1$	$\nu = 2$	$\nu = 4$
$L \to S$, Gemma-3-4B/12B-it							
MMLU-Pro	$0.400_{\pm 0.013}$	$0.584_{\pm 0.013}$	$\underline{\mathbf{0.431}}_{\pm 0.013}$	$\underline{0.429}_{\pm 0.013}$	$\underline{0.416}_{\pm 0.013}$	$\underline{0.412}_{\pm 0.013}$	$\underline{0.429}_{\pm 0.013}$
ARC-C	$0.791_{\pm 0.012}$	$0.933_{\pm 0.007}$	$\underline{\mathbf{0.846}}_{\pm 0.011}$	$\underline{0.838}_{\pm 0.011}$	$\underline{0.832}_{\pm 0.011}$	$\underline{0.828}_{\pm 0.011}$	$\underline{0.841}_{\pm 0.011}$
GSM8K	$0.871_{\pm 0.021}$	$0.883_{\pm 0.020}$	$0.844_{\pm 0.023}$	$0.867_{\pm 0.021}$	$0.863_{\pm 0.022}$	$0.863_{\pm 0.022}$	$\underline{\mathbf{0.902}}_{\pm 0.019}$
CoQA	$0.607_{\pm 0.027}$	$0.657_{\pm 0.026}$	$0.563_{\pm 0.028}$	$0.556_{\pm 0.028}$	$\underline{0.661}_{\pm 0.026}$	$\underline{\mathbf{0.689}}_{\pm 0.025}$	$\underline{0.686}_{\pm 0.025}$
IFEval	$0.715_{\pm 0.019}$	$0.762_{\pm 0.018}$	$0.625_{\pm 0.021}$	$\mathbf{0.654}_{\pm 0.021}$	$0.591_{\pm 0.021}$	$0.632_{\pm 0.021}$	$0.643_{\pm 0.021}$
XQuAD	$0.443_{\pm 0.007}$	$0.629_{\pm 0.006}$	$\underline{0.484}_{\pm 0.007}$	$\underline{0.504}_{\pm 0.007}$	$\underline{0.448}_{\pm 0.007}$	$\underline{\mathbf{0.577}}_{\pm 0.007}$	$\underline{0.527}_{\pm 0.007}$
RepLiQA	$0.908_{\pm 0.015}$	$0.899_{\pm 0.016}$	$0.885_{\pm 0.017}$	$\underline{0.913}_{\pm 0.015}$	$0.891_{\pm 0.016}$	$0.908_{\pm 0.015}$	$\underline{\mathbf{0.922}}_{\pm 0.014}$
MT-Bench-101	$8.88_{\pm 0.16}$	$9.29_{\pm 0.12}$	$8.75_{\pm 0.18}$	$8.80_{\pm 0.16}$	$\mathbf{8.83}_{\pm 0.17}$	$8.73_{\pm 0.19}$	$8.73_{\pm 0.18}$
RULER, 4k	$0.983_{\pm 0.006}$	$1.000_{\pm 0.000}$	$\underline{0.991}_{\pm 0.004}$	$\underline{0.998}_{\pm 0.001}$	$0.971_{\pm 0.008}$	$\underline{0.989}_{\pm 0.004}$	$\underline{\mathbf{0.999}}_{\pm 0.001}$
RULER, 8k	$0.823_{\pm 0.014}$	$0.964_{\pm 0.008}$	$\underline{0.838}_{\pm 0.015}$	$\underline{0.909}_{\pm 0.013}$	$\underline{0.856}_{\pm 0.015}$	$\underline{0.887}_{\pm 0.013}$	$\underline{\mathbf{0.922}}_{\pm 0.012}$
RULER, 16k	$0.700_{\pm 0.012}$	$0.915_{\pm 0.012}$	$\underline{0.718}_{\pm 0.015}$	$\underline{\mathbf{0.786}}_{\pm 0.014}$	$\underline{0.729}_{\pm 0.014}$	$\underline{0.749}_{\pm 0.014}$	$\underline{0.772}_{\pm 0.014}$
$S \to L$, Gemma-3-4B/12B-it							
MMLU-Pro	$0.400_{\pm 0.013}$	$0.584_{\pm 0.013}$	$\underline{0.430}_{\pm 0.013}$	$\underline{\mathbf{0.439}}_{\pm 0.013}$	$0.404_{\pm 0.013}$	$\underline{0.423}_{\pm 0.013}$	$\underline{0.425}_{\pm 0.013}$
ARC-C	$0.791_{\pm 0.012}$	$0.933_{\pm 0.007}$	$0.788_{\pm 0.012}$	$\mathbf{0.794}_{\pm 0.012}$	$0.758_{\pm 0.013}$	$0.765_{\pm 0.012}$	$0.760_{\pm 0.012}$
GSM8K	$0.871_{\pm 0.021}$	$0.883_{\pm 0.020}$	$0.867_{\pm 0.021}$	$0.887_{\pm 0.020}$	$\underline{0.895}_{\pm 0.019}$	$\underline{\mathbf{0.902}}_{\pm 0.019}$	$0.891_{\pm 0.020}$
CoQA	$0.607_{\pm 0.027}$	$0.657_{\pm 0.026}$	$0.543_{\pm 0.028}$	$0.555_{\pm 0.028}$	$0.508_{\pm 0.028}$	$0.538_{\pm 0.028}$	$\mathbf{0.569}_{\pm 0.028}$
IFEval	$0.717_{\pm 0.019}$	$0.765_{\pm 0.018}$	$\mathbf{0.647}_{\pm 0.021}$	$0.645_{\pm 0.021}$	$0.636_{\pm 0.021}$	$0.641_{\pm 0.021}$	$0.640_{\pm 0.021}$
XQuAD	$0.443_{\pm 0.007}$	$0.629_{\pm 0.006}$	$\underline{0.491}_{\pm 0.008}$	$\underline{0.489}_{\pm 0.008}$	$0.445_{\pm 0.008}$	$\underline{0.475}_{\pm 0.008}$	$\underline{\mathbf{0.499}}_{\pm 0.007}$
RepLiQA	$0.908_{\pm 0.015}$	$0.899_{\pm 0.016}$	$0.841_{\pm 0.019}$	$\mathbf{0.883}_{\pm 0.017}$	$0.754_{\pm 0.023}$	$0.872_{\pm 0.018}$	$0.860_{\pm 0.018}$
MT-Bench-101	$8.88_{\pm 0.16}$	$9.29_{\pm 0.12}$	$8.94_{\pm 0.18}$	$9.03_{\pm 0.16}$	$8.86_{\pm 0.17}$	$8.83_{\pm 0.17}$	$\underline{\mathbf{9.13}}_{\pm 0.15}$
RULER, 4k	$0.983_{\pm 0.006}$	$1.000_{\pm 0.000}$	$0.970_{\pm 0.008}$	$0.983_{\pm 0.006}$	$0.951_{\pm 0.010}$	$0.975_{\pm 0.008}$	$\mathbf{0.984}_{\pm 0.006}$
RULER, 8k	$0.823_{\pm 0.014}$	$0.964_{\pm 0.008}$	$0.808_{\pm 0.013}$	$\underline{0.841}_{\pm 0.013}$	$0.770_{\pm 0.014}$	$0.812_{\pm 0.014}$	$\underline{\mathbf{0.844}}_{\pm 0.014}$
RULER, 16k	$0.700_{\pm 0.012}$	$0.915_{\pm 0.012}$	$\underline{0.723}_{\pm 0.013}$	$\underline{\mathbf{0.747}}_{\pm 0.011}$	$0.686_{\pm 0.013}$	$\underline{0.713}_{\pm 0.012}$	$\underline{0.741}_{\pm 0.012}$

J GEMMA-4-E2B/E4B-IT

Table 16 reports cache translation on the Gemma-4-E2B/E4B-it pair in both directions, and Figure 25 draws its mean scores.

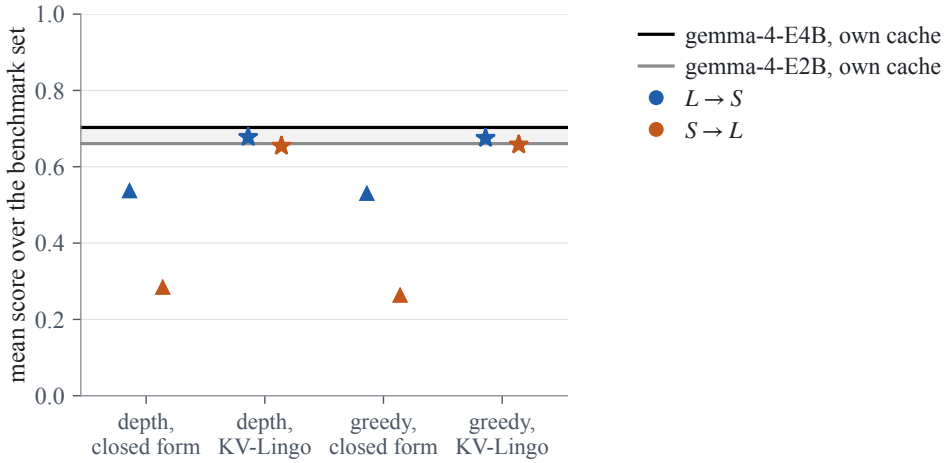


Figure 25: Cache translation with KV-Lingo on the pair Gemma-4-E2B/E4B-it, both directions, for the relative-depth ("depth") and greedy within-group assignments, each as its closed-form fit (triangles) and after self-distillation (stars, seed 42); error bars: standard error over evaluation items (Appendix E.1). Averaged score over MMLU-Pro, ARC-C, LongBench-v2, XQuAD, RULER (mean over 4k–16k) and MT-Bench-101, all reasoning-off, with judge scores rescaled to $[0, 1]$. Per-benchmark scores are in Table 16, which reports MT-Bench-101 and LongBench-v2 reasoning-on instead.

Table 16: **Grouped cache translation on the Gemma-4-E2B/E4B-it pair.** Both native anchors and two within-group source-layer assignments (relative-depth and greedy), each as its closed-form fit and after self-distillation. Trained cells are seed 42. $\pm$: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the best of the four translators on that row. Reasoning-off, except MT-Bench-101 and LongBench-v2, which are reasoning-on (with reasoning off both native anchors fall below LongBench-v2's 25% chance floor). MMLU-Pro and ARC-C are lenient generate-and-parse accuracy; XQuAD is token-F1 over the eleven languages; MT-Bench-101 is a 1–10 judge score; RULER is recall averaged over 4k–16k; LongBench-v2 is accuracy. Note that E2B outscores E4B on MT-Bench-101, so on that row the small model is the harder anchor.

	native		closed form		self-distilled	
Benchmark	small	large	depth	greedy	depth	greedy
$L \to S$, Gemma-4-E2B/E4B-it						
MMLU-Pro	$0.468_{\pm 0.013}$	$0.576_{\pm 0.013}$	$0.346_{\pm 0.012}$	$0.354_{\pm 0.013}$	$\underline{\mathbf{0.481}}_{\pm 0.013}$	$\underline{0.474}_{\pm 0.013}$
ARC-C	$0.897_{\pm 0.009}$	$0.939_{\pm 0.007}$	$0.872_{\pm 0.010}$	$0.879_{\pm 0.010}$	$\underline{\mathbf{0.912}}_{\pm 0.008}$	$\underline{0.905}_{\pm 0.009}$
XQuAD	$0.470_{\pm 0.009}$	$0.546_{\pm 0.009}$	$0.298_{\pm 0.010}$	$0.296_{\pm 0.010}$	$\underline{0.488}_{\pm 0.010}$	$\underline{\mathbf{0.544}}_{\pm 0.010}$
MT-Bench-101	$9.45_{\pm 0.13}$	$9.35_{\pm 0.15}$	$8.94_{\pm 0.19}$	$8.74_{\pm 0.21}$	$9.25_{\pm 0.16}$	$\mathbf{9.36}_{\pm 0.12}$
RULER	$0.996_{\pm 0.001}$	$1.000_{\pm 0.000}$	$0.755_{\pm 0.009}$	$0.717_{\pm 0.009}$	$\mathbf{0.964}_{\pm 0.004}$	$0.931_{\pm 0.005}$
LongBench-v2	$0.304_{\pm 0.023}$	$0.332_{\pm 0.024}$	$0.175_{\pm 0.019}$	$0.190_{\pm 0.020}$	$\mathbf{0.284}_{\pm 0.023}$	$0.266_{\pm 0.022}$
$S \to L$, Gemma-4-E2B/E4B-it						
MMLU-Pro	$0.468_{\pm 0.013}$	$0.576_{\pm 0.013}$	$0.086_{\pm 0.007}$	$0.084_{\pm 0.007}$	$0.446_{\pm 0.013}$	$\mathbf{0.450}_{\pm 0.013}$
ARC-C	$0.897_{\pm 0.009}$	$0.939_{\pm 0.007}$	$0.261_{\pm 0.013}$	$0.116_{\pm 0.009}$	$0.790_{\pm 0.012}$	$\mathbf{0.828}_{\pm 0.011}$
XQuAD	$0.470_{\pm 0.009}$	$0.546_{\pm 0.009}$	$0.012_{\pm 0.002}$	$0.012_{\pm 0.002}$	$\underline{\mathbf{0.509}}_{\pm 0.011}$	$\underline{0.508}_{\pm 0.011}$
MT-Bench-101	$9.45_{\pm 0.13}$	$9.35_{\pm 0.15}$	$7.83_{\pm 0.24}$	$8.00_{\pm 0.23}$	$9.34_{\pm 0.13}$	$\mathbf{9.35}_{\pm 0.13}$
RULER	$0.996_{\pm 0.001}$	$1.000_{\pm 0.000}$	$0.471_{\pm 0.010}$	$0.472_{\pm 0.010}$	$\mathbf{0.972}_{\pm 0.004}$	$0.971_{\pm 0.004}$
LongBench-v2	$0.304_{\pm 0.023}$	$0.332_{\pm 0.024}$	$0.205_{\pm 0.020}$	$0.258_{\pm 0.022}$	$0.284_{\pm 0.023}$	$\mathbf{0.296}_{\pm 0.023}$

K CROSS-TOKENIZER CACHE TRANSLATION: QWEN3-8B/MISTRAL-7B-INSTRUCT-V0.3

This section describes the procedure to apply KV-Lingo translators to the pair Qwen3-8B/Mistral-7B-Instruct-v0.3.

The source prefills its own rendering of the prefix and the translator is applied to its pre-RoPE, pre-normalisation cache token by token, as in the matched case, so the translated cache holds one entry per translated source token. However, we do not apply RoPE canonically to the translated keys, which would induce a mismatch during training between the rotation of the last translated token of the prefix, and the first continuation token. Instead, we assign a target token to every translated token, with the method explained below, and rotate the translated keys according to the position of their assigned target token. That way, some rotations can be identical between translated tokens, and not all target rotations are represented in the set of the translated tokens; but the translated cache has virtually the same number of tokens as the target cache.

