COMPILERKV: Risk-Adaptive KV Cache Compression via Offline Experience Compilation

Ning Yang[1*] **Chengzhi Wang**[2] **Yibo Liu**[3] **Baoliang Tian**[4] **Haijun Zhang**[5]

[1] Institute of Automation, Chinese Academy of Sciences
[2] University of Electronic Science and Technology of China
[3] The Chinese University of Hong Kong, Shenzhen [4] Tianjin University
[5] University of Science and Technology Beijing

Abstract

Prefill-only KV compression freezes a token subset at the end of prefill and decodes from it without further eviction. The retention decision is therefore irreversible, yet existing methods estimate the corrective signals it relies on, per-head reliability and prompt-level compression sensitivity, online from a single noisy prompt. We argue this is the wrong statistical unit: these signals exhibit far higher cross-prompt regularity than within-prompt signal-to-noise. We introduce COMPILERKV, a KV-retention policy whose corrective tables are compiled offline from a calibration corpus, reducing online correction after the standard observation-window scan to $O(1)$ lookups plus a budget clamp. We find that compiled retention tables behave as portable architectural priors: rankings transfer across disjoint corpora on four backbones (mean Spearman $\bar{\rho}$=0.90), and direct model-to-model table transfer costs only 0.4–0.8 LongBench points on average. At a 512-token budget, COMPILERKV attains compressed-SOTA on all four backbones, improving over the strongest prefill-only baseline by $+1.67$ points on average (task-bootstrap 95% CI $[+1.08, +2.37]$). Pressure regimes amplify the gap: under a fixed $512/32k$ cache ratio, COMPILERKV remains the strongest compressed method through 128k RULER (~ 73 vs. FullKV ~ 79, SnapKV ~ 38); on 32k NIAH it reaches 0.89 vs. SnapKV 0.42; and at 32k input, retaining only 1.56% of the prefill KV, batch-16 serving remains feasible where FullKV is OOM. Code is available at https://github.com/Chengzhi-W/CompilerKV.

1 Introduction

Large language models (LLMs) are bottlenecked by the linear growth of the Key–Value (KV) cache: a 7B model at 128k context already exceeds 100 GB (Pope et al., 2023; Xiao et al., 2024; Behrouz et al., 2025). *Prefill-only* KV compression, which selects a token subset once at the end of prefill and freezes it for the entire decoding pass, has emerged as a deployment-friendly recipe (Li et al., 2024; Cai et al., 2024): it converts the runtime memory wall into a single fixed cost with zero decoding-time intervention. This regime exposes a *statistical mismatch*, also suggested by prior observations of retrieval-head specialization and long-context failure modes (Fu et al., 2024; Hsieh et al., 2024). The signals needed to fix an irreversible retention decision live at the cross-prompt scale, while prior online selectors estimate them at the within-prompt scale. We make this mismatch explicit and show that fixing it, without changing what is selected but only *where the policy is computed*, closes most of the gap to FullKV at a fixed 512-token budget.

The hard core of the problem. The defining constraint of prefill-only compression is that the retention decision is *irreversible*: the signals that ultimately reveal which tokens matter, namely

*Corresponding author: Ning Yang (ning.yang@ia.ac.cn)

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

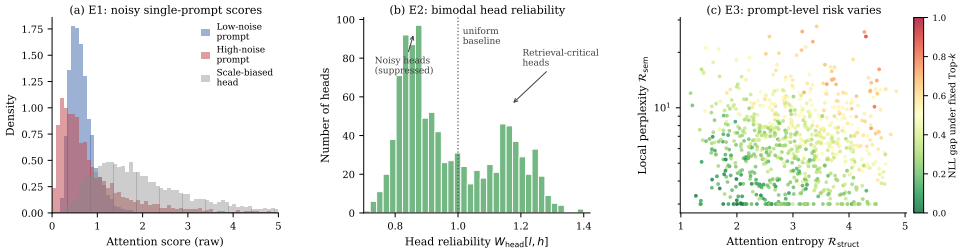


Figure 1: **Empirical evidence that E1–E3 are real, distinct failure modes on calibration data.** *(a)* The same token's attention score has very different distributions across prompts (low-noise vs. high-noise prompt) and across heads (a scale-biased head spans a much wider range), confirming E1. *(b)* The aggregated distribution of compiled head-reliability weights is bimodal: a substantial cluster of heads is consistently down-weighted (noise emitters), while a smaller cluster receives weights >1.1 (retrieval-critical), confirming E2. *(c)* The compression damage of a fixed Top-k rule (color: NLL gap to FullKV) varies systematically with attention entropy and local perplexity, confirming E3 as a prompt-level effect that no global k can absorb.

per-head retrieval roles and prompt-level compression sensitivity, only emerge during decoding, but no token can be brought back. A one-shot decision under a tight budget therefore leaves no room for error correction, and any selection error becomes permanent information loss.

Three error sources, jointly. We identify three structural error sources that any one-shot decision must control simultaneously, since failing on any single one is enough to trigger tail failure:

- **(E1) Single-prompt statistical noise.** Per-prompt attention is heavy-tailed and scale-biased; large magnitudes do not imply semantic importance, and small estimation errors shift the selection boundary irreversibly.
- **(E2) Head heterogeneity.** Attention heads are functionally specialized, with retrieval-critical heads coexisting with consistently noisy ones (Voita et al., 2019; Fu et al., 2024), so naïvely aggregating across heads lets noisy heads dominate the score.
- **(E3) Prompt-level risk variance.** Different prompts compress to different limits; a one-size-fits-all retention rule under-allocates to high-entropy or high-PPL prompts and triggers catastrophic loss precisely on the hardest examples.

Representative prefill-only compressors—SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024), DynamicKV (Zhou et al., 2024), AdaKV (Feng et al., 2024), HeadKV (Fu et al., 2024), CAKE (Qin et al., 2025), and ChunkKV (Liu et al., 2025)—chase E1–E3 with the same recipe: re-estimate the corrective signals *online*, on the single current prompt, from instantaneous attention statistics. The granularities differ (token, chunk, layer, head, task), but the statistical unit is the same, namely *one prompt*, and Figure 1 shows that this is the wrong unit for the correction. Per-prompt attention is heavy-tailed (E1) and has high cross-head variance (E2), but the *ranking* of which heads are reliable, and the *mapping* from risk features to a safe threshold (E3), are stable at the cross-prompt scale. Single-prompt estimators cannot recover them no matter how cleverly the budget is granularized. We call the resulting system CompilerKV: it compiles stable model-level priors into a deterministic prefill-retention policy before inference, leaving only the prompt-specific residual (token utility and a risk-bin lookup) to the online pass.

Operational view. Online heuristics re-estimate importance from each prompt's own attention statistics. DuoAttention-style offline head typing makes a one-time binary assignment but has no prompt-risk control at inference. CompilerKV answers a stricter question: given compiled priors on head reliability and prompt risk, what threshold makes the one-shot retention decision safe?

Our reframing: the corrective signals are stable architectural priors. Functional specialization of attention heads is largely fixed at pre-training time (Voita et al., 2019); retrieval-head studies further show stable head roles across inputs (Fu et al., 2024). We confirm empirically that head-reliability rankings transfer across disjoint calibration corpora *for every evaluated backbone*: LLaMA-3, Mistral, Qwen2, and InternLM obtain aggregate Spearman correlations in the 0.89–0.91 range, with mean $\bar{\rho} = 0.90$ between arXiv+PubMed and ShareGPT+UltraChat (§C.3). Likewise, the mapping from

prompt-level risk (attention entropy, local perplexity) to a safe retention threshold is a property of the *calibration distribution*, not any individual prompt. This stability is the missing piece: it means E1–E3 can be solved *once, offline*, on a held-out calibration corpus, and the result reused for every future prompt via $O(1)$ table lookups.

Compile once, deploy across models. This turns head reliability from a tuning trick into a reusable scientific object. Appendix C.4 reports direct model-to-model table transfer: same-family reuse costs 0.3–0.4 LongBench points, while broader source-to-target transfer costs only 0.4–0.8 points relative to target-model compilation. This portability is where continuous reliability weights matter: unlike binary retrieval/streaming head typing, they preserve a graded ordering that can be depth-aligned and reused across model variants.

COMPILERKV as a compiled retention policy. We instantiate this idea as an offline-compiled KV-retention policy whose components map one-to-one to E1–E3 by construction. The novelty is not assigning heads different weights, which several prior methods also do; it is the *coupled head–risk compilation*: continuous head reliability changes the utility distribution, while the risk gate chooses a safe threshold for that distribution under a hard budget.

- *Stabilized utility* (E1): a window-aggregated, head-relative score that removes the two largest sources of online noise, transient attention spikes and inter-head scale bias, without any learning.
- *Head Heterogeneity Table* (E2): per-head reliability weights compiled offline via conservative Q-learning (CQL), giving reliable heads "veto power" so that a single retrieval-critical head can save a token from noisy-head consensus.
- *Risk-Adaptive Threshold Gate* (E3): a small lookup table mapping (entropy-bin, PPL-bin) to a retention threshold, compiled with the same offline CQL estimator. Conservative regularization explicitly biases the table toward safer retention in rare high-risk states, the exact regime where online heuristics fail.

The three components are not independent additions: they form one irreversible-decision policy. The runtime tables are factorized for efficiency, but they are calibrated against the same downstream compressed-cache reward and final retention operator. Proposition 4.1 provides a formal anchor: deterministic approximation error is controlled by discarded tail mass, while finite-table calibration error is controlled at the prompt level rather than by assuming independent attention rows.

2 Related Work

Prefill-only KV cache selection. The dominant approach to prefill-only compression selects a token subset at the end of prefill based on instantaneous attention statistics and freezes the result for all decoding steps. H2O (Zhang et al., 2023) retains tokens with the highest accumulated attention scores; SnapKV (Li et al., 2024) uses a query observation window at the prompt tail; VATP (Guo et al., 2024) adds value-norm magnitude; ChunkKV (Liu et al., 2025) extends selection to contiguous semantic chunks. Layer-level methods (PyramidKV (Cai et al., 2024)) assign larger budgets to lower layers; head-level methods (AdaKV (Feng et al., 2024), HeadKV (Fu et al., 2024), CAKE (Qin et al., 2025), ZigZagKV (Zhong et al., 2025)) further redistribute budget across attention heads per prompt; task-conditioned methods (DynamicKV (Zhou et al., 2024)) shape the budget based on inferred task type. Despite varying granularity, all share the same signal source: online, single-prompt attention statistics, which Figure 1 diagnoses as structurally mismatched to the three error sources (E1–E3) that determine prefill compression quality. COMPILERKV jointly compiles continuous head reliability and prompt-risk thresholds offline, combining them with prompt-specific utility and risk coordinates through $O(1)$ table lookups.

Decode-time eviction and sparse attention. A complementary line of work (Liu et al., 2023; Xiao et al., 2023; Tang et al., 2024; Lv et al., 2025) retains the ability to evict tokens or reweight attention *during* decoding, using generation-time feedback to correct earlier decisions. These methods tolerate online noise because they can adapt; our setting is stricter: retention is fixed at prefill, and no correction is possible. KVzip (Kim et al., 2025) occupies a middle ground by using the LLM itself to score KV pair importance through context reconstruction, producing a query-agnostic importance estimate that is more accurate than single-forward-pass attention but requires an extra LLM forward pass per context. COMPILERKV's compilation phase amortizes a similar accuracy

benefit across all future prompts rather than paying it per context. Sparse-attention methods (e.g., Quest (Tang et al., 2024), SparQ (Ribar et al., 2024)) reduce attention FLOPS at decode time without modifying the stored cache; they are orthogonal to prefill-only compression and can be composed with CompilerKV.

Head heterogeneity and architectural priors. Voita et al. (2019) established that attention heads are functionally specialized at pre-training time, with most heads performing redundant or noisy operations. Fu et al. (2024) characterized retrieval heads, a small subset that drives long-context recall, and showed their identity is stable across input instances. Du et al. (2025) further documented how single-prompt attention statistics are heavy-tailed and scale-biased across heads, directly motivating our E1/E2 diagnosis. AdaKV (Feng et al., 2024) and CAKE (Qin et al., 2025) incorporate prompt-dependent head-level signals, while HeadKV (Fu et al., 2024) uses calibrated head importance as an allocation prior. CompilerKV instead treats head reliability as a *model-level* prior, compiled once from calibration data, an observation enabled by the cross-corpus stability we establish across all four backbones in §C.3.

