Don't Read Everything: A Curvature-Conditioned Query for Linear Attention

Anonymous ACL submission

Abstract

Linear attention reduces the quadratic cost of softmax attention by maintaining a recurrent fast-weight state, but it consistently lags on in-context retrieval and long-context tasks. Existing remedies act on the write side of memory through gating, delta updates, or kernel feature maps, but the read step is left unchanged: every past key contributes additively to the output, so useful targets are diluted by the bulk of stored vectors. We borrow one specific piece of softmax's geometry to construct a cheap read-time contraction of the query. A second-order Taylor expansion of the softmax log-partition at the isotropic-attention point gives a local quadratic model whose curvature coincides with the running key covariance, a quantity that can be maintained with the same recurrent/chunkwise mechanism as the linear-attention state. The associated linear operator contracts the query along the high-density directions of memory before it reads the state. We call this mechanism Curvature-Conditioned Query (CCQ). CCQ modifies only the read step and is composable with any linear-attention backbone. Attached to GLA and Gated DeltaNet, it improves perplexity, zero-shot downstream accuracy, S-NIAH retrieval at and beyond the training context, length-extrapolation perplexity from 4K to 20K, and LongBench accuracy, at small extra cost.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) has become the default choice for sequence modelling, but its softmax attention scales quadratically with sequence length, which makes long-context training and inference expensive. To mitigate this, linear attention (Katharopoulos et al., 2020) replaces the softmax inner product with a kernelized dot product, reframing the read as a linear RNN with a matrix-valued state and reducing the cost to linear time and constant cache.

Early linear-attention variants underperformed standard Transformers on language modelling, but recent enhancements have closed much of this gap. Data-dependent gating, used by GLA (Yang et al., 2024a), RetNet (Sun et al., 2023), RWKV (Peng et al., 2023), Mamba (Gu and Dao, 2024), and Mamba2 (Dao and Gu, 2024), lets the state forget old entries adaptively. The delta rule, revived by Schlag et al. (2021) and parallelized in Yang et al. (2024b), replaces the additive write with a soft overwrite of the closest stored key, and Gated DeltaNet combines the two (Yang et al., 2025). Despite this progress, linear-attention models still trail softmax on in-context retrieval and long-context tasks (Arora et al., 2024a,b; Hsieh et al., 2024), where the ability to single out one stored item matters most.

We argue that this gap is not only about *what* is written into memory but also about *how* it is read. Softmax attention couples two operations in one expression: a key–query inner product, and a log-sum-exp normalizer that lets the matches compete so the most distinctive key dominates. Linear attention preserves the inner product but discards the normalizer, so every past key keeps a constant additive contribution in the output and useful targets are diluted by the bulk of stored vectors. The remedies above act on the write side, deciding more carefully what enters the recurrent state. They cannot, by construction, react to which directions of memory a future query will read against.

We propose a cheap read-time correction inspired by the local curvature of the missing normalizer. Once the keys are fixed, the log-sum-exp competition term depends only on the query, and a second-order Taylor expansion at the isotropic-attention point shows that its curvature is the running sample covariance of the keys. Applying the associated linear contraction to the query yields a cleaned query that has been pulled away from the high-density directions of memory before it

multiplies the linear-attention state. We call this mechanism Curvature-Conditioned Query (CCQ). The contributions of this paper are as follows:

- We identify the read step as a distinct, under-explored lever for closing the softmax–linear gap, complementary to the write-side remedies that dominate prior work.
- We propose CCQ, a curvature-aware linear contraction built from a single Taylor expansion of softmax's log-partition, that composes additively with any linear-attention write rule.
- Attached without modification to GLA and Gated DeltaNet at 500M and 1.3B parameters, CCQ improves perplexity, zero-shot downstream accuracy, S-NIAH retrieval at and beyond the training context, length-extrapolation perplexity from 4K to 20K, and LongBench accuracy.

2 Preliminary

2.1 Softmax Attention

Given a sequence of d-dimensional input vectors $x_1, \ldots, x_T$, a single-head Transformer (Vaswani et al., 2017) forms queries, keys, and values by linear projections

$$q_t = W_Q x_t, \quad k_t = W_K x_t, \quad v_t = W_V x_t, \quad (1)$$

with $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$, and reads from the past via causal softmax attention:

$$a_{ti} = \frac{\exp(k_i^\top q_t)}{\sum_{j\le t}\exp(k_j^\top q_t)}, \qquad o_t = \sum_{i\le t} a_{ti}\, v_i. \quad (2)$$

Equivalently,

$$\log a_{ti} = k_i^\top q_t - \log \sum_{j\le t}\exp(k_j^\top q_t). \quad (3)$$

This form separates the raw key–query score from the log-partition term, exposing softmax as an inner-product score followed by a data-dependent normalizer. We will use this view in Section 3 to motivate CCQ as a lightweight read-time correction inspired by the local curvature of the second term.

2.2 Linear Attention

Linear attention (Katharopoulos et al., 2020) replaces the exponential kernel $\exp(k_i^\top q_t)$ in (2) with the dot product of a feature map $\phi(k_i)^\top\phi(q_t)$, where $\phi : \mathbb{R}^d \to \mathbb{R}^n$. Dropping the denominator and taking ϕ to be the identity (the simplest and most common choice in modern variants) factors the read into the recurrent form

$$S_t = S_{t-1} + v_t k_t^\top \in \mathbb{R}^{d_v\times d_k}, \qquad o_t = S_t\, q_t. \quad (4)$$

The matrix S_t acts as a fast-weight memory (Schmidhuber, 1992; Schlag et al., 2021): every past key–value pair is written into it once via an outer product and read once per query through a single matrix-vector product. This swaps the $\mathcal{O}(T^2)$ cost and $\mathcal{O}(T)$ KV cache of softmax for $\mathcal{O}(T)$ training cost and an $\mathcal{O}(1)$ cache of fixed size $d_v d_k$, at the price of a finite-rank approximation of the full attention matrix.

The recurrence (4) is the additive baseline; the *backbone* is the choice of write update that replaces it. Gated variants such as GLA (Yang et al., 2024a), RetNet (Sun et al., 2023), and Mamba2 (Dao and Gu, 2024) insert a data-dependent decay before the outer product; delta-rule variants such as DeltaNet (Schlag et al., 2021; Yang et al., 2024b) and Gated DeltaNet (Yang et al., 2025) replace the additive write with a soft overwrite of the closest stored key. Throughout this paper we treat the backbone as a black box: CCQ modifies only how the state is read and is therefore compatible with any update of the form $S_t = \mathrm{Update}(S_{t-1};\, k_t, v_t, \ldots)$.

3 Method

3.1 Motivation: the missing competition of linear attention

Softmax attention (Vaswani et al., 2017) couples two operations in a single closed-form expression: a key–query inner product $k_j^\top q_t$ that scores each past key, and a log-partition

$$A_t(q_t) = \log \sum_{m\le t}\exp(k_m^\top q_t) \quad (5)$$

that normalizes those scores into a probability simplex. The second piece shapes softmax's selection behaviour: when a query overlaps many stored keys, A_t grows and individual matches are relatively discounted, so keys lying away from the bulk of memory tend to dominate the readout.

Linear attention (Katharopoulos et al., 2020) drops the normalizer to gain linear time and constant cache. Its recurrent fast-weight form (Schmid-

huber, 1992; Schlag et al., 2021)

$$S_t \;=\; \sum_{j\le t} v_j k_j^\top, \quad o_t \;=\; S_t q_t \tag{6}$$

preserves the inner-product score but discards the competition: every past key contributes additively forever, regardless of how many neighbours it now shares a direction with. Crowded directions therefore accumulate as a constant background that the query cannot escape, and useful needles drown in the bulk of stored vectors (Arora et al., 2024a,b; Hsieh et al., 2024). Rebuilding A_t exactly would cost the full attention matrix and erase the linear efficiency. The question is whether the geometric flavour of its suppression effect, namely contracting the query along directions where memory is dense, can be captured cheaply enough to fit inside a linear recurrence.

3.2 From competition to a query-side correction

The key observation is that the competition term A_t is a function of q_t alone once the keys are fixed. Whatever softmax does to attenuate crowded directions, it does through this $A_t(q_t)$. Two ideas follow.

Idea 1: replace A_t with its local quadratic shape. Expanding A_t to second order around $q=0$ gives (see Appendix A.2 for the derivation)

$$\begin{aligned} A_t(q) \;=\;& \log t + \mu_t^\top q \\ &+ \tfrac{1}{2}\, q^\top\!\big(\bar{C}_t - \mu_t\mu_t^\top\big) q + \mathcal{O}(\|q\|^3), \end{aligned} \tag{7}$$

with $\mu_t = \frac{1}{t}\sum_{j\le t} k_j$, the running second moment $\bar{C}_t \;=\; \frac{1}{t}\sum_{j\le t} k_j k_j^\top$, and the running sample covariance

$$\Sigma_t \;\triangleq\; \bar{C}_t - \mu_t\mu_t^\top \;=\; \mathrm{Cov}_{\le t}[k]. \tag{8}$$

The constant $\log t$ and the linear term $\mu_t^\top q$ shift every score uniformly and play no role in differentiating one key from another. The quadratic Hessian Σ_t has eigendirections aligned with *directions of high key spread*: top eigenvectors point where stored keys vary the most around their mean.

Idea 2: contract the query along the local curvature, then read. A natural way to use this knowledge is to contract the query along the directions in which A_t rises before letting it interact with the linear memory. The Hessian Σ_t furnishes a curvature-aware linear operator on the query space, and applying $I-\lambda_t\Sigma_t$ to q_t gives the *cleaned query*

$$\begin{aligned} q_t^{\text{clean}} \;&=\; (I - \lambda_t\,\Sigma_t)\, q_t \;=\; q_t \;-\; \lambda_t\,\Sigma_t\, q_t, \\ \lambda_t \;&=\; \sigma\big(W_\lambda q_t + b_\lambda\big). \end{aligned} \tag{9}$$

The scalar $\lambda_t \in (0,1)$ is a query-conditioned gate that controls how aggressively to contract. The matrix Σ_t is maintainable by the same recurrent/chunkwise mechanism as the backbone state through two running statistics: $\bar{C}_t$ has the form of (6) with the value stream replaced by the key stream and no decay, and μ_t is a running sum of keys. Together they cost one additional $d_k\times d_k$ matrix and one d_k-vector per head (see Appendix B).