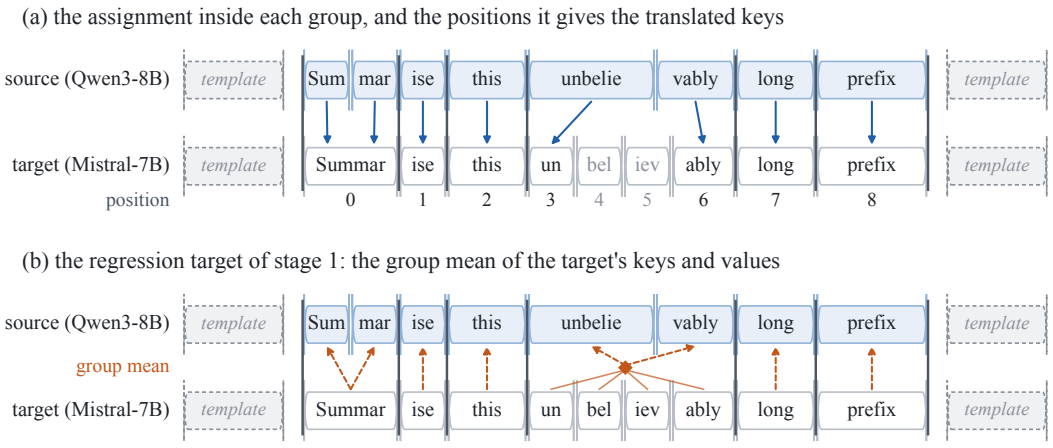


Figure 26: **Cache translation across a tokenizer boundary, on one span of text.** The two models segment *Summarise this unbelievably long prefix* differently; the vertical bars stand at the character offsets where both of them end a token, which delimit the minimal groups. **(a)** Inside a group, the source tokens are spread evenly over the target ones, and the translated keys of a source token are rotated at the position of the target token it is assigned to: a position can carry two translated tokens, another none. **(b)** Stage 1 regresses the keys and values of a source token onto the average of those of the target tokens of its group, which is the assigned token itself in a group holding one token on each side. Boilerplate from the two chat templates is prefilled by the target. The source row is the segmentation Qwen3 actually produces; the target row stands in for a SentencePiece one.

Assigning each source token to a target token. Take a piece of text rendered by the source as the tokens $s_1, \dots, s_p$, by the target as the tokens $t_1, \dots, t_q$, with $p \neq q$ in general. We assign one target token to every source token, as follows. Source tokens are spread evenly over the target ones: for $p \geq 2$, the source token s_i is assigned to $t_{j(i)}$ with $j(i) = 1 + \mathrm{round}\big((i-1)(q-1)/(p-1)\big)$, while a single source token is assigned to t_1; Figure 26(a) shows the result on one span of text. This assignment is non-decreasing. Its two ends are those of the text: s_1 is sent to t_1, s_p to t_q. The position of the assigned target token is the position at which we rotate the translated keys of a source token. Note that some target tokens can be assigned to several source tokens, while others might be assigned to none.

Breaking down the prefix into minimal mismatching groups. We do not apply the rule above to the whole prefix at once. The prefix is first cut at every token at which the source and the target both end at the same character. Two consecutive such offsets delimit a group of tokens covering exactly the same characters on both sides, with at least one token on either side. Cutting at every shared offset makes these groups minimal: none of them can be split into two smaller parts that both cover

Table 17: **Cache translation across a tokenizer boundary**, Qwen3-8B ↔ Mistral-7B-Instruct-v0.3. The *small* model is Mistral-7B-Instruct-v0.3 and the *large* one Qwen3-8B, each shown decoding from its own cache; both blocks are scored on the same evaluation sets, so the two anchor columns repeat. The translator combines the three assigned source layers with one rectangular map per source layer (201M parameters), trained at a single seed. ±: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. MMLU-Pro and ARC-C are lenient generate-and-parse accuracy at a 2048-token budget, GSM8K flexible exact match, IFEval prompt-level strict accuracy, and XQuAD token-F1 pooled over its eleven languages; RepLiQA is judged correctness over the whole 512-item set, answerable and unanswerable together; LongMemEval is judged accuracy averaged over its oracle, 8k and 16k rungs. The *mean* row weights the seven benchmarks equally.

		native		translated
	Benchmark	small	large	self-distilled
$L \to S$	MMLU-Pro	$0.318_{\pm 0.012}$	$0.629_{\pm 0.013}$	$\underline{0.406}_{\pm 0.013}$
	ARC-C	$0.725_{\pm 0.013}$	$0.937_{\pm 0.007}$	$\underline{0.833}_{\pm 0.011}$
	RepLiQA	$0.754_{\pm 0.019}$	$0.795_{\pm 0.018}$	$0.740_{\pm 0.019}$
	LongMemEval	$0.542_{\pm 0.022}$	$0.702_{\pm 0.021}$	$0.455_{\pm 0.022}$
	GSM8K	$0.516_{\pm 0.031}$	$0.922_{\pm 0.017}$	$0.387_{\pm 0.030}$
	IFEval	$0.504_{\pm 0.021}$	$0.804_{\pm 0.017}$	$0.451_{\pm 0.021}$
	XQuAD	$0.549_{\pm 0.007}$	$0.753_{\pm 0.006}$	$0.351_{\pm 0.007}$
	mean	$0.558_{\pm 0.007}$	$0.792_{\pm 0.006}$	$0.517_{\pm 0.007}$
$S \to L$	MMLU-Pro	$0.318_{\pm 0.012}$	$0.629_{\pm 0.013}$	$\underline{0.376}_{\pm 0.013}$
	ARC-C	$0.725_{\pm 0.013}$	$0.937_{\pm 0.007}$	$\underline{0.758}_{\pm 0.013}$
	RepLiQA	$0.754_{\pm 0.019}$	$0.795_{\pm 0.018}$	$0.666_{\pm 0.021}$
	LongMemEval	$0.542_{\pm 0.022}$	$0.702_{\pm 0.021}$	$0.506_{\pm 0.021}$
	GSM8K	$0.516_{\pm 0.031}$	$0.922_{\pm 0.017}$	$\underline{0.852}_{\pm 0.022}$
	IFEval	$0.504_{\pm 0.022}$	$0.805_{\pm 0.017}$	$\underline{0.595}_{\pm 0.021}$
	XQuAD	$0.549_{\pm 0.007}$	$0.753_{\pm 0.006}$	$0.422_{\pm 0.008}$
	mean	$0.558_{\pm 0.007}$	$0.792_{\pm 0.006}$	$\underline{0.596}_{\pm 0.007}$

the same characters on the two sides. Each group is then treated as the text of the previous paragraph, independently of the others. On our fitting corpus, 83% of the groups hold a single token on each side, where the two tokens are simply assigned to each other.

Initialising the translators (stage 1). The closed-form solution of stage 1 (equation 3) takes the keys and values of a source token as input, and the average of the target keys and values in the corresponding minimal mismatching group as regression target (Figure 26(b)); the regression target is the assigned token itself when $p = q = 1$.

Boilerplate tokens are prefilled. Boilerplate tokens from the chat templates are left out of the construction, turn delimiters and generation prompts having no counterpart across the two families; the target prefills them itself.

Results. Table 17 reports both directions on eight benchmarks, and Figure 27 draws their mean.

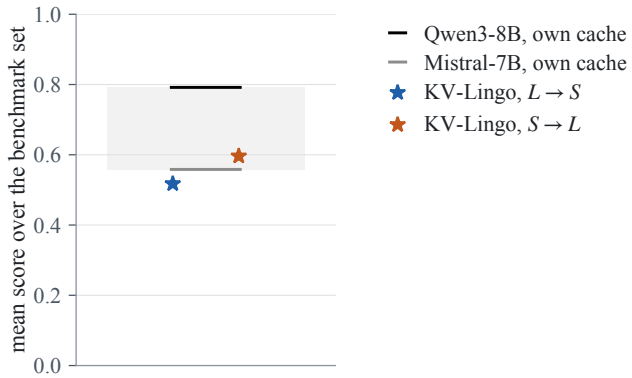


Figure 27: **Cache translation across a tokenizer boundary, Qwen3-8B ↔ Mistral-7B-Instruct-v0.3.** Mean over the seven benchmarks of Table 17, both directions, a single map per direction. Error bars: standard error over evaluation items (Appendix E.1).

L MLA CACHE TRANSLATION

This section complements §4.5; Figure 28 draws the suite mean of Table 18, and Table 19 compares translating the latents with translating the keys and values of an unconverted pair.

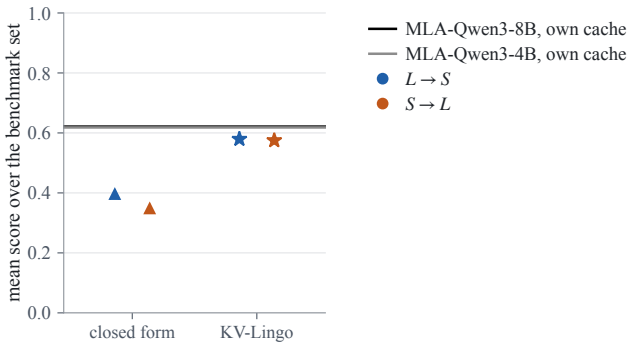


Figure 28: **MLA latent translation, Qwen3-4B/8B converted with CARE.** Mean over the benchmarks of Table 18 other than IFEval, both directions. Triangles are the closed-form fit, stars the self-distilled KV-Lingo map, a single seed. Error bars: standard error over evaluation items (Appendix E.1).

The MLA models. The source and target models have caches of the same shape, holding one 1024-dimensional compressed latent and a 64-dimensional rotary key per token, per layer. The models were converted into MLA models from Qwen3-4B and Qwen3-8B pretrained models, with CARE (Zhou et al., 2026). CARE is calibrated on 256 sequences of 1024 Dolci-Instruct-SFT tokens (Olmo et al., 2025), packed without padding, with a latent rank of 1024, a rotary key of width 64, and the query and key normalisation of Qwen3 kept. Then, as CARE appears to hurt the performance of the native models, we fine-tune them for 120k steps on Dolci-Instruct-SFT. The loss is the cross-entropy of the final assistant turn plus, with weight 1, the KL divergence $\mathrm{KL}(p_{\mathrm{MLA}} \,\|\, p_{\mathrm{GQA}})$ at temperature 1 between the converted model's next-token distribution and that of its own frozen unconverted original, on the same positions. We use AdamW at a learning rate of 10^{-5} with 2% warmup and linear decay, weight decay 10^{-4}, gradient clipping at 1.0 and `bf16`, with one conversation of at most 4096 tokens per GPU on 8 H100 GPUs, and hold out 512 Dolci conversations, selected by content hash, for validation. We report that 120k checkpoint throughout; its native accuracies are the anchor columns of Table 18, 0.759 and 0.819 on ARC-C and 0.306 and 0.351 on MMLU-Pro for the 4B and the 8B. Most of the distance to stock Qwen3-Instruct (0.920 and 0.937 on ARC-C, 0.564 and 0.615 on MMLU-Pro) comes from the recovery fine-tune rather than from the conversion: an unconverted GQA pair given the same fine-tune at the same budget scores 0.852 and 0.900 on ARC-C, 0.417 and 0.486 on MMLU-Pro. The converted pair is read against that matched control.

Table 18: Performance of KV-Lingo on an MLA pair (small: 4B, large: 8B), compared to the performance of each model equipped with its own cache. One map per cell (seed 42). ±: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the better of the two translators on that row. IFEval is prompt-level strict accuracy, and the *mean* row weights the six benchmarks equally. On the mean, KV-Lingo lands about 0.04 below both native models, whose own means differ by only 0.004, so in both directions the switch costs some quality. The closed-form fit beats the self-distilled translator on MMLU-Pro and ARC-C in $L \to S$ and on ARC-C in $S \to L$.

		native		translated	
	Benchmark	small	large	closed form head-mixing	KV-Lingo head-mixing
$L \to S$	MMLU-Pro	$0.306_{\pm 0.012}$	$0.351_{\pm 0.012}$	$\mathbf{0.291}_{\pm 0.012}$	$0.258_{\pm 0.011}$
	ARC-C	$0.759_{\pm 0.013}$	$0.819_{\pm 0.011}$	$\underline{\mathbf{0.785}}_{\pm 0.012}$	$\underline{0.769}_{\pm 0.012}$
	XQuAD	$0.505_{\pm 0.008}$	$0.508_{\pm 0.008}$	$0.381_{\pm 0.008}$	$\mathbf{0.440}_{\pm 0.008}$
	CoQA	$0.639_{\pm 0.027}$	$0.567_{\pm 0.028}$	$0.519_{\pm 0.028}$	$\mathbf{0.577}_{\pm 0.028}$
	IFEval	$0.682_{\pm 0.020}$	$0.691_{\pm 0.020}$	$0.543_{\pm 0.021}$	$\mathbf{0.630}_{\pm 0.021}$
	RepLiQA	$0.881_{\pm 0.017}$	$0.860_{\pm 0.018}$	$0.006_{\pm 0.004}$	$\mathbf{0.854}_{\pm 0.019}$
	mean	$0.629_{\pm 0.007}$	$0.633_{\pm 0.007}$	$0.421_{\pm 0.007}$	$\mathbf{0.588}_{\pm 0.007}$
$S \to L$	MMLU-Pro	$0.306_{\pm 0.012}$	$0.351_{\pm 0.012}$	$0.268_{\pm 0.012}$	$\mathbf{0.292}_{\pm 0.012}$
	ARC-C	$0.759_{\pm 0.013}$	$0.819_{\pm 0.011}$	$\mathbf{0.745}_{\pm 0.013}$	$0.738_{\pm 0.013}$
	XQuAD	$0.505_{\pm 0.008}$	$0.508_{\pm 0.008}$	$0.340_{\pm 0.008}$	$\mathbf{0.436}_{\pm 0.008}$
	CoQA	$0.639_{\pm 0.027}$	$0.567_{\pm 0.028}$	$0.226_{\pm 0.023}$	$\mathbf{0.554}_{\pm 0.028}$
	IFEval	$0.682_{\pm 0.020}$	$0.691_{\pm 0.020}$	$0.429_{\pm 0.021}$	$\mathbf{0.645}_{\pm 0.021}$
	RepLiQA	$0.881_{\pm 0.017}$	$0.860_{\pm 0.018}$	$0.163_{\pm 0.020}$	$\mathbf{0.854}_{\pm 0.019}$
	mean	$0.629_{\pm 0.007}$	$0.633_{\pm 0.007}$	$0.362_{\pm 0.007}$	$\mathbf{0.587}_{\pm 0.007}$

Table 19: **Translating latents against translating the keys and values they compress.** One pair, Qwen3-4B/8B, converted to MLA and, as the baseline, left as it is; both versions given the same recovery fine-tune at two budgets, and their translators fit by the same recipe on the same data, so architecture is the only difference. *Continuation KL* is the held-out KL to the target's own next-token distribution, the quantity stage 2 minimises, reported for the closed-form fit and after self-distillation. *Accuracy − own native* is the translated cache's generate-and-parse accuracy minus the target's own, so 0 is a lossless translation and positive means the translation helped. One seed per cell (seed 42). ±: standard error over evaluation items, paired between the translated and the native run (Appendix E.1); the KLs carry none, as per-sequence KLs were not stored. Because the KL is measured against each target's own cache and the target differs between the two directions, the columns are commensurable in meaning rather than against a fixed reference, but the closed-form pair, which involves no optimiser, separates the two versions the same way and by more.

		continuation KL				accuracy − own native			
		closed form		self-distilled		MMLU-Pro		ARC-C	
Recovery	Dir.	latent	K/V	latent	K/V	latent	K/V	latent	K/V
60k	$L \to S$	2.613	0.117	0.103	0.023	$+0.011_{\pm 0.013}$	$+0.004_{\pm 0.013}$	$+0.023_{\pm 0.014}$	$+0.045_{\pm 0.010}$
60k	$S \to L$	0.977	0.098	0.136	0.021	$-0.095_{\pm 0.012}$	$-0.086_{\pm 0.013}$	$-0.090_{\pm 0.014}$	$-0.048_{\pm 0.010}$
120k	$L \to S$	3.033	0.117	0.152	0.021	$-0.049_{\pm 0.013}$	$+0.047_{\pm 0.013}$	$+0.010_{\pm 0.013}$	$+0.053_{\pm 0.010}$
120k	$S \to L$	1.651	0.103	0.109	0.021	$-0.059_{\pm 0.013}$	$-0.087_{\pm 0.013}$	$-0.081_{\pm 0.012}$	$-0.061_{\pm 0.010}$

M COMPARISON WITH DROIDSPEAK

DroidSpeak (Liu et al., 2024a) learns no map: the target recomputes a contiguous group of its own layers from a hidden state of the source, and adopts the source's keys and values verbatim everywhere else, which requires the source and the target caches to have the same shape, and the source and target hidden dimensions to be the same. None of our pairs satisfies both conditions. We therefore adapt DroidSpeak to Qwen3-0.6B/1.7B, whose caches have the same shape (28 layers, 8 key-value heads of dimension 128) but whose hidden widths differ (1024 vs. 2048). The source's embedding cache therefore cannot be reused, so the recomputed group must start at layer 0: the target prefills layers 0 to $j-1$ itself and reuses the source's cache for layers j to 27. We pick j with DroidSpeak's own offline profiler, which selects the cheapest group closing all but 5% of the output-KL gap between verbatim reuse and the target's own cache. Scoring is our reasoning-off protocol on ARC-C and MMLU-Pro.

