# SIDEKV: SIDE-INFORMATION-AWARE CONTEXT COMPRESSION FOR KV CACHE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Long-context inference in large language models (LLMs) incurs substantial memory and computation through the key–value (KV) cache. Learned context compression replaces a long historical block with a compact representation, but existing approaches typically construct this representation from the discarded context alone, despite correlated context remaining available to the predictor. We introduce SideKV, a side-information-aware context-compression framework that conditions the rate of a learned summary on the retained context. A lightweight encoder compresses the discarded block into summary vectors, while a conditional rate model exploits the retained context as side information; the pretrained LLM remains frozen. Across 13 LLMs, with 320 of 1312 cache slots retained, conditioning on side information reduces the variational rate to $15\%$–$67\%$ of that of an otherwise identical unconditional model and lowers excess predictive distortion on 11 of the 13 LLMs. The distortion advantage also persists across additional corpora, rate–distortion settings, and correlated time series. Among learned compressed representations, SideKV substantially improves over mean pooling while avoiding adaptation of the pretrained predictor, and achieves a mean excess distortion of $0.050$ bits/token. It requires only $0.25\times$ full-context prefill arithmetic and $0.30\times$ time to first token at the main setting. On LongBench, SideKV also achieves the highest mean score among the learned compressed representations evaluated. These results show that exploiting retained context as side information is a promising direction for lightweight KV-cache compression.

## 1 INTRODUCTION

Large language models are deployed in conversational agents, retrieval-augmented generation and long-term memory in dialogue (Tavakoli et al., 2026), and current models and agentic workflows run over contexts of a million tokens or more (Gemini Team, 2024; Meta AI, 2025; DeepSeek-AI, 2026). During inference the key–value (KV) cache stores the keys and values of every previous token, and its size grows with the context length and the batch. For Llama-3.1-8B at 128K tokens and a batch of four it takes $81\%$ of the model's $85$ GB footprint in FP16 (Bhatnagar et al., 2026), which weight quantization leaves untouched. Existing methods shrink it in three ways. Eviction keeps a subset of the original entries (Xiao et al., 2024; Li et al., 2024; Kim et al., 2025; An et al., 2026), and quantization keeps every entry at a lower precision (Liu et al., 2024; Zandieh et al., 2026). Learned context compression replaces a discarded block with a few summary vectors, at ratios from $2\times$ to $500\times$ (Ge et al., 2024; Zhang et al., 2025; Li et al., 2025).

However, learned compression determines the summary from the discarded block alone. Under common cache policies the model keeps an initial prefix and a recent window, and within a document this retained context is correlated with the block that goes. Ignoring this correlation wastes capacity on information the retained context already carries. Learned compression also fixes the size of its summary in advance, without measuring that redundancy.

To address these issues, we propose SideKV, a side-information-aware context compression framework. SideKV contains an encoder that compresses the discarded block into $k$ Gaussian summary vectors, and a conditional rate model that takes the retained context and gives the prior of these vectors. The rate of the summary is the divergence between the encoder's distribution and this prior. Specifically, content that the retained context already predicts takes few bits under the prior, and the



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

encoder carries the part of the block the retained context lacks. We train the encoder and the rate model against the loss of the LLM on the continuation, and the LLM itself stays frozen.

We evaluate SideKV on 13 LLMs from Qwen, Llama, Gemma, OLMoE, DeepSeek and Mistral, with 320 of 1312 cache slots. SideKV improves the rate–prediction trade-off over an otherwise identical unconditional baseline, and the advantage holds on four additional corpora and two correlated time series. It outperforms eviction on all 13 LLMs and achieves the lowest distortion among eviction and three selection methods on 7 of them. Averaged over the 13 LLMs, its distortion is 33% below that of ReST-KV, one of the selection methods, with $4\times$ less arithmetic and an $18\times$ shorter TTFT on a GPU. It also recovers on average 71% of the improvement that 500xCompressor achieves over eviction. The compressed cache runs up to $36\times$ faster than the full context at a 4096-token block.

Our main contributions are as follows.

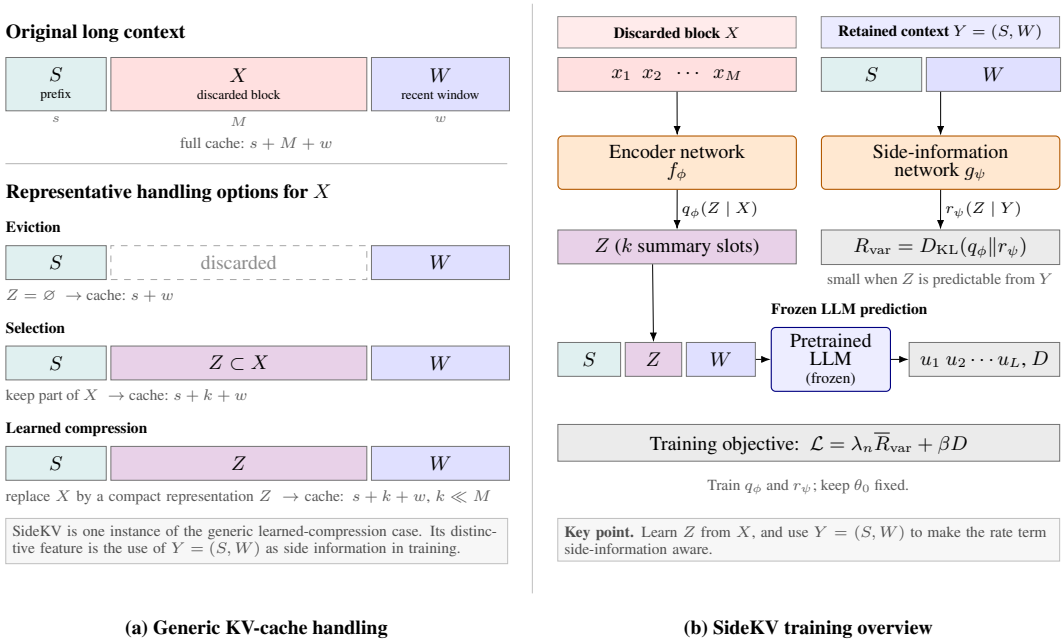
- We propose SideKV, a context compression framework in which a conditional rate model takes the retained context as side information, the summary carries a measured rate, and the pretrained LLM stays frozen.
- We show that the conditional rate model lowers the rate below that of an otherwise identical unconditional prior on all 13 LLMs, and the distortion in all 28 cells of four additional corpora and on two correlated time series.
- We compare SideKV with ten systems on 13 frozen LLMs at 320 slots. SideKV reaches the lowest mean distortion of the systems that keep the LLM frozen without running the LLM over the discarded block. Its prefill takes 0.24 to 0.26 of the arithmetic of the full context against 1.00 to 2.00 for the selection methods, and on a GPU 0.30 of its time to first token against 1.34 to 5.51. No system at 320 slots is both lighter in arithmetic and lower in distortion.