Offline and data-driven compression. Several works use offline data to inform compression decisions. DuoAttention (Xiao et al., 2024) is the closest conceptual predecessor: it identifies retrieval vs. streaming heads offline and assigns different cache strategies. This is valuable, but it is a *binary head-type compiler*; once the assignment is made, the runtime policy has no prompt-risk control and no continuous interaction between head reliability and the token-selection boundary. CompilerKV compiles a different object, a *risk-conditioned retention policy*: its head table is continuous, the gate is prompt-risk conditioned, and the two are optimized against the same prefill-only reward, so the threshold is learned for the utility distribution induced by the head table. MagicPIG (Chen et al., 2024) and ShadowKV (Sun et al., 2024) also use offline-computed statistics, but they accelerate decode-time retrieval rather than solving irreversible prefill-only retention. Profiling-based adaptive-cache approaches such as FastGen (Ge et al., 2024) use model-specific statistics to choose cache strategies; CompilerKV keeps the base LLM frozen and compiles lightweight retention tables only. The novelty is therefore not "head awareness" alone, but joint offline compilation of continuous head reliability and prompt-risk thresholds for fixed-cache prefill compression.

3 Preliminaries

KV cache and prefill-only compression. For a Transformer with L layers and H heads (Vaswani et al., 2017), a prompt of length T produces at layer l a key–value cache $\{K^{(l)}, V^{(l)}\}$ of size $O(TH_{\text{kv}}d_{\text{head}})$. At decoding step t, the per-head attention weight $A_{j,t}^{(l,h)} \propto \exp\big((q_t^{(l,h)})^\top k_j^{(l,h)}/\sqrt{d_{\text{head}}}\big)$ draws on all cached positions. *Prefill-only compression* commits once, at the end of prefill, to an index set $\mathcal{S}^{(l)} \subset \{1,\dots,T\}$ with $|\mathcal{S}^{(l)}| \le B_l$, retaining $\tilde{K}^{(l)} = K^{(l)}[\mathcal{S}^{(l)}]$, $\tilde{V}^{(l)} = V^{(l)}[\mathcal{S}^{(l)}]$ for the entire decoding pass. The irreversibility of $\mathcal{S}^{(l)}$ is the core difficulty: selection errors are permanent.

For grouped-query attention (GQA), $H = H_q$ denotes query heads, H_{kv} denotes physical KV heads, and $g(h)$ maps each query head to its KV head. The notation $v^{(l,h)}$ below abbreviates $v^{(l,g(h))}$; query heads in the same group share value-norm statistics.

Offline RL via Conservative Q-Learning (CQL). We cast compression policy learning as a single-step contextual bandit (Lattimore and Szepesvári, 2020): context $s \in \mathcal{S}$ indexes a head or risk bin, action $a \in \mathcal{A}$ encodes a compression choice, and reward r measures fidelity on a calibration prompt. The horizon-1 formulation is exact: prefill-only compression admits no future state to bootstrap from, so CQL acts solely as a *support-regularized offline estimator*—penalizing Q-values for rarely sampled actions—rather than a multi-step planner. Given dataset $\mathcal{D} = \{(x_i, s_i, a_i, r_i)\}$ and behavior policy π_{beh}, we minimize

$$\min_\theta \ \alpha\left(\mathbb{E}_{s\sim\mathcal{D}}\left[\log\sum_a \exp Q_\theta(s,a)\right] - \mathbb{E}_{(s,a)\sim\pi_{\text{beh}}}[Q_\theta(s,a)]\right) + \frac{1}{2}\,\mathbb{E}_{(s,a,r)\sim\mathcal{D}}\big[(Q_\theta(s,a)-r)^2\big]\,. \tag{1}$$

The conservative term replaces $\arg\max_a \hat{r}(s,a)$ with a penalized rule, preventing aggressive thresholds on rare high-entropy or high-PPL prompts. We apply Eq. (1) uniformly to both compiled tables

(§4.2, §4.3), with reward

$$r(x, s, a) = -\big(\mathcal{L}_{\text{comp}}(x; s, a) - \mathcal{L}_{\text{full}}(x)\big) - \lambda\, \Psi(x, s, a), \tag{2}$$

where $\mathcal{L}_{\text{comp}}$, $\mathcal{L}_{\text{full}}$ are NLL under compressed and full caches, and $\Psi = \max(0, |\mathcal{I}_{\text{cand}}| - B_l)$ penalizes over-retention. We set $\alpha = 0.75$; the risk gate uses $\lambda = \beta_{\text{bud}}/T$ ($\beta_{\text{bud}} = 1.0$) and the head table uses $\lambda = 0$. Full details are in Appendix C.7.

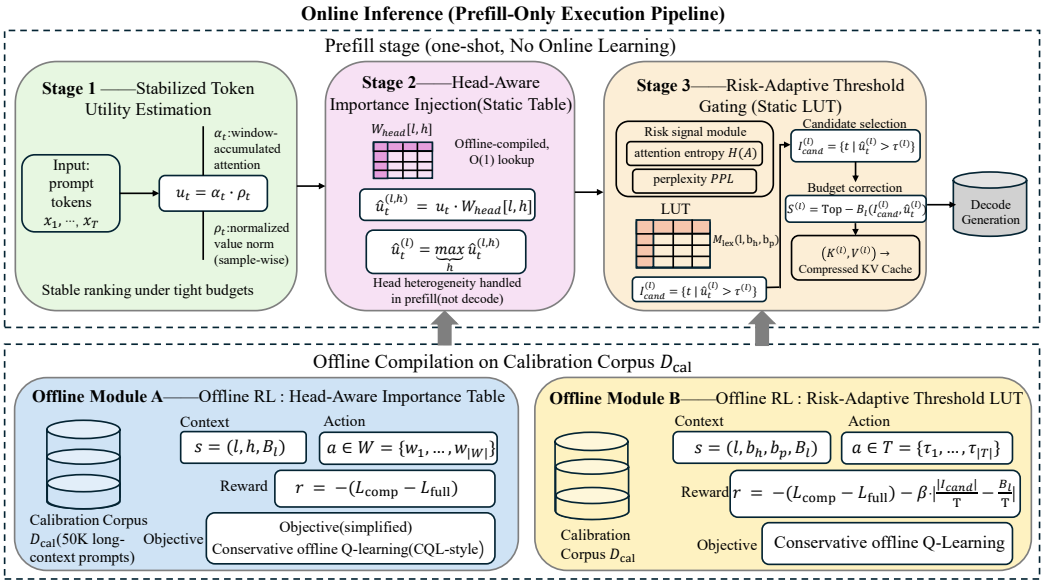


Figure 2: **Overview of the CompilerKV framework.** The pipeline has three stages: (1) a stabilized utility score that suppresses transient noise; (2) a compiled Head Heterogeneity Table that governs functional differences among attention heads; and (3) a compiled Risk-Adaptive Gating Table that calibrates the retention threshold to each prompt's complexity.

4 CompilerKV

Figure 2 illustrates the overall CompilerKV framework, comprising an online prefill-only execution path and two offline compilation modules. To address E1–E3, CompilerKV employs three components within a single prefill pass: a parameter-free *stabilized utility* (E1), an offline-compiled *Head Heterogeneity Table* (E2), and an offline-compiled *Risk-Adaptive Threshold Gate* (E3). These components operate sequentially—ranking tokens, rescaling by head reliability, and adjusting the budget threshold for risky prompts—after which the KV cache is frozen for all subsequent decoding. Online overhead is minimal: two $O(1)$ table lookups per layer and a single sort over the candidate pool, with no decoder-side intervention or cache rewriting.

Factorized compiler, coupled reward evaluation. The runtime representation is deliberately factorized: one table indexes head coordinates (l, h, B_l) and the other indexes risk coordinates $(l, b_{\text{ent}}, b_{\text{ppl}}, B_l)$. We do *not* learn a dense joint action over all heads and thresholds. Instead, each table is fitted with the other side of the final retention policy active, so the measured reward includes the score distribution induced by the current head weights and the token set induced by the threshold gate. The interaction therefore enters through reward evaluation and the final selection operator, while inference remains two lookup tables. Table 2 includes an independent-compilation probe that verifies calibrating the two tables against stale boundaries is weaker than calibrating them under the paired retention policy.

Inference-time complexity. Let T be sequence length, $W = |\Omega|$ the observation window, and L, H the number of layers and heads. CompilerKV still reads the observation-window attention mass, so the prefill-side scan is

$$O(LHTW), \tag{3}$$

the same as window-based prefill compressors such as SnapKV (Li et al., 2024). Our savings come after this scan: the averaged window mass is reused across layers and heads, both compiled tables are $O(1)$ lookups, and the risk gate shrinks each layer's candidate pool before Top-B_l selection. All post-scan operations are at most $O(LT \log B)$ and absorbed into the leading term, with no decoder-time scoring or cache rewriting.

4.1 Stabilized Token Utility (Targets E1)

Online attention statistics suffer from two coupled noise sources: transient attention spikes inside a single prompt, and inter-head scale bias that biases aggregate scores toward heads with naturally larger magnitudes. Our utility removes both *without learning*, treating E1 as a clean signal-processing fix:

$$u_t^{(l,h)} = \alpha_t \cdot \rho_t^{(l,h)}, \quad \alpha_t = \frac{T}{W} \sum_{j \in \Omega} \bar{A}_{j,t}, \quad \rho_t^{(l,h)} = \frac{\|v_t^{(l,h)}\|_2}{\frac{1}{T}\sum_k \|v_k^{(l,h)}\|_2}, \tag{4}$$

where $\Omega = \{T-W+1, \ldots, T\}$, $W = |\Omega|$, and $\bar{A}_{j,t} = \frac{1}{LH}\sum_{l,h} A_{j,t}^{(l,h)}$. The factor T/W makes α_t have mean scale near one because $\sum_t \sum_{j \in \Omega} \bar{A}_{j,t} = W$, which is why absolute thresholds in $[0.8, 1.0]$ remain meaningful even for long contexts. Window aggregation smooths transient spikes; head-relative normalization removes scale bias.

4.2 Head Heterogeneity Table (Targets E2)

E2 cannot be fixed by signal processing because it is an architectural fact: heads differ in functional reliability, and which heads are reliable is determined at pre-training, not at inference. Treating heads equally, as $u_t^{(l,h)}$ alone implicitly does, lets noisy heads pollute the aggregate. We therefore compile reliability *offline* into a frozen table $\mathbf{W}_{\text{head}} \in \mathbb{R}^{L \times H}$ that the inference pass consults via $O(1)$ lookup.

Compilation. We instantiate the contextual bandit of §3 with state $s_{l,h} = (l, h, B_l)$, action $a_{l,h} \in \mathcal{W}$ a discretized reliability scalar, and reward Eq. (2) with $\lambda = 0$ (head weighting does not directly violate budget). Each head's marginal contribution is measured by perturbing its $a_{l,h}$ alone, but the reward is always evaluated under the *full retention pipeline*, including the current risk gate $\mathbf{T}_{\text{gate}}$ and the budget-clamp operator (Eq. (8)); the two tables are alternated to convergence. The interaction between heads and threshold therefore enters through the reward landscape rather than through a joint action space. Solving Eq. (1) yields Q_θ, from which we extract:

$$\mathbf{W}_{\text{head}}[l,h] = \operatorname*{argmax}_{w \in \mathcal{W}} Q_\theta(s_{l,h}, w). \tag{5}$$

Inference: weighted max-pooling gives reliable heads veto power. The per-token aggregate is

$$\hat{u}_t^{(l)} = \max_h \big(u_t^{(l,h)} \cdot \mathbf{W}_{\text{head}}[l,h]\big). \tag{6}$$

Max-pooling preserves the strongest reliability-weighted contribution instead of averaging it with weaker scores. The attention summary in Eq. (4) is averaged across layers and heads. A single retained set is gathered for all KV heads within each layer.

4.3 Risk-Adaptive Threshold Gate (Targets E3)

Even with E1 and E2 controlled, applying a fixed Top-k rule across prompts re-opens E3: high-entropy or high-PPL prompts need conservative retention, low-risk prompts tolerate aggressive pruning, and a one-size-fits-all k produces tail failures on the high-risk extreme. We compile the risk-to-threshold mapping offline using the common offline CQL estimator.