Table 20: **DroidSpeak against KV-Lingo, Qwen3-0.6B/1.7B, reasoning-off.** ARC-C (1172 items) and MMLU-Pro (1400 items) accuracy. *Compute* is the target-side cost before decoding starts, in analytic MACs per token as a fraction of the target's own prefill. In $L \to S$ the profiler selects all 28 layers, so DroidSpeak coincides with the target's own cache by construction. KV-Lingo is seed 42. $\pm$: standard error over evaluation items (Appendix E.1); the $L \to S$ verbatim cell is the $j = 0$ point of Table 21, on its subsample.

	$S \to L$ (target 1.7B)			$L \to S$ (target 0.6B)		
	ARC-C	MMLU-Pro	compute	ARC-C	MMLU-Pro	compute
Target, own cache	$0.770_{\pm 0.012}$	$0.379_{\pm 0.013}$	1.00	$0.568_{\pm 0.014}$	$0.234_{\pm 0.011}$	1.00
Source alone	$0.568_{\pm 0.014}$	$0.234_{\pm 0.011}$	—	$0.770_{\pm 0.012}$	$0.379_{\pm 0.013}$	—
DroidSpeak, verbatim ($j = 0$)	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	0.00	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	0.00
DroidSpeak, profiler	$0.770_{\pm 0.012}$	$0.381_{\pm 0.013}$ ($j = 26$)	0.93	$0.568_{\pm 0.014}$	$0.234_{\pm 0.011}$ ($j = 28$)	1.00
Closed-form	$0.497_{\pm 0.015}$	$0.191_{\pm 0.011}$	0.03	$0.640_{\pm 0.014}$	$0.210_{\pm 0.011}$	0.07
KV-Lingo	$0.467_{\pm 0.015}$	$0.238_{\pm 0.011}$	0.03	$0.616_{\pm 0.014}$	$0.219_{\pm 0.011}$	0.07

Table 21: **DroidSpeak accuracy against the recompute boundary j.** The target recomputes layers 0 to $j-1$ and reuses the source's cache above. Subsampled to 400 ARC-C and 420 MMLU-Pro items; the last row is the target's own cache on the same items. $\pm$: standard error over evaluation items (Appendix E.1).

	$S \to L$		$L \to S$	
j	ARC-C	MMLU-Pro	ARC-C	MMLU-Pro
0	$.000_{\pm .000}$	$.000_{\pm .000}$	$.000_{\pm .000}$	$.000_{\pm .000}$
4	$.000_{\pm .000}$	$.000_{\pm .000}$	$.000_{\pm .000}$	$.000_{\pm .000}$
8	$.000_{\pm .000}$	$.000_{\pm .000}$	$.000_{\pm .000}$	$.000_{\pm .000}$
12	$.018_{\pm .007}$	$.005_{\pm .003}$	$.000_{\pm .000}$	$.002_{\pm .002}$
16	$.075_{\pm .013}$	$.021_{\pm .007}$	$.000_{\pm .000}$	$.002_{\pm .002}$
18	$.170_{\pm .019}$	$.050_{\pm .011}$	$.003_{\pm .003}$	$.005_{\pm .003}$
20	$.410_{\pm .025}$	$.136_{\pm .017}$	$.500_{\pm .025}$	$.124_{\pm .016}$
22	$.768_{\pm .021}$	$.360_{\pm .023}$	$.548_{\pm .025}$	$.221_{\pm .020}$
24	$.760_{\pm .021}$	$.419_{\pm .024}$	$.543_{\pm .025}$	$.233_{\pm .021}$
26	$.775_{\pm .021}$	$.393_{\pm .024}$	$.555_{\pm .025}$	$.257_{\pm .021}$
own	$.758_{\pm .021}$	$.410_{\pm .024}$	$.565_{\pm .025}$	$.257_{\pm .021}$

Table 20 gives the headline comparison and Table 21 the full range of recompute boundaries. Three observations stand out. *Verbatim reuse fails completely.* At $j = 0$ the target almost never emits a parseable answer, and accuracy stays near zero, below chance, up to $j = 18$ in both directions. The target's own accuracy is first recovered at $j = 22$, that is, after recomputing 79% of its layers, so the pair offers no cheap operating point. *DroidSpeak's accuracy is capped by the target's.* Everything it does not reuse, it recomputes with the target's own stack, so it cannot exceed the target; in $L \to S$ its

profiler recomputes the whole stack and saves nothing. KV-Lingo instead brings part of the source's advantage along with the cache, and outscores the 0.6B target on ARC-C (0.616 vs. 0.568). *In $S \to L$, the two methods sit at opposite ends of the cost range.* DroidSpeak matches the target's accuracy at 93% of its prefill, whereas KV-Lingo costs 3% but loses accuracy. DroidSpeak does win on transfer size, since it ships only the 2 layers it reuses where a translator ships all 28. DroidSpeak was designed for fine-tuned variants of one base model, whose caches are nearly interchangeable; these results show that the premise does not hold across model sizes.

N DENSE-TO-MOE CACHE TRANSLATION

This section details the experiment of §4.6, where Qwen3-4B prefills and Qwen3-30B-A3B decodes from the translated cache, and the reverse translator, from the MoE's cache to the 4B's.

The pair. Qwen3-30B-A3B (Yang et al., 2025a) replaces the feed-forward block of each of its 48 layers by 128 experts, of which each token is routed to 8: it activates 3.3B of its 30.5B parameters per token. Its attention has 4 key-value heads of dimension 128, so $d_{\mathrm{tgt}} = 512$, against 8 heads of the same dimension in each of the 36 layers of Qwen3-4B, so $d_{\mathrm{src}} = 1024$; the two models share a tokenizer. For the KV-Lingo translator, each target layer reads $\nu = 3$ source layers, assigned by relative depth (§3.1), through one 512×3072 matrix per side, for 151M parameters in all. The reverse translator assigns three of the MoE's layers to each layer of the 4B the same way, through one 1024×1536 matrix per side, for 113M parameters.

Training. The translator is trained as in Appendix E, on the long-context mixture of Appendix C, with one change: stage 1 fits the closed-form map under the attention-weighted norm equation 7, the *attention mass* norm of Appendix O, rather than the Frobenius norm. That fit alone reaches a held-out KL of 0.207. Stage 2 then runs 5000 steps of equation 4 on 8 B200 GPUs, one conversation per GPU and step. Of the learning rates $\{10^{-6}, 3{\times}10^{-6}, 10^{-5}\}$, $3{\times}10^{-6}$ ends at the lowest held-out KL, 0.0634 (Table 23), and is the translator evaluated below. The reverse translator follows the same recipe: its closed-form fit starts at 0.304, and of the learning rates $\{3{\times}10^{-5}, 10^{-4}, 3{\times}10^{-4}\}$, 10^{-4} ends lowest, at 0.0858.

Evaluation. The translated caches are scored by the same harness and on the same items as the Qwen3-4B/8B pair (Table 4), with every benchmark run in both reasoning modes. Table 22 reports the whole suite, and Figure 29 averages it by task group. The reverse translator falls short of the 4B on its own cache in all but four of its 36 cells, and its RULER recall collapses to 0.07–0.28.

Table 22: **Qwen3-4B and Qwen3-30B-A3B: the whole evaluation suite, both directions and both reasoning on and off.** *4B* and *MoE* decode from their own caches; *4B→MoE* is the MoE decoding from the translated cache of the 4B, and *MoE→4B* the 4B decoding from the translated cache of the MoE. Underline: the translated cache beats the 4B on its own cache. Metrics as in Table 8, with GSM8K scored by flexible exact match, and RULER and LongMemEval given per context length rather than averaged. The eight benchmarks above the middle rule have prefixes of at most 3k tokens. The translators are trained on prefixes of up to 16k tokens, so 32k is extrapolation.

	reasoning-off				reasoning-on			
	own cache		translated		own cache		translated	
Benchmark	4B	MoE	4B→MoE	MoE→4B	4B	MoE	4B→MoE	MoE→4B
MMLU-Pro	0.548	0.674	0.525	0.368	0.607	0.736	0.606	0.437
ARC-C	0.918	0.948	0.916	0.849	0.938	0.970	0.959	0.905
GSM8K	0.914	0.930	0.898	0.863	0.820	0.859	0.863	0.836
IFEval	0.771	0.839	0.745	0.632	0.758	0.845	0.795	0.706
CoQA	0.722	0.664	0.675	0.702	0.612	0.625	0.614	0.613
RepLiQA	0.915	0.905	0.934	0.795	0.911	0.938	0.939	0.900
XQuAD	0.613	0.688	0.464	0.444	0.564	0.693	0.598	0.427
MT-Bench-101	8.87	9.40	9.25	8.91	9.06	9.51	9.30	8.93
RULER 4k	1.000	1.000	0.766	0.278	0.968	0.950	0.544	0.174
RULER 8k	1.000	1.000	0.704	0.249	0.953	0.972	0.548	0.171
RULER 16k	0.998	1.000	0.638	0.211	0.927	0.972	0.524	0.150
RULER 32k	0.994	0.998	0.531	0.078	0.937	0.987	0.476	0.070
LongMemEval oracle	0.722	0.784	0.674	0.622	0.830	0.851	0.797	0.720
LongMemEval 8k	0.648	0.717	0.612	0.563	0.817	0.810	0.707	0.643
LongMemEval 16k	0.617	0.658	0.576	0.486	0.699	0.774	0.645	0.512
LongMemEval 32k	0.548	0.576	0.440	0.409	0.586	0.717	0.535	0.419
LongBench-v2	0.405	0.362	0.397	0.388	0.310	0.422	0.345	0.250

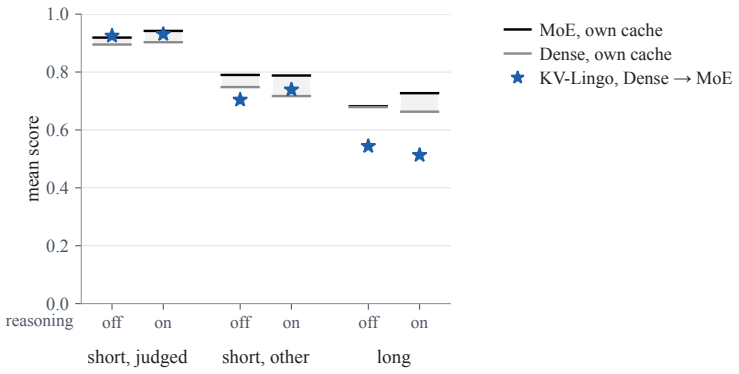


Figure 29: **Dense→MoE mean scores, by task group and reasoning mode.** Qwen3-4B (*Dense*, grey) and Qwen3-30B-A3B (*MoE*, black) decode from their own caches; the stars are the MoE decoding from the KV-Lingo translation of the dense model's cache. *Short, judged* averages the two judge-scored short-context benchmarks, RepLiQA and MT-Bench-101, with the 1–10 judge ratings rescaled to 0–1; *short, other* averages the six others, MMLU-Pro, ARC-C, GSM8K, IFEval, CoQA and XQuAD; *long* averages RULER, LongMemEval and LongBench-v2, with RULER and LongMemEval first averaged over their context lengths. Per-benchmark scores are in Table 22.

A non-linear translator. To test whether linearity costs this pair anything, we add a two-layer MLP branch to the linear map of each target layer and side, $\widehat{K}_i^\ell = T_K^\ell k_i + W_2\,\mathrm{GELU}(W_1 k_i + b_1) + b_2$, where $k_i \in \mathbb{R}^{3d_{\mathrm{src}}}$ stacks $K_{\mathrm{src},i}^{\ell'}$ over $\ell' \in \mathcal{E}_\ell$ and the hidden width is 4096, and likewise for values. W_2 and b_2 start at zero, so stage 2 starts from the same closed-form fit, and the recipe is otherwise unchanged, the learning rate being shared by both parts (Table 23). From the 4B to the MoE, the MLP brings the translator to 1.56B parameters, 10× the linear map, and buys nothing: at its best learning rate it ends at 0.0644, against 0.0634 for the linear map, and takes 1.8× as long to train. In the reverse direction, at 0.87B parameters (8×), it ends 2% below the best linear map, at 0.0842 against 0.0858, and takes 1.5× as long to train.

Table 23: **Linear against non-linear translators**, Qwen3-4B and Qwen3-30B-A3B, both directions. Held-out KL of equation 4 during stage 2, one seed per run; step 0 is the closed-form fit, shared by every run of a direction. *Time* is the wall-clock of stage 2 on 8 B200 GPUs. The 4B → MoE MLP at 3×10^{-5} was stopped at step 2000, at 0.150 against 0.073 for the linear map at 3×10^{-6}. In the reverse direction, the MLP's best learning rate is the larger of the two it was run at, so its optimum is not bracketed. **Bold**: the lowest final KL of each direction.

			held-out KL at step				
Translator	Parameters	Learning rate	0	1000	2500	5000	Time
Qwen3-4B → Qwen3-30B-A3B							
linear	151M	10^{-6}	0.207	0.0749	0.0689	0.0659	3.8 h
		3×10^{-6}	0.207	0.0846	0.0706	**0.0634**	3.8 h
		10^{-5}	0.207	0.1154	0.0894	0.0742	3.8 h
linear + MLP	1.56B	10^{-6}	0.207	0.0768	0.0690	0.0654	6.9 h
		3×10^{-6}	0.207	0.0871	0.0729	0.0644	6.9 h
		3×10^{-5}	0.207	0.1818	—	—	—
Qwen3-30B-A3B → Qwen3-4B							
linear	113M	3×10^{-5}	0.304	0.1078	0.0965	0.0921	3.2 h
		10^{-4}	0.304	0.1027	0.0933	0.0858	3.2 h
		3×10^{-4}	0.304	0.1217	0.1097	0.0891	3.2 h
linear + MLP	0.87B	3×10^{-6}	0.304	0.1212	0.1029	0.0989	4.9 h
		3×10^{-5}	0.304	0.0997	0.0892	**0.0842**	4.9 h

Timing. Table 24 times both models and the translation on an Apple M3 Ultra with MLX, in `bf16`, at batch size 1. The MoE's prefill takes 2.2× as long as the 4B's at 64 tokens and 1.07× at 1024,

while it decodes 1–2% faster, and the translator adds 2.6–4.4% to the 4B's prefill in `fp32`, 2.5–3.4% in `bf16`. As in Appendix Y, the translated path's time to first token also counts the MoE's forward pass on the last prompt token, one decoding step, timed as the inverse of its decoding speed. The table prices the two uses of the translator. In a switch, the 4B has already served the conversation and holds the cache, so the MoE's first token comes after the translation and that decoding step: 15.8 ms at 64 tokens and 31.7 ms at 1024, 9.6 to 13.3× sooner than after its own prefill. As a prefill accelerator, the 4B first prefills the prompt itself, so its prefill joins the path: 1.8× sooner at 64 tokens, and no sooner at 1024, where the 4B prefills about as slowly as the MoE. Letting the 4B emit the first token instead, since its prefill predicts it, would remove the MoE's step and bring the first token to the 4B's prefill time, 2.2× sooner than the MoE's at 64 tokens. Our evaluation does not cover that protocol, in which the first token of every answer is the 4B's.