The linear-attention read is then performed unchanged, but with the cleaned query:

$$o_t \;=\; S_t\, q_t^{\text{clean}} \;=\; S_t q_t \;-\; \lambda_t\, S_t\Sigma_t\, q_t. \tag{10}$$

The first term is the original linear-attention output. The second is a structured, context-conditioned correction that subtracts a read along the high-variance directions of memory, weighted by λ_t. Because S_t is never modified, CCQ slots onto a backbone's read without changing its write rule; we instantiate this with GLA and Gated DeltaNet in Sec. 4.1.

What the correction actually penalizes. Substituting (9) into the bilinear score reveals the selection pattern in score space:

$$s_{tj}^{\text{CCQ}} \;=\; k_j^\top q_t \;-\; \lambda_t\, k_j^\top \Sigma_t\, q_t. \tag{11}$$

The first term is the original inner-product score; the second penalizes a candidate key by the amount that k_j and q_t co-vary in the empirical distribution of past keys. Equivalently, $k_j^\top\Sigma_t q_t \;=\; \mathrm{Cov}_{\le t}[k_j^\top k,\, k^\top q_t]$, so a key is penalized only when its alignment with a typical stored key correlates with that stored key's alignment with the query. Keys whose alignment with memory is uncorrelated with the query's read direction are left untouched. CCQ therefore biases the read toward directions that are less represented in the running key covariance, favouring isolated targets. The strength of this bias is set per-token by the bounded gate λ_t, so the model can dial the correction up where it helps and down where the standard inner-product score is already sufficient.

When does CCQ help retrieval? On a toy retrieval model with $\Sigma_t \approx \rho uu^\top$ dominated by a single high-variance direction u, the CCQ-corrected

margin between a target key $k_\star$ (alignment $\alpha = \langle k_\star, u\rangle$) and a distractor k_d (alignment β) gains the term $\lambda\rho(\beta - \alpha)(u^\top q)$ relative to the un-cleaned margin (Eq. (26), derived in Appendix G). CCQ therefore widens the target margin whenever the target lies off the high-variance direction *and* the query overlaps with that direction, and reduces to identity in the boundary cases (target collinear with u, or query orthogonal to u). The diagnostic in Appendix H is consistent with the favourable regime in the cases we probe: needle keys sit off the high-variance subspace, putting the cleaning operator on the configuration it was designed for.

3.3 Properties of the construction

We now make precise the local link between CCQ and softmax that the rest of the paper relies on. Restricted to the isotropic-attention point $q = 0$, the Hessian of A_t is the sample covariance of the keys,

$$\nabla_q^2 A_t(0) \;=\; \Sigma_t \;=\; \mathrm{Cov}_{\le t}[k], \qquad (12)$$

which is the classical exponential-family identity that the Hessian of the log-partition equals the covariance of the sufficient statistic (Wainwright and Jordan, 2008); the proof is reproduced in Appendix A.2. The cleaning operator $I - \lambda_t\Sigma_t$ in (9) is therefore a curvature-aware contraction of the query along the local softmax geometry: directions in which Σ_t has large eigenvalues are shrunk by a factor close to $1 - \lambda_t$, while directions orthogonal to memory are passed through unchanged. Where softmax uses the full nonlinear A_t at every query, CCQ uses only its second-order Taylor expansion at a single fixed point ($q = 0$) through running statistics that fit into the same recurrent/chunkwise mechanism as the backbone state (at the cost of one $d_k \times d_k$ matrix and one d_k-vector per head).

This Hessian–covariance correspondence has three immediate consequences. First, the construction is *adaptive*: both the suppression direction $\Sigma_t q_t$ and its strength λ_t are query-dependent, so CCQ produces a different cleaning geometry at every token and every head while using only a pair of shared running statistics per head. Second, it is *local to the keys*: the correction is built from second-order statistics of the keys, not from the query distribution, so Σ_t literally records how memory is filling up irrespective of where the next query lands. Third, it is *decoupled from the write rule*: the cleaning acts on q_t only, so the state recurrence $S_t = \mathrm{Update}(S_{t-1}; k_t, v_t, \ldots)$ is untouched. This makes the construction architecturally compatible with any linear-attention write rule; we evaluate it on GLA and Gated DeltaNet.

Empirical check on the underlying assumption. The construction rests on the assumption that retrieval-relevant keys live *off* the high-variance directions of memory, while distractor keys cluster inside them. We test this directly on pretrained models by planting a unique needle sentence in a packed context of ∼120 filler sentences and measuring, for the keys captured at *every* layer, the fraction of each key's centred energy that lives in the top-16 eigenvector subspace of the running key covariance Σ. To avoid single-prompt artefacts we average across several distinct needle scenarios (different novel tokens, sentences, and queries); see Appendix H for the full setup. On both Qwen3-4B-Instruct[1] (softmax) and a Gated DeltaNet 500M (linear attention), needle keys project less onto the top-16 subspace than distractor keys at **100% of layers in both models**. The effect is consistent across prompts (per-layer standard deviation ∼0.02) and stronger in the linear-attention model ($\bar{\Delta}_\mu = -0.21$ vs -0.11, Fig. 1). Pooling tokens across every layer and every prompt (Fig. 2) gives a direct view of the underlying distributions: the unique-needle histogram sits clearly to the left of the distractor histogram in both models, so the per-layer Δ_μ in Fig. 1 is not a thin mean-of-means artefact but reflects a genuine population-level shift of the needle keys away from the high-variance subspace.

Stability. With unit-norm keys and the sigmoid-gated $\lambda_t \in (0, 1)$, the cleaning operator $I - \lambda_t\Sigma_t$ has spectrum in $(0, 2)$ at every position, independent of sequence length, and acts as a soft anisotropic contraction in the eigenbasis of Σ_t that attenuates magnitude along high-variance directions without annihilating any direction of q_t. The operator-norm bound and spectrum argument are in Appendix F.

4 Experiments

4.1 Experimental Setup

Training. We pretrain all models from scratch as autoregressive language models on the same FineWeb-Edu (Penedo et al., 2024) subset, with the same tokenizer (32K BPE), context length (4K),

[1] https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507

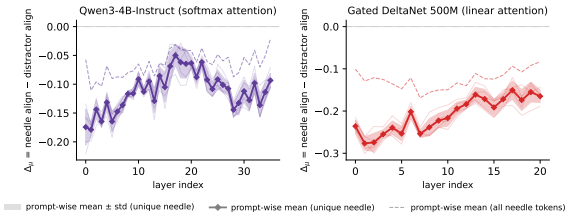

Figure 1: Layer-by-layer geometric separation, aggregated over a collection of needle-in-a-haystack prompts (Appendix H). Each curve plots Δ_μ = needle align − distractor align on the top-16 eigenvector subspace of the centred key covariance Σ, head-averaged. The solid line is the prompt-wise mean and the shaded band is $\pm$ one prompt-wise standard deviation; the dashed line uses all 17 needle-span tokens (including common words like "the", "is") as a control. The unique-token Δ is negative at *every* layer of both models, with no prompt-wise sign flip ($\sigma \approx 0.02$). The effect is strongest in early layers and weakens with depth, but never crosses zero. The linear-attention model shows roughly $2\times$ the separation of the softmax model at every depth; we report this as a single-pair observation and do not claim a causal interpretation, since the two models differ in scale, training data, and head dimension as well as in attention type.

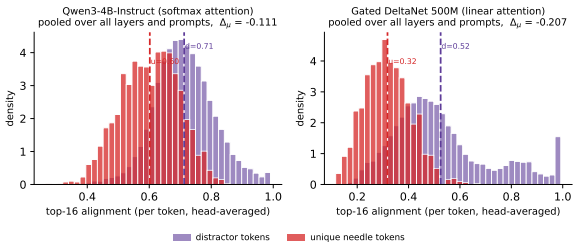

Figure 2: Per-token distribution of top-16 alignment values for one representative layer per model (Qwen3 layer 4, Gated DeltaNet layer 8). Distractor tokens (purple) form a continuous distribution; the truly unique needle tokens (red vertical bars) cluster well to the left of the distractor mean. The means are separated by -0.156 (Qwen3) and -0.258 (Gated DeltaNet) at these layers.

Scale	Model	Layers	Hidden	Tokens
500M	Transformer	35	1024	15B
500M	Mamba2	65	1024	15B
500M	GLA	21	1024	15B
500M	GLA-Hedgehog	21	1024	15B
500M	Gated DeltaNet	21	1024	15B
500M	CCQ-GLA	21	1024	15B
500M	CCQ-Gated DeltaNet	21	1024	15B
1.3B	Transformer	24	2048	40B
1.3B	Mamba2	48	2048	40B
1.3B	GLA	21	2048	40B
1.3B	GLA-Hedgehog	21	2048	40B
1.3B	Gated DeltaNet	21	2048	40B
1.3B	CCQ-GLA	21	2048	40B
1.3B	CCQ-Gated DeltaNet	21	2048	40B

Table 1: Per-model layer counts, hidden sizes, and token budgets at the 500M and 1.3B scales.

optimizer, and token budget within each scale. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1, peak learning rate 10^{-3}, 1024-step linear warmup, and cosine decay to zero. Mixed precision is bfloat16. The global batch size is 524,288 tokens per step at the 1.3B scale and 262,144 tokens per step at the 500M scale. We report two scales: at *500M* every model is trained for 15B tokens and at *1.3B* for 40B tokens. The training objective is standard next-token cross entropy with the fused norm and fused cross-entropy kernels. Per-model layer counts, hidden sizes, and token budgets are listed in Table 1; the full optimizer settings are reproduced in Table 6 of Appendix D.

Baselines. We compare CCQ to five backbones spanning softmax attention, state-space, and linear-attention families: a Llama-style **Transformer**, **Mamba2** (Dao and Gu, 2024), gated linear attention (**GLA** (Yang et al., 2024a)), **GLA-Hedgehog** (GLA with the Hedgehog learned-feature map (Zhang et al., 2024)), and **Gated DeltaNet** (Yang et al., 2025). CCQ is attached on top of GLA and Gated DeltaNet, giving *CCQ-GLA* and *CCQ-Gated DeltaNet*; the only difference between a backbone and its CCQ variant is the read step. All variants share SwiGLU MLPs, RMSNorm, tied input/output embeddings.