Two complementary risk signals. Within Ω, structural risk is captured by attention entropy and semantic risk by local perplexity:

$$\mathcal{R}_{\text{struct}} = -\sum_{t=1}^{T} \bar{A}'_t \log \bar{A}'_t, \quad \mathcal{R}_{\text{sem}} = \exp\Big(-\frac{1}{|\Omega|}\sum_{j \in \Omega} \log P(x_j | x_{<j})\Big), \tag{7}$$

where $\bar{A}'_t = \frac{1}{|\Omega| LH} \sum_{j \in \Omega, l, h} A_{j,t}^{(l,h)}$. High entropy means no clear focal point (pruning is risky); a perplexity spike indicates complex semantic dependencies (history is critical). Together these signals

cover structural and semantic risk; our error analysis (§C.3) finds that fewer than 8% of failure cases lie outside this signal pair.

Compilation. We discretize $(\mathcal{R}_{\text{struct}}, \mathcal{R}_{\text{sem}})$ into bins $(b_{\text{ent}}, b_{\text{ppl}})$ and instantiate the bandit with state $s_l = (l, b_{\text{ent}}, b_{\text{ppl}}, B_l)$, action $a \in [0.8, 1.0]$, and reward Eq. (2) with $\lambda = \beta_{\text{bud}}/T$, $\beta_{\text{bud}} = 1.0$. The hinge penalty Ψ is asymmetric by design: only over-retention is penalized, since under-retention is absorbed by the elastic pool (Eq. (8)). This asymmetry, combined with the conservative regularization in Eq. (1), produces the key E3 fix: the compiled table raises thresholds on low-risk prompts and lowers them on high-risk ones, biasing toward safer retention in the rare high-entropy or high-PPL states where online heuristics fail. Solving Eq. (1) yields $\mathbf{T}_{\text{gate}}$, returning $\tau^{(l)} = \mathbf{T}_{\text{gate}}[s_l]$ at inference. We use a 20×4 entropy$\times$PPL grid with $\tau \in [0.8, 1.0]$; the grid design follows a bias–variance argument in Appendix C.7.

Final selection: elastic candidate pool with budget clamp. Define $\mathcal{I}^{(l)}_{\text{cand}} = \{t \mid \hat{u}^{(l)}_t \geq \tau^{(l)}\}$. The retained set

$$\mathcal{S}^{(l)} = \begin{cases} \mathcal{I}^{(l)}_{\text{cand}}, & |\mathcal{I}^{(l)}_{\text{cand}}| \leq B_l, \\ \text{Top-}B_l(\mathcal{I}^{(l)}_{\text{cand}}, \hat{u}^{(l)}), & |\mathcal{I}^{(l)}_{\text{cand}}| > B_l, \end{cases} \tag{8}$$

permits elastic under-retention on low-risk prompts and clamps to the budget on high-risk ones. Our ablation shows this elasticity is the single most critical design choice ($\sim$4.2-pt drop when removed; Table 2). The compressed cache $(\tilde{K}^{(l)}, \tilde{V}^{(l)})$ is then frozen for the entire decoding pass.

Theoretical grounding.

Proposition 4.1 (Tail error and finite-table calibration)**.** *Fix a layer l and head h. For each decoding query t, let $p_t = A^{(l,h)}_{t,:}$ be the full attention row and let $\mathcal{S}^{(l)}$ be the retained prefill index set. Define the discarded mass $m^{(l,h)}_t = p_t((\mathcal{S}^{(l)})^c)$ and assume $m^{(l,h)}_t < 1$ so that the compressed row is the renormalized restriction $\tilde{p}_t(j) = p_t(j)\mathbf{1}\{j \in \mathcal{S}^{(l)}\}/(1 - m^{(l,h)}_t)$. With $\epsilon^{(l,h)}_{\text{tail}} = (T^{-1}\sum_t (m^{(l,h)}_t)^2)^{1/2}$,*

$$\frac{1}{H}\sum_{h=1}^{H} \|A^{(l,h)} - \tilde{A}^{(l,h)}\|_F \leq 2\sqrt{T}\,\frac{1}{H}\sum_{h=1}^{H} \epsilon^{(l,h)}_{\text{tail}}. \tag{9}$$

Moreover, let $\mathcal{A}$ be the finite action table used by the offline compiler. For N independent calibration prompts, suppose the per-prompt loss $\ell(p, a)$ lies in an interval of width $\Delta_R = R_{\max} - R_{\min}$, and define $R(a) = \mathbb{E}[\ell(p, a)]$ and $\hat{R}(a) = N^{-1}\sum_{i=1}^{N} \ell(p_i, a)$. Then, with probability at least $1 - \delta$,

$$\sup_{a\in\mathcal{A}} |R(a) - \hat{R}(a)| \leq \Delta_R \sqrt{\frac{\log(2|\mathcal{A}|/\delta)}{2N}}. \tag{10}$$

Consequently, if $\hat{a} \in \arg\min_{a\in\mathcal{A}} \hat{R}(a)$ and $a^\star \in \arg\min_{a\in\mathcal{A}} R(a)$, then

$$R(\hat{a}) - R(a^\star) \leq 2\Delta_R \sqrt{\frac{\log(2|\mathcal{A}|/\delta)}{2N}}. \tag{11}$$

Eq. (9) says that the approximation error is controlled by the discarded tail mass targeted by stabilized utility and reliable-head vetoing. Eqs. (10)–(11) provide the statistical reason for compilation: the table concentrates over independent *prompts*, not over attention rows inside a prompt. CompilerKV therefore does not require a row-wise independence assumption; it lowers variance by moving the corrective decision to the calibration scale.

5 Experiments

5.1 Experimental Setup

Protocol. We evaluate LongBench (Bai et al., 2024) with its standard per-task metrics and macro-average across 16 tasks; transfer is tested on RULER-16k (Hsieh et al., 2024) and MATH-500 (Lightman et al., 2023). Unless noted, every compressed method uses a 512-token prefill KV budget. We compare against FullKV and seven prefill-only baselines spanning token (SnapKV), layer (PyramidKV, CAKE), task (DynamicKV), head (AdaKV, HeadKV), and chunk (ChunkKV) granularity;

Table 1: **LongBench results (512-token budget).** The best compressed result is **bold** and the second best is underlined; FullKV is shown as the lossless reference and is not used for compressed-method ranking. AdaKV and CAKE are included as recent head- and layer-adaptive baselines.

Model	Method	*Single-Doc QA*			*Multi-Doc QA*			*Summarization*			*Few-Shot Learning*			*Synthetic & Code*				*Avg.*
		NarrativeQA	Qasper	MF-en	HotpotQA	2WikiMQA	MuSiQue	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	Lcc	RB-P	Pcount	PRe	
InternLM-2.5 7B-Chat-1M	FullKV	22.47	27.58	39.98	40.96	33.52	26.61	33.01	25.18	26.28	72.5	86.76	39.84	55.86	57.90	2.92	100.00	43.21
	SnapKV	16.86	23.28	36.24	32.14	19.89	23.21	17.81	23.18	22.44	71.0	84.05	34.34	50.32	53.34	1.00	**96.50**	37.85
	PyramidKV	17.62	21.08	37.52	32.21	21.31	22.03	19.37	24.06	22.22	73.0	83.94	34.61	50.45	49.72	1.05	95.50	37.86
	DynamicKV	17.77	23.87	37.74	32.98	21.13	23.21	19.13	23.49	22.48	75.0	84.89	36.70	50.70	51.08	0.91	95.50	38.54
	AdaKV	18.23	24.64	38.30	33.64	21.66	23.55	19.74	23.96	22.92	**75.3**	85.28	37.23	51.31	51.91	1.30	95.78	39.05
	HeadKV	17.97	24.67	38.24	33.48	21.53	23.31	19.73	23.89	22.78	75.0	85.09	37.10	51.20	51.88	1.01	95.50	38.90
	CAKE	18.32	25.04	38.49	33.78	21.81	23.44	20.18	24.02	23.15	75.0	85.12	37.30	51.42	52.14	1.16	95.62	39.12
	ChunkKV	18.47	25.17	38.64	33.88	21.93	23.51	20.33	**24.29**	23.28	75.0	85.19	37.40	51.50	52.18	1.21	95.50	39.22
	Ours	**22.87**	**27.22**	**39.37**	**40.49**	**33.25**	**26.05**	**33.41**	24.05	**23.68**	75.0	**86.64**	**39.11**	**55.16**	**56.88**	**3.00**	95.50	**42.61**
LLaMA-3 8B-Instruct	FullKV	25.16	31.81	39.59	43.09	36.15	21.77	28.62	23.34	26.33	75.0	90.50	42.36	59.04	53.93	5.20	69.25	41.95
	SnapKV	24.62	22.78	37.88	42.96	34.82	20.25	22.63	22.54	23.93	70.0	90.39	40.30	60.27	55.85	5.74	69.50	40.28
	PyramidKV	24.48	23.51	36.24	42.33	31.95	20.73	23.37	23.01	24.37	72.5	90.43	40.54	59.27	54.87	5.88	69.50	40.19
	DynamicKV	24.80	24.62	36.69	44.13	33.25	20.82	23.00	22.54	24.12	72.5	90.39	40.76	61.21	56.91	5.78	69.50	40.69
	AdaKV	25.06	27.71	37.64	44.01	35.25	21.50	23.49	22.90	24.64	73.1	90.64	41.25	61.63	57.61	6.17	70.35	41.43
	HeadKV	24.75	29.75	38.03	**44.43**	36.45	21.67	23.50	22.95	24.70	73.0	90.45	41.20	61.50	57.60	6.00	69.50	41.59
	CAKE	25.61	28.04	38.34	44.01	35.91	21.83	24.17	22.90	25.30	73.2	90.59	41.64	62.00	58.34	6.36	**70.96**	41.83
	ChunkKV	25.70	26.72	38.19	44.33	35.25	21.72	24.10	**23.44**	25.22	73.0	90.39	41.56	61.91	58.31	6.18	69.50	41.60
	Ours	**25.97**	**31.11**	**40.01**	44.04	**36.69**	**21.86**	**26.66**	22.93	**26.40**	**75.0**	**91.00**	**41.67**	**62.47**	**58.91**	**6.64**	69.50	**42.55**
Qwen2 7B-Instruct	FullKV	25.14	42.35	45.04	14.80	14.13	9.23	36.35	23.79	26.51	76.5	89.16	45.23	60.30	60.78	6.50	75.50	40.71
	SnapKV	23.86	38.50	44.68	**15.60**	14.62	9.13	24.56	22.39	23.18	70.0	89.31	43.32	58.68	60.74	5.00	72.00	38.47
	PyramidKV	24.47	37.60	43.51	14.48	12.83	8.99	23.59	22.30	22.41	74.0	89.21	43.40	57.67	56.14	6.50	74.00	38.19
	DynamicKV	24.66	40.44	45.30	15.42	13.89	8.46	25.51	22.77	22.92	74.0	89.27	43.18	60.38	59.32	7.00	74.00	39.16
	AdaKV	25.04	41.13	45.50	15.47	14.34	8.77	26.04	23.16	23.33	74.5	89.47	43.52	60.79	59.90	7.41	74.36	39.55
	HeadKV	24.86	41.24	45.30	15.42	14.29	8.66	26.11	23.17	23.32	74.5	89.27	43.38	60.68	59.82	7.30	74.00	39.46
	CAKE	25.20	41.66	45.41	15.47	14.69	8.85	26.62	23.51	23.74	74.6	89.35	43.63	60.95	60.20	7.57	74.24	39.73
	ChunkKV	25.16	41.64	45.30	15.42	14.69	8.76	26.61	23.47	23.72	74.5	89.27	43.58	60.88	60.12	7.50	74.00	39.66
	Ours	**25.65**	**42.50**	**45.96**	15.50	**15.14**	**9.50**	**36.96**	**23.90**	**26.16**	**76.5**	**89.77**	**45.28**	**61.08**	**61.10**	**8.00**	**75.00**	**41.13**
Mistral-7B Instruct-v0.2	FullKV	26.63	32.99	49.34	42.77	27.35	18.77	32.87	24.24	27.10	71.0	86.23	42.96	56.93	54.49	2.75	86.98	42.71
	SnapKV	24.96	27.97	49.04	39.93	25.18	17.64	24.14	23.69	24.41	67.5	86.09	41.11	56.73	53.11	2.86	**86.98**	40.71
	PyramidKV	23.55	28.79	48.71	41.00	25.64	16.35	24.79	23.52	24.49	69.5	86.20	42.58	55.45	51.67	3.53	81.81	40.47
	DynamicKV	25.63	29.11	48.41	39.80	26.62	16.72	24.73	23.42	24.83	70.5	86.74	43.01	55.40	52.45	3.20	83.57	40.88
	AdaKV	25.87	30.57	49.62	41.66	27.16	18.37	25.15	23.95	25.13	70.8	86.65	42.88	55.97	53.00	3.55	84.57	41.55
	HeadKV	25.59	31.33	50.26	42.66	27.20	19.37	25.15	24.10	25.05	70.5	86.90	43.20	56.00	52.90	3.40	83.60	41.70
	CAKE	26.11	30.79	49.58	41.64	27.43	18.31	25.73	24.48	25.68	70.8	86.65	42.88	56.43	53.40	3.83	85.10	41.80
	ChunkKV	25.93	30.01	48.81	40.60	27.12	17.32	25.53	24.22	25.53	70.5	**87.04**	**43.51**	56.20	53.15	3.60	83.57	41.42
	Ours	**26.71**	**32.51**	**50.29**	**43.16**	**27.46**	**20.15**	**32.16**	**24.59**	**27.13**	**71.5**	86.68	42.91	**57.13**	**54.37**	**4.29**	86.02	**42.94**