Table 24: **Latency on an Apple M3 Ultra**, using MLX in `bf16`, batch size 1. Decoding speed is averaged over 256 tokens generated after the prompt. In a switch, the MoE's first token comes after the `fp32` translation of the 4B's cache and one decoding step of the MoE; as a prefill accelerator, after the 4B's prefill as well. *Speedup* divides the MoE's prefill by that time.

Prompt tokens	64	128	256	512	1024
Qwen3-4B prefill (ms)	67.9	94.9	140.4	248.7	393.7
Qwen3-30B-A3B prefill (ms)	151.8	188.6	215.8	287.9	421.4
translation, `fp32` (ms)	1.8	3.1	5.7	11.0	17.2
translation, `bf16` (ms)	1.7	2.7	4.7	7.3	13.1
Qwen3-4B decoding (tokens/s)	70.1	70.0	69.6	68.6	67.9
Qwen3-30B-A3B decoding (tokens/s)	71.5	71.3	70.7	69.6	68.9
switch, first token (ms)	15.8	17.1	19.8	25.3	31.7
speedup	9.63×	11.02×	10.89×	11.36×	13.29×
prefill accelerator, first token (ms)	83.6	112.0	160.2	274.0	425.4
speedup	1.81×	1.68×	1.35×	1.05×	0.99×

O MEASURING THE DISTANCE BETWEEN CACHES

Closed-form translators with other norms. Stage 1 of our training recipe measures the reconstruction residual of equation 3 in the Frobenius norm, weighting every position of the prefix equally. However, this choice of norm is not always aligned with downstream performance, as Table 25 shows. We propose the *attention-weighted norm*, which replaces the uniform weighting of the Frobenius norm by the average attention each prefix position receives from the continuation's queries. For $\Delta \in \mathbb{R}^{d\times n}$ with columns $\delta_1, \dots, \delta_n$, we define

$$\|\Delta\|^2_{\text{attn}} \;=\; \sum_{i=1}^{n} \bar{a}_i \,\|\delta_i\|_2^2 \;=\; \big\|\Delta\,\mathrm{diag}(\bar{a})^{1/2}\big\|_F^2, \quad \bar{a}_i \;=\; \frac{1}{m}\sum_{j=1}^{m} a_{j,i}, \tag{7}$$

where $a_{j,i} \;=\; \mathrm{softmax}_i\big(q_j^\top k_i/\sqrt{d_h}\big)$ is the attention that the j-th continuation query q_j places on prefix position i. The norm follows from a first-order expansion of the attention output. The target consumes the translated cache only through attention, which at a continuation query q outputs $\sum_{i=1}^{n} a_i v_i$ at each head. Perturbing the cache by $(\delta k_i, \delta v_i)$ changes that output, to first order, by $\sum_i a_i\,\delta v_i \;+\; \sum_i a_i\Big(\delta s_i - \sum_j a_j\,\delta s_j\Big)v_i$, with $\delta s_i \;=\; q^\top \delta k_i/\sqrt{d_h}$. Both the key and the value residual contribute to the output, weighted by that position's attention weight a_i, and averaging over the continuation's queries gives the weights $\bar{a}$ of equation 7. A prefix token the continuation barely attends to therefore contributes little to the output, regardless of how badly it is reconstructed, whereas the Frobenius norm charges it as much as, for example, an attention sink.

When fitting a closed-form translator (stage 1 of training) with the attention norm instead of the Frobenius norm, we obtain a significantly better off-the-shelf validation loss and downstream performance (Table 26). However, when running self-distillation (i.e., stage 2) from that improved initialisation then absorbs most of that advantage: both initialisations lead to a similar validation loss (Figure 30). For this reason, we keep the uniform fit as our default initialisation, as it is cheaper to compute; while the closed-form translator fitted with the attention-weighted norm can be used without subsequent training in computationally-bound settings. The rest of this appendix reports the comparison in full.

Table 25 shows that a better cache fit does not guarantee better downstream accuracy: post-k-norm translators have the higher R^2 but decode worse.

Table 25: **The MSE objective is not always aligned with downstream performance: the example of pre- versus post-k-norm.** Closed-form translators fitted before and after the target's k-norm, on both Qwen3 pairs in both directions. R^2 is measured against the target's own cache on held-out prefixes of Nemotron-SFT-Instruction-Following-Chat-v2. Accuracy is that of the target decoding from that cache. Better capture point in bold. Post-norm translators have a better R^2 fit, but worse downstream performance: the MSE objective is not always aligned with downstream performance.

	pre-k-norm			post-k-norm		
Source → target	R^2	MMLU-Pro	ARC-C	R^2	MMLU-Pro	ARC-C
Qwen3-1.7B → 0.6B	0.805	**0.257**	**0.671**	**0.852**	0.202	0.591
Qwen3-0.6B → 1.7B	0.797	**0.218**	**0.544**	**0.801**	0.079	0.262
Qwen3-8B → 4B	0.849	**0.541**	**0.930**	**0.882**	0.469	0.859
Qwen3-4B → 8B	0.847	**0.526**	0.912	**0.867**	0.497	**0.914**

The family, and why it is small. Stage 1 of equation 3 charges every position equally: the Frobenius residual is $\sum_i \|\delta_i\|_2^2$ over the columns δ_i of $MK^\ell_{\text{src}} - K^\ell_{\text{tgt}}$. Replacing each term by a quadratic form $\delta_i^\top \Omega_i \delta_i$ with $\Omega_i \succeq 0$ gives one generalised least-squares translator per choice of $\{\Omega_i\}_i$, of which the attention-weighted norm of equation 7 is the case $\Omega_i = \bar{a}_i I$. The obvious choices all collapse. If $\Omega_t \equiv \Omega$ is the same at every token and full rank, it factors out of the normal equations and the minimiser is *exactly* the Frobenius solution, whatever Ω is: whitening the target's cache, or any fixed re-weighting of its coordinates, is a no-op. Only a metric that varies across tokens moves the map. Among those we take the form of equation 7, $\Omega_t = w_t I$, one scalar weight per position, which turns equation 3 into a weighted least-squares problem solved by the same second-moment statistics with w_t inserted in both sums. We compare four families: *Uniform* ($w_t = 1$) is our default.

Sink follows StreamingLLM (Xiao et al., 2024) in up-weighting the first four positions by a factor of $s \in \{2, 4, 8, 16, 32\}$. *Attention mass* takes $w_i = \bar{a}_i$ of equation 7, the attention the target's own heads pay to position i, averaged over the continuation's queries and over heads on the fitting corpus. Its *attention-Fisher* refinement weights by $\bar{a}_i(1 - \bar{a}_i)$, the diagonal of the softmax Jacobian, which discounts a position whose attention is already saturated and therefore cannot respond to a change in its key.

Table 26: **Fitting norms on the Qwen3-1.7B-Base/0.6B-Base pair**, $L \to S$, short prefixes. Left: the closed-form translator on its own, next-token KL against native decoding. Right: the same maps after stage 2, run to convergence (15k steps, one seed, norms paired on a common validation batch). The two columns are measured on different continuation lengths and should be read down, not across.

fitting norm	off-the-shelf KL	after stage 2
Frobenius (uniform)	0.175	0.0195
sink $\times 2$	0.134	0.0196
sink $\times 4$	0.103	0.0194
sink $\times 8$	0.085	0.0195
sink $\times 16$	0.078	0.0196
sink $\times 32$	0.076	0.0198
attention mass	0.064	**0.0186**
attention Fisher	0.064	**0.0187**

Off the shelf, the norm matters a great deal. The norms rank strictly, attention $<$ sink $<$ Frobenius (Table 26, left): the next-token KL against native decoding falls from 0.175 to 0.064, a 63% reduction, and the continuation perplexity gap to native decoding from 1.77 to 0.25 (native 12.87). The Fisher correction buys nothing over raw attention mass, and both beat the sink family at its plateau, so what matters is up-weighting every heavily read key rather than the first few. Uniform cache MSE is essentially unchanged across norms (0.102 against 0.103 on keys): the norm is invisible to reconstruction error and visible only in decoding, one more instance of the mismatch that motivates stage 2.

The effect grows with context. Refitted under the long-context Instruct recipe of our main results ($S \to L$, prefixes packed to 12–16k), the attention-weighted translator cuts the validation KL of equation 4 from 1.49 to 0.26, putting the continuation cross-entropy within 0.04 nats of the target prefilling the context itself, against 1.56 nats for the Frobenius fit. Downstream that separates a usable training-free translator from an unusable one: on RULER the Frobenius translator essentially fails to retrieve (0.10–0.35 recall from 4 to 32k) where the attention-weighted one reaches 0.60–0.84, against 0.72–0.96 for the source decoding on its own cache; on MMLU-Pro and ARC-C it moves from 0.191 and 0.497, below the source's own accuracy, to 0.235 and 0.549, level with it (0.234 and 0.568; the target on its own cache scores 0.379 and 0.770).

Stage 2 absorbs most of the difference. Fine-tuned to convergence, and again at twice the budget so that a head start is not read as a floor, the sink family lands on the Frobenius level while the attention norms sit 3–4% below it (Table 26, right; 0.0183 against 0.0188 at the doubled budget, a gap flat over the second half of training); Figure 30 draws the two endpoint runs. What survives unambiguously is speed: the attention-weighted initialisation reaches in 250 steps the validation KL the Frobenius one needs 5000 steps for. The long-context pair agrees: after stage 2 the two are within 0.03 recall at every RULER length, and the attention norm keeps a small multiple-choice edge ($+0.01$ on MMLU-Pro, $+0.10$ on ARC-C, with the Frobenius-initialised translator failing to terminate on 20% of ARC items against 4%). The attention weights cost one extra attention pass over the fitting corpus, computed by a query-blocked softmax reduction to stay feasible at 16k, and nothing at inference: the translator is the same object.

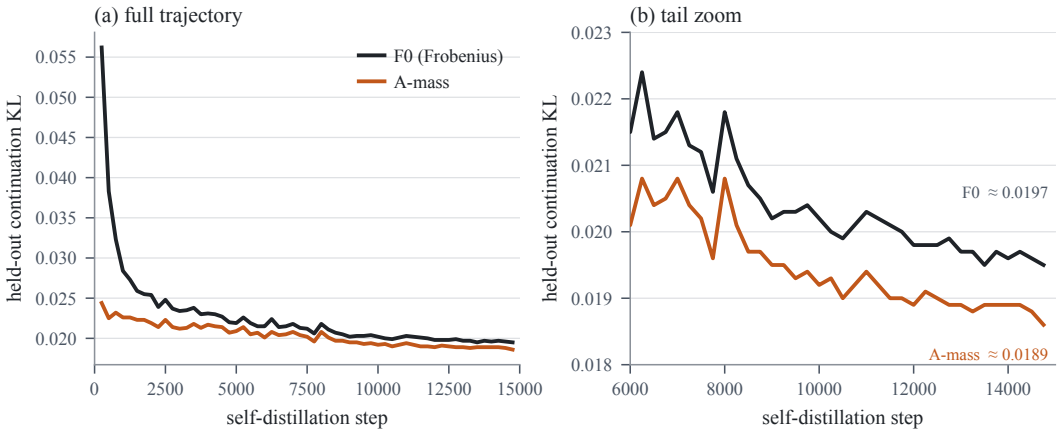


Figure 30: **Stage 2 from two of the eight closed-form initialisations.** Held-out continuation KL against native decoding over 15k self-distillation steps; Qwen3-1.7B-Base $\rightarrow$ Qwen3-0.6B-Base, $L \rightarrow S$, one seed, both runs paired on a common validation batch. F0 is the Frobenius row of Table 26 and A-mass the attention-mass one; the five sink initialisations track F0 and the attention-Fisher one tracks A-mass, so only the two endpoints are drawn and the table carries the rest. (a) shows the full run, (b) zooms the tail, where the attention-weighted initialisation holds about 0.0008 below the level the Frobenius run settles into.

P THE IMPACT OF REASONING ON CACHE TRANSLATION

Figure 31 gives the reasoning-off and reasoning-on scores, and so the reasoning boost, for each model on its own cache and for the translated cache. On the 4B/8B pair the translated cache tracks the target's own boost in both directions. The 0.6B/1.7B pair splits: translating into the larger model amplifies the boost ($+0.102$ against $+0.018$), translating into the smaller one dampens it ($+0.029$ against $+0.045$).

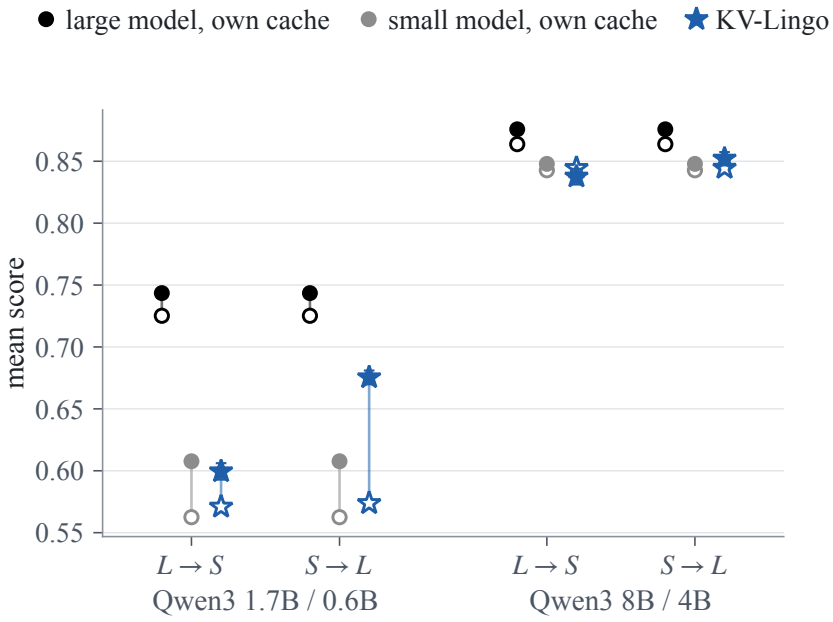


Figure 31: **Reasoning off against reasoning on, both shape-preserving Qwen3 pairs.** Mean score over four equally weighted benchmark families (MMLU-Pro, ARC-C, MT-Bench-101 rescaled from its 1–10 rating, and RULER over 4/8/16k), with reasoning off (open markers) and on (filled), so the segment joining a pair of markers is that condition's reasoning boost. A translated cache is decoded by the small model in $L \to S$ and by the large one in $S \to L$, and is therefore read against grey and black respectively. Translated caches are the mean over three seeds. Error bars, in both reasoning modes and on the native models too, are the standard error over evaluation items (Appendix E.1).

Q MULTI-TURN CACHE TRANSLATION

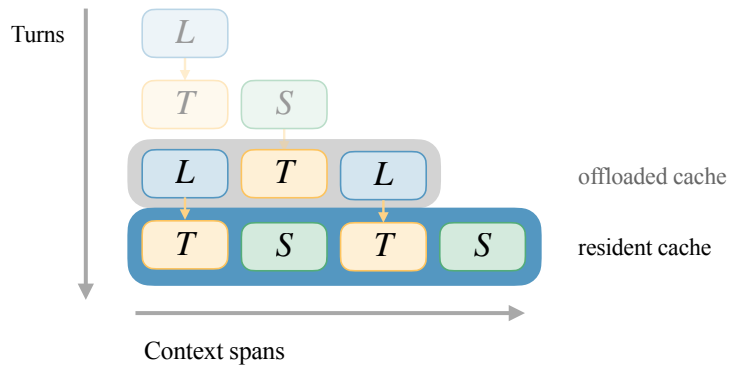


Figure 32: **KV-Lingo multi-turn protocol.** L and S alternate over a shared context. Each row is the cache line of the model that decodes at that turn, and each block one turn's span of it: a block labelled L or S holds the cache that model wrote itself, a block labelled T the translation of a span the other model wrote. At each switch, the outgoing model's line is offloaded, and the incoming model's line is extended by translating only the span written since it last decoded; here, at turn 4, S decodes from its resident line while L's line waits off the accelerator. No span passes through more than one translator.