Evaluation. We probe four axes of model quality: *(i)* common-sense reasoning and language modelling (`lm-evaluation-harness` (Gao et al., 2021), 8 zero-shot tasks plus LAMBADA and WikiText); *(ii)* synthetic in-context retrieval (NIAH and RULER-style tasks of Hsieh et al. (2024) at 4K and 8K); *(iii)* length-extrapolation perplexity from 4K to 20K on six long-context corpora; and *(iv)* long-context understanding (14-task English LongBench (Bai et al., 2024), zero-shot). The per-family task selection, context lengths, metrics, and batch sizes are listed in Appendix E.

4.2 Language Modelling and Downstream Accuracy

Table 2 reports the language-modelling and zero-shot downstream-accuracy numbers. At 500M, CCQ-Gated DeltaNet attains the best WikiText perplexity (25.59) and the best 7-task Avg (49.87%) among the recurrent models, improving over its

backbone by -0.74 Wiki PPL and $+0.89$ Avg; CCQ-GLA improves over GLA on LAMBADA accuracy ($+2.72$) and Wiki PPL (-1.10). At 1.3B, CCQ-GLA attains the best WikiText perplexity (16.92), LAMBADA perplexity (11.39), LAMBADA accuracy (47.82), and HellaSwag (59.14), improving on its GLA backbone by -1.88 Wiki PPL, -0.92 LMB PPL, and $+3.70$ HellaSwag. CCQ-Gated DeltaNet attains the best 7-task Avg (56.24) and best SIQA (41.71), improving on its Gated DeltaNet backbone by $+0.60$ Avg, with CCQ-GLA a close second on Avg (56.10).

4.3 Synthetic Needle-in-a-Haystack

Table 3 resolves the single-needle tasks by context length: *S-NIAH-1* (pass-key), *S-NIAH-2* (number), and *S-NIAH-3* (uuid). 1K/2K is in-distribution, 8K is pure length extrapolation; S-NIAH-3 collapses past 4K for every model. Difficulty grows from S-NIAH-1 to S-NIAH-3 as the needle overlaps more with distractor tokens, and within each task as context grows.

CCQ rescues both backbones at the lengths where they fail. At 500M, GLA collapses on S-NIAH-1 past 1K but CCQ-GLA recovers to $97.6/56.4/20.6$ at 2K/4K/8K, and CCQ-Gated DeltaNet takes the best 8K columns on S-NIAH-2 ($+26$ over backbone) and S-NIAH-3 4K. At 1.3B the gains sharpen: CCQ-GLA adds $+49.0$ on S-NIAH-1 4K, $+29.4$ on S-NIAH-2 8K, and $+35.6$ on S-NIAH-3 4K; CCQ-Gated DeltaNet adds $+40.6$ on S-NIAH-3 4K, and CCQ-GLA wins S-NIAH-2 4K/8K and S-NIAH-3 1K across *all seven models*. The kernel-feature-map baseline GLA-Hedgehog *worsens* its backbone at every length, confirming read-side cleaning is memory-aware rather than a generic inner-product change. The Transformer leads in-distribution S-NIAH-3 at 1.3B but collapses to 0 at 8K, where the recurrent CCQ variants retain a substantial fraction of in-distribution accuracy.

4.4 Length Extrapolation

We compute token-level perplexity on a 4K-to-20K sweep in 2K steps (nine points) over six long-context corpora (GovReport, QMSum, NarrativeQA, Qasper, PG19, CodeParrot); since all models are trained at 4K, the 6K–20K points are pure extrapolation (Appendix E).

At 500M (Fig. 3), CCQ-Gated DeltaNet attains the lowest PPL on every dataset at every length, with the margin over its backbone widening as

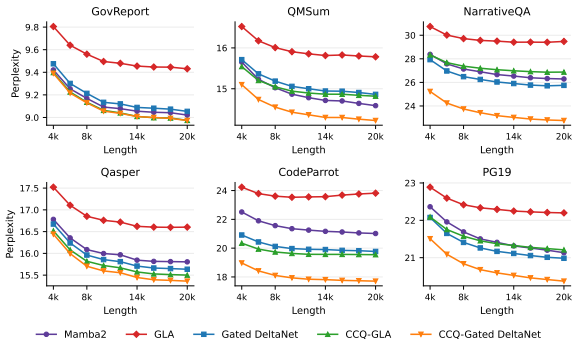

Figure 3: Length-extrapolation PPL at 500M over six corpora and nine lengths (4K–20K in 2K steps). Lower is better. Five recurrent models shown (Mamba2, GLA, Gated DeltaNet, CCQ-GLA, CCQ-Gated DeltaNet); the Transformer and GLA-Hedgehog are off-range and omitted. Y-axes are per-panel auto-scaled to the shown curves.

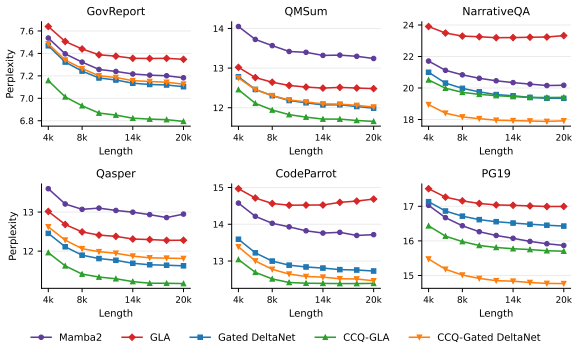

Figure 4: Length-extrapolation PPL at 1.3B on the same six corpora and nine lengths as Fig. 3. Five recurrent models shown; Transformer and GLA-Hedgehog are off-range and omitted.

context grows; CCQ-GLA sits well below GLA, closing most of the gap to the stronger recurrent baselines. The Transformer (4K-only training) and GLA-Hedgehog (steeply rising curves) are off-range and not plotted.

At 1.3B (Fig. 4) the two CCQ variants split the panels at 20K: CCQ-GLA wins GovReport, QMSum, Qasper, and CodeParrot; CCQ-Gated DeltaNet wins NarrativeQA and PG19. Each CCQ variant beats its own backbone on every panel by 1–2.5 PPL at 20K. The Transformer 1.3B is off-range (CodeParrot 20K > 1700).

4.5 Long-Context Understanding

Table 4 reports zero-shot accuracy on the 14 English LongBench tasks. We only report at the 1.3B scale (Appendix E). CCQ-GLA attains the best 14-task Avg (15.77) among the recurrent

Scale	Model	Wiki PPL↓	LMB PPL↓	LMB acc↑	PIQA acc	Hella. acc_n	Wino. acc	ARC-e acc_n	ARC-c acc_n	SIQA acc	BoolQ acc	Avg. acc
500M parameters / 15B tokens												
Recurrent models												
500M	Mamba2	26.22	29.37	34.62	67.30	44.15	52.57	55.13	29.27	39.92	55.84	49.17
500M	GLA	27.27	32.11	34.02	66.38	41.93	52.57	52.23	28.58	38.43	59.66	48.54
500M	GLA-Hedgehog	32.47	43.74	29.50	66.21	40.48	51.85	53.37	28.33	39.41	**61.71**	48.77
500M	Gated DeltaNet	26.33	29.10	33.88	67.85	43.48	**53.67**	55.56	28.67	39.41	54.19	48.98
500M	**CCQ-GLA**	26.17	25.77	36.74	68.01	43.45	53.35	**56.82**	29.78	39.87	56.85	49.73
500M	**CCQ-Gated DeltaNet**	**25.59**	**22.93**	**38.19**	**68.28**	**44.61**	53.35	56.14	**30.55**	**40.23**	55.96	**49.87**
Attention models												
500M	Transformer	331.69	30.47	34.04	67.03	43.12	54.93	54.80	30.12	38.08	58.01	49.44
1.3B parameters / 40B tokens												
Recurrent models												
1.3B	Mamba2	18.17	13.55	45.55	72.58	57.00	56.91	64.60	36.52	41.61	56.97	55.17
1.3B	GLA	18.80	12.31	47.02	72.52	55.44	**59.67**	66.54	**38.91**	39.15	58.01	55.75
1.3B	GLA-Hedgehog	19.07	15.17	43.08	**72.91**	55.90	56.43	**67.76**	38.65	40.69	59.17	55.93
1.3B	Gated DeltaNet	17.98	13.11	45.57	71.27	56.66	57.22	65.91	36.35	41.45	**60.58**	55.64
1.3B	**CCQ-GLA**	**16.92**	**11.39**	**47.82**	72.03	**59.14**	59.43	64.23	36.95	41.20	59.69	56.10
1.3B	**CCQ-Gated DeltaNet**	18.01	12.52	46.13	71.33	56.82	58.17	66.46	38.82	**41.71**	60.37	**56.24**
Attention models												
1.3B	Transformer	41.55	14.31	45.02	71.22	53.44	56.27	61.49	36.01	42.58	61.16	54.60

Table 2: Language modelling perplexity and zero-shot common-sense reasoning accuracy. Hella.: HellaSwag; Wino.: WinoGrande. *acc_n* denotes normalized accuracy where applicable. The Avg column is the unweighted mean over PIQA, Hella., Wino., ARC-e, ARC-c, SIQA, BoolQ (seven tasks); rows missing any task are reported as –. **Bold** marks the best recurrent number per scale. Cells marked — were not evaluated.

models, improving over its GLA backbone by $+1.70$ Avg, with CCQ-Gated DeltaNet a close second (14.78, $+0.98$ over its own backbone) and ahead of Mamba2 by $+2.64$. CCQ-GLA leads the QA columns (Qasper, MFQ, 2WM) and the GovReport/TREC/RepoBench-P axes; CCQ-Gated DeltaNet leads MultiNews, TriviaQA, and SAMSum. Both CCQ variants improve on their respective backbones across the longer-output tasks (GovReport, MultiNews, SAMSum) and on the few-shot TREC/TriviaQA columns by 4–14 points. The Transformer baseline collapses on every long-input column (Avg 4.80) because its 4,096-token training window is much shorter than the LongBench inputs.