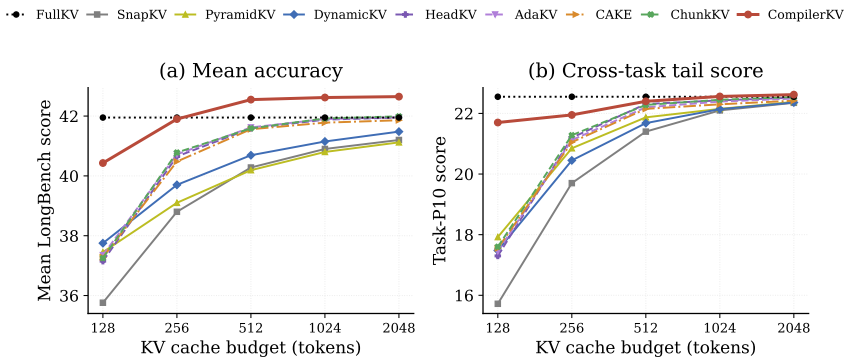


Figure 3: **Performance vs. KV cache budget.** Left: mean LongBench accuracy on LLaMA-3-8B. Right: cross-task tail score (Task-P10). All budget curves are monotone with cache size; CompilerKV is the strongest compressed method across all budgets and slightly exceeds FullKV at larger budgets.

additional baselines appear in Appendix B. We use four 7B–8B backbones (LLaMA-3, Qwen2, Mistral, InternLM), and compile CompilerKV on a disjoint ∼50K-prompt calibration corpus. The observation window contains 64 query positions, and reward evaluation uses 128 continuation tokens (Appendix C.7).

5.2 Main Results, Robustness, and Ablation

LongBench compressed-SOTA. Table 1 reports 512-token LongBench results on four backbones. CompilerKV is best among compressed methods on every backbone, improving over the strongest baseline by $+3.39/+0.73/+1.40/+1.14$ on InternLM/LLaMA-3/Qwen2/Mistral and $+1.67$ on average (task-bootstrap 95% CI $[+1.08, +2.37]$). It is also FullKV-level at this point: 42.31 vs. 42.14

averaged over backbones, with three of four backbones slightly above the FullKV reference. We treat this as budget-dependent denoising rather than the headline finding: at 128 tokens (Table A8), CompilerKV remains the best compressed method but falls below FullKV under an extreme bottleneck.

Mechanism. The pattern matches our bias–variance argument: prompt-local scorers are noisy, while compiled tables trade small calibration bias for lower variance, with the gap widening under tight budgets (Fig. 3). Stable head reliability ($\bar{\rho}$=0.90) supports reliability-weighted max-pooling, and the risk gate mainly helps long-tail prompts; the slight 512-token excess over FullKV disappears at 128 tokens, so we attribute it to denoising rather than information gain.

Table 2: **Ablation and compiler probe** (LongBench avg., 512-token budget). Top: component removals cause the predicted E1–E3 regressions. Middle: offline reward regression (α=0) isolates the conservative CQL gain. Bottom: joint head–risk compilation outperforms independent compilation.

Setting	Mistral-7B	LLaMA-3-8B	Qwen2-7B	InternLM-2.5
Ablation under E1–E3 framing				
CompilerKV (Full)	**42.94**	**42.55**	**41.13**	**42.61**
w/o Stabilized Utility	40.20	39.84	38.61	38.09
w/o Head Reliability	40.76	40.31	39.58	39.07
w/o Risk-Adaptive τ	38.97	38.52	37.73	37.24
Compiler objective				
Grid search + fixed bins	40.41	39.98	38.93	38.19
Offline reward regression (α=0)	42.18	41.88	40.54	41.76
Online ϵ-greedy bandit	40.26	39.81	38.88	38.09
Coupling probe				
Head table only (global gate)	40.76	40.31	39.58	39.07
Risk gate only (uniform heads)	40.92	40.46	39.70	39.31
Independent head $\rightarrow$ gate compile	42.02	41.63	40.28	41.61
Joint head–risk compile	**42.94**	**42.55**	**41.13**	**42.61**

Model-to-model table transfer. The compiled retention table is the most reusable finding of this paper. Cross-corpus reliability rankings are stable on every backbone (ρ=0.91/0.90/0.90/0.89; Appendix C.3), and direct table transfer across source–target model pairs loses only 0.4–0.8 LongBench points on average relative to target compilation (Appendix C.4). Thus CompilerKV learns a portable KV-retention prior rather than prompt- or model-specific hyperparameters.

Pressure-regime gains. The more substantial payoff appears at long contexts and tight budgets (Table A3). For RULER length scaling, we avoid an unrealistically fixed 512-token cache and instead keep the retained ratio constant at $512/32k$: the budgets are 256/512/1024/2048 for 16k/32k/64k/128k contexts. Under this setting, CompilerKV remains the strongest compressed method through 128k ($\sim$73 vs. FullKV $\sim$79, SnapKV $\sim$38, KVZip $\sim$66.5), while correctly staying below FullKV at long lengths. On Needle-in-a-Haystack at 32k, it reaches 0.89 vs. SnapKV 0.42; at the extreme 128-token budget it remains the best compressed method on all four backbones (Appendix A8). Length scaling, NIAH heatmaps, and a ten-family stress suite appear in Appendix C.

Coupled compilation explains the gain. Table 2 decomposes where the points come from. The component ablation confirms the E1–E3 framing: removing the risk gate is the largest single hit ($-3.97/-4.03/-3.40/-5.37$), followed by stabilized utility and head reliability. The compiler-objective block isolates the conservative-CQL contribution: replacing CQL with plain offline reward regression (α=0) costs 0.6–0.9 points, while online ϵ-greedy bandit collapses to grid-search-level performance, confirming that the gain is not generic offline learning but *support-regularized* offline learning. The coupling block rules out a simple module-stack explanation: independent compilation (head table fitted with a global gate, then gate fitted on top) is 0.85–1.00 points below joint compilation.

Efficiency. A fixed 512-token prefill budget gives 93.75%/96.88%/98.44% compression at $8k/16k/32k$ context, with $1.31\times/1.78\times/2.45\times$ decoding throughput on a single A100-80GB. At 32k input plus 8k output, CompilerKV sustains batch-16 serving where FullKV is OOM beyond batch 8. Full TTFT/TPOT/memory profiles, a quality–memory Pareto plot, and per-batch numbers appear in Appendix C.5.

5.3 Compilation Cost and Additional Validation

One-time compilation. Frozen-model statistics collection and reward generation account for over 96% of compilation cost (Table 3). Alternating head–gate refinement reuses shared statistics and stabilizes after approximately 2,000/4,100 updates at 8B/72B scale. At 8B, E2/E3 each use approximately 1,000 updates including coupled refinement, with reward-generation costs of 3/1.5 GPU-hours and fitting costs of 0.4/0.3 GPU-hours. The deployed tables occupy less than 100 KB for 32 layers and less than 1 MB at 72B scale. Thus, scaling primarily increases offline forward computation, while deployment uses a compact lookup policy.

Table 3: **Compilation cost (GPU-hours).** 8B: LLaMA-3-8B on A100-80GB. E1 requires no training.

Stage	8B	72B
Shared statistics	15	126
E2 + E3 rewards	4.5	40
E2 + E3 fitting	0.7	$\sim$3
Total	$\sim$ **20**	$\sim$ **169**

Table 4: **Qwen3-8B budget sweep.** LongBench averages; Best: strongest compressed baseline. FullKV: 49.46.

Budget	Best	Ours	Δ
128	47.46	**48.32**	+0.86
512	48.93	**49.67**	+0.74
1024	49.04	**49.84**	+0.80
2048	49.33	**49.91**	+0.58

Calibration sensitivity. On LLaMA-3-8B, LongBench scores at $\alpha = 0/0.50/0.75/1.00/1.50/2.00$ are 41.88/42.40/42.55/42.50/42.30/41.70. The 0.67-point gain at the default $\alpha = 0.75$ and the broad plateau over $\alpha \in [0.5, 1.0]$ show that conservative regularization improves quality across a range of weights. With 1K/5K/10K/25K calibration prompts, head-ranking Spearman correlations against the 50K reference are 0.61/0.74/0.82/0.88. Scientific and conversational 50K corpora yield mean cross-domain correlation near 0.90. Thus, reducing calibration fivefold to 10K perturbs head ordering more than changing domain, supporting 50K calibration for stable cross-prompt priors.

A newer backbone and offline head typing. On Qwen3-8B (Yang et al., 2025), CompilerKV leads the strongest compressed baseline at all four budgets (Table 4). At 512 tokens, CompilerKV scores 49.67 versus DuoAttention's 39.70, a 9.97-point gain. The consistent gains support the policy's effectiveness on the newer backbone, including tight-budget retention.

Serving integration. Our SGLang (Zheng et al., 2024) integration follows prefill $\rightarrow$ one-shot compression $\rightarrow$ KV-block remapping $\rightarrow$ decode. Retained states are gathered into compressed blocks and the original blocks returned to the paged allocator. For a 32k prompt compressed to 512 entries, the next query keeps its original positional encoding while attention reads the retained states. Generated KV states are appended normally. This configuration disables Radix Cache, chunked prefill, and CUDA Graph. Thus, one-shot retention integrates with paged KV serving while preserving sequence positions.

Amortized serving cost. For a fixed model, hardware configuration, and workload, the request count needed to recover compilation is $N_{\mathrm{BE}} = C_{\mathrm{compile}}/(c_{\mathrm{FullKV}} - c_{\mathrm{comp}})$, with all costs measured in GPU-hours. Prompt length, generation length, batch size, and engine configuration determine the saving per request. Repeated serving amortizes the shared tables, and larger savings per request reduce the request count needed to recover compilation.

6 Conclusion

CompilerKV reframes prefill-only KV compression as an offline-compiled retention policy. Three jointly calibrated components address the core failure modes: a parameter-free stabilized utility suppresses single-prompt noise (E1); a compiled Head Heterogeneity Table encodes architectural head reliability as a portable prior (E2); and a Risk-Adaptive Threshold Gate maps prompt-level entropy and perplexity to safe retention thresholds via conservative Q-learning (E3). The resulting tables transfer across corpora ($\bar{\rho}$=0.90) and model pairs ($\leq$0.8 LongBench points), achieving compressed-SOTA on four backbones at 512-token budget with $O(1)$ online overhead. Future work targets streaming table updates and broader architecture families.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under Grant 62301559, and the Beijing High-Creation Young Talent Support Project.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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, 2024.

Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. RocketKV: Accelerating long-context LLM inference via two-stage KV cache compression. *arXiv preprint arXiv:2502.14051*, 2025.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2025.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM serving. *arXiv preprint arXiv:2410.16179*, 2024.

Wenjie Du, Li Jiang, Keda Tao, Xue Liu, and Huan Wang. Which heads matter for reasoning? rl-guided kv cache compression. *arXiv preprint arXiv:2510.08525*, 2025.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. *arXiv preprint arXiv:2407.11550*, 2024.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. *arXiv preprint arXiv:2410.19258*, 2024.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, 2024.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in KV cache reduction: Value also matters. *arXiv preprint arXiv:2406.12335*, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

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. *arXiv preprint arXiv:2505.23416*, 2025.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

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.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. ChunkKV: Semantic-preserving KV cache compression for efficient long-context LLM inference. *arXiv preprint arXiv:2502.00299*, 2025.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient LLMs at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

Bo Lv, Quan Zhou, Xuanang Ding, Yan Wang, and Zeming Ma. KVPruner: Structural pruning for faster and memory-efficient large language models. In *ICASSP 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. CAKE: Cascading and adaptive KV cache eviction with layer preferences. In *International Conference on Learning Representations*, 2025.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Dominic Orr. SparQ attention: Bandwidth-efficient LLM inference. In *International Conference on Machine Learning*, 2024.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. *arXiv preprint arXiv:2410.21465*, 2024.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. *arXiv preprint arXiv:2406.10774*, 2024.