Figure 32 illustrates the two strategies compared in §4.7: the naive one, which translates the whole prefix again at every switch, and KV-Lingo, where each model retains the spans it decoded and the translated spans of the other model. At a switch, the incoming model rebuilds its cache by translating the other model's newly generated cache since the last switch, so each span is translated at most once.

Keeping one full cache line per model doubles the cache memory of a single model for an equal-shape pair such as Qwen3-4B/8B, since each line covers the whole prefix. Only the active line needs to sit on the accelerator, however. The idle line is not read at all until the next switch, so it gains nothing from being resident, and it can be offloaded. It then moves once per switch, an $O(n)$ copy of a cache that already exists, whereas full re-prefill recomputes that cache with a forward pass whose attention costs $O(n^2)$; the copy should stay far cheaper at the long contexts where memory matters. Both directions of the copy can also be hidden: the outgoing line is offloaded while the incoming model decodes, and the incoming line is reloaded while the new span is being translated. The accelerator therefore holds the same cache as when serving a single model.

Figures 33 and 34 repeat the CoQA comparison of Figure 4 on QuAC and MultiChallenge, with the same pair, protocol and translators. Both agree with CoQA: the naive strategy falls below full re-prefill within a few turns (from turn 3 on QuAC, and in most bins on MultiChallenge), while KV-Lingo stays with full re-prefill to the end of each sweep: turn 8 on QuAC, the 9–10 bin on MultiChallenge.

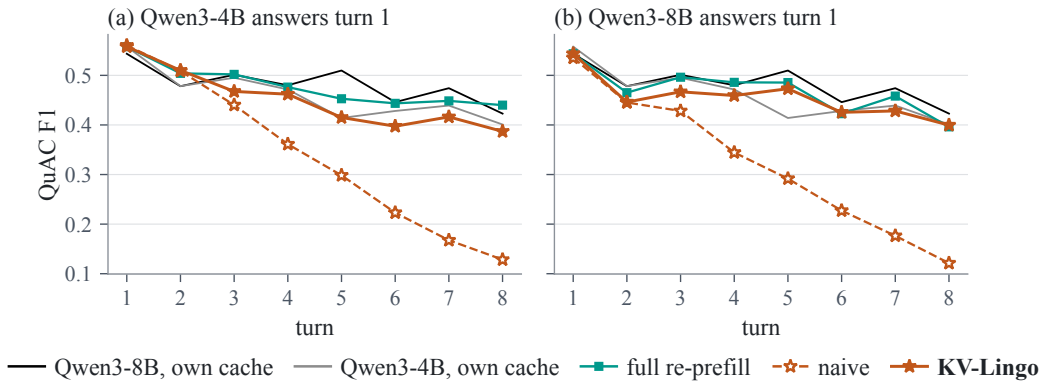


Figure 33: **Multi-turn model switching on QuAC.** Span F1 against turn index, same pair and protocol as Figure 4, 256 frozen conversations truncated to depth 8. Full re-prefill is exact and costs a native prefill over the whole history at every switch; KV-Lingo translates only the span the target's cache lacks. Translated conditions are the mean over three seeds; error bars, on every condition including re-prefill, are the standard error over evaluation items (Appendix E.1).

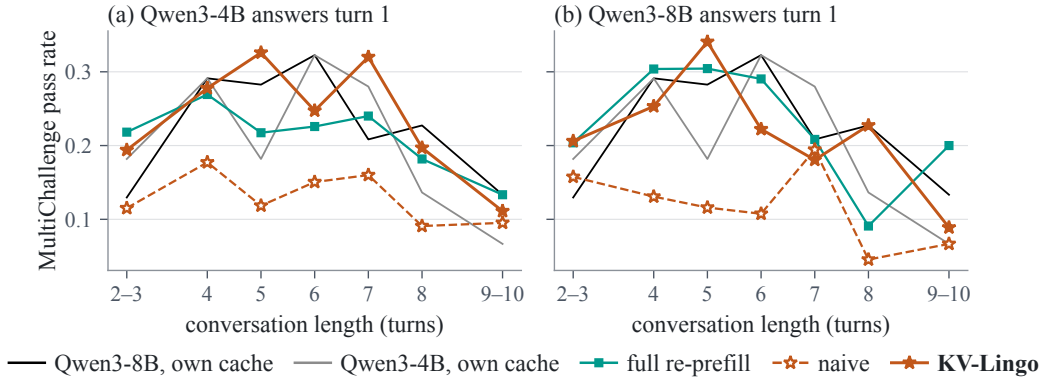


Figure 34: **Multi-turn model switching on MultiChallenge.** Judged pass rate against conversation length, same pair and protocol as Figure 4. The history is pre-written and fixed, so every condition reaches the final prompt over a byte-identical sequence and the two models' caches split the tokens roughly evenly, unlike CoQA and QuAC, where the passage sits in the turn-1 block. Bins are disjoint item sets of the size annotated at the foot of each panel, so the depth axis varies across items rather than within one. Translated conditions are the mean over three seeds; error bars, on every condition including re-prefill, are the standard error over evaluation items (Appendix E.1).

R PRE-k-NORM VS. POST-k-NORM TRANSLATION

Keys can be captured before key normalisation (pre-k-norm, pre-norm in §3.1) or after it and before RoPE (post-k-norm, post-norm in §3.1). We fit the same translator at both capture points, holding everything else fixed (both Qwen3 pairs, both directions, one training recipe, the same evaluation items), and compare them on the whole downstream suite. Only keys are affected: values are captured at the same point either way. Table 27 summarises the difference per evaluation.

Table 27: **Post-k-norm minus pre-k-norm capture, per evaluation.** Self-distilled translator, over the two Qwen3 pairs in both directions. A cell is one pair and direction and, for long-context evaluations, one context length, scored on that evaluation's own metric. $\pm$: standard error of the mean difference over evaluation items, accounting for the items the cells of a row share (Appendix E.1).

evaluation	cells	post-k-norm worse in	mean Δ
RULER, reasoning-off (4–32k)	16	14	$-0.027_{\pm 0.002}$
RULER, reasoning-on (4–32k)	16	12	$-0.026_{\pm 0.005}$
LongMemEval, reasoning-on, four rungs	16	12	$-0.010_{\pm 0.005}$
LongMemEval, reasoning-on, multi-turn at 16k	4	4	$-0.027_{\pm 0.010}$
LongBench-v2, reasoning-off	4	3	$-0.010_{\pm 0.007}$
RepLiQA, reasoning-on, judged accuracy and token-F1	8	4	$-0.004_{\pm 0.004}$
MMLU-Pro and ARC-C	8	4	$+0.021_{\pm 0.003}$

Post-k-norm loses on long context and gains on multiple choice. Pooling the long-context evaluations (RULER and LongBench-v2 scored reasoning-off, LongMemEval and its multi-turn variant reasoning-on), post-k-norm is the worse capture point in 33 of 40 cells, mean $-0.019_{\pm 0.003}$ (sign test $p < 10^{-4}$), and no single cell is catastrophic: the worst is $-0.063_{\pm 0.012}$, on RULER at 16k. Taken one cell at a time, the difference exceeds two standard errors in 12 of the 40 cells, 11 worse and 1 better. On MMLU-Pro and ARC-C the sign flips: post-k-norm is better by $0.021_{\pm 0.003}$ on average, although it wins in only four of the eight cells.

S PRE-k-NORM VERSUS POST-ROPE TRANSLATION

Keys can also be captured after the rotary embedding has been applied (post-RoPE), rather than before key normalisation (pre-k-norm, called pre-norm in §3.1). We fit the same translator at both capture points, holding the model pairs, directions, objective and training recipe fixed, and compare them on the downstream suite (Table 28) and on RULER, at four evaluation context budgets and for translators fitted with three different training prefix caps (Figure 35). Short-context scores are mixed, but at long context pre-k-norm is clearly ahead: on RULER at 32k it reaches 0.65–0.98 against 0.27–0.85 for post-RoPE. Figure 35 shows why: post-RoPE collapses beyond its training prefix cap.

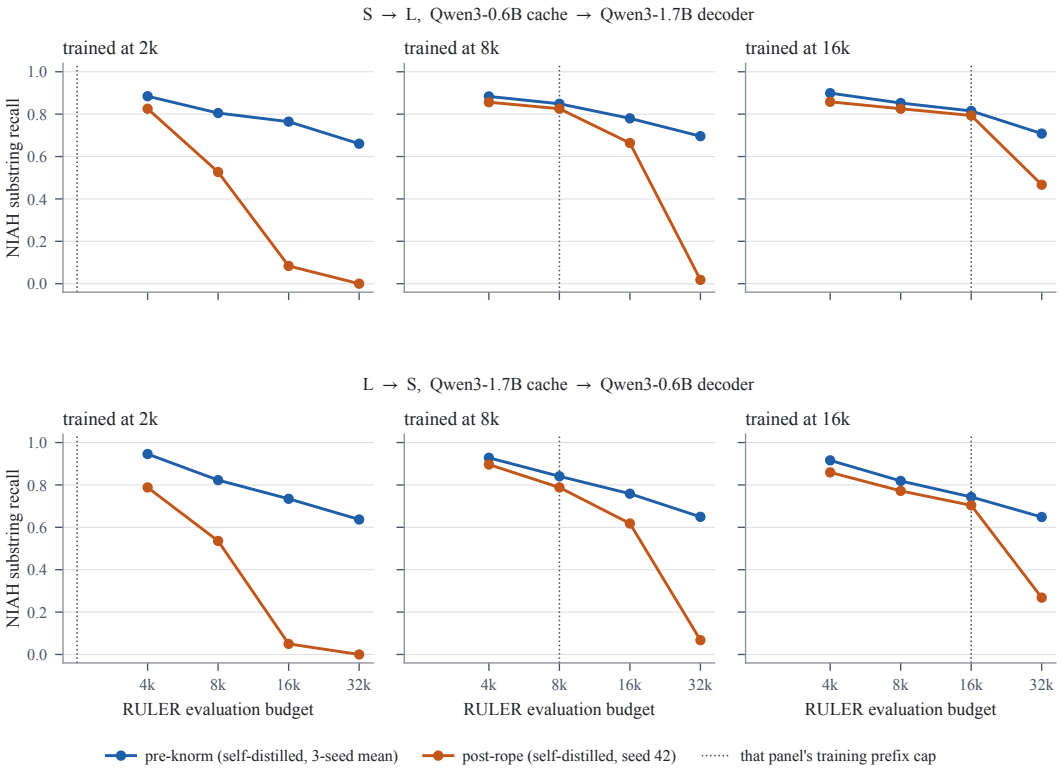


Figure 35: **Where the two capture points diverge: extrapolation past the training prefix cap.** RULER NIAH substring recall against evaluation context budget on the Qwen3-0.6B/1.7B pair, both directions. One panel per training prefix cap; the dotted line marks that cap, so everything to its right is extrapolation. Both curves are the same self-distilled translator fitted at the two capture points, pre-k-norm averaged over three seeds and post-RoPE at seed 42, its only seed; error bars are the standard error over evaluation items (Appendix E.1). Post-RoPE matches pre-k-norm within the cap and falls away beyond it, to near-zero recall at 32k for the 2k and 8k caps, while pre-k-norm degrades smoothly across the whole range.

Table 28: **Pre-k-norm against post-RoPE capture on the evaluation suite**, translators fitted at a 16k training prefix cap, seed 42 (post-RoPE's only seed), reasoning-off except GSM8K. $\pm$: standard error over evaluation items (Appendix E.1). Underline: beats the small model, in $S \to L$ by more than one standard error. **Bold**: the better capture point.