## 2 Related Work

**Eviction and selection.** StreamingLLM keeps an initial prefix and a recent window (Xiao et al., 2024), and H2O drops entries by cumulative attention (Zhang et al., 2023). SnapKV selects entries by the attention of an observation window (Li et al., 2024), and ReST-KV scores each removal by the layer-wise output error it would cause (An et al., 2026). KVzip scores entries by the quality of context reconstruction, and shows that query-aware eviction degrades when a cache is reused (Kim et al., 2025). VECTOR rebuilds a value from its key (Lin et al., 2026), and IndexMem compresses the evicted entries into a latent memory whose readout is added to the attention output as a residual (Yang et al., 2026). SideKV instead encodes the discarded block into a compact summary.

**Quantization.** KIVI quantizes keys per channel and values per token (Liu et al., 2024), and TurboQuant reaches near-optimal distortion for online vector quantization (Zandieh et al., 2026). KVTC adds transform coding and an entropy coder (Staniszewski & Łańcucki, 2026), and Laus et al. (2026) allocate bits by reverse water-filling over whitened features. STAR-KV decomposes the keys head by head and the values jointly, with the rank chosen by soft thresholding (Bhatnagar et al., 2026). RDKV places eviction and quantization at two ends of one bit allocation, and a bit-width that falls to zero drops the entry (Zhang et al., 2026). SideKV replaces the discarded block by $k$ summary vectors and penalizes their variational rate.

**Context compression.** Gist tokens compress a prompt into a few input positions (Mu et al., 2023), the in-context autoencoder writes it into memory slots (Ge et al., 2024), and AutoCompressors pass summary vectors to the input of the next segment (Chevalier et al., 2023). In the last two the model computes the keys and values of those positions from one input vector a slot, and gisting caches the keys and values of the gist positions directly. Feldman & Artzi (2025) compare summaries of this kind, and mean pooling outperforms the causal compression token. Activation Beacon instead condenses the keys and values of every layer, through attention parameters added to the predictor (Zhang et al., 2025), and 500xCompressor hands the predictor the same tensors for $k$ memory tokens (Li et al., 2025). SAC aggregates the context into the keys and values of anchor tokens (Liu et al., 2026). Activation Beacon adds modules inside the predictor and 500xCompressor and SAC train adapters on it, where SideKV leaves the predictor untouched. These methods fix the size of




**(a) Generic KV-cache handling** **(b) SideKV training overview**

Notation: $S$, $X$ and $W$ denote token blocks with lengths $s$, $M$ and $w$. $Z$ is a generic representation replacing $X$.

Figure 1: Generic KV-cache handling and SideKV training. **(a)** The four ways of handling a block under one partition $[S \mid X \mid W]$. **(b)** The encoder network $f_\phi$ and the side-information network $g_\psi$ output the parameters of $q_\phi(Z \mid X)$ and $r_\psi(Z \mid Y)$, whose divergence is the variational rate surrogate.

the summary in advance and place no constraint on its rate. A fixed number of slots bounds the dimension of the summary and not the information it carries.

**Rate and side information.** The variational information bottleneck constrains the rate of a learned representation against an unconditional prior (Alemi et al., 2017). Nagle et al. (2024) study prompt compression as a distortion–rate problem, with log-loss among their distortion measures, and Colaco & Lahjouji (2026) place eviction, quantization, prompt pruning and agent memory under one compaction objective. Coding with side information at the decoder conditions that prior on a correlated source, and goes back to Wyner & Ziv (1976). Learned coders with decoder side information recover binning (Özyılkan et al., 2023), give the conditional-prior rate a one-shot operational meaning (Phan et al., 2024) and scale to images (Whang et al., 2024). Concurrent work codes the cache conditioned on the preceding context losslessly (Magarshak, 2026), or as sequential Wyner–Ziv coding with the next query as decoder side information (Kim, 2026). SideKV constrains the rate of a lossy summary of the KV cache, and conditions that constraint on the retained context.

# 3 Method

## 3.1 Problem formulation and notation

We partition a long context into three consecutive token sequences,

$$[\,S \mid X \mid W\,], \qquad s = |S|, \quad M = |X|, \quad w = |W|, \tag{1}$$

where $S$ is a retained prefix, $X$ is the historical block to be compressed, and $W$ is a retained recent window. For any token sequence $A$, $|A|$ denotes its length. Thus, $s$, $M$, and $w$ denote the lengths of $S$, $X$, and $W$, respectively. We let $Z$ denote the representation that replaces $X$ for the predictor, so that the predictor takes $[\,S \mid Z \mid W\,]$ as input. As shown in Figure 1(a), the full context corresponds to $Z = X$, eviction to $Z = \varnothing$, selection retains a subset $Z \subset X$, and learned compression constructs a compact representation $Z$ from $X$.

A continuation $U = (u_1, \ldots, u_T)$ of $T$ tokens follows the context. The pretrained predictor $p_{\theta_0}$ remains frozen throughout, and predictive distortion is the mean negative log-likelihood of the continuation,

$$D(Z) \;=\; -\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^{T} \log_2 p_{\theta_0}(u_t \mid u_{<t}, S, Z, W)\right]. \tag{2}$$

Thus, $D(X)$ is the distortion under the full context, $D(\varnothing)$ under eviction, and $D(Z)$ under a compressed representation. We define the excess distortion relative to the full context as

$$\Delta D_{XZ} \;:=\; D(Z) - D(X). \tag{3}$$

A positive $\Delta D_{XZ}$ indicates a loss relative to the full context, whereas a negative value indicates a lower predictive loss. Unless stated otherwise, the experimental tables report $\Delta D_{XZ}$ on paired spans. The cache budget of each system is reported as the number of retained slots. Appendix D summarizes the notation.

### 3.2 Side-information-aware learned compression

The block $X$ is discarded while $Y = (S, W)$ remains available to the predictor, and the retained context can make part of $X$ predictable. For example, if $S$ shows a person preparing to sprint and $W$ shows the same person reaching the destination, the missing segment $X$ is largely predictable as the intervening running motion. The representation $Z$ is a compact summary of this missing segment. It therefore need not spend the same rate on information already implied by the retained context $Y$.