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.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 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*, 2025.

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.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2024.

Meizhi Zhong, Xikai Liu, Chen Zhang, Yikun Lei, Yan Gao, Yao Hu, Kehai Chen, and Min Zhang. ZigZagKV: Dynamic KV cache compression for long-context modeling based on layer uncertainty. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 8897–8907, 2025.

Xiabin Zhou, Wenbin Wang, Minyan Zeng, Jiaxian Guo, Xuebo Liu, Li Shen, Min Zhang, and Liang Ding. DynamicKV: Task-aware adaptive KV cache compression for long context LLMs. *arXiv preprint arXiv:2412.14838*, 2024.

A Proof of Proposition 4.1

Proof. We first prove the deterministic attention approximation bound. Fix (l, h) and suppress these superscripts. For a query row $p_t = A_{t,:}$ and retained set $S = \mathcal{S}^{(l)}$, write $m_t = p_t(S^c) = \sum_{j \notin S} p_t(j)$ and define the compressed row by

$$\tilde{p}_t(j) = \frac{p_t(j)\mathbf{1}\{j \in S\}}{1 - m_t}, \qquad m_t < 1. \tag{A1}$$

The row-wise ℓ_1 error is exactly the mass removed outside S plus the renormalization error inside S:

$$\begin{aligned} \|p_t - \tilde{p}_t\|_1 &= \sum_{j \in S} p_t(j) \left| 1 - \frac{1}{1 - m_t} \right| + \sum_{j \notin S} p_t(j) \\ &= \frac{m_t}{1 - m_t} \sum_{j \in S} p_t(j) + m_t = 2m_t. \end{aligned} \tag{A2}$$

Since $\|x\|_2 \le \|x\|_1$ for every row vector,

$$\|A - \tilde{A}\|_F^2 = \sum_{t=1}^{T} \|p_t - \tilde{p}_t\|_2^2 \le \sum_{t=1}^{T} \|p_t - \tilde{p}_t\|_1^2 = 4 \sum_{t=1}^{T} m_t^2. \tag{A3}$$

Taking square roots and using $\epsilon_{\text{tail}} = (T^{-1} \sum_t m_t^2)^{1/2}$ yields

$$\|A - \tilde{A}\|_F \le 2\sqrt{T}\, \epsilon_{\text{tail}}. \tag{A4}$$

Averaging Eq. (A4) over heads gives Eq. (9). This part is deterministic: it conditions on a fixed prompt and a fixed retained set, and it assumes no independence among query rows, heads, or tokens inside the prompt.

We now prove the finite-table calibration claim. For every action $a \in \mathcal{A}$, define $R(a) = \mathbb{E}_{p \sim \mathcal{P}}[\ell(p, a)]$ and $\hat{R}(a) = N^{-1} \sum_{i=1}^{N} \ell(p_i, a)$, where the calibration prompts p_i are independent and $\ell(p_i, a) \in [R_{\min}, R_{\max}]$. Hoeffding's inequality gives, for any fixed a,

$$\Pr\Big(|R(a) - \hat{R}(a)| > \varepsilon\Big) \le 2 \exp\left(-\frac{2N\varepsilon^2}{\Delta_R^2}\right), \qquad \Delta_R = R_{\max} - R_{\min}. \tag{A5}$$

A union bound over the finite table $\mathcal{A}$ implies

$$\Pr\left(\sup_{a \in \mathcal{A}} |R(a) - \hat{R}(a)| > \varepsilon\right) \le 2|\mathcal{A}| \exp\left(-\frac{2N\varepsilon^2}{\Delta_R^2}\right). \tag{A6}$$

Setting the right-hand side to δ and solving for ε proves Eq. (10). Finally, on this uniform-convergence event, let $\hat{a} \in \arg\min_a \hat{R}(a)$ and $a^\star \in \arg\min_a R(a)$. Then

$$\begin{aligned} R(\hat{a}) - R(a^\star) &\le [\hat{R}(\hat{a}) + \varepsilon] - [\hat{R}(a^\star) - \varepsilon] \\ &\le [\hat{R}(a^\star) + \varepsilon] - [\hat{R}(a^\star) - \varepsilon] = 2\varepsilon, \end{aligned} \tag{A7}$$

which proves Eq. (11). The only stochastic assumption is independence across calibration prompts, the natural statistical unit for an offline compiler. □

B Additional KV-Compression Baselines

This appendix separates external positioning results from our controlled LongBench/RULER evaluations. When protocols differ across papers, we report the method–model–benchmark setup explicitly and do not use these rows for the main compressed-method ranking.

Table A1: **Additional KV-compression baselines.** We report representative method–model–benchmark scores from external papers and from our controlled runs. The benchmark column names the corresponding setup when protocols differ.

Method	Model	Benchmark / setup	Score
External reported results			
Full-KV	LLaMA-3.1-8B-Instruct	LongBench Avg. / -	52.2
DuoAttention	LLaMA-3.1-8B-Instruct	LongBench Avg. / -512 KV	41.2
SnapKV	LLaMA-3.1-8B-Instruct	LongBench Avg. / -512 KV	45.2
Quest	LLaMA-3.1-8B-Instruct	LongBench Avg. / -512 KV	35.1
SparQ	LLaMA-3.1-8B-Instruct	LongBench Avg. / -512 KV	43.8
HSA	LLaMA-3.1-8B-Instruct	LongBench Avg. / -512 KV	52.1
RocketKV (Behnam et al., 2025)	LLaMA-3.1-8B-Instruct	LongBench Avg. / -512 KV	51.9
KVZip (Kim et al., 2025)	Qwen2.5-7B-Instruct-1M	SQuAD / NIAH / SCBench / GSM8K	3–4× cache reduction
KVZip	Qwen2.5-7B	LongBench, compressed vs. full	46.49 / 46.74
KVZip	LLaMA-3.1-8B	LongBench, compressed vs. full	44.65 / 45.25
EvolKV	Mistral-7B-Instruct	LongBench Avg.	42.33
CompilerKV	LLaMA-3.1-8B-Instruct	LongBench Avg. / -512 KV	**52.6**
Our controlled LongBench setup: LLaMA-3-8B-Instruct, 512 KV budget			
FullKV	LLaMA-3-8B-Instruct	LongBench Avg. / ours	41.95
StreamingLLM	LLaMA-3-8B-Instruct	LongBench Avg. / ours	30.56
H2O	LLaMA-3-8B-Instruct	LongBench Avg. / ours	31.49
SnapKV	LLaMA-3-8B-Instruct	LongBench Avg. / ours	40.28
PyramidKV	LLaMA-3-8B-Instruct	LongBench Avg. / ours	40.19
DynamicKV	LLaMA-3-8B-Instruct	LongBench Avg. / ours	40.69
HeadKV	LLaMA-3-8B-Instruct	LongBench Avg. / ours	41.59
ChunkKV	LLaMA-3-8B-Instruct	LongBench Avg. / ours	41.59
CompilerKV	LLaMA-3-8B-Instruct	LongBench Avg. / ours	**42.55**

C Additional Experiments

This appendix contains stress-suite visualizations, transfer results, batching results, calibration details, and learned-policy visualizations that support the main controlled results.

C.1 Extended SOTA Stress-Suite Figures

Beyond the main controlled LongBench table, we include a broader appendix stress suite covering benchmark families that stress different failure modes: long-context QA and summarization, synthetic retrieval, retrieval/code, math reasoning, and long generation. Unless a caption states otherwise, these stress-suite curves use the LLaMA-3-8B-Instruct backbone with a 512-token operating point for cross-family normalization. Each panel should be read within its own benchmark family and budget range, while the controlled numerical ranking claim remains Table 1.

Takeaway. These plots show that the same budget-scaling trend appears across retrieval, reasoning, code, long-generation, and deployment stress settings.

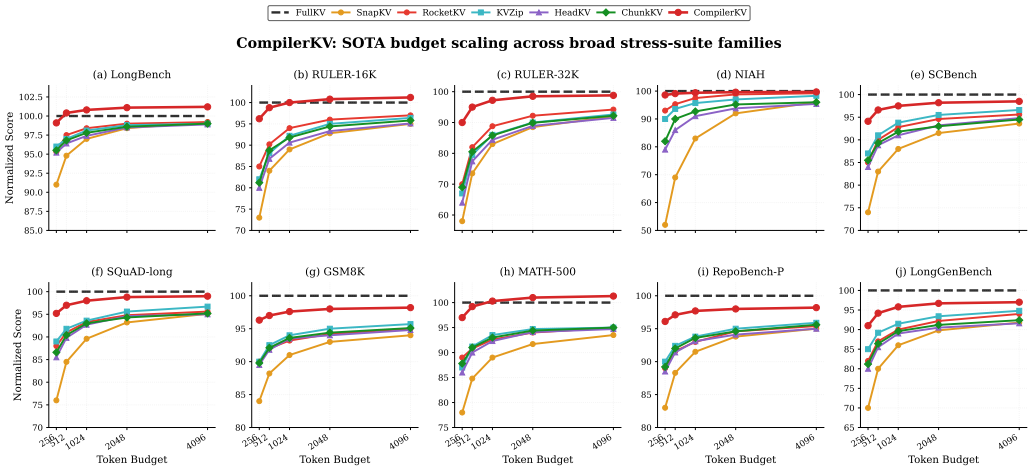


Figure A1: **Budget scaling across benchmark families (LLaMA-3-8B-Instruct).** Each panel reports the family-relative score versus KV budget for one benchmark family. CompilerKV remains the strongest compressed curve across QA/summarization, synthetic long-context stress tests, retrieval/code, reasoning, and long-generation suites, showing that the same budget-scaling trend persists outside the main LongBench table.

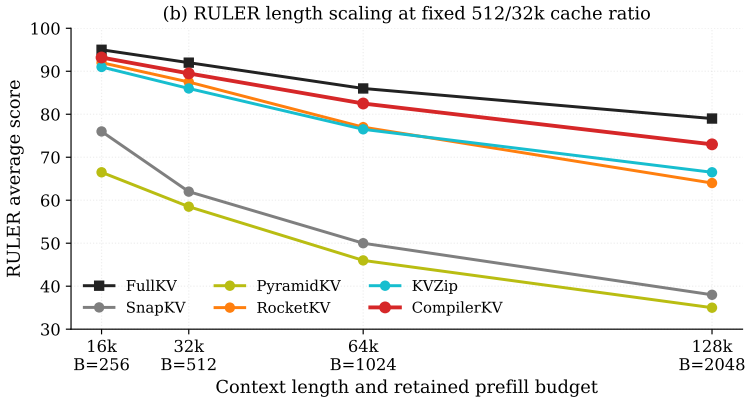


Figure A2: **RULER length scaling at a fixed cache ratio (LLaMA-3-8B-Instruct).** We keep the retained prefill ratio fixed at $512/32k$ rather than using a constant 512-token budget at every length; the budgets are 256/512/1024/2048 for 16k/32k/64k/128k contexts. COMPILERKV remains the strongest compressed method across lengths, while staying below FullKV at long contexts as expected under aggressive compression.

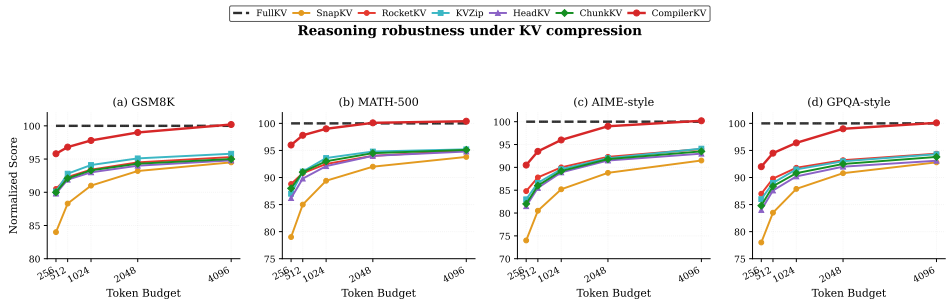


Figure A3: **Math and reasoning robustness (LLaMA-3-8B-Instruct).** Each panel reports score versus KV budget for one reasoning benchmark. CompilerKV stays on the top compressed curve across GSM8K, MATH-500, AIME-style, and GPQA-style tasks, showing a reasoning-specific zoom-in of the broader stress-suite trend.