	native		translated	
Benchmark	small	large	pre-k-norm	post-RoPE
$L \to S$, Qwen3-0.6B/1.7B				
MMLU-Pro	$0.234_{\pm 0.011}$	$0.379_{\pm 0.013}$	$0.219_{\pm 0.011}$	$\underline{\mathbf{0.249}}_{\pm 0.011}$
ARC-C	$0.568_{\pm 0.014}$	$0.770_{\pm 0.012}$	$\underline{0.616}_{\pm 0.014}$	$\underline{\mathbf{0.660}}_{\pm 0.014}$
GSM8K	$0.641_{\pm 0.030}$	$0.746_{\pm 0.027}$	$\underline{\mathbf{0.695}}_{\pm 0.029}$	$0.641_{\pm 0.030}$
CoQA	$0.475_{\pm 0.026}$	$0.672_{\pm 0.026}$	$\mathbf{0.469}_{\pm 0.026}$	$0.386_{\pm 0.026}$
IFEval	$0.579_{\pm 0.021}$	$0.662_{\pm 0.020}$	$\mathbf{0.529}_{\pm 0.021}$	$0.497_{\pm 0.022}$
RepLiQA	$0.766_{\pm 0.023}$	$0.921_{\pm 0.014}$	$\underline{\mathbf{0.811}}_{\pm 0.021}$	$\underline{0.790}_{\pm 0.022}$
MT-Bench-101	$6.11_{\pm 0.26}$	$7.95_{\pm 0.22}$	$\mathbf{6.05}_{\pm 0.26}$	$5.90_{\pm 0.28}$
LongMemEval, oracle	$0.396_{\pm 0.025}$	$0.558_{\pm 0.025}$	$\underline{\mathbf{0.486}}_{\pm 0.025}$	$\underline{0.468}_{\pm 0.025}$
LongMemEval, 32k	$0.247_{\pm 0.022}$	$0.365_{\pm 0.024}$	$\underline{\mathbf{0.342}}_{\pm 0.024}$	$0.082_{\pm 0.014}$
RULER, 4k	$0.964_{\pm 0.008}$	$0.994_{\pm 0.004}$	$\mathbf{0.928}_{\pm 0.011}$	$0.859_{\pm 0.012}$
RULER, 32k	$0.717_{\pm 0.013}$	$0.891_{\pm 0.013}$	$\mathbf{0.651}_{\pm 0.011}$	$0.268_{\pm 0.015}$
$L \to S$, Qwen3-4B/8B				
MMLU-Pro	$0.564_{\pm 0.013}$	$0.615_{\pm 0.013}$	$\underline{\mathbf{0.569}}_{\pm 0.013}$	$0.561_{\pm 0.013}$
ARC-C	$0.920_{\pm 0.008}$	$0.937_{\pm 0.007}$	$\underline{\mathbf{0.925}}_{\pm 0.008}$	$0.918_{\pm 0.008}$
GSM8K	$0.824_{\pm 0.024}$	$0.824_{\pm 0.024}$	$\underline{\mathbf{0.863}}_{\pm 0.022}$	$0.824_{\pm 0.024}$
CoQA	$0.721_{\pm 0.024}$	$0.712_{\pm 0.023}$	$0.721_{\pm 0.024}$	$\underline{\mathbf{0.722}}_{\pm 0.023}$
IFEval	$0.787_{\pm 0.018}$	$0.802_{\pm 0.017}$	$0.769_{\pm 0.018}$	$\mathbf{0.771}_{\pm 0.018}$
RepLiQA	$0.916_{\pm 0.015}$	$0.935_{\pm 0.013}$	$\underline{\mathbf{0.935}}_{\pm 0.013}$	$\underline{0.929}_{\pm 0.014}$
MT-Bench-101	$9.00_{\pm 0.19}$	$9.15_{\pm 0.16}$	$\mathbf{8.97}_{\pm 0.19}$	$8.96_{\pm 0.18}$
LongBench-v2	$0.388_{\pm 0.045}$	$0.431_{\pm 0.046}$	$\underline{\mathbf{0.448}}_{\pm 0.046}$	$\underline{\mathbf{0.448}}_{\pm 0.046}$
LongMemEval, oracle	$0.715_{\pm 0.023}$	$0.725_{\pm 0.023}$	$0.704_{\pm 0.023}$	$\mathbf{0.712}_{\pm 0.023}$
LongMemEval, 32k	$0.542_{\pm 0.025}$	$0.535_{\pm 0.025}$	$\underline{\mathbf{0.548}}_{\pm 0.025}$	$0.465_{\pm 0.025}$
RULER, 4k	$1.000_{\pm 0.000}$	$0.998_{\pm 0.003}$	$0.998_{\pm 0.003}$	$\mathbf{1.000}_{\pm 0.000}$
RULER, 32k	$0.995_{\pm 0.003}$	$0.996_{\pm 0.003}$	$\mathbf{0.968}_{\pm 0.008}$	$0.792_{\pm 0.014}$
$S \to L$, Qwen3-0.6B/1.7B				
MMLU-Pro	$0.234_{\pm 0.011}$	$0.379_{\pm 0.013}$	$\mathbf{0.238}_{\pm 0.011}$	$0.216_{\pm 0.011}$
ARC-C	$0.568_{\pm 0.014}$	$0.770_{\pm 0.012}$	$0.467_{\pm 0.015}$	$\mathbf{0.509}_{\pm 0.015}$
GSM8K	$0.641_{\pm 0.030}$	$0.746_{\pm 0.027}$	$\underline{0.711}_{\pm 0.028}$	$\underline{\mathbf{0.738}}_{\pm 0.028}$
CoQA	$0.475_{\pm 0.026}$	$0.672_{\pm 0.026}$	$\underline{0.535}_{\pm 0.027}$	$\underline{\mathbf{0.607}}_{\pm 0.027}$
IFEval	$0.580_{\pm 0.021}$	$0.662_{\pm 0.020}$	$\mathbf{0.538}_{\pm 0.021}$	$0.505_{\pm 0.022}$
RepLiQA	$0.766_{\pm 0.023}$	$0.921_{\pm 0.014}$	$\underline{0.859}_{\pm 0.018}$	$\underline{\mathbf{0.878}}_{\pm 0.017}$
MT-Bench-101	$6.11_{\pm 0.26}$	$7.95_{\pm 0.22}$	$\underline{7.19}_{\pm 0.23}$	$\underline{\mathbf{7.48}}_{\pm 0.22}$
LongMemEval, oracle	$0.396_{\pm 0.025}$	$0.558_{\pm 0.025}$	$0.378_{\pm 0.025}$	$\underline{\mathbf{0.442}}_{\pm 0.025}$
LongMemEval, 32k	$0.247_{\pm 0.022}$	$0.365_{\pm 0.024}$	$\mathbf{0.219}_{\pm 0.021}$	$0.170_{\pm 0.019}$
RULER, 4k	$0.964_{\pm 0.008}$	$0.994_{\pm 0.004}$	$\mathbf{0.901}_{\pm 0.009}$	$0.858_{\pm 0.007}$
RULER, 32k	$0.717_{\pm 0.013}$	$0.891_{\pm 0.013}$	$\mathbf{0.707}_{\pm 0.011}$	$0.467_{\pm 0.016}$
$S \to L$, Qwen3-4B/8B				
MMLU-Pro	$0.564_{\pm 0.013}$	$0.615_{\pm 0.013}$	$\mathbf{0.571}_{\pm 0.013}$	$0.542_{\pm 0.013}$
ARC-C	$0.920_{\pm 0.008}$	$0.937_{\pm 0.007}$	$0.915_{\pm 0.008}$	$\mathbf{0.917}_{\pm 0.008}$
GSM8K	$0.824_{\pm 0.024}$	$0.824_{\pm 0.024}$	$\mathbf{0.840}_{\pm 0.023}$	$0.777_{\pm 0.026}$
CoQA	$0.721_{\pm 0.024}$	$0.712_{\pm 0.023}$	$\mathbf{0.738}_{\pm 0.024}$	$0.733_{\pm 0.023}$
IFEval	$0.787_{\pm 0.018}$	$0.800_{\pm 0.017}$	$0.784_{\pm 0.018}$	$\mathbf{0.786}_{\pm 0.018}$
RepLiQA	$0.916_{\pm 0.015}$	$0.935_{\pm 0.013}$	$\underline{\mathbf{0.949}}_{\pm 0.012}$	$\underline{0.944}_{\pm 0.012}$
MT-Bench-101	$9.00_{\pm 0.19}$	$9.15_{\pm 0.16}$	$9.00_{\pm 0.18}$	$\underline{\mathbf{9.19}}_{\pm 0.16}$
LongBench-v2	$0.388_{\pm 0.045}$	$0.431_{\pm 0.046}$	$0.405_{\pm 0.046}$	$\mathbf{0.414}_{\pm 0.046}$
LongMemEval, oracle	$0.715_{\pm 0.023}$	$0.725_{\pm 0.023}$	$\mathbf{0.699}_{\pm 0.023}$	$0.694_{\pm 0.023}$
LongMemEval, 32k	$0.542_{\pm 0.025}$	$0.535_{\pm 0.025}$	$\mathbf{0.527}_{\pm 0.025}$	$0.458_{\pm 0.025}$
RULER, 4k	$1.000_{\pm 0.000}$	$0.998_{\pm 0.003}$	$0.998_{\pm 0.003}$	$\mathbf{1.000}_{\pm 0.000}$
RULER, 32k	$0.995_{\pm 0.003}$	$0.996_{\pm 0.003}$	$\mathbf{0.979}_{\pm 0.007}$	$0.854_{\pm 0.011}$

T MOVING THE SWITCH POINT

We move the point at which the cache changes hands. Table 29 translates only the document of each long-context task and lets the target prefill the question itself: self-distilled translators stay close to the whole-prompt protocol, the largest drops being 0.07 on LongBench-v2 in $L \to S$ and 0.05 on RULER at 32k in $S \to L$, while the closed-form fit collapses in $L \to S$. Table 30 instead switches after the source's reasoning trace, so the target only writes the answer: the 0.6B target then matches the 1.7B source on MMLU-Pro and ARC-C and exceeds it on GSM8K.

Two readings of Table 29 are worth noting. LongMemEval is scored there on 256 questions per rung, so its absolute values are not comparable with those of Table 8; the difference between translating the whole prefix and the context only is, since it is paired per question. And the large movements in the closed-form columns are empty-answer failures rather than retrieval failures: in every cell that collapses, under either protocol, the closed-form fit leaves a large fraction of the items (up to 0.96) with no parseable answer at all, whereas the trained maps answer on every item of every cell.

Table 29: **Translating the context alone instead of the whole prompt.** The four evaluation sets that carry a document followed by a question about it, on the Qwen3-4B/8B pair, reasoning-off, both directions. *whole* is the protocol used everywhere else in the paper: the source prefills the entire rendered prompt and the map translates all of it. *context* translates the document alone, and the target prefills the question, the assistant header and the empty reasoning block on top of the translated cache before answering. The two protocols are scored on the same items, so they pair per item. Translated cells are the mean over three seeds; $\pm$: standard error over evaluation items (Appendix E.1). *mixed* adds a cross-entropy term to the self-distillation loss (Appendix U). Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the best translator on that row, across both protocols. RULER is substring recall, LongBench-v2 accuracy on the untruncated subset ($n = 116$), LongMemEval and RepLiQA judged accuracy.

	native		closed form		KV-Lingo		mixed	
Benchmark	small	large	whole	context	whole	context	whole	context
$L \to S$, Qwen3-4B/8B								
RULER, 4k	$1.000_{\pm 0.000}$	$0.998_{\pm 0.003}$	$0.994_{\pm 0.004}$	$0.823_{\pm 0.011}$	$\mathbf{0.998}_{\pm 0.003}$	$\mathbf{0.998}_{\pm 0.001}$	$\mathbf{0.998}_{\pm 0.002}$	$0.996_{\pm 0.002}$
RULER, 8k	$1.000_{\pm 0.000}$	$0.998_{\pm 0.003}$	$0.989_{\pm 0.005}$	$0.706_{\pm 0.016}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$	$0.999_{\pm 0.001}$
RULER, 16k	$0.998_{\pm 0.003}$	$0.999_{\pm 0.001}$	$0.960_{\pm 0.009}$	$0.350_{\pm 0.017}$	$\mathbf{0.997}_{\pm 0.001}$	$0.992_{\pm 0.004}$	$0.995_{\pm 0.002}$	$0.983_{\pm 0.004}$
RULER, 32k	$0.995_{\pm 0.003}$	$0.996_{\pm 0.003}$	$0.897_{\pm 0.010}$	$0.168_{\pm 0.011}$	$0.972_{\pm 0.007}$	$0.949_{\pm 0.009}$	$\mathbf{0.977}_{\pm 0.007}$	$0.966_{\pm 0.007}$
LongMemEval oracle	$0.582_{\pm 0.031}$	$0.633_{\pm 0.030}$	$0.504_{\pm 0.031}$	$0.129_{\pm 0.021}$	$0.559_{\pm 0.028}$	$\mathbf{0.570}_{\pm 0.029}$	$\mathbf{0.570}_{\pm 0.027}$	$0.549_{\pm 0.029}$
LongMemEval @8k	$0.594_{\pm 0.031}$	$0.621_{\pm 0.030}$	$0.543_{\pm 0.031}$	$0.066_{\pm 0.016}$	$\underline{\mathbf{0.599}}_{\pm 0.027}$	$0.572_{\pm 0.028}$	$0.587_{\pm 0.026}$	$0.572_{\pm 0.028}$
LongMemEval @16k	$0.465_{\pm 0.031}$	$0.547_{\pm 0.031}$	$0.434_{\pm 0.031}$	$0.000_{\pm 0.000}$	$\underline{\mathbf{0.466}}_{\pm 0.027}$	$0.449_{\pm 0.028}$	$0.464_{\pm 0.027}$	$0.452_{\pm 0.028}$
LongMemEval @32k	$0.410_{\pm 0.031}$	$0.387_{\pm 0.030}$	$0.324_{\pm 0.029}$	$0.012_{\pm 0.007}$	$0.379_{\pm 0.027}$	$0.375_{\pm 0.028}$	$0.372_{\pm 0.026}$	$\mathbf{0.382}_{\pm 0.027}$
LongBench-v2	$0.388_{\pm 0.045}$	$0.431_{\pm 0.046}$	$0.388_{\pm 0.045}$	$0.138_{\pm 0.032}$	$\underline{0.445}_{\pm 0.044}$	$0.376_{\pm 0.043}$	$\underline{\mathbf{0.448}}_{\pm 0.043}$	$0.362_{\pm 0.042}$
RepLiQA	$0.916_{\pm 0.015}$	$0.935_{\pm 0.013}$	$\underline{\mathbf{0.944}}_{\pm 0.012}$	$0.629_{\pm 0.026}$	$\underline{0.934}_{\pm 0.011}$	$\underline{0.924}_{\pm 0.012}$	$\underline{0.942}_{\pm 0.011}$	$\underline{0.936}_{\pm 0.011}$
$S \to L$, Qwen3-4B/8B								
RULER, 4k	$1.000_{\pm 0.000}$	$0.998_{\pm 0.003}$	$0.999_{\pm 0.001}$	$0.917_{\pm 0.012}$	$0.999_{\pm 0.001}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$
RULER, 8k	$1.000_{\pm 0.000}$	$0.998_{\pm 0.003}$	$0.998_{\pm 0.003}$	$0.943_{\pm 0.010}$	$\mathbf{0.999}_{\pm 0.001}$	$\mathbf{0.999}_{\pm 0.001}$	$\mathbf{0.999}_{\pm 0.001}$	$\mathbf{0.999}_{\pm 0.001}$
RULER, 16k	$0.998_{\pm 0.003}$	$0.999_{\pm 0.001}$	$0.978_{\pm 0.007}$	$0.875_{\pm 0.011}$	$\mathbf{0.997}_{\pm 0.001}$	$0.981_{\pm 0.006}$	$\mathbf{0.997}_{\pm 0.002}$	$0.987_{\pm 0.004}$
RULER, 32k	$0.995_{\pm 0.003}$	$0.996_{\pm 0.003}$	$0.864_{\pm 0.012}$	$0.801_{\pm 0.012}$	$\mathbf{0.980}_{\pm 0.006}$	$0.936_{\pm 0.010}$	$0.973_{\pm 0.007}$	$0.920_{\pm 0.010}$
LongMemEval oracle	$0.582_{\pm 0.031}$	$0.633_{\pm 0.030}$	$0.527_{\pm 0.031}$	$0.484_{\pm 0.031}$	$0.564_{\pm 0.028}$	$\underline{\mathbf{0.611}}_{\pm 0.028}$	$0.572_{\pm 0.028}$	$0.579_{\pm 0.028}$
LongMemEval @8k	$0.594_{\pm 0.031}$	$0.621_{\pm 0.030}$	$0.520_{\pm 0.031}$	$0.500_{\pm 0.031}$	$0.555_{\pm 0.027}$	$0.561_{\pm 0.028}$	$0.572_{\pm 0.027}$	$\mathbf{0.599}_{\pm 0.028}$
LongMemEval @16k	$0.465_{\pm 0.031}$	$0.547_{\pm 0.031}$	$0.395_{\pm 0.031}$	$0.391_{\pm 0.031}$	$0.409_{\pm 0.027}$	$\mathbf{0.487}_{\pm 0.028}$	$0.443_{\pm 0.027}$	$0.461_{\pm 0.028}$
LongMemEval @32k	$0.410_{\pm 0.031}$	$0.387_{\pm 0.030}$	$0.297_{\pm 0.029}$	$0.266_{\pm 0.028}$	$0.333_{\pm 0.026}$	$0.365_{\pm 0.027}$	$0.372_{\pm 0.027}$	$\mathbf{0.410}_{\pm 0.027}$
LongBench-v2	$0.388_{\pm 0.045}$	$0.431_{\pm 0.046}$	$0.319_{\pm 0.043}$	$0.371_{\pm 0.045}$	$0.385_{\pm 0.041}$	$\mathbf{0.399}_{\pm 0.043}$	$\mathbf{0.399}_{\pm 0.043}$	$0.376_{\pm 0.044}$
RepLiQA	$0.916_{\pm 0.015}$	$0.935_{\pm 0.013}$	$0.274_{\pm 0.024}$	$\underline{0.930}_{\pm 0.014}$	$\underline{0.936}_{\pm 0.012}$	$\underline{0.938}_{\pm 0.011}$	$\underline{0.937}_{\pm 0.011}$	$\underline{\mathbf{0.940}}_{\pm 0.011}$

Table 30: **Handing the cache over after the reasoning trace rather than at the prompt.** Qwen3-1.7B → Qwen3-0.6B, reasoning-on. Under *prompt* the source prefills the prompt, the map translates it, and the target does its own thinking and answering, the protocol used everywhere else in the paper. Under *trace* the source also generates its chain of thought on its own cache, the map translates the prompt and the trace together, and the target only writes the answer that follows the closing reasoning marker. *text* is the reference for the second protocol: the same target answering from its *own* prefill of prompt + trace, which is what an errorless map would hand it. Translated cells are the mean over three seeds; ±: standard error over evaluation items (Appendix E.1). Underline: the translated cache beats the small model; this pair is read $L \to S$ throughout, so no standard-error margin applies. **Bold**: the best translator on that row, across both switch points. MMLU-Pro and ARC-C are letter accuracy with a parse failure counted incorrect, GSM8K flexible exact match. On MMLU-Pro, the *trace* variants are helped by a format effect worth about 6 accuracy points (asked only to commit an answer, the target emits a parseable one about twice as reliably as either model reasoning and concluding in one turn), so matching the source there is not a sign of better reasoning.

	native			closed form		KV-Lingo	
Benchmark	small	large	text	prompt	trace	prompt	trace
MMLU-Pro	$0.319_{\pm 0.012}$	$0.473_{\pm 0.013}$	$0.508_{\pm 0.013}$	$0.001_{\pm 0.001}$	$\underline{0.335}_{\pm 0.012}$	$0.260_{\pm 0.009}$	$\mathbf{\underline{0.473}}_{\pm 0.013}$
ARC-C	$0.690_{\pm 0.014}$	$0.873_{\pm 0.010}$	$0.875_{\pm 0.010}$	$0.009_{\pm 0.003}$	$0.666_{\pm 0.014}$	$\underline{0.726}_{\pm 0.011}$	$\mathbf{\underline{0.870}}_{\pm 0.010}$
GSM8K	$0.641_{\pm 0.030}$	$0.746_{\pm 0.027}$	$0.848_{\pm 0.023}$	$0.031_{\pm 0.011}$	$0.297_{\pm 0.029}$	$\underline{0.686}_{\pm 0.025}$	$\mathbf{\underline{0.862}}_{\pm 0.020}$

U OBJECTIVE ABLATION: CROSS-ENTROPY AND MIXED OBJECTIVES

We compare the self-distillation loss equation 4 with two alternatives.