5 Related Works

Linear attention (Katharopoulos et al., 2020) replaces the $\mathcal{O}(T^2)$ pairwise softmax of the Transformer (Vaswani et al., 2017) with a recurrent fast-weight memory (Schmidhuber, 1992; Schlag et al., 2021), yielding $\mathcal{O}(T)$ training and $\mathcal{O}(1)$ cache inference. The Hopfield perspective (Ramsauer et al., 2021) makes the limitation visible: vanilla linear attention writes with a Hebbian rule whose capacity is bounded, which shows up as a recall gap against softmax (Arora et al., 2024a,b; Hsieh et al., 2024). Existing remedies group by which part of the read or write pipeline they modify.

Data-dependent gating. GLA (Yang et al., 2024a) adds a learned per-key forget gate; RetNet (Sun et al., 2023) uses fixed exponential decay; RWKV (Peng et al., 2023) interleaves a time mixing decay with a channel-mixing MLP; Mamba (Gu and Dao, 2024) and Mamba2 (Dao and Gu, 2024) cast the decay as a selective state-space model conditioned on the input; xLSTM (Beck et al., 2024) revisits LSTM gating with matrix-valued cells. These methods control *what stays* in the state, but the read remains a single matrix–vector product.

Delta rule and its gated extension. A second line replaces the additive write with a soft overwrite of the matched key. The delta rule was revived by Schlag et al. (2021) and parallelised across sequence length by Yang et al. (2024b). Gated DeltaNet (Yang et al., 2025) combines a forget gate with the delta update and closes most of the recall–retention gap. The delta family has stronger capacity than Hebbian writes but still reads through the unmodified inner product.

Online-learning view of the write. A recent line interprets the recurrent state as the solution to a small online learning problem and refines the write accordingly: MesaNet (von Oswald et al., 2026) solves an online least-squares problem at each step, TTT (Sun et al., 2025) parameterises the state by a small nonlinear predictor updated at test time, and Titans (Behrouz et al., 2026) adds a long-term memory module trained at inference. These deliver sharper writes at the cost of nonlinear updates or per-token inner-loop work, and again leave the read step unchanged.

		S-NIAH-1 (pass-key)				S-NIAH-2 (number)				S-NIAH-3 (uuid)		
Scale	Model	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K
500M	Mamba2	**100.0**	**100.0**	97.6	68.4	**100.0**	**100.0**	**97.6**	45.2	**68.6**	**54.8**	28.8
500M	GLA	88.0	26.2	11.0	5.2	99.8	94.8	41.6	13.8	39.8	16.0	4.2
500M	GLA-Hedgehog	76.6	55.8	36.6	0.0	93.8	85.6	16.4	0.0	11.8	3.4	0.4
500M	Gated DeltaNet	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	84.6	24.8	56.2	54.2	30.2
500M	**CCQ-GLA**	**100.0**	97.6	56.4	20.6	99.8	**100.0**	93.6	27.8	11.0	41.4	3.6
500M	**CCQ-Gated DeltaNet**	**100.0**	**100.0**	**100.0**	99.6	**100.0**	99.0	87.0	**50.8**	67.6	53.4	**33.6**
500M	Transformer	**100.0**	**100.0**	97.8	0.0	40.2	98.2	91.2	0.0	12.2	7.2	12.6
1.3B	Mamba2	99.8	**100.0**	**100.0**	99.2	**100.0**	**100.0**	79.6	55.8	69.4	47.6	27.4
1.3B	GLA	**100.0**	86.4	47.8	13.0	**100.0**	99.4	93.6	31.2	66.0	51.6	29.2
1.3B	GLA-Hedgehog	92.4	96.0	75.0	6.2	99.8	98.4	62.4	3.6	49.2	23.8	1.8
1.3B	Gated DeltaNet	**100.0**	99.8	**100.0**	**99.8**	**100.0**	**100.0**	83.4	56.6	64.4	56.0	6.0
1.3B	**CCQ-GLA**	**100.0**	**100.0**	96.8	42.6	**100.0**	**100.0**	**99.4**	**60.6**	**94.4**	90.4	64.8
1.3B	**CCQ-Gated DeltaNet**	**100.0**	99.8	**100.0**	98.2	**100.0**	**100.0**	86.0	58.4	66.4	59.2	46.6
1.3B	Transformer	**100.0**	**100.0**	99.8	0.0	99.8	**100.0**	99.0	0.0	93.0	**94.4**	**82.2**

Table 3: Synthetic needle-in-a-haystack accuracy as a function of context length, on the 500M and 1.3B checkpoints. S-NIAH-1: pass-key retrieval. S-NIAH-2: number in haystack. S-NIAH-3: uuid in haystack (reported up to 4K; uuid retrieval collapses past that for every model we tested). All values are percentages. **Bold** marks the best per column at each scale. The 8K columns probe pure length extrapolation past the 4K training window; the Transformer's collapse at 8K is expected since its training context is only 4K.

		Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Code		
Scale	Model	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP	Avg.
1.3B parameters / 40B tokens																
Recurrent models																
1.3B	Mamba2	**2.29**	5.75	13.30	**6.03**	7.51	2.90	7.39	**18.13**	11.10	36.00	22.25	8.52	19.25	23.33	13.13
1.3B	GLA	2.20	4.78	13.51	4.94	9.32	**3.32**	7.46	16.63	11.20	24.75	22.85	23.88	25.26	26.81	14.07
1.3B	GLA-Hedgehog	1.08	3.47	9.95	3.91	4.70	2.22	7.03	11.06	9.03	8.00	19.31	7.88	18.52	18.96	8.94
1.3B	Gated DeltaNet	2.12	5.28	13.43	5.77	7.78	3.12	7.11	17.39	11.33	23.00	21.01	25.68	**24.62**	25.60	13.80
1.3B	**CCQ-GLA**	2.08	**6.10**	**14.81**	5.18	**9.90**	3.27	**9.11**	18.10	12.28	**39.50**	26.09	24.79	23.26	**26.37**	**15.77**
1.3B	**CCQ-Gated DeltaNet**	2.24	3.91	13.26	3.39	4.63	2.51	8.76	17.99	**12.76**	36.50	**28.28**	**27.75**	23.68	21.29	14.78
Attention models																
1.3B	Transformer	0.28	2.58	6.61	0.66	1.55	0.00	3.47	3.65	7.54	6.50	3.92	2.56	14.41	13.41	4.80

Table 4: LongBench accuracy on 14 English tasks, grouped by task family, at the 1.3B scale. The 500M models are omitted because at that scale the per-task standard error is comparable to the spread between models and the ranking is dominated by noise. The Avg column is the unweighted mean over the 14 tasks. NQA: NarrativeQA. QQA: Qasper. MFQ: MultiField QA. HQA: HotpotQA. 2WM: 2WikiMultiQA. Mus: MuSiQue. GvR: GovReport. QMS: QMSum. MNs: MultiNews. TRC: TREC. TQA: TriviaQA. SSM: SAMSum. LCC: LCC. RBP: RepoBench-P. All values are percentages; **bold** marks the best recurrent number.

Kernel feature maps. A third group changes the similarity function before any write: Performer (Choromanski et al., 2021) uses random features to approximate the softmax kernel; RFA (Peng et al., 2021) applies a similar idea with explicit normalisation; T2R (Kasai et al., 2021) converts a pretrained Transformer into a linear RNN by learning a feature map; Hedgehog (Zhang et al., 2024) learns an MLP feature map that mimics the softmax similarity to recover the spiky kernel linear attention typically loses. These reshape how keys and queries are compared but still use a single inner-product read against the state.

Position of CCQ. CCQ acts at a stage none of these families touch. The write recurrence is left exactly as the chosen backbone defines it, and only the query used to read from memory is modified, using both the current query and the running second-order statistics of the stored keys. The cleaning is therefore memory-aware in a way that input-only query projections cannot be. Because it touches only the read, CCQ composes additively with any of the write-side mechanisms above; in our experiments we attach it without modification to GLA and Gated DeltaNet.

6 Conclusion

We introduced CCQ, a read-side correction that contracts the query along the high-density directions of memory using the running key covariance, read off the local Hessian of softmax's log-partition at the isotropic-attention point. Attached without modification to GLA and Gated DeltaNet, CCQ improves perplexity, downstream accuracy, S-NIAH 8K, length-extrapolation PPL, and LongBench at both 500M and 1.3B.

Limitations

We evaluated CCQ only at 500M and 1.3B parameters with up to 40B training tokens, only on two chunkwise linear-attention backbones (GLA and Gated DeltaNet), and only on perplexity, S-NIAH, length-extrapolation PPL, and LongBench; behaviour at 7B+ scale, on Mamba/Mamba2-style selective state-space layers, with RWKV-style channel mixing, or on agentic and multi-turn benchmarks is not characterised.

References

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

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. 2024b. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR.

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

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2024. xLSTM: Extended long short-term memory. In *Advances in Neural Information Processing Systems*.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2026. Titans: Learning to memorize at test time. *Advances in Neural Information Processing Systems*, 38:113506–113543.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2021. A framework for few-shot language model evaluation.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495. Association for Computational Linguistics.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*.

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

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021. Finetuning pretrained transformers into RNNs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10630–10643. Association for Computational Linguistics.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2024. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, and 15 others. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077. Association for Computational Linguistics.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021.

Random feature attention. In *International Conference on Learning Representations*.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2021. Hopfield networks is all you need. In *International Conference on Learning Representations*.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9355–9366. PMLR.

Jürgen Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. 2025. Learning to (Learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 57503–57522. PMLR.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. 2026. MesaNet: Sequence modeling by locally optimal test-time training. In *The Fourteenth International Conference on Learning Representations*.

Martin J. Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2024a. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024b. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. 2024. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations*.

Appendix

A Derivations

This appendix collects the calculations referenced in Section 3. The notation matches the main text: $q_t \in \mathbb{R}^d$ is the query at step t, $\{k_m\}_{m \le t}$ are the stored keys, and $A_t(q) = \log \sum_{m \le t} \exp(k_m^\top q)$ is the softmax log-partition.

A.1 Why *curvature*?

The word *curvature* in the title refers to the Hessian $\nabla_q^2 A_t(q)$ of the softmax log-partition. For a twice differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, the gradient ∇f captures the local slope and the Hessian $\nabla^2 f$ the local quadratic shape: directions where $\nabla^2 f$ is large are directions in which f rises steeply.