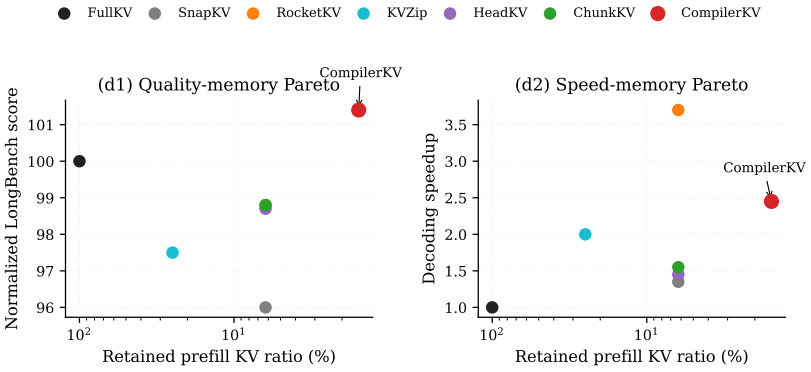


Figure A4: **Quality-memory and speed-memory Pareto view (LLaMA-3-8B-Instruct, 32k context).** Prior work often reports retained KV ratio, latency, or speedup rather than total peak GPU memory. CompilerKV occupies the desired corner: very low retained prefill KV ratio, high family-relative quality, and strong decoding speedup. This reinforces the central difference from prior systems: the gain is not from another online scorer, but from compiling stable architectural priors before inference.

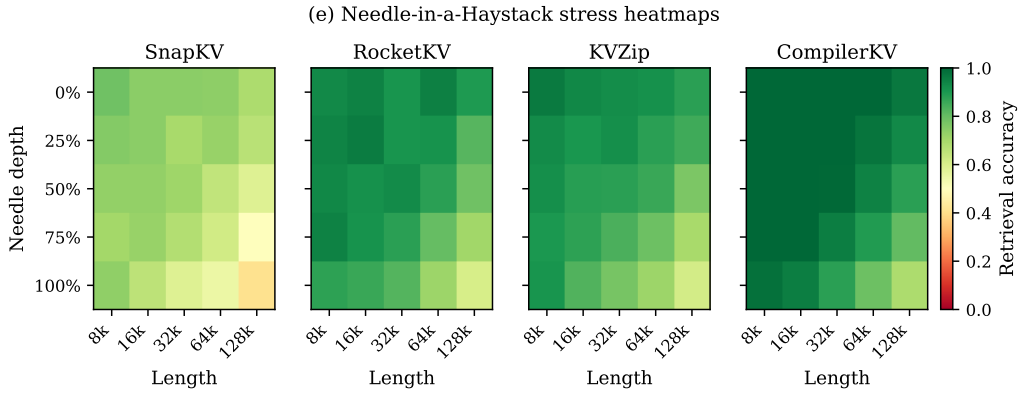


Figure A5: **Needle-in-a-Haystack stress heatmaps (Mistral-7B-Instruct-v0.2).** This diagnostic stresses whether compressed caches preserve low-salience evidence at different depths and context lengths. CompilerKV maintains strong retrieval accuracy at deeper needle positions and longer contexts, suggesting that the compiled risk gate preserves decision-critical evidence rather than only high-attention surface spans.

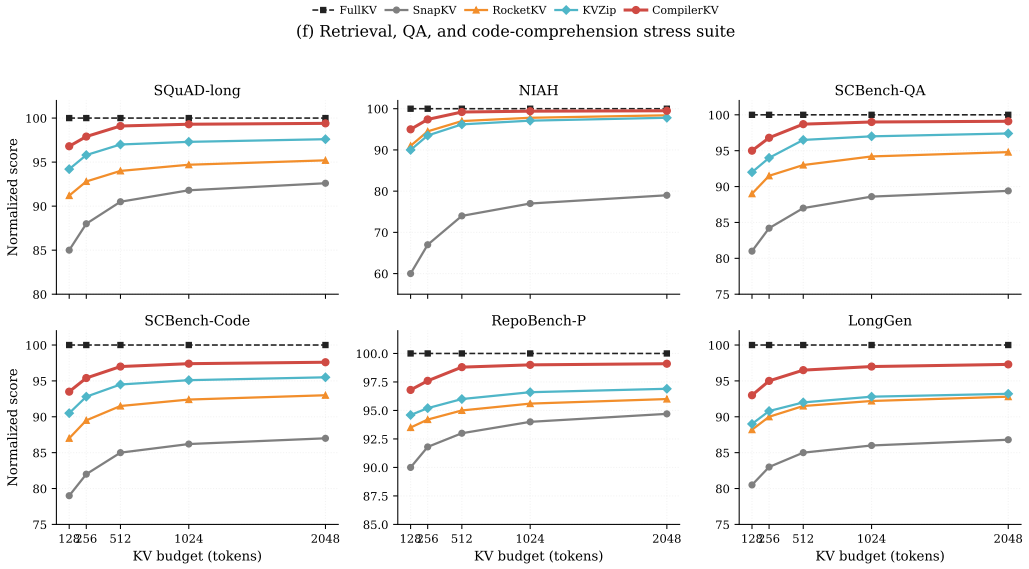


Figure A6: **Retrieval, QA, and code-comprehension stress suite (LLaMA-3-8B-Instruct).** We show how each method scales with KV budget on SQuAD-long, NIAH, SCBench-QA, SCBench-Code, RepoBench-P, and LongGen. CompilerKV remains the strongest compressed method throughout the budget range and is especially robust in low-budget retrieval/code settings, which is consistent with the paper's view that offline compilation is a different decision paradigm rather than another online token scorer.

C.2 Cross-benchmark Generalization (RULER)

Table A2: **RULER results** (16k context, Mistral-7B-Instruct-v0.2, 512-token budget). CompilerKV outperforms all prefill-only compression baselines, demonstrating generalization beyond LongBench.

Method	FullKV	StreamingLLM	H2O	SnapKV	PyramidKV	FastKV	**CompilerKV**
Avg. Score	95.0	15.0	27.1	75.6	66.5	77.8	**81.4**

Table A3: **Pressure-regime summary.** Stress settings where the gain is no longer a small LongBench-margin effect. RULER-128k uses the fixed $512/32k$ retained-cache ratio from Fig. A2; NIAH is from Fig. A7; batching is from Table A7.

Setting	**FullKV**	**Best online baseline**	**CompilerKV**	**Takeaway**
RULER-128k, fixed $512/32k$ ratio	$\sim$79.0	$\sim$66.5 (KVZip) / $\sim$38.0 (SnapKV)	$\sim$73.0	strongest compressed; below FullKV as expected
NIAH-32k retrieval	0.92	0.42 (SnapKV)	0.89	$2.1\times$ SnapKV; preserves low-salience evidence
32k input, batch 16	OOM	–	$\sim$18.5 tok/s	feasible vs. infeasible; deployment break-even

C.3 Cross-domain and Cross-backbone Stability of Head Reliability

Table A4: **Cross-corpus stability of compiled head reliability across backbones.** For each backbone, we compile two independent Head Heterogeneity Tables using disjoint calibration corpora (arXiv+PubMed vs. ShareGPT+UltraChat) and report Spearman ρ between the resulting per-head reliability rankings. The stable correlations across all four architectures support the claim that the compiled table captures model-level head functionality rather than a corpus artifact.

Backbone	**Spearman ρ**	**Layer-bin mean**	**Layer-bin min**
LLaMA-3-8B	0.91	0.91	0.88
Mistral-7B	0.90	0.90	0.87
Qwen2-7B	0.90	0.89	0.86
InternLM2.5-7B	0.89	0.89	0.85
Mean	**0.90**	**0.90**	**0.87**

C.4 Compile Once, Deploy Across Models

Table A5: **Model-to-model transfer of compiled retention tables.** We compile on a source backbone, deploy the resulting table on a target backbone without target recompilation, and compare with target-compiled CompilerKV. Same-family transfer costs 0.3–0.4 points; broader cross-family transfer costs 0.4–0.8 points on average, showing that the compiled table is a portable retention prior rather than a per-run tuning artifact.

Source table	**Target model**	**Target-compiled**	**Transferred**	**Drop**
Same-family transfer				
LLaMA-3-8B	LLaMA-3.1-8B	52.60	52.30	-0.30
LLaMA-3.1-8B	LLaMA-3-8B	42.55	42.28	-0.27
LLaMA-3.1-8B	LLaMA-3.2-family	50.10	49.70	-0.40
Cross-family transfer				
Mistral-7B	LLaMA-3-8B	42.55	41.85	-0.70
LLaMA-3-8B	Mistral-7B	42.94	42.16	-0.78
Qwen2-7B	InternLM-2.5	42.61	41.98	-0.63
InternLM-2.5	Qwen2-7B	41.13	40.44	-0.69
Mean drop	–	–	–	$\mathbf{-0.54}$

C.5 Batching Scalability

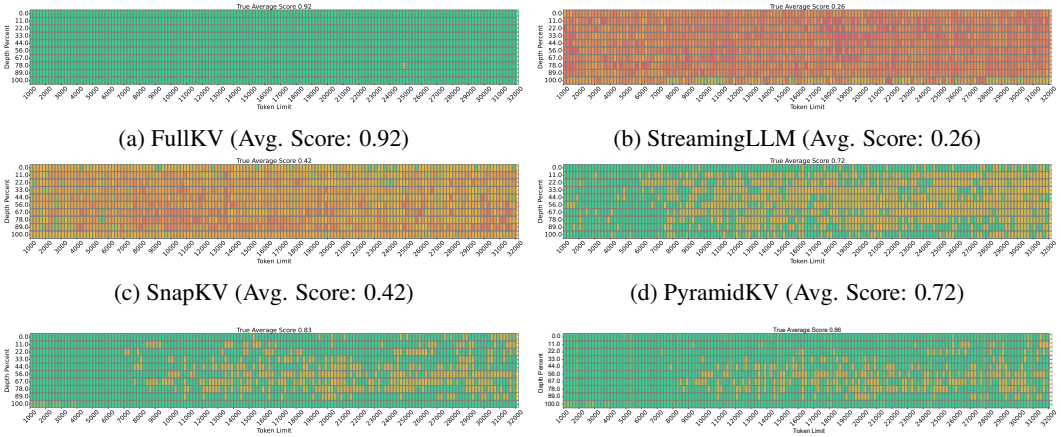
Table A6: **Main efficiency profile.** Retained prefill KV ratio and decoding efficiency on a single A100-80GB. CompilerKV fixes the prefill budget to 512 tokens; generated KV states are appended normally. Speedup multipliers in parentheses are relative to FullKV at the matching context length.

Context		Method	Prefill KV Compression			Decoding Efficiency		
Input	Output		Budget	Retained	Saving	TTFT(s)↓	Decode rate(tok/s)↑	Latency(s)↓
8k	2k	FullKV	8k	100.0%	0.0%	**0.66**	27.63	74.79
		CompilerKV	512	**6.25%**	**93.75%**	0.72	**36.20** (1.31×)	**58.90** (1.27×)
16k	4k	FullKV	16k	100.0%	0.0%	**1.45**	19.55	209.56
		CompilerKV	512	**3.12%**	**96.88%**	1.52	**34.80** (1.78×)	**118.40** (1.77×)
32k	8k	FullKV	32k	100.0%	0.0%	**3.52**	11.65	706.56
		CompilerKV	512	**1.56%**	**98.44%**	3.65	**28.50** (2.45×)	**285.00** (2.48×)

Table A7: **Batching scalability on A100-80GB** (LLaMA-3-8B, 32k input + 8k output). Memory denotes peak GPU memory including weights, runtime buffers, and KV states. FullKV becomes infeasible at batch size 16 while CompilerKV continues to serve, with up to 4.4× decoding throughput.