Cross-entropy. The cross-entropy (CE) objective fits the translator to the ground-truth continuation rather than to the target's own predictions:

$$\mathcal{L}_{\mathrm{CE}}(T) \;=\; -\,\mathbb{E}_{(x_{1:n},\,y_{1:m})}\,\frac{1}{m}\sum_{j=1}^{m}\log p_{\mathrm{tgt}}\big(y_j \mid T(\mathcal{C}_{\mathrm{src}}(x_{1:n})),\; y_{<j}\big). \tag{8}$$

CE improves strongly on the closed-form fit, but self-distillation beats it even on the perplexity of the corpus both are trained on: on the from-scratch pair, CE leaves a gap of 0.048–0.138 nats to native decoding, against 0.031–0.101 for self-distillation (Table 32). CE is also fragile: at long context or reasoning-on, it drives the decoder off its own manifold (unclosed reasoning blocks, repetition loops), collapsing RULER recall and MT-Bench scores where self-distillation translators keep working.

Remark 2. Chen et al. (2026) train their translators with cross-entropy on continuations generated by the target model, while we use ground-truth continuations, typically generated by more capable language models. Training on target continuations might mitigate the brittleness we report.

Mixed objective. The mixed objective adds the CE term to the self-distillation loss, $\mathcal{L}_{\mathrm{mix}} = \mathcal{L} + \lambda\,\mathcal{L}_{\mathrm{CE}}$ with $\lambda = 1$. Mixed translators are reported in the "mixed" columns of Tables 8 and 9 (both Qwen3 pairs, both directions, the whole evaluation suite) and Table 29 (long-context benchmarks). They perform about as well as self-distillation: neither dominates, and most gaps between the two are within one standard error of their difference over evaluation items.

V CACHE TRANSLATION WITH QWEN3-LIKE MODELS PRETRAINED FROM SCRATCH

The Qwen3 checkpoints we translate between come from a heavily engineered pretraining pipeline whose data and recipe are only partly disclosed (Yang et al., 2025a), so the success of a linear translator could stem from that pipeline rather than from the architecture. To rule this out, we pretrain a 0.6B/1.7B pair from scratch with the Qwen3 architectures, under a recipe we fully control: 35B tokens each, independent initialisations and no distillation. We train the same translator with the same recipe as for the released pair. The closed-form translator suffers on the from-scratch pair, leaving 1.8–4.7× the released pair's perplexity gap, whether because of the recipe or of the much smaller training budget. Self-distillation closes most of that difference, so translatability does not depend on the Qwen3 pretraining pipeline.

Setup. The two models have exactly the Qwen3-0.6B and Qwen3-1.7B architectures ($N = 28$ layers, $H = 8$ key-value heads of width $d_h = 128$, so $d = 1024$), and therefore the same translator shape as the released pair. We pretrain each on 35B tokens of Nemotron-CC (Su et al., 2025), Chinchilla-optimal for the larger model (Hoffmann et al., 2022), with the cl100k_base tokenizer of tiktoken rather than Qwen3's; the two share this corpus and tokenizer but no checkpoint or initialisation. Both follow one recipe: sequences of 4096 tokens in batches of 256 (≈1.05M tokens, ≈33k steps), AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, 2000 warmup steps, a constant learning rate decayed to zero over the last 30% of the steps, gradient clipping at 1.0, a z-loss of weight 10^{-4} and `bf16` mixed precision. The peak learning rate, 3.79×10^{-3}, and the weight decay, 0.2, are set for a base model of width 512 and depth 32 and transferred to each model by CompleteP scaling (Dey et al., 2026). We cannot replicate the Qwen3 post-training, so we compare against the released Qwen3-0.6B/1.7B-Base pair rather than the Instruct checkpoints of our main results, which leaves pretraining as the only difference between the two pairs. Base models have no chat template, so both pairs' translators are trained on Nemotron-CC, with samples of a 256-token prefix and a 256-token continuation: the split translator with head mixing, the identity layer assignment, a closed-form stage 1 and 5000 self-distillation steps. The released pair uses the learning rate selected per direction, the from-scratch pair a single rate. We measure continuation perplexity on a 64-token forced continuation after a 256-token prefix, over 128 documents of Nemotron-CC and 128 of WikiText (Merity et al., 2016), and zero-shot accuracy on BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), the two ARC splits (Clark et al., 2018) and WinoGrande (Sakaguchi et al., 2021), at most 2000 examples each. The from-scratch models are much weaker than the released ones, so each translator is compared with its own pair's native decoding rather than on raw scores.

Results. We compare the log-perplexity of the target's continuation under the translated and the native cache. Over $L \to S$ and $S \to L$, each on Nemotron-CC and WikiText (Figure 36a), the closed-form translator leaves a gap of 0.24, 0.44, 0.30 and 0.56 nats on the from-scratch pair against 0.13, 0.20, 0.06 and 0.14 on the released pair, 1.8–4.7× larger. Self-distillation cuts the from-scratch gap by 5–9×, to 0.031, 0.067, 0.043 and 0.101, which leaves 0.017 to 0.037 nats between the two pairs. Downstream (Figure 36b), the from-scratch translator costs 0.014 and 0.029 average accuracy in the two directions in closed form, and 0.006 and 0.001 after self-distillation, against −0.020 and 0.026 for the released pair's self-distilled translator (a negative value means the translated cache beats the target's own). After self-distillation, neither pair is clearly easier to translate. The measurements of Table 1 are consistent with this: the from-scratch pair is barely harder to fit linearly (R^2_{lin} of 0.752 and 0.651 on keys and values, against 0.766 and 0.658) but has more nonlinear headroom (ρ of 0.935 and 0.903 against 0.965 and 0.945), which a least-squares fit pays for and an objective on the target's outputs can work around.

Head mixing. The two from-scratch models start from independent initialisations, so their heads have no reason to correspond. Table 31 uses this pair to compare KV-Lingo, which mixes heads, with the head-wise MLP translator of Chen et al. (2026), at matched parameters and FLOPs and at 4× the parameters, all trained with the same recipe. KV-Lingo is best in all twelve rows, and on perplexity (Table 32) the head-wise MLP leaves 1.7–3.4× our gap at matched parameters and still 1.5–2.3× at 4×. On released Qwen3 pairs, whose heads are aligned, head-wise translators instead match head-mixing ones (Appendix G).

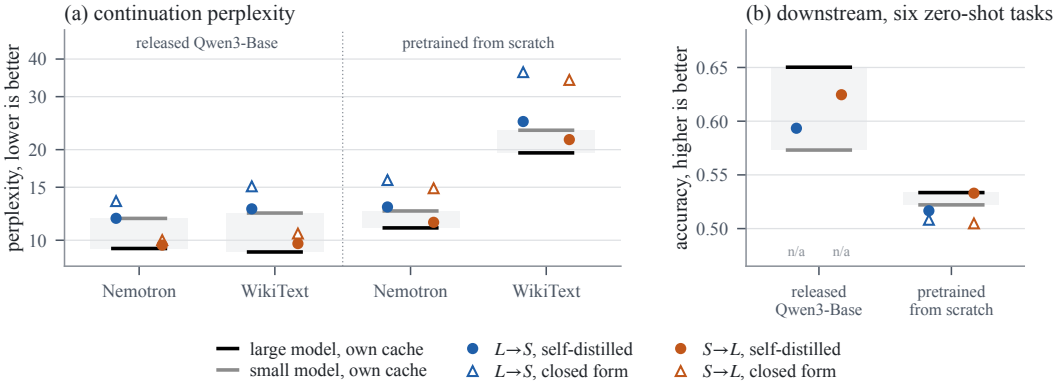


Figure 36: **The same translator and recipe on the released Qwen3-**0.6**B/**1.7**B-Base pair and on a** 0.6**B/**1.7**B pair pretrained from scratch.** Both panels report absolute scores. Each pair's two native models are short horizontal lines (black the larger, grey the smaller) with the band between them shaded, and the cost of translation is the distance from a marker to *its own* target's line (grey for $L \to S$, black for $S \to L$). Triangles are the closed-form translator (stage 1), circles the self-distilled one (stage 2). *(a)* Continuation perplexity on Nemotron-CC, which the translators are fitted on, and on WikiText; the axis is logarithmic, so equal vertical distances are equal log-perplexity gaps in both pairs. *(b)* Accuracy averaged over the six zero-shot tasks; the released pair's closed-form translator was not run on this suite. Every self-distilled translator is a single run (seed 42). Error bars in *(b)* are the standard error over evaluation items (Appendix E.1), binomial from the aggregate accuracies since only aggregate scores were stored; *(a)* has none, as per-document perplexities were not stored.

Table 31: **KV-Lingo against the head-wise MLP translator on a pair with unaligned heads.** The from-scratch 0.6B/1.7B pair of Appendix V, whose two models start from independent initialisations. KV-Lingo against the head-wise gated MLP of Chen et al. (2026) at matched parameters and FLOPs (59.0M against our 58.7M) and at 4× that budget (235.9M), all trained with the same two-stage recipe and 5000 self-distillation steps. Underline: the translated cache beats the small model, and in $S \to L$ by more than one standard error. **Bold**: the best translator on that row. BoolQ and WinoGrande are accuracy, HellaSwag, PIQA and the two ARC splits length-normalised accuracy; zero-shot, 2000 examples per task, one map per cell. ±: standard error over evaluation items (Appendix E.1), binomial from the aggregate accuracy, since per-item outcomes were not stored for these runs. On BoolQ the large model scores below the small one.

		native		translated		
	Benchmark	small	large	KV-Lingo	head-wise MLP 1.0×	head-wise MLP 4.0×
$L \to S$	BoolQ	$0.523_{\pm 0.011}$	$0.496_{\pm 0.011}$	$\underline{\mathbf{0.547}}_{\pm 0.011}$	$0.510_{\pm 0.011}$	$0.519_{\pm 0.011}$
	HellaSwag	$0.457_{\pm 0.011}$	$0.484_{\pm 0.011}$	$\mathbf{0.448}_{\pm 0.011}$	$0.439_{\pm 0.011}$	$0.443_{\pm 0.011}$
	PIQA	$0.691_{\pm 0.011}$	$0.708_{\pm 0.011}$	$\mathbf{0.685}_{\pm 0.011}$	$0.677_{\pm 0.011}$	$0.683_{\pm 0.011}$
	ARC-E	$0.578_{\pm 0.011}$	$0.636_{\pm 0.011}$	$\mathbf{0.557}_{\pm 0.011}$	$0.509_{\pm 0.011}$	$0.529_{\pm 0.011}$
	ARC-C	$0.306_{\pm 0.013}$	$0.338_{\pm 0.014}$	$\mathbf{0.293}_{\pm 0.013}$	$0.262_{\pm 0.013}$	$0.276_{\pm 0.013}$
	WinoGrande	$0.529_{\pm 0.014}$	$0.557_{\pm 0.014}$	$\underline{\mathbf{0.534}}_{\pm 0.014}$	$0.523_{\pm 0.014}$	$0.525_{\pm 0.014}$
$S \to L$	BoolQ	$0.523_{\pm 0.011}$	$0.496_{\pm 0.011}$	$\mathbf{0.511}_{\pm 0.011}$	$0.419_{\pm 0.011}$	$0.458_{\pm 0.011}$
	HellaSwag	$0.457_{\pm 0.011}$	$0.484_{\pm 0.011}$	$\mathbf{0.465}_{\pm 0.011}$	$0.462_{\pm 0.011}$	$0.460_{\pm 0.011}$
	PIQA	$0.691_{\pm 0.011}$	$0.708_{\pm 0.011}$	$\underline{\mathbf{0.706}}_{\pm 0.011}$	$0.698_{\pm 0.011}$	$\underline{0.702}_{\pm 0.011}$
	ARC-E	$0.578_{\pm 0.011}$	$0.636_{\pm 0.011}$	$\underline{\mathbf{0.612}}_{\pm 0.011}$	$0.561_{\pm 0.011}$	$0.569_{\pm 0.011}$
	ARC-C	$0.306_{\pm 0.013}$	$0.338_{\pm 0.014}$	$\underline{\mathbf{0.328}}_{\pm 0.014}$	$0.309_{\pm 0.013}$	$0.306_{\pm 0.013}$
	WinoGrande	$0.529_{\pm 0.014}$	$0.557_{\pm 0.014}$	$\mathbf{0.541}_{\pm 0.014}$	$0.538_{\pm 0.014}$	$0.533_{\pm 0.014}$

Table 32: **Continuation perplexity on the from-scratch pair**, for the translators of Table 31, plus KV-Lingo's closed-form initialisation and KV-Lingo trained with cross-entropy (CE) instead of self-distillation (Appendix U). The last five columns are the gap $\ln(\text{translated PPL}) - \ln(\text{native PPL})$ in nats, so zero is native and lower is better; *native PPL* is the target's own continuation perplexity on that corpus. A 64-token continuation after a 256-token prefix, 128 documents, one map per cell. No standard errors are shown: per-document perplexities were not stored. **Bold**: the lowest gap on that row.

			KV-Lingo (ours)			head-wise MLP	
	Corpus	native PPL	closed form	CE	self-distilled	1.0×	4.0×
$L \to S$	Nemotron	12.51	0.236	0.048	**0.031**	0.085	0.069
	WikiText	23.20	0.444	0.106	**0.067**	0.229	0.155
$S \to L$	Nemotron	10.99	0.303	0.060	**0.043**	0.074	0.065
	WikiText	19.51	0.558	0.138	**0.101**	0.181	0.170

Table 33: **Cache translation on two Qwen3-Base pairs.** The released Qwen3-0.6B/1.7B-Base and Qwen3-4B/8B-Base pairs, both directions, with KV-Lingo, the identity layer assignment and the recipe of §4.1. KV-Lingo cells are the mean over three seeds; the closed-form fit is a single solve. ±: standard error over evaluation items (Appendix E.1), binomial from the aggregate accuracy since only aggregate scores were stored for these runs (for KV-Lingo, taken at the three-seed mean); SQuAD-v1 token-F1 is not binary, so its cells carry none. Base models have no chat template, so both pairs are fitted on plain text and every task is run 5-shot on at most 2000 examples. Underline: the translated cache beats the small model, in $S \to L$ by more than one standard error. **Bold**: the better translator on that row. ARC-C is length-normalised accuracy, SQuAD-v1 token-F1, BoolQ and WinoGrande accuracy. The native columns repeat within a pair. On the 4B/8B $L \to S$ SQuAD-v1 row, the underlined 0.906 is 0.9063 against the small model's 0.9057.

			native		translated	
		Benchmark	small	large	closed form	KV-Lingo
Qwen3 0.6B/1.7B	$L \to S$	ARC-C	$0.441_{\pm 0.015}$	$0.550_{\pm 0.015}$	$\mathbf{0.429}_{\pm 0.014}$	$0.423_{\pm 0.014}$
		SQuAD-v1	0.826	0.881	0.785	**0.821**
		BoolQ	$0.720_{\pm 0.010}$	$0.799_{\pm 0.009}$	$\underline{\mathbf{0.783}}_{\pm 0.009}$	$\underline{0.772}_{\pm 0.009}$
		WinoGrande	$0.597_{\pm 0.014}$	$0.658_{\pm 0.013}$	$0.597_{\pm 0.014}$	$\underline{\mathbf{0.611}}_{\pm 0.014}$
	$S \to L$	ARC-C	$0.441_{\pm 0.015}$	$0.550_{\pm 0.015}$	$\underline{0.472}_{\pm 0.015}$	$\underline{\mathbf{0.493}}_{\pm 0.015}$
		SQuAD-v1	0.826	0.881	0.746	**0.794**
		BoolQ	$0.720_{\pm 0.010}$	$0.799_{\pm 0.009}$	$0.696_{\pm 0.010}$	$\mathbf{0.705}_{\pm 0.010}$
		WinoGrande	$0.597_{\pm 0.014}$	$0.658_{\pm 0.013}$	$0.597_{\pm 0.014}$	$\mathbf{0.610}_{\pm 0.014}$
Qwen3 4B/8B	$L \to S$	ARC-C	$0.645_{\pm 0.014}$	$0.683_{\pm 0.014}$	$0.624_{\pm 0.014}$	$\mathbf{0.636}_{\pm 0.014}$
		SQuAD-v1	0.906	0.915	0.884	$\underline{\mathbf{0.906}}$
		BoolQ	$0.860_{\pm 0.008}$	$0.877_{\pm 0.007}$	$\underline{0.870}_{\pm 0.008}$	$\underline{\mathbf{0.871}}_{\pm 0.008}$
		WinoGrande	$0.713_{\pm 0.013}$	$0.777_{\pm 0.012}$	$\underline{0.735}_{\pm 0.012}$	$\underline{\mathbf{0.737}}_{\pm 0.012}$
	$S \to L$	ARC-C	$0.645_{\pm 0.014}$	$0.683_{\pm 0.014}$	$0.613_{\pm 0.014}$	$\mathbf{0.646}_{\pm 0.014}$
		SQuAD-v1	0.906	0.915	0.861	**0.896**
		BoolQ	$0.860_{\pm 0.008}$	$0.877_{\pm 0.007}$	$\mathbf{0.857}_{\pm 0.008}$	$0.854_{\pm 0.008}$
		WinoGrande	$0.713_{\pm 0.013}$	$0.777_{\pm 0.012}$	$\underline{0.725}_{\pm 0.013}$	$\underline{\mathbf{0.741}}_{\pm 0.012}$

Released Base pairs. Tables 33 and 34 report the released Base pairs on their own, at three seeds: downstream accuracy for both Qwen3-Base pairs in both directions, and the cost of the same translators on corpora they were not fitted on. Averaged over the four tasks, the $S \to L$ translated cache lands within 0.005 of the small model that prefilled it, not near the large target.