As shown in Figure 1(b), SideKV uses two trainable networks. The encoder network $f_\phi$ takes the discarded block $X$ and produces the parameters of the encoder distribution $q_\phi(Z \mid X)$, from which the representation $Z$ is generated. The side-information network $g_\psi$ takes the retained context $Y$ and produces the parameters of the conditional model $r_\psi(Z \mid Y)$, which models how predictable the learned representation is from the retained context. SideKV then uses the discrepancy between these two distributions as a rate penalty: information in $Z$ that is predictable from $Y$ is penalized less, encouraging $Z$ to emphasize information not already explained by the retained context, while the predictive objective asks $Z$ to preserve information useful for the continuation. The predictor remains frozen, and only $\phi$ and $\psi$ are trained.

The encoder network that generates $Z$ observes $X$ and not $Y$, which gives $q_\phi(z \mid x, y) = q_\phi(z \mid x)$ and makes $Z - X - Y$ a Markov chain. The relation does not make $X$ and $Y$ independent, since they are generally correlated parts of one context. It states only that $Y$ is not used to generate $Z$. The retained context enters instead through the conditional model $r_\psi(Z \mid Y)$ of the rate surrogate.

### 3.3 Distribution-agnostic variational rate surrogate

Let $q_\phi(Z \mid X)$ be the distribution the encoder induces and $r_\psi(Z \mid Y)$ a tractable conditional model, neither of them tied to a particular class of distributions. We define the *variational rate surrogate*

$$R_{\mathrm{var}}(\phi, \psi) = \mathbb{E}_{X,Y}\left[D_{\mathrm{KL}}\left(q_\phi(Z \mid X) \,\|\, r_\psi(Z \mid Y)\right)\right]. \tag{4}$$

With logarithms to base two, $R_{\mathrm{var}}$ is in bits over the block, and $\overline{R}_{\mathrm{var}} = R_{\mathrm{var}}/M$ is its value per discarded token. Eq. (4) is a training penalty with an information-theoretic reading, not a conventional bit rate in source coding. A part of the representation that $Y$ already predicts keeps the two distributions close and adds little to the penalty, and a part that $Y$ does not explain adds more.

The data distribution and the encoder induce the joint distribution $q_\phi(x, y, z) = p_{\mathrm{data}}(x, y)\, q_\phi(z \mid x)$, and $I_q$ denotes mutual information under it. Its conditional latent distribution is

$$q_\phi(z \mid y) = \int p_{\mathrm{data}}(x \mid y)\, q_\phi(z \mid x)\, \mathrm{d}x. \tag{5}$$

A derivation in Appendix D shows that

$$R_{\mathrm{var}}(\phi, \psi) = I_q(X; Z \mid Y) + \mathbb{E}_Y\left[D_{\mathrm{KL}}\left(q_\phi(Z \mid Y) \,\|\, r_\psi(Z \mid Y)\right)\right], \tag{6}$$

and so

$$R_{\mathrm{var}}(\phi, \psi) \geq I_q(X; Z \mid Y). \tag{7}$$

For a fixed encoder, equality holds when the conditional model exactly matches the encoder-induced conditional distribution,

$$r_\psi(z \mid y) = q_\phi(z \mid y). \tag{8}$$

Note that in Eq. (6), the first term measures the information the summary preserves about the block beyond the retained context. The second term measures the mismatch of the conditional model. A drop in the trained surrogate can come from either term.

By the same derivation, an unconditional model $r_\psi(Z)$ gives

$$R_{\text{var}}^{\text{uncond}} = I_q(X;Z) + D_{\text{KL}}(q_\phi(Z) \| r_\psi(Z)). \tag{9}$$

Under $Z - X - Y$ the chain rule gives $I_q(X;Z \mid Y) = I_q(X;Z) - I_q(Z;Y) \le I_q(X;Z)$. The floor of the conditional surrogate sits below the unconditional one by the information the summary shares with the retained context. A trained pair of networks also includes the mismatch term of Eq. (6), and Appendix D derives both statements.

### 3.4 Gaussian implementation and the frozen predictor

The formulation does not restrict the choice of statistical distribution. For practical implementation, we use factorized diagonal Gaussians, which allow reparameterized training and a closed-form KL divergence. For each summary slot $j$,

$$q_\phi(z_j \mid X) = \mathcal{N}\big(z_j; \mu_{\phi,j}(X), \operatorname{diag} \sigma^2_{\phi,j}(X)\big),\ r_\psi(z_j \mid Y) = \mathcal{N}\big(z_j; \tilde{\mu}_{\psi,j}(Y), \operatorname{diag} \tilde{\sigma}^2_{\psi,j}(Y)\big), \tag{10}$$

where $\mu_{\phi,j}, \tilde{\mu}_{\psi,j} \in \mathbb{R}^{d_z}$ are means and $\sigma_{\phi,j}, \tilde{\sigma}_{\psi,j} \in \mathbb{R}^{d_z}_+$ elementwise standard deviations. Both networks output a log-variance, from which the standard deviation is taken, and the sample is $z_j = \mu_{\phi,j}(X) + \sigma_{\phi,j}(X) \odot \epsilon_j$ with $\epsilon_j \sim \mathcal{N}(0, I)$. A latent vector is no cache entry, and each $z_j$ becomes one input-layer token,

$$e_j \ = \ \bar{c}\, \frac{B_\phi z_j}{\|B_\phi z_j\|_2}, \qquad \bar{c} = \mathbb{E}_v \|\text{emb}_{\theta_0}(v)\|_2, \tag{11}$$

with $B_\phi \in \mathbb{R}^{d \times d_z}$ and $\bar{c}$ the mean norm of the predictor's token embeddings, and the frozen predictor takes $[\,S \mid e_{1:k} \mid W\,]$ through its own layers. The hand-over is $d_z$ numbers a slot at one layer, against the keys and values of every layer that a cache-level summary passes over. The SideKV encoder processes $X$ once to construct the summary, after which the frozen predictor processes only $[\,S \mid e_{1:k} \mid W\,]$. Summary construction is outside the autoregressive decoding loop and runs once per compressed block. A more expressive conditional model could provide a tighter approximation to $q_\phi(Z \mid Y)$ and reduce the variational mismatch of Eq (6), and Appendix D.6 gives the divergence in closed form and Appendix B the architecture.