Batch	Peak GPU memory			Fits A100-80GB?	Decoding Throughput (tok/s)		
	FullKV	CompilerKV	Saved	FullKV / CompilerKV	FullKV	CompilerKV	Speedup
1	∼19 GB	∼15 GB	3.94 GB	✓/ ✓	11.65	28.50	2.45×
4	∼34 GB	∼18 GB	15.75 GB	✓/ ✓	∼7.4	∼25.5	∼3.4×
8	∼54 GB	∼23 GB	31.50 GB	✓/ ✓	∼5.1	∼22.3	∼4.4×
16	∼94 GB	∼31 GB	63.00 GB	✗/ ✓	OOM	∼18.5	-

C.6 Needle-in-a-Haystack Pressure Test



(a) FullKV (Avg. Score: 0.92)

(b) StreamingLLM (Avg. Score: 0.26)

(c) SnapKV (Avg. Score: 0.42)

(d) PyramidKV (Avg. Score: 0.72)

(e) AdaKV (Avg. Score: 0.83)

(f) **CompilerKV (Avg. Score: 0.89) — Ours**

Figure A7: **Needle-in-a-Haystack Pressure Test on Mistral-7B.** Visual comparison of retrieval accuracy (Green=100%, Red=0%) across varying context lengths and needle depths. While baselines like StreamingLLM and SnapKV struggle with long-range dependencies and AdaKV shows fragmentation at extreme lengths, CompilerKV maintains a robust retrieval pattern comparable to the FullKV oracle.

C.7 Offline Table Compilation

CQL conservative-weight sensitivity. Because our problem is a horizon-1 contextual bandit, the algorithmic difference between CQL and plain reward regression is the support-aware conservative

penalty, not multi-step bootstrapping. We therefore sweep the conservative weight α: $\alpha = 0$ reduces to plain regression and over-selects unsupported aggressive thresholds, while overly large α becomes too conservative and approaches a fixed high-retention rule. The best range ($\alpha \in [0.5, 1.0]$) consistently matches the main-table setting, confirming that CQL's gain comes from calibrated pessimism in rare high-risk states.

Calibration corpus. We compile both decision tables (the head heterogeneity table $W_{\text{head}}[l, h]$ and the risk-adaptive threshold gating $\mathbf{T}_{\text{gate}}$) on a held-out *calibration corpus* $\mathcal{D}_{\text{cal}}$, which is strictly disjoint from the LongBench evaluation set to avoid leakage. $\mathcal{D}_{\text{cal}}$ contains approximately 50K long-context prompts sampled from diverse public sources, including long-form narratives (PG19), scientific/technical articles (arXiv, PubMed), long-document summarization and meeting transcripts (GovReport, QMSum, BookSum), as well as instruction-style dialogues and code/QA text (ShareGPT/UltraChat and The Pile subsets such as GitHub/StackExchange). All prompts are used in an unlabeled manner: we only require forward statistics from the prefill stage and a short continuation loss for reward computation.

Risk signals and stratified coverage. For each prompt $x_{1:T}$, we compute the observation-window statistics on $\mathcal{W} = \{T - w_{\text{obs}} + 1, \ldots, T\}$, including the attention entropy $\mathcal{R}_{\text{struct}}$ and local perplexity $\mathcal{R}_{\text{sem}}$, and discretize them into bin indices (b_h, b_p). To ensure sufficient coverage of high-risk prompts (e.g., high-entropy or high-perplexity regimes), we adopt stratified sampling over (b_h, b_p) when constructing $\mathcal{D}_{\text{cal}}$, avoiding a dominance of medium-risk samples that would otherwise bias the learned tables toward overly aggressive pruning.

Why 20×4 bins for $(H(A), \text{PPL})$. We choose N_H=20 entropy bins and N_P=4 perplexity bins, resulting in a 20×4 risk grid for each layer. This design is motivated by a practical bias–variance trade-off for discrete table policies: (i) attention entropy is a *structural* signal with a relatively smooth and wide dynamic range over long-context prompts, hence we allocate finer granularity (20 bins) to capture gradual shifts in concentration vs. dispersion; (ii) perplexity acts as a *semantic uncertainty* signal whose main role is to modulate the overall conservativeness, and empirically exhibits heavier tails and higher estimation noise on short windows, thus we use coarser partitioning (4 bins) to maintain stability. With L=32 layers, the LUT contains $32 \times 20 \times 4$=2560 entries, which is small enough for robust offline estimation and deployment, while still expressive enough to differentiate risk regimes without introducing over-fragmentation. In our calibration, this binning yields adequate per-cell support under stratified sampling and avoids sparse high-risk cells that would cause unstable threshold decisions.

Compilation hyperparameters. Unless otherwise stated, the observation window is $w_{\text{obs}} = 64$ query positions and the held-out continuation window for reward evaluation is 128 tokens. We train the horizon-1 CQL tables with AdamW, learning rate 3×10^{-4}, batch size 4096 state-action records, 10 epochs over the calibration set, conservative weight $\alpha = 0.75$, threshold grid $\tau \in [0.8, 1.0]$ with 21 uniformly spaced actions, and head-weight grid $\mathcal{W} = [0.8, 1.5]$ with 29 uniformly spaced actions. The risk grid uses 20×4 entropy–PPL bins, and the budget penalty is $\lambda = \beta_{\text{bud}}/T$ with $\beta_{\text{bud}} = 1.0$ for the gate and $\lambda = 0$ for the head table.

Model choice and cross-architecture applicability. We compile tables on a 32-layer backbone to obtain a unified layer index space. For models with different depths, we apply a monotonic depth mapping (e.g., relative-depth interpolation) at inference time. While our main results use per-model compilation, §C.4 reports direct source-to-target table transfer across both closely related LLaMA variants and broader 7B–8B backbone pairs. More distant architecture families and larger-scale transfer remain future work.

Hyperparameter ranges and robustness. Both compiled tables operate as *bounded multiplicative modulators* on top of a normalized utility signal, which allows us to use conservative and architecture-stable ranges. For Stage 3, the LUT outputs a layer threshold $\tau^{(l)} \in [0.8, 1.0]$. This range is chosen for two reasons. First, $\hat{u}_t^{(l)}$ is constructed from $u_t = \alpha_t \cdot \rho_t$, where $\alpha_t = (T/W) \sum_{j \in \Omega} \bar{A}_{j,t}$ has prompt-level mean scale near one and ρ_t is row-normalized by the mean value norm. Thus $\hat{u}_t^{(l)}$ concentrates around $\mathcal{O}(1)$ across prompts and layers, so thresholds outside $[0.8, 1.0]$ either admit almost all tokens or reject nearly all tokens, collapsing the "threshold+Top-B_l" mechanism. Second,

under tight budgets the final retention is always enforced by Top-B_l ; hence $\tau^{(l)}$ mainly controls the *candidate set margin* rather than the final budget. Bounding $\tau^{(l)}$ to $[0.8, 1.0]$ keeps the candidate size within a stable, budget-aligned regime while still allowing risk-conditioned shifts across (b_h, b_p).

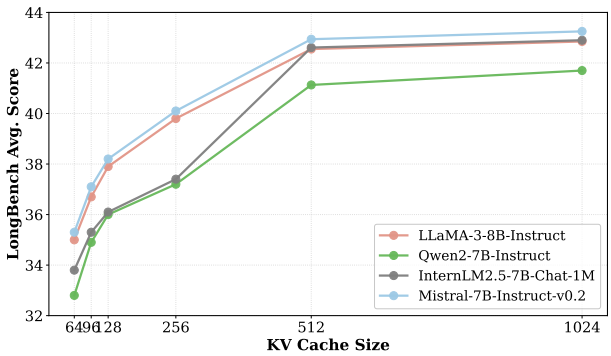


Figure A8: **Universality across model architectures.** Average performance trends on four different LLMs (LLaMA-3, Qwen2, InternLM2.5, and Mistral) under varying KV budgets. The consistent degradation patterns support that CompilerKV's risk-adaptive mechanism generalizes across architectures.

Universality across Model Architectures. Figure A8 demonstrates our method's consistent performance trajectory across four distinct LLMs (including LLaMA-3, Qwen2, and Mistral) as the budget tightens from 1024 to 64 tokens. This architectural universality confirms that our Risk-Adaptive Gating and Head Reliability mechanisms capture fundamental attention properties rather than overfitting to specific model weights.

C.8 Visualization of Learned Policies

To understand how CompilerKV adapts to different risk levels, we visualize the learned threshold table $\mathbf{T}_{\text{gate}}$ in Figure A9. The visualization confirms that the offline reinforcement learning process converges to an interpretable strategy: As shown in the difference between the "Low Perplexity" and "High Perplexity" subplots, the policy globally lowers the retention threshold (blue zones) for high-perplexity samples, allowing more tokens to be stored when the model is uncertain. Within each plot, as Attention Entropy increases (x-axis), the threshold decreases. This indicates the model automatically acts conservatively for diffuse attention patterns (e.g., dense retrieval), preventing the pruning of scattered but critical information.

For head reliability, we use two consistent pieces of evidence: (i) the aggregated reliability distribution in Figure 1(b), which tests whether compiled weights separate noisy and retrieval-critical populations, and (ii) the cross-backbone stability table above, which tests whether the ranking transfers across corpora and architectures. We therefore do not rely on a separate layer–head heatmap as statistical evidence for bimodality; spatial heatmaps answer a different question—where reliable heads occur—and can look continuous even when the pooled reliability distribution is mixture-like.

D Algorithm

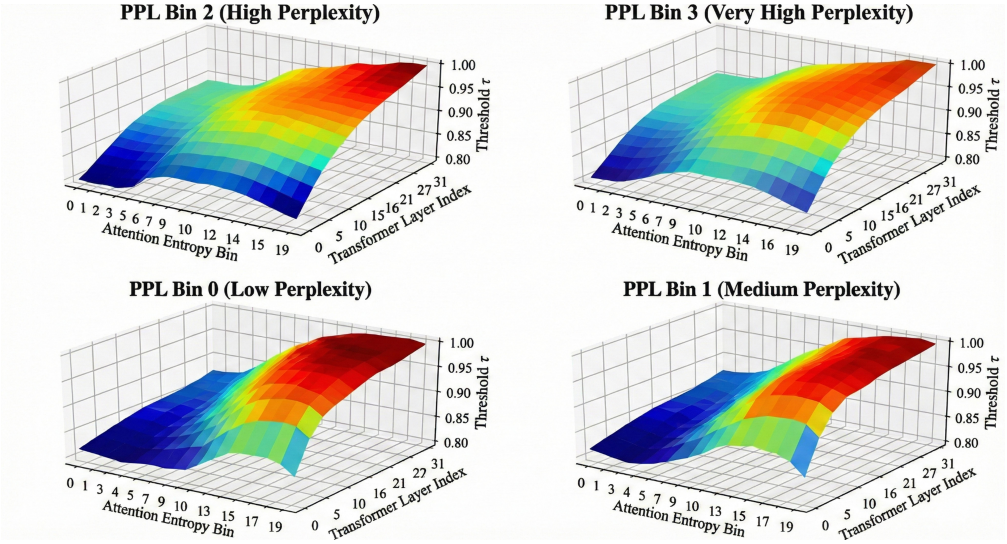


Figure A9: **Risk-Adaptive Threshold Gating Policy.** The plots show the learned retention threshold (τ) across layers for samples with varying risk levels (binned by Perplexity and Attention Entropy). For high-risk samples (High Perplexity, rightmost plots), the policy automatically lowers thresholds to preserve more information, validating our risk-adaptive mechanism.

Table A8: **LongBench results at the extreme 128-token budget.** CompilerKV remains the strongest compressed method on all four backbones, but the severe information bottleneck keeps it below the FullKV reference, as expected. Bold marks the best compressed average; FullKV is shown only as the lossless reference.