Table 34: **The Base-pair translators off their fitting distribution.** Relative continuation-perplexity gap (translated − native)/native, in percent, for the self-distilled translators of Table 33 on Nemotron-CC, which they are fitted on, and four corpora they never see. Mean over three seeds; a 256-token continuation after a 256-token prefix, 128 documents per corpus; zero is native and lower is better. No standard errors are shown: per-document perplexities were not stored. The gap is relative because absolute perplexity varies six-fold across corpora (2.4 on code to 14.7 on books). The worst of the twenty cells costs 6.5% and the median 2.1%, so the translators do not rely on their fitting corpus. The closed-form fit was not run on this suite.

		Nemotron	WikiText	Code	Books	arXiv
Qwen3 0.6B/1.7B	$L \to S$	+0.03	+3.23	+4.51	+2.18	+2.74
	$S \to L$	+2.58	+6.50	+5.92	+5.50	+5.07
Qwen3 4B/8B	$L \to S$	+0.28	+1.41	+0.62	+1.20	+1.08
	$S \to L$	+1.04	+0.36	+1.51	+2.57	+2.03

W SPLIT VS. JOINT MAPS

A joint map translates the stacked $[K; V]$ with a single $2d_{\mathrm{tgt}} \times 2d_{\mathrm{src}}$ matrix. KV-Lingo's split map is the joint map with its key-to-value and value-to-key blocks set to zero, leaving T_K^ℓ and T_V^ℓ, which halves the parameters. We compare the two on the Qwen3-0.6B/1.7B-Base and 4B/8B-Base pairs in both directions, with the recipe of §3.2 on Nemotron text and one seed per map (Table 35). The split map loses almost nothing: its validation KL is higher by only 0.0008–0.0027 nats per token, below two seed standard deviations of the split map itself (0.0010–0.0024 over three seeds); its continuation log-perplexity is higher by at most 0.008 nats on Nemotron and WikiText; and its BoolQ accuracy differs by at most 0.013, within the sampling error of the 2000-item evaluation (two-proportion z-test, $p \geq 0.28$). KV-Lingo therefore translates keys and values independently throughout.

Table 35: **Split against joint maps on the Qwen3 Base pairs**, after self-distillation, one seed per map. Validation KL is the forward KL from the target decoding from its own cache to the target decoding from the translated cache, in nats per token over 2048 Nemotron continuation tokens. $\Delta \ln \mathrm{PPL}$ is the split map's continuation log-perplexity minus the joint map's, in nats, over 128 documents per corpus with a 256-token prefix and a 256-token continuation. BoolQ is 0-shot accuracy on 2000 items.

		validation KL		$\Delta \ln \mathrm{PPL}$		BoolQ	
Pair	Direction	split	joint	Nemotron	WikiText	split	joint
0.6B/1.7B	$L \to S$	0.0202	0.0175	$+0.0037$	$+0.0076$	0.740	0.729
	$S \to L$	0.0265	0.0257	$+0.0012$	$+0.0039$	0.716	0.709
4B/8B	$L \to S$	0.0143	0.0120	$+0.0013$	$+0.0045$	0.818	0.807
	$S \to L$	0.0175	0.0164	$+0.0018$	$+0.0009$	0.831	0.844

X CACHE TRANSLATION ACROSS LANGUAGES

Figure 37 reports the per-language XQuAD scores behind the multilingual paragraph of §4.2. The translators are fitted on an English-only stream (the English subset of Nemotron-SFT-Instruction-Following-Chat-v2), which makes the ten other languages unseen at training time.

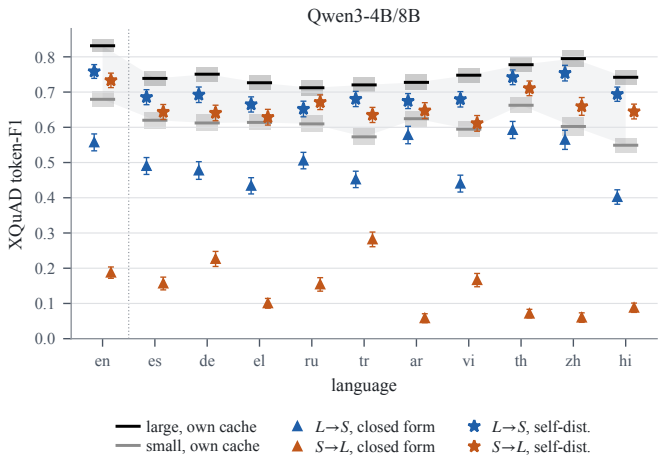


Figure 37: **XQuAD token-F1 by language, both translation directions, Qwen3-4B/8B.** Horizontal bars indicate the small model's and large model's scores, with the band between them shaded. Both directions are then overlaid: $L \to S$ in blue, $S \to L$ in orange. Stars are the score of KV-Lingo translators, averaged over three seeds; triangles are the closed-form baseline. Error bars, on the native scores and the closed-form fits too, are the standard error over evaluation items (Appendix E.1). These translators have been fitted on English-only data (from Nemotron-SFT-Instruction-Following-Chat-v2), so English, left of the dotted line, is their training language and the other ten are not. Reasoning-off; 256 questions per language. The distilled translators outperform the small model in most of the languages, showing striking OOD generalisation.

Y COMPUTATIONAL GAINS OF CACHE TRANSLATION

We measure time to first token, decoding throughput and peak memory in each scenario of §2.2.

Time to first token. Re-prefilling runs the target over the n tokens of the prefix, and its last position predicts the first new token. Translation replaces that forward pass with two operations. The translator applies, at each token and for each target layer, one matrix of size $d_{\mathrm{tgt}} \times \nu d_{\mathrm{src}}$ to the keys and one to the values: $4\nu\, d_{\mathrm{src}} d_{\mathrm{tgt}}$ FLOPs per token and target layer, and no attention. On the shape-preserving Qwen3 pairs ($d_{\mathrm{src}} = d_{\mathrm{tgt}} = 1024$, $\nu = 1$), that is 117 MFLOP per token into the 28-layer models and 151 MFLOP into the 36-layer ones, against 0.88, 2.8, 7.3 and 13.9 GFLOP for the weight matrices of Qwen3-0.6B, 1.7B, 4B and 8B: 7.5, 24, 48 and 92 times fewer, before counting the prefill's attention. A head-wise translator is $H = 8$ times cheaper still. The translated cache holds keys and values but no prediction, so the target then runs one forward pass over the current token, that is, one decoding step, whose cost grows with n only through reading the cache. The two paths reach the first token after $t_{\mathrm{prefill}}(n)$ and after $t_T(n) + t_{\mathrm{step}}(n)$.

Table 24 times these terms on an Apple M3 Ultra (MLX, `bf16`, batch size 1), for Qwen3-4B and Qwen3-30B-A3B (Figure 5c). A decoding step of the MoE, the inverse of its decoding speed, takes 14.0 ms at 64 tokens and 14.5 ms at 1024, against 152 and 421 ms for its prefill, and the translation takes 1.8 to 17.2 ms. A switch from the 4B, which already holds the cache, therefore reaches the MoE's first token 9.6× sooner than re-prefilling at 64 tokens, 10.9 to 11.4× sooner at 128 to 512, and 13.3× sooner at 1024. Used instead as a prefill accelerator (§4.6), the 4B first prefills the prompt itself, which puts its own prefill on the path: the first token then comes 1.8× sooner at 64 tokens, and no sooner at 1024, where the 4B prefills about as slowly as the MoE.

On an H100, we serve the targets with vLLM (Kwon et al., 2023) and divide the target's time to first token by the translation time plus one vLLM decoding step, in the four directions of the two shape-preserving Qwen3 pairs (Tables 37 and 36). The translation is timed outside vLLM, as matrix products in PyTorch. The switch then reaches the first token 4.8 to 5.6× sooner already at 2k tokens, and 12.0×, 14.8×, 23.2× and 28.9× sooner at 32k into the 0.6B, 1.7B, 4B and 8B (Figure 5a,b). As a same-stack check, we also time both paths end to end with Hugging Face Transformers (Table 36). That harness spends a fixed 29 to 40 ms on every decoding step, against 1.8 to 8.5 ms under vLLM, an overhead that hides the gain at short prefixes; past a few thousand tokens the prefill's arithmetic takes over, and the gap grows with n as under vLLM. Reading ν source layers per target layer multiplies the translation cost by ν (Figure 22).

Decoding throughput. When the cache is translated before starting to decode, after the switch the target decodes at its native speed, as the translated cache has the target's own shape: on the harness of Table 36, the 0.6B/1.7B pair decodes at the same 34 tokens/s with either cache, at every length.

Peak memory. Translation also skips the prefill's activations: at 32k tokens on the 0.6B/1.7B pair, the path with KV-Lingo peaks at 8.7 GB against 9.5 GB (into 0.6B) and 10.3 GB (into 1.7B) for re-prefill.

Table 36: **Components of the switch cost on an H100**, the two shape-preserving Qwen3 pairs, Hugging Face Transformers (`bf16`, batch 1, both paths on the same stack): translation, one decoding step of the target, and its re-prefill, in ms, every term measured at every length. The translation times enter the speedups of Figure 5a. This harness spends a fixed 29 to 40 ms on every decoding step, against 1.8 to 8.5 ms with vLLM (Table 37), so ratios taken within it understate the gain.

	into the smaller model			into the larger model		
n	translate	step	re-prefill	translate	step	re-prefill
Qwen3-0.6B/1.7B						
2048	3.5	29.2	29.9	3.5	29.0	30.0
4096	4.3	29.4	34.0	4.3	29.0	49.6
8192	7.9	29.4	75.7	7.9	29.6	107.0
16384	15.2	29.4	189.2	15.5	29.7	250.9
32768	30.8	29.6	547.3	30.7	29.6	671.6
Qwen3-4B/8B						
2048	4.5	38.6	51.6	4.5	39.2	74.1
4096	5.5	38.8	104.7	5.5	40.4	149.9
8192	10.1	38.5	230.7	10.1	40.4	325.3
16384	19.9	38.9	569.0	19.9	39.7	759.0
32768	39.9	39.3	1577.6	39.4	39.7	1981.8

Table 37: **Serving with vLLM on an H100**, `bf16`, one request at a time: median time to first token / median time per output token, in ms, over 16 requests per prompt length, each generating 128 tokens. Time to first token is measured at the client. The last row is Gemma-3-12B given the 4 KV heads of Gemma-3-4B, with random weights, which times its decoding step against a cache of the 4B's width under a memory budget.

Prompt tokens	2k	4k	8k	16k	32k
Qwen3-0.6B	30 / 1.8	38 / 1.9	58 / 2.1	123 / 2.4	405 / 2.9
Qwen3-1.7B	31 / 2.6	37 / 2.6	67 / 2.8	145 / 3.1	506 / 3.6
Qwen3-4B	44 / 4.6	54 / 4.7	111 / 4.9	275 / 5.3	1065 / 6.0
Qwen3-8B	61 / 7.0	72 / 7.1	150 / 7.3	543 / 7.7	1384 / 8.5
Gemma-3-4B	41 / 4.7	51 / 4.7	92 / 4.7	179 / 4.7	561 / 4.7
Gemma-3-12B	90 / 11.1	101 / 11.1	345 / 10.9	681 / 11.1	1451 / 11.4
Gemma-3-12B, 4 KV heads	84 / 10.4	97 / 10.4	190 / 10.4	656 / 10.4	1361 / 10.5

Z DECODING UNDER A MEMORY BUDGET WITHOUT MATERIALISING THE TARGET CACHE

In the memory-budget scenario of §2.2, the context is kept as the cache of the small model S and translators are merged to queries, so that L can directly decode from the small cache. This requires a linear and post-RoPE translation map. In that case, linearity carries the map across the inner product: $\langle q, T_K^\ell \bar{K}^\ell_{\mathrm{src},i} \rangle = \langle T_K^{\ell\top} q, \bar{K}^\ell_{\mathrm{src},i} \rangle$, $i \in [n]$, where q is a query of the target at the current step. The same holds for values: the attention weights of a target head pool the values of S first, and the map is applied to the pooled vector, folding into the target's output projection. The map is then applied once per layer and decoding step rather than once per cached token.

To get throughput acceleration, however, the translators have to be learned head-wise. A head-wise map pulls each target query back onto the source heads it reads from, so the scores are computed against keys of the width of one head of S; a head-mixing map pulls it back onto the whole cache width of S, and every target head then pays for all heads of S at every cached token.

Our experiments show that head-wise translators land significantly below the score of the large model, when small and large have different cache shapes — which is required in this setting. Hence, Table 38 times this decoding path as a proof-of-concept on Gemma-3-12B, against a cache narrowed to the width of Gemma-3-4B, using head-wise post-RoPE translators with random weights.

Table 38: **Decoding under a memory budget**, Gemma-3-12B in 4-bit on Apple M3 Ultra (MLX), random weights and cache. The full cache has 8 KV heads, the narrowed one 4, the width of Gemma-3-4B, at unchanged depth; the translator is a random per-head map applied to the query and to the attention output. Each cell generates 128 tokens, best of 3; ranges span two machines. Throughput changes are relative to the full cache. Peak memory is that of the whole process during decoding (weights, cache and temporaries).

		decoding throughput			peak memory (GB)
context	batch	full (tokens/s)	narrowed	narrowed + translator	full → narrowed + translator
8k	1	62–65	+9–10%	−5%	9.9 → 9.1
8k	8	160–171	+33–37%	+22–25%	21.6 → 14.9
32k	1	53	+16–20%	+1–5%	13.9 → 11.0
32k	8	110–115	+44–49%	+36–40%	50.2 → 29.4