### 3.5 Training objective

The encoder is trained to balance predictive quality against the variational rate surrogate. Let

$$D_\phi = -\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \log_2 p_{\theta_0}(u_t \mid u_{<t}, S, Z, W)\right] \tag{12}$$

denote the predictive distortion induced by the stochastic representation $Z \sim q_\phi(Z \mid X)$. The training objective is

$$\mathcal{L}(\phi,\psi) = \lambda_n\, \overline{R}_{\text{var}}(\phi,\psi) + \beta\, D_\phi, \qquad \theta_0 \text{ fixed}, \tag{13}$$

with $\overline{R}_{\text{var}} = R_{\text{var}}/M$ the surrogate per discarded token, $\beta = T/M$, a scheduled coefficient $\lambda_n = \min(1,\, n/500)$ at training step $n$, and one sample of $\epsilon$ per block for (12). Multiplied by $M$, the objective at $\lambda_n = 1$ is the block surrogate plus the summed continuation log-loss, both computed in nats by the implementation. We also train at a quarter and four times that weight.

Only the SideKV parameters $\phi$ and $\psi$ are updated. The encoder receives gradients through both the predictive-distortion term and the variational-rate surrogate, whereas the side-information network is trained through the surrogate-rate term. The pretrained language model remains frozen. The

unconditional variant of (9) uses the same encoder with a single learned Gaussian per slot in place of $r_\psi(Z \mid Y)$, and the two arms differ in the rate model alone.

We report $R_{\text{var}}$ in bits over the block and the distortion of Equation 2 in bits per token of $U$; Appendix D gives the conversion.

# 4 Experiments

## 4.1 Setup

**Models and corpora.** We use 13 public pretrained LLMs from 0.5B to 16B parameters from the Qwen2.5, Qwen3, Llama-3.1, Llama-3.2, Gemma-3, OLMoE, DeepSeek-V2 and Mistral series, all kept frozen, covering dense, mixture-of-experts, latent-attention and interleaved local-global architectures. The main corpus is SlimPajama (Soboleva et al., 2023), and seven of the LLMs are also evaluated on WikiText (Merity et al., 2017), PG19 (Rae et al., 2020), five public-domain novels and Python source.

**Baselines.** Every baseline runs on the same frozen LLM and spans (Appendix A). Eviction drops the block and keeps the prefix and the window, 288 slots, as StreamingLLM (Xiao et al., 2024) does. Selection keeps 320 original entries by SnapKV (Li et al., 2024), KVzip (Kim et al., 2025) or ReST-KV (An et al., 2026). Quantization keeps every entry of the block at four bits by KIVI (Liu et al., 2024) or TurboQuant (Zandieh et al., 2026), in the memory of 544 and 572 slots. Summary vectors replace the block by mean pooling, the operator of Feldman & Artzi (2025) applied without training, or by 500xCompressor (Li et al., 2025), SAC (Liu et al., 2026) or Activation Beacon (Zhang et al., 2025), the last two on the LLM each was released for.

## 4.2 Effect of side information

We first ask whether explicitly exploiting the retained context as side information improves the rate–prediction trade-off over an otherwise identical unconditional model. SideKV and the unconditional-prior baseline share the encoder, latent dimension, frozen LLM, data, and optimisation procedure, and differ only in the rate model, $r_\psi(Z \mid Y)$ versus $r_\psi(Z)$. The comparison therefore isolates the effect of exploiting the retained context $Y$ from the other components of the design. Let $R_{\text{u}}$ and $R_{\text{SI}}$ denote the variational rates of the unconditional and side-information-aware models, respectively, and define

$$\rho_R = R_{\text{SI}}/R_{\text{u}} \times 100\%, \quad G_D = \Delta D^{\text{u}}_{XZ} - \Delta D^{\text{SI}}_{XZ}. \tag{14}$$

Thus, smaller $\rho_R$ and positive $G_D$ indicate an advantage from side information.

Conditioning on $Y$ lowers the variational rate on all 13 LLMs (Table 1). The retained rate is only 15%–67% of that under the unconditional prior, showing that the retained context provides substantial predictive information about the compressed representation. Side information also reduces $\Delta D_{XZ}$ on 11 of the 13 LLMs. Appendix E further shows that the distortion advantage persists on 4 additional corpora, across 3 rate–distortion weights, and on 2 correlated time series.

## 4.3 Comparison with learned compressed representations

We next compare SideKV with methods that replace the discarded context by a learned compact representation. These methods fall into two different categories. MeanPool and SideKV keep the pretrained predictor frozen, whereas 500xCompressor, SAC, and Activation Beacon train adapters or additional components inside the predictor. The distinction is important because predictor adaptation provides additional modelling capacity and changes the training cost and deployment requirements.

Table 2 therefore reports the two categories separately. Across the 13 LLMs for which a common comparison is available, SideKV substantially reduces $\Delta D_{XZ}$ relative to MeanPool while keeping the predictor frozen. 500xCompressor achieves lower excess distortion on average, but under a predictor-adapted setting; notably, the downstream LongBench results in Section 4.5 show that SideKV attains a higher mean task score.

On the predictor-specific releases, SideKV remains competitive with predictor-adapted learned compressors: it outperforms SAC at both reported ratios and layouts on Llama-3.2-1B, while Activation

Table 1: Effect of side information on rate and prediction loss on SlimPajama. $R_{\mathrm{u}}$ and $R_{\mathrm{SI}}$ denote the variational rates of the unconditional and side-information-aware models, respectively, and $\rho_R \triangleq 100R_{\mathrm{SI}}/R_{\mathrm{u}}\%$ is the percentage of the unconditional rate retained with side information. $\Delta D^{\mathrm{u}}_{XZ}$ and $\Delta D^{\mathrm{SI}}_{XZ}$ denote the excess predictive distortions of the unconditional and side-information-aware models, respectively, and $G_D \triangleq \Delta D^{\mathrm{u}}_{XZ} - \Delta D^{\mathrm{SI}}_{XZ}$ is the reduction in excess distortion. Smaller $\rho_R$ and larger $G_D$ indicate greater benefit from side information.