For A_t, this curvature has a concrete meaning. Directions in which many stored keys cluster make A_t rise quickly (because adding δ to a key direction multiplies many of the $\exp(k_m^\top q)$ summands at once). Conversely, directions where no key has been stored leave A_t flat. Softmax exploits this asymmetry to suppress crowded matches and let isolated ones survive. CCQ borrows only this geometric signal—the directions in which A_t rises steeply—and uses it to contract the query at read time, without evaluating exp.

A.2 Taylor expansion of A_t

Expanding A_t to second order around $q = 0$ gives

$$\begin{aligned} A_t(q) \;=\; & A_t(0) + \nabla A_t(0)^\top q \\ & + \tfrac{1}{2}\, q^\top \nabla^2 A_t(0)\, q + \mathcal{O}(\|q\|^3). \end{aligned} \tag{13}$$

A direct computation of $\nabla A_t(0)$ and $\nabla^2 A_t(0)$ proceeds as follows. Define the softmax distribution

$$p_m(q) \;=\; \frac{\exp(k_m^\top q)}{\sum_{j \le t} \exp(k_j^\top q)}, \qquad m \le t, \tag{14}$$

so that $A_t(q)$ is the log-normalizer of p. The standard exponential family identities for the log-partition function (Wainwright and Jordan, 2008) give

$$\nabla A_t(q) = \mathbb{E}_p[k] \;=\; \sum_{m \le t} p_m(q)\, k_m, \tag{15}$$

$$\begin{aligned} \nabla^2 A_t(q) &= \mathrm{Cov}_p[k] \\ &= \sum_{m \le t} p_m(q)\, k_m k_m^\top - \mathbb{E}_p[k]\, \mathbb{E}_p[k]^\top. \end{aligned} \tag{16}$$

At the isotropic-attention point $q = 0$, every key receives equal weight $p_m(0) = 1/t$, so

$$\nabla A_t(0) \;=\; \mu_t \;=\; \tfrac{1}{t} \sum_{m \le t} k_m, \tag{17}$$

$$\nabla^2 A_t(0) \;=\; \bar{C}_t - \mu_t \mu_t^\top \;=\; \mathrm{Cov}_{\le t}[k], \tag{18}$$

which is Eq. (12) in the main text. The two-term Hessian is exactly the running sample covariance $\Sigma_t \triangleq \bar{C}_t - \mu_t \mu_t^\top$ used by the cleaning operator throughout the paper.

A.3 Derivation of the cleaning operator

The Hessian identity $\nabla_q^2 A_t(0) = \Sigma_t$ supplies a positive semidefinite operator whose eigendirections are precisely the high-density directions of memory. We use it to define a linear *cleaning operator* on the query space directly, without appealing to a loss or to gradient descent:

$$q_t^{\mathrm{clean}} \;\triangleq\; (I - \lambda_t\, \Sigma_t)\, q_t \;=\; q_t - \lambda_t\, \Sigma_t\, q_t, \tag{19}$$

with a data-dependent gate $\lambda_t = \sigma(W_\lambda q_t + b_\lambda) \in (0, 1)$. The construction is read off the Hessian: along each eigendirection of Σ_t with eigenvalue σ_i, the operator multiplies the query component by $1 - \lambda_t \sigma_i$, so high-curvature directions of A_t are contracted more strongly than low-curvature ones, with λ_t setting the overall strength. The form $I - \lambda_t \Sigma_t$ is the unique linear correction that realizes this anisotropic contraction in the Hessian eigenbasis with a single scalar of freedom. With unit-norm keys, $\|\Sigma_t\|_2 \le \|\bar{C}_t\|_2 \le 1$, so the cleaning operator $I - \lambda_t \Sigma_t$ has spectrum in $(1 - \lambda_t,\, 1) \subset (0, 2)$. No query direction is annihilated. Since Σ_t is symmetric positive semidefinite, $I - \lambda_t \Sigma_t$ is itself symmetric and the cleaning is a soft *anisotropic contraction* in the eigenbasis of Σ_t; it rescales eigen-directions but does not rotate them. This is why the cleaning is stable at every position and at every sequence length, and why CCQ behaves well even when λ_t approaches 1.

A.4 Score-space form of the correction

Substituting $q_t^{\mathrm{clean}} = q_t - \lambda_t \Sigma_t q_t$ into the bilinear score $s_{tj} = k_j^\top q_t^{\mathrm{clean}}$ and expanding $\Sigma_t = \bar{C}_t - \mu_t \mu_t^\top$,

$$\begin{aligned} s_{tj}^{\mathrm{CCQ}} &= k_j^\top q_t - \lambda_t\, k_j^\top \Sigma_t\, q_t \\ &= k_j^\top q_t - \tfrac{\lambda_t}{t} \sum_{i \le t} (k_j^\top k_i)(k_i^\top q_t) \\ &\qquad + \lambda_t\, (k_j^\top \mu_t)(\mu_t^\top q_t). \end{aligned} \tag{20}$$

This is Eq. (11) in the main text. The first term is the standard inner-product score. The second term penalizes a candidate key k_j when it aligns with a stored direction k_i *and* the query already activates that direction; the third term adds back the rank-one contribution of the mean direction μ_t, so that only the spread around μ_t (not the mean itself) is suppressed.

B Chunkwise computation of the running covariance

CCQ needs $\Sigma_t q_t = \bar{C}_t q_t - \mu_t(\mu_t^\top q_t)$ at every position t. We compute the two terms by chunkwise scans that share the same structure as the backbone's chunkwise update.

Second-moment term $\bar{C}_t q_t$. For training in the chunkwise mode of Yang et al. (2024b,a), partition the sequence into chunks $[1{:}C], [C{+}1{:}2C], \ldots$ of fixed length C. For a query at position t in chunk r,

$$\bar{C}_t\, q_t \;=\; \tfrac{1}{t}\left(S_r^{\text{prev}} + I_{[r,t]}^{\text{intra}}\right) q_t, \qquad (21)$$

where $S_r^{\text{prev}} = \sum_{j \le rC} k_j k_j^\top$ is the chunk-prefix sum and $I_{[r,t]}^{\text{intra}} = \sum_{rC < j \le t} k_j k_j^\top$ is the intra-chunk contribution. S_r^{prev} has the form of the linear-attention state for the kernel feature map $\phi(k) = k$ and the value stream taken to be the key stream; we maintain it as a separate $d_k \times d_k$ running matrix alongside the backbone state. The intra-chunk term is the dense $Q_{[r]}\,(K_{[r]}^\top K_{[r]} \odot M)$-style block of Yang et al. (2024a). Length normalization by $1/t$ is applied after the product.

Mean term $\mu_t(\mu_t^\top q_t)$. The same chunk decomposition applies to the running mean: $M_r^{\text{prev}} = \sum_{j \le rC} k_j$ is carried as a d_k-vector alongside S_r^{prev}, and the intra-chunk prefix sum $I_{[r,t]}^{\text{intra},1} = \sum_{rC < j \le t} k_j$ is computed by a single cumulative sum within the chunk. Letting $\tilde{M}_{[r,t]} \triangleq M_r^{\text{prev}} + I_{[r,t]}^{\text{intra},1}$ denote the running key sum at position t, the mean correction reduces to one inner product and one outer product per position:

$$\mu_t(\mu_t^\top q_t) \;=\; \tfrac{1}{t^2}\, \tilde{M}_{[r,t]}\left(\tilde{M}_{[r,t]}^\top q_t\right). \qquad (22)$$

In practice both terms fuse with the backbone's existing chunkwise kernel: one extra reduction per chunk materializes S_r^{prev} and M_r^{prev}, and within the chunk we add one cumulative sum, one inner product, and one outer product per query position. Training throughput remains close to the underlying backbone.

Algorithm 1 CCQ chunkwise forward pass (training)

Require: sequence $\{x_t\}_{t=1}^T$, chunk size C, backbone state $S_0 = 0$, key second-moment state $S_0^K = 0$, key mean state $M_0 = 0$, position $t \leftarrow 0$

for each chunk $r = 0, 1, \ldots$ of length C **do**
  $Q_{[r]}, K_{[r]}, V_{[r]} \leftarrow$ project $\{x_{rC+1}, \ldots, x_{(r+1)C}\}$
  $\bar{Q}_{[r]} \leftarrow \ell_2$-normalize $Q_{[r]}$
  $\bar{K}_{[r]} \leftarrow \ell_2$-normalize $K_{[r]}$
  // CCQ read-side cleaning
  $P^{\text{prev}} \leftarrow S_r^K\, \bar{Q}_{[r]}^\top$ ▷ inter-chunk: prefix sum of $kk^\top$ times queries
  $P^{\text{intra}} \leftarrow (\bar{Q}_{[r]} \bar{K}_{[r]}^\top \odot M)\, \bar{K}_{[r]}$ ▷ intra-chunk dense block
  $\text{CumK}_{[r]} \leftarrow M_r + \textsc{cumsum}(\bar{K}_{[r]})$ ▷ running key sum at every position
  $D \leftarrow \text{diag}(1/t,\, 1/(t{+}1),\, \ldots,\, 1/(t{+}C))$ ▷ length normalization
  $\text{MeanCorr}_{[r]} \leftarrow D^2 \odot \text{CumK}_{[r]}\,(\text{CumK}_{[r]}^\top \bar{Q}_{[r]})$ ▷ $\mu_t(\mu_t^\top q_t)$, fused
  $\Lambda \leftarrow \sigma(\bar{Q}_{[r]} W_\lambda + b_\lambda)$ ▷ per-token gate
  $Q_{[r]}^{\text{clean}} \leftarrow \bar{Q}_{[r]} - \Lambda \odot \big(D\,(P^{\text{prev}} + P^{\text{intra}}) - \text{MeanCorr}_{[r]}\big)$
  $O_{[r]}, S_{r+1} \leftarrow \textsc{BackboneUpdate}(Q_{[r]}^{\text{clean}}, K_{[r]}, V_{[r]}, S_r)$
  $S_{r+1}^K \leftarrow S_r^K + \bar{K}_{[r]}^\top \bar{K}_{[r]}$ ▷ update second-moment state
  $M_{r+1} \leftarrow M_r + \sum_j \bar{K}_{[r],j}$ ▷ update mean state
  $t \leftarrow t + C$
end for
return $\{O_{[r]}\}$

C Algorithms

Algorithm 1 gives the chunkwise forward pass used during training. The two CCQ-specific lines are highlighted; the rest of the loop is exactly the backbone's chunkwise update and is left as an opaque BACKBONEUPDATE so the same procedure applies to GLA, Gated DeltaNet, or any other linear-attention variant. Algorithm 2 gives the per-token recurrent version used at inference time, with $\bar{C}_t$ maintained as a second recurrent state alongside the backbone state S_t.