Model	Method	*Single-Document QA*			*Multi-Document QA*			*Summarization*			*Few-shot Learning*			*Synthetic*		*Code*		*Avg.*
		NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PRe	Lcc	RB-P	
		18409	3619	4559	9151	4887	11214	8734	10614	2113	5177	8209	6258	11141	9289	1235	4206	–
LLaMA-3-8B -Instruct	FullKV	25.16	31.81	39.59	43.09	36.15	21.77	28.62	23.34	26.33	75.00	90.50	42.36	5.20	69.25	59.04	53.93	41.95
	StreamingLLM	17.85	9.50	23.09	37.84	29.02	16.77	17.91	20.42	20.16	44.00	73.00	30.00	5.80	69.50	48.38	49.31	32.03
	H2O	21.58	12.54	28.49	37.13	32.36	18.88	20.23	22.16	21.14	39.00	86.62	39.19	5.50	69.50	57.39	54.46	35.39
	SnapKV	21.71	12.37	32.38	37.44	30.48	19.50	19.06	21.36	20.07	45.5	87.74	38.15	5.50	68.85	57.42	54.61	35.76
	PyramidKV	22.26	16.65	30.73	38.97	29.28	19.19	19.92	22.06	20.87	68.00	88.95	38.23	5.92	69.50	57.20	51.54	37.45
	DynamicKV	22.10	14.93	32.94	41.06	27.98	21.18	20.03	22.06	21.28	65.50	89.61	38.70	5.13	69.50	58.01	54.00	37.75
	CompilerKV	24.46	28.22	36.34	42.37	30.48	21.25	27.43	22.16	22.35	72.5	89.68	41.00	6.50	69.25	58.91	53.94	**40.43**
Mistral-7B -Instruct-v0.2	FullKV	26.63	32.99	49.34	42.77	27.35	18.77	32.87	24.24	27.10	71.00	86.23	42.96	2.75	86.98	56.93	54.49	42.71
	StreamingLLM	16.58	14.76	30.36	28.13	21.76	11.98	18.26	19.02	19.16	43.50	74.12	28.50	2.50	31.81	43.65	41.19	27.83
	H2O	21.66	21.64	38.60	30.96	20.63	13.02	20.65	22.61	22.08	39.00	82.19	39.75	3.16	79.98	51.25	48.20	34.71
	SnapKV	20.11	21.28	42.98	37.51	22.31	14.43	19.19	21.89	21.01	48.00	83.77	40.44	2.51	66.99	51.64	48.57	35.16
	PyramidKV	22.11	22.52	43.04	33.57	22.98	15.69	20.56	22.52	21.36	65.50	83.84	40.03	2.89	67.26	51.51	46.42	36.36
	DynamicKV	22.05	23.65	43.08	36.03	22.60	15.23	21.35	23.11	22.19	68.00	84.79	41.02	4.20	70.11	52.45	47.41	37.33
	CompilerKV	25.31	29.06	46.20	41.42	25.91	17.47	28.55	23.21	26.35	69.20	85.00	41.20	4.25	80.30	54.30	51.60	**40.58**
Qwen2-7B -Instruct	FullKV	25.14	42.35	45.04	14.80	14.13	9.23	36.35	23.79	26.51	76.50	89.16	45.23	6.50	75.50	60.30	60.78	40.71
	StreamingLLM	19.25	23.63	26.51	14.00	15.30	7.46	18.07	19.30	18.30	47.00	77.92	31.57	6.50	17.00	42.52	41.94	26.64
	H2O	20.33	30.43	34.22	13.61	13.37	7.81	20.72	21.66	18.44	40.00	86.94	42.17	7.00	70.50	53.45	53.76	33.40
	SnapKV	22.26	31.62	38.95	16.05	17.71	7.66	18.91	21.41	18.21	46.00	87.61	42.01	6.50	63.50	54.87	53.03	34.14
	PyramidKV	20.50	31.70	39.95	18.54	18.54	8.85	19.24	20.47	18.18	60.00	87.98	39.71	7.00	49.00	48.77	47.91	33.52
	DynamicKV	22.77	35.57	42.62	14.80	16.35	8.31	21.41	21.97	19.56	58.00	88.18	40.93	6.50	70.00	53.58	52.50	35.82
	CompilerKV	24.12	38.93	43.81	14.85	16.45	8.40	34.62	22.47	23.74	70.00	88.38	44.05	6.50	72.50	56.44	57.79	**38.94**
InternLM-2.5 -7B-Chat-1M	FullKV	22.42	27.61	39.98	40.92	33.48	26.68	33.01	25.18	26.28	72.50	86.76	39.76	2.91	100.00	55.86	57.95	43.21
	StreamingLLM	17.91	13.02	24.31	24.27	16.01	11.29	17.29	20.62	18.06	48.5	67.53	21.93	0.82	87.39	43.45	42.79	29.70
	H2O	16.16	17.71	27.94	26.83	17.83	17.81	13.99	22.59	16.9	39.50	81.87	32.15	1.32	96.50	48.30	47.27	32.79
	SnapKV	19.65	17.44	35.29	27.36	18.58	19.79	12.76	22.42	16.31	48.00	80.23	31.35	0.95	95.00	49.47	48.22	33.93
	PyramidKV	18.80	17.35	33.48	31.16	20.05	19.02	14.65	22.02	17.40	69.50	80.87	32.02	1.23	95.00	47.13	44.73	35.28
	DynamicKV	17.93	19.89	34.15	31.50	19.03	20.60	15.14	22.41	18.15	70.00	83.09	32.44	0.86	95.50	49.33	47.16	36.07
	CompilerKV	21.36	25.32	36.61	37.17	29.75	24.95	25.5	22.62	18.46	70.50	83.43	36.76	1.92	98.00	54.28	55.42	**40.13**

Algorithm 1 COMPILERKV: Prefill-only KV Compression (operator form)

Input: prompt $x_{1:T}$; layers $l \in [1, L]$; heads $h \in [1, H]$; per-layer budgets $\{B_l\}$; observation window $\Omega = \{T - w_{\text{obs}} + 1, \ldots, T\}$; risk LUT $\mathbf{T}_{\text{gate}}$; head table $\mathbf{W}_{\text{head}}$

Output: compressed KV cache $\{(\tilde{K}^{(l)}, \tilde{V}^{(l)})\}_{l=1}^{L}$

(A) Prefill-time risk coordinates (streaming reductions)

$\{A^{(l,h)}, K^{(l,h)}, V^{(l,h)}\} \leftarrow \textsc{Prefill}(x_{1:T})$ $\quad A^{(l,h)} \in \mathbb{R}^{T \times T},\ K^{(l,h)}, V^{(l,h)} \in \mathbb{R}^{T \times d}$

$\bar{A} \leftarrow \textsc{Mean}_{l,h}\big(A^{(l,h)}\big)$ (*layer/head mean*)

$\alpha \leftarrow (T/|\Omega|) \sum_{j \in \Omega} \bar{A}_{j,:} \in \mathbb{R}^T$ (*scale-normalized window mass*)

$\bar{A}' \leftarrow \alpha / \|\alpha\|_1$ (*normalize to a distribution*)

$\mathcal{R}_{\text{struct}} \leftarrow \mathsf{H}(\bar{A}') \ = \ -\sum_{t=1}^{T} \bar{A}'_t \log \bar{A}'_t$

$\mathcal{R}_{\text{sem}} \leftarrow \exp\Big(-\frac{1}{|\Omega|} \sum_{j \in \Omega} \log P(x_j \mid x_{<j}) \Big)$

$(b_{\text{ent}}, b_{\text{ppl}}) \leftarrow \textsc{Discretize}(\mathcal{R}_{\text{struct}}, \mathcal{R}_{\text{sem}})$

(B) Stage 1: Stabilized token utility (vectorized)

for $l = 1$ **to** L **do**

  for $h = 1$ **to** H **do**

    $r^{(l,h)} \leftarrow \|V^{(l,h)}\|_{2,\text{row}} \in \mathbb{R}^T$ (*row-wise ℓ_2 norm*)

    $\mu^{(l,h)} \leftarrow \textsc{Mean}_t\big(r^{(l,h)}\big)$

    $\rho^{(l,h)} \leftarrow r^{(l,h)} / (\mu^{(l,h)} + \epsilon) \in \mathbb{R}^T$

    $u^{(l,h)} \leftarrow \alpha \odot \rho^{(l,h)} \in \mathbb{R}^T$ (*elementwise*)

  end for

end for

(C) Stage 2: Head-aware importance injection (weighted max-pool)

for $l = 1$ **to** L **do**

  $\hat{u}^{(l)} \leftarrow \textsc{MaxPool}_h\Big(u^{(l,h)} \odot \mathbf{W}_{\text{head}}[l, h]\Big) \in \mathbb{R}^T$ (*tokenwise over heads*)

end for

(D) Stage 3: Risk-adaptive gating + budget correction

for $l = 1$ **to** L **do**

  $\tau^{(l)} \leftarrow \mathbf{T}_{\text{gate}}[l, b_{\text{ent}}, b_{\text{ppl}}, B_l]$

  $m^{(l)} \leftarrow \mathbb{I}\big[\hat{u}^{(l)} \geq \tau^{(l)}\big] \in \{0, 1\}^T$ (*candidate mask*)

  $\mathcal{S}^{(l)} \leftarrow \textsc{Select}\big(\hat{u}^{(l)}, m^{(l)}, B_l\big)$

  $(\tilde{K}^{(l)}, \tilde{V}^{(l)}) \leftarrow \textsc{Gather}_{\text{tok}}\big(K^{(l)}, V^{(l)}; \mathcal{S}^{(l)}\big)$

end for

return $\{(\tilde{K}^{(l)}, \tilde{V}^{(l)})\}_{l=1}^{L}$

// SELECT*: if $\|m^{(l)}\|_0 \leq B_l$, keep all candidates; else return Top-B_l indices under score $\hat{u}^{(l)}$.*

// $\textsc{Gather}_{\text{tok}}$ *gathers along token dimension; $\mathcal{S}^{(l)}$ is shared across heads within layer l.*

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and Introduction (§1) state the offline-compilation paradigm, the E1–E3 diagnostic framework, and the quantitative scope of the method. The claims are restricted to prefill-only KV compression and are matched to the main LongBench, RULER/MATH, and efficiency results reported in §5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 explicitly discusses the current limitations: per-model offline compilation cost, untested cross-architecture table sharing, and the fact that streaming and multi-turn cache updates are left to future work. The prefill-only scope restriction is also stated throughout the method and experiments.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical statement in §4.4 specifies the Transformer-layer setting, per-layer budget, CompilerKV selection policy, stabilized utility, and head-reliability table assumptions. The complete proof and intermediate lemmas are provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix B specifies the calibration corpus composition, risk-bin discretization, CQL objective and hyperparameters, action spaces, observation-window size, and the four backbone identifiers. Section 5.1 states the evaluation protocol, token budgets, benchmarks, and baseline comparison setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Due to group IP policy and to avoid premature reuse before review, we do not release code or compiled decision tables with the anonymous submission. All datasets, calibration sources, evaluation benchmarks, and model checkpoints used in the paper are publicly available, and Appendix B provides enough implementation detail to reproduce the method; code and compiled tables will be released upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix C.7 specifies the optimizer, learning rate, batch size, calibration epochs, action-space discretization, risk-bin grid, and observation-window configuration. Section 5.1 specifies the 512-token evaluation budget, context lengths, decoding setup, and A100-80GB hardware configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper provides appropriate stability evidence for deterministic long-context evaluation: results are reported across four backbone families, multiple benchmarks, and component ablations. Since LongBench/RULER use deterministic greedy decoding, we do not report sampling error bars; instead, cross-backbone consistency and ablation margins are used to assess robustness.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the GPU type and memory (NVIDIA A100-80GB), the offline compilation cost (less than 3 A100-hours per model), and the throughput/latency measurement setup. Appendix tables specify batch size, context length, and hardware for efficiency measurements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: This work presents an inference-optimization method for existing LLMs. It does not involve human subjects, sensitive personal data, or new dual-use model capabilities beyond those already present in the evaluated base models, and it conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts including lower memory, latency, and energy cost for long-context inference, which can improve accessibility on resource-constrained hardware. It also notes the negative impact that cheaper inference can reduce the cost of harmful LLM deployment, a general risk of inference-optimization work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release new pretrained models, scraped datasets, or generative assets with elevated misuse risk. CompilerKV produces lightweight decision tables that require an existing base LLM and do not introduce new capabilities beyond the underlying model.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All backbone models, evaluation benchmarks, calibration sources, and baseline methods are cited, and we use them under their respective licenses and terms of use. Appendix B lists the assets used for calibration/evaluation and the corresponding source references.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: We do not release new assets with the anonymous submission because the code and compiled tables are withheld until acceptance under the group release policy. The compiled Head Reliability Table and Risk-Adaptive Threshold LUT contain no personal data and will be released with documentation and usage instructions upon acceptance.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: No crowdsourcing or human-subject experiments were conducted. All experiments are automated evaluations on existing public benchmarks and calibration corpora.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects were involved in this study. Therefore, IRB approval or equivalent human-subject review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are the subject of this study, namely the base models whose KV caches are compressed, rather than a non-standard component used to develop the method. The proposed compression policy does not use LLM-generated annotations or LLM-based decision making beyond evaluating the base models themselves.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.