| | Variational rate | | | Excess distortion (bits/token) | | |
|---|---|---|---|---|---|---|
| Predictor | $R_{\mathrm{u}}$ | $R_{\mathrm{SI}}$ | $\rho_R \downarrow$ | $\Delta D^{\mathrm{u}}_{XZ}$ | $\Delta D^{\mathrm{SI}}_{XZ}$ | $G_D \uparrow$ |
| Qwen2.5-0.5B | 0.21 | 0.10 | 48% | +0.125 | +0.109 | +0.016 |
| Qwen2.5-7B | 1.25 | 0.19 | 15% | +0.028 | −0.078 | +0.106 |
| Qwen2.5-14B | 0.22 | 0.14 | 64% | +0.119 | +0.097 | +0.022 |
| Qwen3-8B | 0.98 | 0.19 | 19% | +0.068 | −0.045 | +0.113 |
| Llama-3.2-1B | 0.36 | 0.12 | 33% | +0.068 | +0.042 | +0.026 |
| Llama-3.2-3B | 0.50 | 0.11 | 22% | +0.068 | +0.032 | +0.035 |
| Llama-3.1-8B | 0.66 | 0.26 | 39% | +0.097 | +0.068 | +0.029 |
| Gemma-3-1B | 3.73 | 0.94 | 25% | +0.209 | −0.031 | +0.240 |
| Gemma-3-4B | 1.50 | 1.01 | 67% | +0.106 | +0.111 | −0.005 |
| Gemma-3-12B | 2.09 | 0.95 | 45% | +0.078 | +0.145 | −0.068 |
| OLMoE-1B-7B | 0.32 | 0.09 | 28% | +0.131 | +0.117 | +0.014 |
| DeepSeek-V2-Lite | 0.40 | 0.13 | 33% | +0.004 | −0.038 | +0.042 |
| Mistral-7B | 0.26 | 0.07 | 27% | +0.128 | +0.118 | +0.009 |

Table 2: Comparison with learned context-compression methods. Excess predictive distortion $\Delta D_{XZ} \downarrow$ is reported separately for methods that keep the pretrained predictor frozen and methods that train components inside the predictor. Panel (a) gives the common comparison across 13 LLMs; panels (b) and (c) report predictor-adapted methods on the LLMs supported by their released weights. Avg. is over the 13 LLMs in panel (a).

**(a) Common comparison across 13 LLMs**

| | Qwen2.5 | | | Qwen3 | Llama-3.2 | | Llama-3.1 | Gemma-3 | | | OLMoE | DeepSeek | Mistral | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | 0.5B | 7B | 14B | 8B | 1B | 3B | 8B | 1B | 4B | 12B | 1B-7B | V2-Lite | 7B | Avg. |
| *Frozen predictor* | | | | | | | | | | | | | | |
| MeanPool | 0.136 | 2.274 | 0.167 | 0.116 | 0.157 | 0.140 | 0.113 | 0.553 | 0.833 | 2.721 | 0.132 | 0.087 | 0.125 | 0.581 |
| **SideKV** | **0.109** | **−0.078** | **0.097** | **−0.045** | **0.042** | **0.032** | **0.068** | **−0.031** | **0.111** | **0.145** | **0.117** | **−0.038** | **0.118** | **0.050** |
| *Training inside predictor* | | | | | | | | | | | | | | |
| 500xC. | 0.097 | −0.144 | 0.084 | −0.107 | −0.003 | −0.018 | 0.065 | −0.190 | −0.131 | −0.114 | 0.113 | −0.095 | 0.116 | −0.025 |

**(b) Predictor-adapted SAC release: Llama-3.2-1B**

| Method | Cache | $\Delta D_{XZ} \downarrow$ |
|---|---|---|
| eviction | 288 | +0.123 |
| SAC 5×, its layout | 262 | +0.226 |
| SAC 15×, its layout | 87 | +0.285 |
| SAC 5×, our layout | 493 | +0.125 |
| SAC 15×, our layout | 357 | +0.140 |
| **SideKV** | 320 | **+0.102** |

**(c) Predictor-adapted Activation Beacon release: Qwen2-7B-Instruct**

| Method | Cache | $\Delta D_{XZ} \downarrow$ |
|---|---|---|
| eviction | 288 | +0.114 |
| Activation Beacon | 320 | **+0.056** |
| **SideKV** | 320 | +0.107 |

Beacon achieves a lower $\Delta D_{XZ}$ on Qwen2-7B-Instruct (0.056 versus 0.107). The training and implementation costs are reported in the next subsection.

Appendices F and G provide the full 13-LLM comparison and further comparisons with predictor-adapted compressors, respectively.

### 4.4 Implementation Complexity and Efficiency

This subsection compares the computational and training costs. Unlike Section 4.3, which focuses on learned representations, this comparison includes eviction, selection, quantization, and learned-compression methods as these approaches reach their compressed representations through substantially different computation paths. Table 3 reports prefill arithmetic, time to first token (TTFT), trainable parameters, cache budget, and mean excess predictive distortion $\Delta D_{XZ}$.

Selection methods such as SnapKV, KVzip, and ReST-KV first run the LLM over the full prompt before selecting cache entries, and therefore retain the full-context prefill cost; KVzip additionally requires a second pass. Quantization likewise processes the full prompt and reduces the precision

Table 3: Arithmetic and TTFT are relative to full-context processing on Qwen2.5-7B; trainable parameters are reported for Qwen2.5-14B where applicable. † denotes evaluation on the predictor supported by the released weights. Mean $\Delta D_{XZ}$ is reported for context.

| System | Full LLM Prefill | Trained params | Cache slots | Arith. ↓ | TTFT ↓ | Mean $\Delta D_{XZ}$ ↓ |
|---|---|---|---|---|---|---|
| *full context* | ✓ | 0 | 1312 | 1.00 | 1.00 | 0 |
| *Eviction and selection* | | | | | | |
| eviction | × | 0 | 288 | 0.22 | 0.27 | 0.388 |
| SnapKV | ✓ | 0 | 320 | 1.00 | 1.34 | 0.083 |
| KVzip | ✓ | 0 | 320 | 2.00 | 2.49 | 0.143 |
| ReST-KV | ✓ | 0 | 320 | 1.00 | 5.51 | 0.075 |
| *Quantization* | | | | | | |
| KIVI | ✓ | 0 | 1312 (4-bit) | 1.00 | 1.04 | 0.002 |
| TurboQuant | ✓ | 0 | 1312 (4-bit) | 1.00 | 2.41 | 0.006 |
| *Trained inside the LLM* | | | | | | |
| 500xCompressor | ✓ | 201.5M (1.36%) | 320 | 1.00 | 1.17 | −0.025 |
| SAC† | ✓ | 13.8M (1.12%) | 493 | 1.00 | 3.05 | 0.125 |
| Activation Beacon† | ✓ | 462.6M (6.07%) | 320 | 1.02 | 1.67 | 0.056 |
| *Summary vectors, LLM frozen* | | | | | | |
| MeanPool | × | 0 | 320 | 0.24 | 0.29 | 0.581 |
| **SideKV** (ours) | × | 18.3M (0.12%) | 320 | 0.25 | 0.30 | 0.050 |

Table 4: Downstream performance on LongBench. Average score over sixteen tasks with fifty samples per task, using a 1024-token discarded block and a 256-token retained window; higher is better. "uncond." denotes SideKV with an unconditional prior, and the final column reports the mean over the evaluated predictors. Bold marks the best score within each method category.