In Alg. 1, the colour-highlighted lines are the only additions over a standard backbone forward pass; everything else (projections, BACKBONEUPDATE, output computation) is

Algorithm 2 CCQ recurrent inference (one token)

Require: input x_t, backbone state S_{t-1}, second-moment state $\bar{C}_{t-1}$, mean state μ_{t-1}, position t

$q_t, k_t, v_t \leftarrow$ project x_t
$\bar{q}_t \leftarrow q_t / \|q_t\|_2$
$\bar{k}_t \leftarrow k_t / \|k_t\|_2$
$\bar{C}_t \leftarrow \bar{C}_{t-1} + \frac{1}{t}(\bar{k}_t \bar{k}_t^\top - \bar{C}_{t-1})$ ▷ running-mean update of $kk^\top$
$\mu_t \leftarrow \mu_{t-1} + \frac{1}{t}(\bar{k}_t - \mu_{t-1})$ ▷ running-mean update of k
$\Sigma_t \leftarrow \bar{C}_t - \mu_t \mu_t^\top$ ▷ centred covariance / Hessian at $q = 0$
$\lambda_t \leftarrow \sigma(W_\lambda \bar{q}_t + b_\lambda)$
$q_t^{\text{clean}} \leftarrow \bar{q}_t - \lambda_t \Sigma_t \bar{q}_t$
$o_t, S_t \leftarrow \textsc{BackboneUpdate}(q_t^{\text{clean}}, k_t, v_t, S_{t-1})$
return o_t, new state $(S_t, \bar{C}_t, \mu_t)$

reused verbatim. In Alg. 2, the cache at inference time is the triple $(S_t, \bar{C}_t, \mu_t)$: the backbone state, the running key second moment, and the running key mean. They have sizes $d_v d_k$, d_k^2, and d_k respectively, so the total cache remains $\mathcal{O}(1)$ in sequence length.

Wall-clock overhead. Table 5 reports the per-step wall-clock time of a single attention/recurrent layer for each backbone and its CCQ variant, measured on one H200 at the 1.3B layer shape ($H = 2048$). We report forward + backward at the 4K training context (training-step cost) and forward-only at both the 4K training context and 8K extrapolated context (inference-step cost). Each entry is the mean ± standard deviation across 40 repetitions after 5 warm-up iterations, using `torch.cuda.synchronize` around every step. The CCQ overhead is ∼1.2 ms on top of GLA and ∼2.2 ms on top of Gated DeltaNet for a 4K training step, a flat additive constant dominated by the extra $d_k \times d_k$ statistic and the cleaning projection. Since the per-layer cost in the full model is dominated by the SwiGLU MLP (roughly 2× the attention block), the end-to-end training throughput penalty is materially smaller than the layer-level percentages suggest.

D Training Hyperparameters

Table 6 reports the training hyperparameters used across all runs. All variants share a 4K training context, SwiGLU MLPs, RMSNorm, tied input/output embeddings, and the chunkwise training mode;

Model	fwd+bwd	forward only	
	4K	4K	8K
GLA	2.28±0.02	0.75±0.02	0.96±0.01
CCQ-GLA	3.50±0.03	1.20±0.02	1.37±0.01
Gated DeltaNet	6.26±0.02	1.95±0.01	2.98±0.03
CCQ-Gated DeltaNet	8.06±0.03	2.24±0.01	3.67±0.03

Table 5: Per-layer wall-clock time (milliseconds) on one H200 at the 1.3B layer shape (batch 1, hidden 2048). *fwd+bwd 4K* is the training-step cost at the 4K context; *forward only* is the inference cost at the 4K training length and the 8K extrapolated length. Each cell is mean ± standard deviation across 40 repetitions after 5 warm-up steps. CCQ adds a flat ∼1–2 ms per step that does not grow with context.

CCQ adds only the read-side cleaning operator of Sec. 3.2 on top of its backbone. We use the head dimensions recommended in the backbone papers (head dim 256 for GLA, Gated DeltaNet at 500M and 1.3B); for the value side we use `expand_v` = 2, so the value heads have twice the dimension of the key/query heads. We inherit the remaining optimizer settings of Yang et al. (2025). The CCQ-specific projection W_λ is a single linear layer per head with bias initialised so that $\sigma(b_\lambda) = 0.1$; this keeps the initial cleaning strength small while leaving room for the gate to learn. Per-model layer counts and token budgets are listed in Table 1 of the main text.

Hardware. All models are trained on a single node with 8× NVIDIA H200 GPUs using FSDP with mixed-precision bfloat16 and the chunkwise training kernel of Sec. B. The same node is used for all evaluations (Sec. E): CSR/LM, S-NIAH, length-extrap PPL, and LongBench. The CCQ-specific $\Sigma_t q_t$ correction adds one $d_k \times d_k$ matrix and one d_k-vector per head to the backbone state, so the recurrent cache size during inference is within a few percent of the unmodified backbone at the scales we report.

E Evaluation Details

This appendix expands the four evaluation families referenced in Sec. 4.1.

(i) Common-sense reasoning and language modelling. We use `lm-evaluation-harness` (Gao et al., 2021) for the zero-shot tasks (no few-shot prompting, no in-context examples). Reported tasks and metrics: PIQA (acc), HellaSwag (acc_norm), WinoGrande (acc), ARC-Easy

Setting	Value
Corpus	FineWeb-Edu (subset)
Tokenizer	32K BPE
Context length	4096
Precision	bfloat16
Optimizer	AdamW
(β_1, β_2)	$(0.9,\ 0.95)$
Weight decay	0.1
Peak learning rate	10^{-3}
Warmup steps	1024
Schedule	cosine decay to 0

Table 6: Training hyperparameters shared across all models.

(acc_norm), ARC-Challenge (acc_norm), BoolQ (acc), SocialIQA (acc), LAMBADA OpenAI (perplexity and acc), and WikiText word perplexity. The 7-task Avg column of Table 2 is the unweighted mean of PIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, SIQA, and BoolQ.

(ii) Synthetic in-context retrieval. Single-needle NIAH tasks of Hsieh et al. (2024), with `max_length=8,192`, batch size 8 at 1.3B and 32 at 500M. Table 3 resolves the three single-needle tasks at 1K, 2K, 4K, and 8K (S-NIAH-1 pass-key, S-NIAH-2 number, S-NIAH-3 uuid; S-NIAH-3 stops at 4K because uuid retrieval collapses past that for every model we tested).

(iii) Length-extrapolation perplexity. For each (dataset, length L) the dataset's text is tokenized into a single stream, reshaped into non-overlapping windows of L tokens, and PPL is reported as the exponential of the token-weighted mean cross-entropy across windows. Nine evaluation lengths from 4K to 20K in 2K steps on six long-context corpora (GovReport, QMSum, NarrativeQA, Qasper, PG19, CodeParrot). All models are trained at $L = 4$K, so the 6K–20K points are pure extrapolation. The figure in the main text (Fig. 3) shows the five recurrent models whose perplexity is within a comparable range; the Transformer and GLA-Hedgehog curves are off-range on every panel and are not shown.

(iv) Long-context understanding (LongBench). The 14 English tasks of LongBench (Bai et al., 2024): NarrativeQA, Qasper, MultiField QA, HotpotQA, 2WikiMultiQA, MuSiQue (multi-doc QA); GovReport, QMSum, MultiNews (summarisation); TREC (few-shot classification); TriviaQA, SAMSum (single-doc QA, dialogue); LCC, RepoBench-P (code completion). Each task uses its official metric (F1 for QA, ROUGE-L for summarisation, classification accuracy for TREC, code-edit similarity for the code tasks). The 14-task Avg is the unweighted mean over the per-task scores. We only place the 1.3B block in the main text because at the 500M scale the per-task standard error is comparable to the between-model spread.

F Additional details on stability

The cleaning operator $q \mapsto (I - \lambda_t \Sigma_t)q$ is only well behaved when both Σ_t and λ_t are bounded. The two design choices in Eq. (9) guarantee this independently of sequence length.

Operator-norm bound on Σ_t. For unit-norm keys, the raw second moment $C_t = \sum_{j \le t} k_j k_j^\top$ has trace t, so its operator norm grows linearly in t. The sample second moment $\bar{C}_t = C_t/t$ has operator norm bounded by 1, since

$$\begin{aligned} \|\bar{C}_t\|_2 &= \sup_{\|x\|=1} \frac{1}{t} \sum_{j \le t} (k_j^\top x)^2 \\ &\le \sup_{\|x\|=1} \frac{1}{t} \sum_{j \le t} \|k_j\|^2 \|x\|^2 = 1. \end{aligned} \tag{23}$$

The centred covariance is dominated by the second moment: $\Sigma_t = \bar{C}_t - \mu_t \mu_t^\top \preceq \bar{C}_t$, so $\|\Sigma_t\|_2 \le \|\bar{C}_t\|_2 \le 1$.

Spectrum of the cleaning operator. With $\lambda_t \in (0, 1)$ and $\|\Sigma_t\|_2 \le 1$, the operator $I - \lambda_t \Sigma_t$ has spectrum in $(1 - \lambda_t,\ 1) \subset (0, 2)$. Hence $\|q_t^{\text{clean}}\|_2 \le \|q_t\|_2$ and no direction is annihilated. Bypassing either bound (using C_t instead of $\bar{C}_t$, or λ_t without the sigmoid) breaks this guarantee and we observed training instabilities on long sequences in early ablations.

G Derivation of the retrieval-margin proposition

This appendix derives the retrieval-margin shift quoted in Sec. 3.2 as Eq. (26), and spells out the three interpretable regimes referenced in the body.