| | Qwen2.5 | | | Qwen3 | Llama-3.1 | Gemma-3 | | Mistral | |
|---|---|---|---|---|---|---|---|---|---|
| System | 0.5B | 7B | 14B | 8B | 8B | 4B | 12B | 7B | Mean |
| *full context* | 20.1 | 34.9 | 37.6 | 8.1 | 34.6 | 40.2 | 44.8 | 27.2 | 30.9 |
| *Trained inside the LLM* | | | | | | | | | |
| 500xC. | 10.3 | 16.6 | 15.7 | 8.1 | 14.5 | 12.8 | 17.8 | 22.0 | 14.7 |
| *Summary vectors, LLM frozen* | | | | | | | | | |
| MeanPool | **15.7** | 2.8 | 14.8 | 7.0 | 16.5 | 24.4 | 23.6 | 21.4 | 15.8 |
| uncond. | 15.6 | 14.8 | 15.8 | **7.1** | 16.9 | 22.8 | 23.7 | 21.5 | 17.3 |
| **SideKV** | 14.8 | **17.8** | **16.2** | **7.1** | **22.9** | **24.7** | **25.1** | **22.0** | **18.8** |

rather than the number of KV entries. Predictor-adapted learned compressors, including 500xCompressor, SAC, and Activation Beacon, also require full-LLM processing and introduce additional trainable parameters inside the predictor. In contrast, MeanPool and SideKV construct compact input-level summaries without running the frozen LLM over the discarded block.

At the main setting, SideKV requires $0.25\times$ the full-context prefill arithmetic and $0.30\times$ the TTFT while using 320 cache slots. Its computational cost is therefore close to MeanPool, but with substantially lower predictive distortion. Compared with predictor-adapted learned compressors, SideKV also uses fewer trainable parameters while keeping the pretrained LLM unchanged. These comparisons highlight the potential of SideKV as a lightweight, low-complexity approach to KV-cache compression, combining reduced prefill cost with competitive predictive performance.

Appendix H further reports measured prefill throughput across batch sizes and continuation-scoring latency and peak memory across all 13 LLMs and longer block lengths.

### 4.5 Downstream Performance and Robustness

We further evaluate whether the compressed representations remain useful for downstream long-context tasks using LongBench (Bai et al., 2024). Table 4 reports the average over 16 tasks on 8 LLMs at the main compression setting. Among the learned representations shown in Table 4, SideKV achieves the highest mean score, 18.8, compared with 15.8 for MeanPool and 14.7 for 500xCompressor. The unconditional variant reaches 17.3, indicating that the benefit of side information also carries beyond continuation log-loss to downstream task performance. Interestingly,

500xCompressor achieves a lower predictive distortion in Section 4.3, but obtains a lower mean LongBench score of 14.7. This indicates that continuation log-loss and downstream task accuracy capture different aspects of the information preserved by a compressed representation. Full-context inference remains substantially stronger, with a mean score of 30.9, showing that aggressive compression still loses task-relevant information. Overall, the LongBench results suggest that SideKV preserves useful long-context information while keeping the pretrained predictor frozen.

Appendix I broadens the LongBench comparison to eviction, selection, and quantization methods. Appendix J further tests robustness across corpora, retained-window sizes, longer discarded blocks, and matched cache budgets, while Appendix K examines whether a single compressed representation remains useful when reused across multiple subsequent continuations.

## 5 Limitations and Discussion

The present formulation has several limitations. First, the encoder and conditional rate model use diagonal Gaussian distributions for tractability. More expressive conditional distributions may better capture the structure of the latent representation and further improve the rate–prediction trade-off.

Second, the current training objective is a practical surrogate motivated by the rate–distortion formulation, rather than an optimised objective derived specifically for downstream language prediction. Further theoretical and empirical study of the weighting and interaction between the rate and prediction terms may lead to a better operating point and more stable optimisation.

Third, the reported rate is a variational surrogate rather than the length of an implemented compressed bitstream. A natural next step is therefore to quantize the learned representation and combine the conditional model with practical entropy or Wyner–Ziv-style coding. Such coding could also be combined with conventional KV quantization to obtain further compression.

Finally, the current implementation uses a fixed summary size and operating point. Allowing the number of summary vectors or coding rate to adapt to the redundancy between the discarded block and the retained context is another promising direction for future work.

## 6 Conclusion

This work studies KV-cache compression from a side-information perspective: the context retained by the decoder can also be used to reduce the information that must be carried by a learned summary of the discarded context. Across a broad set of frozen LLMs, exploiting this side information consistently lowers the variational rate and generally improves predictive distortion over an otherwise identical unconditional model. The advantage persists across additional corpora, rate–distortion settings, and correlated time series, showing that it is not tied to a single dataset or operating point.

SideKV also provides a favorable practical trade-off. It preserves useful long-context information on downstream tasks while avoiding adaptation of the pretrained predictor, and substantially reduces prefill computation relative to methods that process the full discarded block. A single compressed representation can also remain effective across multiple later continuations, supporting its use as a reusable cache summary. Overall, these results suggest that side-information-aware compression is a promising direction for lightweight KV-cache reduction and can be combined with stronger learned summaries or practical coding schemes.

### AI use statement

We used generative AI tools (Claude, GPT) to assist in modifying code, polish tables and figures, and help interpret results. We also used them to edit the paper for readability, to draft parts of the text, and to search the literature. We did not use generative AI tools to generate synthetic data, to formulate mathematical claims or to write proofs. We take full responsibility for the final content of this manuscript.

## REFERENCES


Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *ICLR*, 2017.

Yongqi An, Chang Lu, Kuan Zhu, Tao Yu, Chaoyang Zhao, Hong Wu, Ming Tang, and Jinqiao Wang. ReST-KV: Robust KV cache eviction with layer-wise output reconstruction and spatial-temporal smoothing. In *ICLR*, 2026.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *ACL*, 2024.

Priyansh Bhatnagar, Ashkan Moradifirouzabadi, Se-Hyun Yang, SeungJae Lee, Jungwook Choi, and Mingu Kang. STAR-KV: Low-rank KV cache compression via soft thresholding for adaptive rank control. In *ICML*, 2026.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *EMNLP*, 2023.

Ashwin Gerard Colaco and Nada Lahjouji. What to keep, what to forget: A rate–distortion view of memory compaction in LLMs and agents. *arXiv preprint arXiv:2607.08032*, 2026.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006.

Marco Cuturi. PEMS-SF. UCI Machine Learning Repository, https://doi.org/10.24432/C52G70, 2011.

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

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression. In *ICLR*, 2024.

Yair Feldman and Yoav Artzi. No mean feat: Simple, strong baselines for context compression. *arXiv preprint arXiv:2510.20797*, 2025.

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 *ICLR*, 2024.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *NeurIPS*, 2025.

Munsik Kim. Polynomial context-truncation sensitivity in autoregressive language models: Sequential Wyner–Ziv bounds for KV cache compression. *arXiv preprint arXiv:2605.25085*, 2026.

Hannah Laus, Claudio Mayrink Verdun, Hao Wang, Flavio du Pin Calmon, and Felix Krahmer. KV cache compression through the lens of transform coding. *arXiv preprint arXiv:2608.14191*, 2026.

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. In *NeurIPS*, 2024.

Zongqian Li, Yixuan Su, and Nigel Collier. 500xCompressor: Generalized prompt compression for large language models. In *ACL*, 2025.

Yuping Lin, Jiayuan Ding, Yue Xing, Pengfei He, Jiliang Tang, and Subhabrata Mukherjee. A simple plug-in for improving eviction-based KV cache compression. *arXiv preprint arXiv:2605.23258*, 2026.

Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Chunyang Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and Jingbo Zhu. Autoencoding-free context compression for LLMs via contextual semantic anchors. In *ICLR*, 2026.

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

Gregory Magarshak. Sequential KV cache compression via probabilistic language tries: Beyond the per-vector Shannon limit. *arXiv preprint arXiv:2604.15356*, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *NeurIPS*, 2023.

Alliot Nagle, Adway Girish, Marco Bondaschi, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. Fundamental limits of prompt compression: A rate-distortion framework for black-box language models. In *NeurIPS*, 2024.

Ezgi Özyılkan, Johannes Ballé, and Elza Erkip. Learned Wyner–Ziv compressors recover binning. In *ISIT*, 2023.

Buu Phan, Ashish Khisti, and Christos Louizos. Importance matching lemma for lossy compression with side information. In *AISTATS*, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *ICLR*, 2020.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. Cerebras blog, https://www.cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023.

Konrad Staniszewski and Adrian Łańcucki. KV cache transform coding for compact storage in LLM inference. In *ICLR*, 2026.

Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J. Ross Mitchell. Beyond a million tokens: Benchmarking and enhancing long-term memory in LLMs. In *ICLR*, 2026.

Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, https://doi.org/10.24432/C58C86, 2015.

Jay Whang, Alliot Nagle, Anish Acharya, Hyeji Kim, and Alexandros G. Dimakis. Neural distributed source coding. *IEEE Journal on Selected Areas in Information Theory*, 5:493–508, 2024.

Aaron D. Wyner and Jacob Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, 1976.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

Xintong Yang, Hao Gu, Binxing Xu, Lujun Li, Bei Liu, Jiacheng Liu, Qiyuan Zhu, Yike Guo, and Sirui Han. IndexMem: Learned KV-cache eviction with latent memory for long-context LLM inference. In *ICML*, 2026.

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

Junkai Zhang, Hang Guo, Luca Benini, and Yawei Li. RDKV: Rate-distortion bit allocation for joint eviction and quantization of the KV cache. *arXiv preprint arXiv:2605.08317*, 2026.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with Activation Beacon. In *ICLR*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023.

Ruida Zhou, Chao Tian, and Suhas Diggavi. An information-theoretic approach to understanding transformers' in-context learning of variable-order Markov chains. In *AISTATS*, 2026.

# A Implementation of the Comparison Systems

All methods are evaluated on the same spans with the same frozen LLM, each at its own cache budget. Methods that evict or quantize entries of the KV cache after prefill are compared with a full-context reference computed in the same way, and methods that construct a new input sequence are compared with the standard full context.

**Eviction.** The 32-token prefix and the 256-token window are re-encoded without the block, so the window takes the positions directly after the prefix. StreamingLLM likewise assigns positions within the cache (Xiao et al., 2024).

**SnapKV.** We follow Li et al. (2024) with the observation window of 32 and the max-pooling kernel of 7 used in their LongBench experiments. The attention scores of the last 32 queries on each block position are summed and max-pooled, and each key/value head of each layer keeps its top-$k$ positions. Under grouped-query attention, we average the scores of the query heads that share a key/value head. On LongBench, which retains tokens, not per-head cache entries, we average the scores over layers and heads before pooling.

**KVzip.** Following Kim et al. (2025), we pass a repeat prompt, *Repeat the previous context:*, followed by the block through the LLM with the prefilled cache, and score each entry by the maximum attention it receives during this pass, taken over the query heads that share its key/value head. Each key/value head of each layer keeps its top-$k$ entries. On LongBench, which retains tokens, the attention of this pass is averaged over layers and heads.

**ReST-KV.** We score each entry by the layer-wise output reconstruction indicator of An et al. (2026), with the exponential moving average over the last 32 queries ($\alpha$=0.3) and the adaptive spatial window ($\beta$=2000). Under grouped-query attention, we average the scores of the query heads that share a key/value head, and each key/value head keeps its top-$k$ entries. No hyperparameter is tuned per model.

**Quantization.** As in Liu et al. (2024), we quantize the keys of the block per channel and the values per token, uniformly between their extremes at four bits. The prefix and the retained window stay in bfloat16.

**TurboQuant.** We apply Algorithm 1 of Zandieh et al. (2026) to the values and Algorithm 2, their unbiased inner product quantizer, to the keys. Both randomly rotate each vector after scaling it to unit norm and quantize each coordinate with their Lloyd–Max codebook. As in their Section 4.3, the channels are split into outlier and non-outlier sets, with a quarter of them at one bit more. The budget depends on the head dimension and ranges from 572 to 608 slots across the LLMs, and the tables print the smallest.

**Activation Beacon.** We load the public checkpoint of Zhang et al. (2025) on its base LLM, Qwen2-7B-Instruct. Its window and stride are set to the length of the prefix plus the block, and its sink size to the length of the prefix. The block is then distilled into the keys and values of beacon tokens at the compression ratio $M/k$, one of the ratios it was trained on, and the prefix and the window stay uncompressed.

**SAC.** We load the pretrained adapters that Liu et al. (2026) released for the base Llama-3.2-1B at $5\times$ and $15\times$. As in their default strategy, the compressed text is divided into equal chunks, one per 5 or 15 tokens, and the middle token of each chunk is selected as an anchor token. In its own layout, SAC compresses the whole context into the keys and values of the anchor tokens. In the layout of this paper, the anchors are selected from the block and placed where the block was, and the prefix and the window stay uncompressed.