Setup. Suppose the running key covariance is dominated by a single high-variance direction $u \in \mathbb{R}^d$, so that $\Sigma_t \approx \rho\, u u^\top$ for some $\rho \in (0, 1]$. Let $k_\star$ be a target key with $\alpha = \langle k_\star, u \rangle$, let k_d be a distractor key with $\beta = \langle k_d, u \rangle$, and let q be the current query.

Score difference. The CCQ-corrected scores are

$$\begin{aligned} s_\star^{\mathrm{CCQ}} &= k_\star^\top q - \lambda\, k_\star^\top \Sigma_t\, q \\ &\approx k_\star^\top q - \lambda\rho\,(k_\star^\top u)(u^\top q) \\ &= k_\star^\top q - \lambda\rho\,\alpha\,(u^\top q), \qquad (24) \\ s_d^{\mathrm{CCQ}} &= k_d^\top q - \lambda\, k_d^\top \Sigma_t\, q \\ &\approx k_d^\top q - \lambda\rho\,\beta\,(u^\top q). \qquad (25) \end{aligned}$$

Subtracting gives the margin shift

$$s_\star^{\mathrm{CCQ}} - s_d^{\mathrm{CCQ}} \;=\; (s_\star - s_d) + \lambda\rho\,(\beta-\alpha)\,(u^\top q). \qquad (26)$$

Tight-cluster specialization. If distractors lie tightly along u, then $\beta \approx 1$ and the margin shift simplifies to $\lambda\rho\,(1-\alpha)\,(u^\top q)$.

Decomposing the query. Writing $q = k_\star + \epsilon$ for a query that aims at the target plus some perturbation ϵ, we have $u^\top q = \alpha + u^\top \epsilon$, so the margin shift becomes

$$\Delta \;=\; \lambda\rho\,(1-\alpha)\,(\alpha + u^\top \epsilon). \qquad (27)$$

The three regimes referenced in the body follow directly:

(i) If $\alpha < 1$ and $\alpha + u^\top \epsilon > 0$, then $\Delta > 0$ and CCQ widens the target margin. Concretely, this happens when the target is not collinear with the high-variance direction and the query has a positive overlap with it.

(ii) If $\alpha \to 1$, the factor $(1-\alpha)$ vanishes and $\Delta \to 0$: the target lies along the high-variance direction, so CCQ contracts target and distractors equally and the margin is unchanged.

(iii) If $u^\top q \approx 0$, then $\Delta \approx 0$ regardless of α: the query has no projection along u, so the cleaning operator has nothing to suppress in the first place.

Remarks. The analysis is first-order in ρ and ignores sub-leading contributions from other eigenvectors of Σ_t and from distractors not collinear with u. Including those terms preserves the qualitative picture: Δ has the same sign as $(\beta-\alpha)(u^\top q)$ and vanishes at the same boundary cases.

H Empirical validation of the alignment assumption

The CCQ derivation assumes that retrieval-relevant keys live *off* the high-variance directions of memory, while distractor keys cluster inside them. This appendix documents the diagnostic that checks this assumption directly on pretrained models, as referenced in Sec. 3.3 and Fig. 1.

Prompts. To avoid drawing conclusions from a single phrasing we use a collection of distinct needle scenarios that share the same structural template but differ in the novel token, the carrying sentence, and the query template (passwords, access codes, magic words, courier phrases, parcel identifiers). Each prompt concatenates ≈ 120 filler sentences drawn from a fixed pool of generic English sentences, inserts the needle in the middle, and ends with a question targeting the novel token. The distractor pool is rotated between variants so each prompt sees a different filler arrangement. After tokenization, each prompt is 1500–1700 tokens and the needle span itself is 17–19 tokens.

A representative excerpt (with most distractors elided) is shown below; **bold** marks the needle sentence and *italic* marks the trailing query:

> The cat sat on the mat and watched the rain through the window. On Tuesday morning the conference room was unusually quiet. She poured the coffee slowly, careful not to wake the dog. ⟨… ∼ 60 more filler sentences …⟩ **Please remember this very carefully: the secret password is FLAMINGO47.** ⟨… ∼60 more filler sentences …⟩ The librarian stamped the books and slid them across the counter. *Q: What is the secret password? A: The secret password is*

Unique-token filter. The needle span necessarily includes generic tokens (e.g. "the", "is", "Please") that also appear all over the distractor context. Reporting the alignment over the full needle span therefore mixes truly novel signal with generic backbone tokens. We filter the needle set to token-ids that appear at most twice in the full prompt, which keeps only the genuinely novel sub-words (the FLAMINGO47, QUARTZ-2199 etc. pieces and a few rare punctuation tokens). Fig. 1 plots both the unique-token mean (solid) and the raw all-needle mean (dashed) so the effect of this filter is visible.

Targets. Two models are probed. **Qwen3-4B-Instruct-2507** (softmax attention; 36 layers) is loaded in bfloat16. **Gated DeltaNet 500M** (linear attention; 21 layers) is the checkpoint we used as a baseline in Sec. 4.1. For each model we sweep *every* layer, since different layers play different roles (Geva et al., 2021) (early = lexicon / syntax, middle = semantic content, late = task-specific routing) and a single-layer report would not be representative of the whole network.

Capture procedure. A forward hook is attached to the selected layer's W_K projection. After one

forward pass over the needle-in-a-haystack prompt, the captured tensor is reshaped to (T, H, d_k) where T is the token length and H is the number of key heads (with grouped-query attention, H is the number of *kv* heads). Each per-head key vector is then ℓ_2-normalized to match CCQ's standing assumption on the key magnitudes.

Alignment measure. For each head we compute the centred covariance $\Sigma^{(h)} = \frac{1}{T-1}\sum_t (k_t^{(h)} - \bar{k}^{(h)})(k_t^{(h)} - \bar{k}^{(h)})^\top$ and its top-16 eigenvectors $U^{(h)} \in \mathbb{R}^{d_k \times 16}$ (by eigh on a symmetric $d_k \times d_k$ matrix, $d_k \in \{128, 256\}$ for our targets). For every token t, the head-averaged *top-16 alignment* is

$$a_t^{(h)} = \frac{\|U^{(h)\top}(k_t^{(h)} - \bar{k}^{(h)})\|^2}{\|k_t^{(h)} - \bar{k}^{(h)}\|^2}, \qquad (28)$$

the fraction of the centred key's energy that sits in the high-variance subspace. We then average $a_t^{(h)}$ separately over (i) the tokens that fall inside the needle sentence and (ii) the remaining context tokens (excluding the trailing query tokens), obtaining the needle mean μ_{needle} and distractor mean $\mu_{\text{distractor}}$ for that head and layer. The *alignment gap* reported throughout this section is their signed difference,

$$\Delta_\mu \triangleq \mu_{\text{needle}} - \mu_{\text{distractor}}, \qquad (29)$$

so $\Delta_\mu < 0$ means needle tokens project *less* onto the top-16 high-variance subspace than distractor tokens do — the geometric configuration CCQ assumes. $|\Delta_\mu|$ measures how strongly the two groups are separated, and the sign indicates which group lives off the high-variance axes. The per-layer values plotted in Fig. 1 are Δ_μ averaged across heads.

Results. Fig. 1 shows Δ_μ for every layer of each model, averaged over the prompt collection. The unique-token Δ_μ is **negative at every single layer of both models** (36/36 for Qwen3, 21/21 for Gated DeltaNet) with a per-layer prompt-wise standard deviation around 0.02. The overall depth-averaged separation is $\bar{\Delta}_\mu = -0.111$ for Qwen3-4B-Instruct and $\bar{\Delta}_\mu = -0.209$ for Gated DeltaNet, roughly $2\times$ larger in the linear-attention model. Both curves share the same depth shape: strongest in the earliest layers, weakest near the middle, and partially recovering in the later layers, without ever crossing zero.

The depth shape needs a careful reading because the three regions have different mechanisms behind them, consistent with known transformer-layer specialization (Geva et al., 2021). At *early layers* keys are still close to token embeddings, so a rare token like `FLAMINGO47` occupies its own near-orthogonal direction simply by virtue of being rare — and any rare token would do the same. The strong negative Δ_μ at depth 0–3 is therefore partly a *lexical-rarity* signal and is *not* by itself strong evidence for the CCQ premise. The middle layers are the more demanding test: there each key has been integrated with its sentence context, so token identity has been smeared into shared semantic directions and the early-layer rarity artefact is gone. Despite that smearing, Δ_μ is still negative in the middle (≈ -0.05 for Qwen3, ≈ -0.20 for Gated DeltaNet); the needle key continues to sit off the high-variance subspace of memory even when it shares semantic content with the surrounding sentences. This is the regime CCQ is designed to act on. The partial recovery of $|\Delta_\mu|$ in the late layers is consistent with task-specific features re-separating the novel token at the answer position.

Fig. 2 additionally plots the per-token distribution of alignment values, pooled over every layer and prompt. The distractor and unique-needle histograms are clearly separated in both models, providing a direct geometric visualization of the statistic that Fig. 1 resolves by depth.

Per-(layer, head) breakdown. Figures 5 and 6 resolve the same statistic at the finest granularity available: one cell per *(layer, head)* pair, with no averaging across layers. Each heatmap has rows indexing layers (top = deepest, bottom = shallowest) and columns indexing attention heads. The cell colour and annotated value report Δ_μ (unique-needle mean minus distractor mean) for that specific (layer, head) pair, pooled across all prompts. Blue cells (the dominant colour in both models) indicate the configuration CCQ assumes; red cells indicate the opposite. The pooled-across-cells distribution shape itself is shown in Fig. 2, so the heatmap focuses exclusively on the separation statistic.

The geometric premise — needle keys sit *off* the high-variance subspace of memory — holds at the cell level, not just on average. Almost every (ℓ, h) cell in both models shows $\Delta_\mu < 0$; the few cells with $\Delta_\mu \gtrsim 0$ are isolated and concentrated in the small handful of mid-network layers where the depth-averaged separation in Fig. 1 is at its weakest. The linear-attention model continues to show a uni-

formly larger separation than the softmax model, head by head and layer by layer. This per-cell consistency is the strongest evidence we can offer for the claim short of training a new model.

Interpretation. The signal is in the right direction at all probed layers of both architectures. The fact that linear-attention keys show a stronger separation matches the intuition that linear attention has no softmax denominator to lean on and must learn this distinction in the key space itself. The diagnostic does not by itself prove that CCQ improves retrieval (only the language-modelling, S-NIAH, length-extrapolation, and LongBench tables in Sec. 4.2, 4.3, 4.4, and 4.5 can do that), but it shows that the underlying geometric premise of the derivation in Sec. 3.3 is realized in trained models rather than only in the toy retrieval analysis.

I Modelling choices and limitations

We collect here the modelling choices that simplify CCQ's local quadratic model, and the limitations they imply.

Why we use the unweighted covariance. The exact softmax Hessian away from $q=0$ is the $p(q)$-*weighted* covariance $\mathrm{Cov}_{p(q)}[k]$, which re-weights keys by their current attention mass. CCQ approximates this q-dependent operator by the unweighted running covariance Σ_t, gaining a pair of shared running statistics per head and avoiding any $\exp(\cdot)$ evaluation. The cost is that variance is measured over the marginal distribution of past keys, not over the conditional distribution induced by the current query; we view this as a coarse but cheap surrogate, and the empirical results suggest it is faithful enough for retrieval-heavy tasks. A q-dependent extension (e.g. a low-rank local reweighting of Σ_t toward $\mathrm{Cov}_{p(q)}[k]$) is a natural follow-up.

Interaction with decayed backbones. When the backbone forgets, the effective memory read from S_t is not exactly $\sum v_j k_j^\top$ but a decayed sum $\sum \gamma_{j,t} v_j k_j^\top$ for some per-key factor $\gamma_{j,t} \in (0, 1]$. CCQ uses an *undecayed* covariance Σ_t, which can over-penalize directions that the backbone has already forgotten. We did not observe this to hurt at our scales (CCQ-Gated DeltaNet improves on both WikiText perplexity and average downstream accuracy at 500M and 1.3B), but a decay-aligned covariance Σ_t^γ computed from γ-weighted running statistics is a principled variant and a useful direction for future work, particularly for backbones with aggressive forgetting.

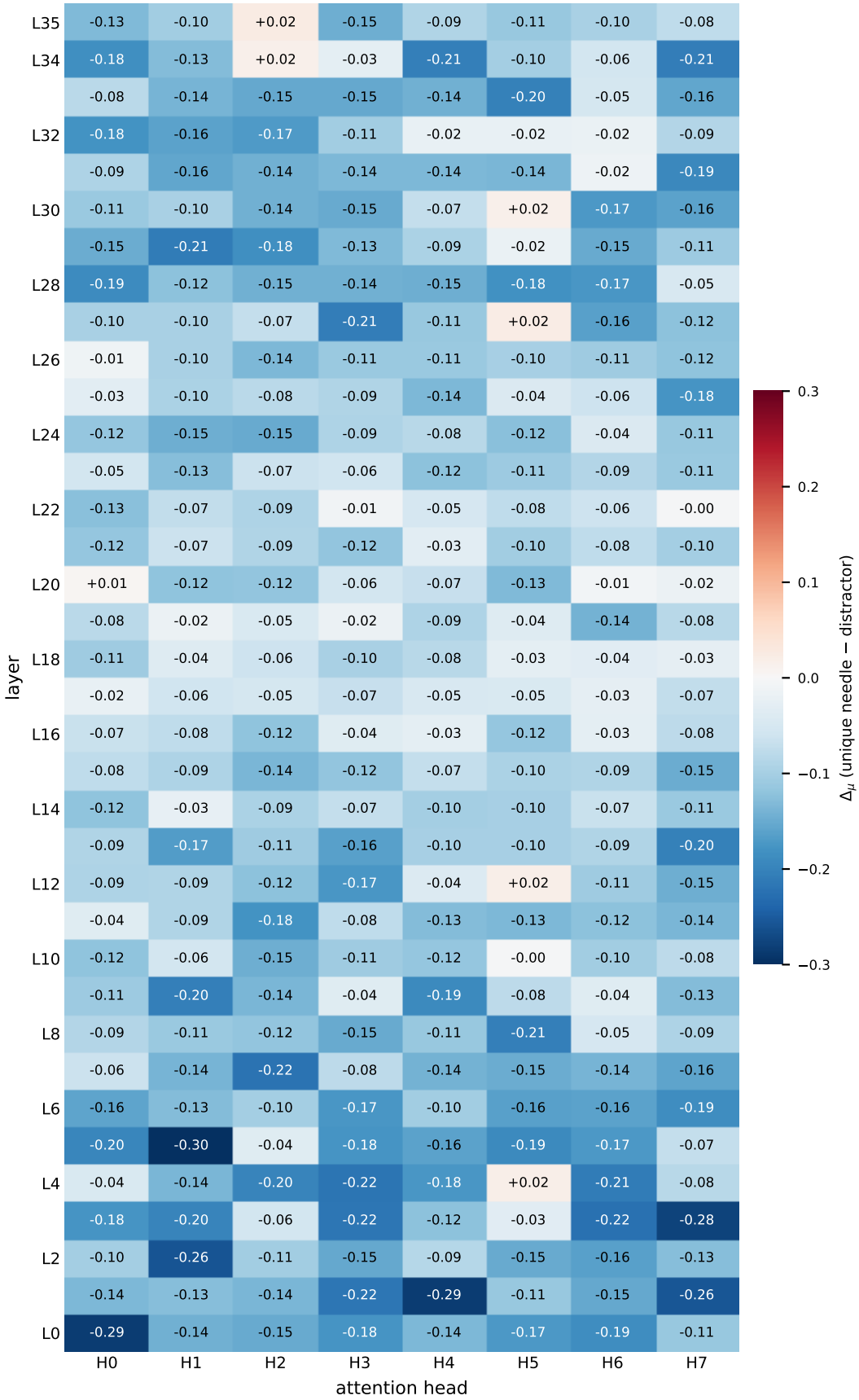

Figure 5: Per-(layer, head) Δ_μ heatmap for Qwen3-4B-Instruct (36 layers $\times$ 8 key heads). Each cell is $\Delta_\mu = \mu_{\text{unique-needle}} - \mu_{\text{distractor}}$ on top-16 alignment, pooled across all prompts. Diverging colormap centred at zero; blue = negative (matches the CCQ premise), red = positive. Annotated value is Δ_μ for that cell.

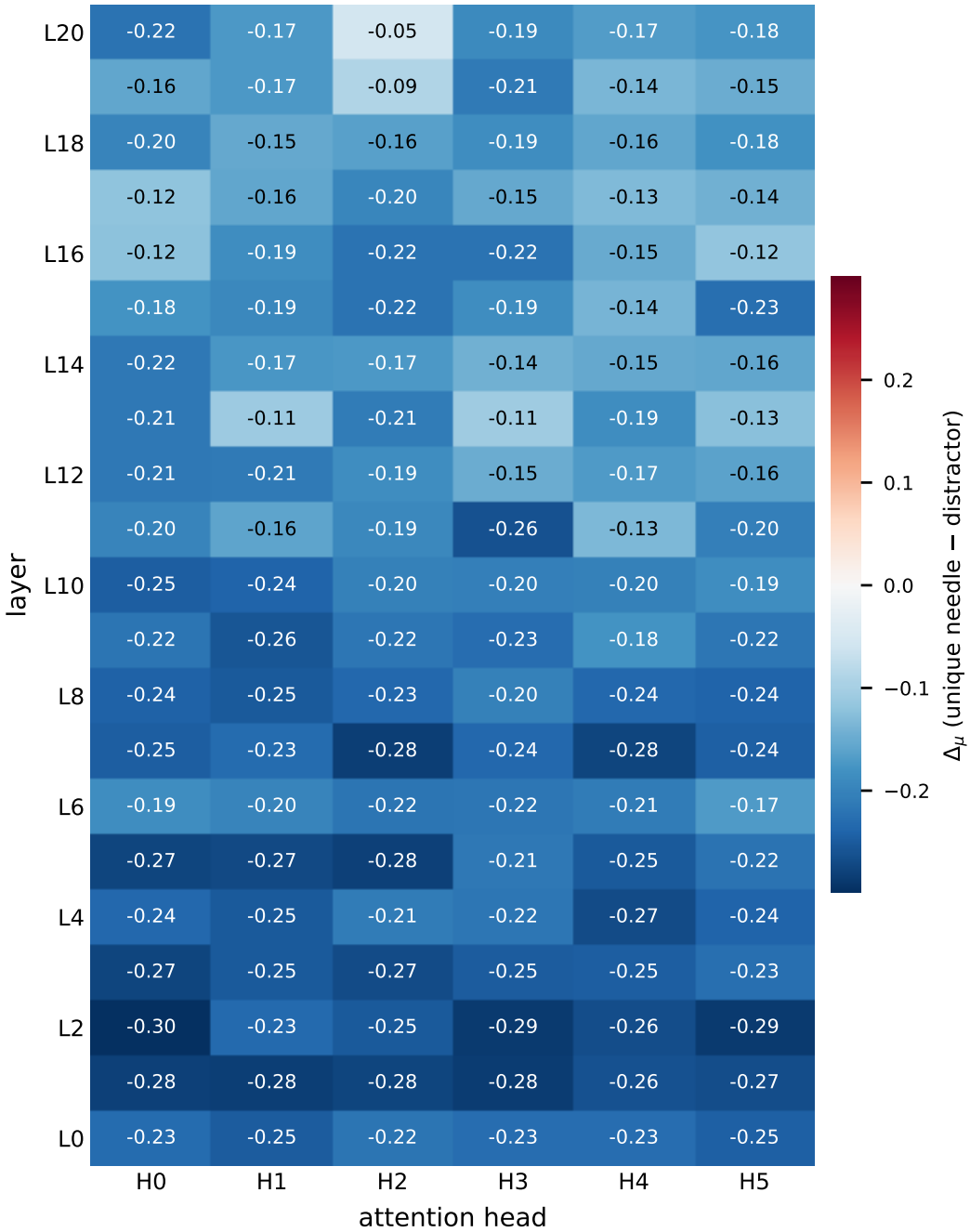

Figure 6: Per-(layer, head) Δ_μ heatmap for Gated DeltaNet 500M (21 layers $\times$ 6 heads). Same colormap and pooling convention as Fig. 5. The separation is visibly larger and more uniform than in the softmax model: every (layer, head) cell is strongly negative.