**Mean pooling.** The token embeddings of the block are split into $k$ contiguous groups and averaged within each group, and the $k$ averages take the place of the block between the prefix and the window. The operator is the mean pooling of Feldman & Artzi (2025).

**500xCompressor.** As in Li et al. (2025), the encoder is the frozen LLM with a trainable LoRA adapter (Hu et al., 2022) on its attention projections. The adapter has rank 128. The encoder reads the block followed by $k$ compressed tokens, and the keys and values of the compressed tokens at every layer are placed in the cache between the prefix and the window. The LLM then runs on this cache with the adapter off. We train the compressed tokens and the adapter for 3000 steps on the same distortion as SideKV.

## B Experimental Details and Hyperparameters

**Data.** The main corpus joins end to end 4000 SlimPajama (Soboleva et al., 2023) documents of at least 4000 characters each. The WikiText corpus joins the non-empty lines of the training split of WikiText-2 (Merity et al., 2017) in its raw form (`wikitext-2-raw-v1`), 10.9 MB of text and 2.5M Qwen2.5 tokens in total. The PG19 corpus takes books from the training split of PG19 (Rae et al., 2020), 4.8M Qwen2.5 tokens in total. The novels corpus concatenates the Project Gutenberg plain-text files of *War and Peace*, *Great Expectations*, *Wuthering Heights*, *Emma* and *Ulysses* (eBooks 2600, 1400, 768, 158 and 4300) without further cleaning, 7.5 MB of text and 1.8M Qwen2.5 tokens in total. The Python corpus concatenates, in path order, the 1,910 source files larger than 1 KiB in the PyTorch 2.8.0 and NumPy 2.0.2 packages installed under Python 3.9, 44 MB of text and 10.9M Qwen2.5 tokens in total. At the main layout a span has 1568 tokens, covering the prefix, the block, the window and the continuation. Every span starts at a uniformly drawn position and can cross a document boundary.

**SideKV.** The encoder maps block embeddings to dimension 512, followed by a 2-layer, 8-head Transformer. We average features within each of $k$ contiguous groups and output a per-slot mean and log-variance, clipped to $[-8, 8]$. The latent summary is sampled per slot, projected back to the LLM hidden size via a single linear layer, and rescaled by $\bar{c}$ (Eq. 11), calculated as the average norm over the first 4,096 vocabulary tokens. The rate model uses the same architecture on the prefix and retained window; the unconditional variant replaces it with learned slot priors. SideKV adds 13.7M to 18.3M parameters across tested LLMs (6.9M to 9.3M without the rate model). Models are trained for 3,000 steps with AdamW (weight decay 0.01, gradient clip 1.0), keeping the base LLM frozen. We run the base LLM in `bfloat16` and SideKV in `float32`. We use a learning rate of $10^{-4}$ for models below 7B parameters and $5 \times 10^{-5}$ for larger models and 500xCompressor. Per-device batch size is 4 for models under 7B, 2 for 7B models, and 1 for Gemma-3.

**The rate term.** Training minimizes Equation 13 following the schedule $\lambda_n$ (Section 3.5), with no lower clamp on $\overline{R}_{\text{var}}$. The weight $\beta = T/M$ is 0.25 at the main layout, and Appendix E.2 also trains at $4T/M$ and $T/(4M)$.

**Compute.** Experiments were performed using Intel Data Center GPU Max 1550 accelerators, for a total of approximately 10,000 GPU-hours.

## C Information-Theoretic Background

Standard definitions used throughout, following Cover & Thomas (2006).

**Entropy and conditional entropy.** For a discrete random variable $X$ with distribution $p$, the entropy $H(X) = -\sum_x p(x) \log_2 p(x)$ is the minimum expected number of bits per symbol needed to describe $X$ losslessly. The conditional entropy $H(X \mid Y) = \mathbb{E}_y H(X \mid Y{=}y)$ measures the uncertainty about $X$ that remains once $Y$ is seen, and $H(X \mid Y) \le H(X)$ with equality exactly when the two are independent.

**Kullback–Leibler divergence.** For distributions $q$ and $r$, $D_{\mathrm{KL}}(q \,\|\, r) = \sum_z q(z) \log_2 \frac{q(z)}{r(z)} \ge 0$, with equality exactly when $q = r$. It is the excess code length incurred when data distributed as $q$ is encoded with a code designed for $r$. With logarithms to base two the divergence is in bits. The paper uses it as a variational rate surrogate and implements no quantizer, entropy coder or transmitted bitstream, and the quantity is not an operational source-coding rate.

**Mutual information.** $I(X;Z) = H(X) - H(X \mid Z)$ is the information $Z$ conveys about $X$. The conditional version $I(X;Z \mid Y) = H(X \mid Y) - H(X \mid Z,Y)$ is the information $Z$ conveys about $X$ beyond the information $Y$ already supplies.

**Prediction is compression.** Language models are themselves effective compressors (Delétang et al., 2024), and trained transformers track a universal compressor in context (Zhou et al., 2026). A predictor that assigns probability $\hat{p}$ to the symbol that actually occurs can drive an arithmetic coder that uses $-\log_2 \hat{p}$ bits on it, and the total over a sequence exceeds $\sum_t -\log_2 \hat{p}_t$ by at most two bits (Cover & Thomas, 2006). Log-loss is a code length, and the distortion and the rate surrogate are both measured in bits for that reason.

**Gaussian KL in closed form.** The implementation computes the rate between diagonal Gaussians, for which

$$D_{\mathrm{KL}}\big(\mathcal{N}(\mu,\sigma^2) \,\|\, \mathcal{N}(\mu_0,\sigma_0^2)\big) = \ln \tfrac{\sigma_0}{\sigma} + \tfrac{\sigma^2+(\mu-\mu_0)^2}{2\sigma_0^2} - \tfrac{1}{2} \quad \text{nats}, \tag{15}$$

summed over the $k \times d_z$ lattice and divided by $M \ln 2$ to report bits per token of the block.

## D Derivation of the Variational Rate Surrogate

Table 5 collects the notation of Section 3.

Table 5: Core notation used in Section 3.

| Symbol | Meaning |
|---|---|
| $S$ | retained prefix sequence, with length $s$ |
| $X$ | historical block to be compressed, with length $M$ |
| $W$ | retained recent window, with length $w$ |
| $Y = (S, W)$ | retained context used as side information |
| $Z$ | generic representation replacing $X$ |
| $U$ | continuation to be predicted, with length $T$ |
| $\theta_0$ | frozen pretrained language-model parameters |
| $\phi$ | parameters of the encoder network |
| $\psi$ | parameters of the side-information network |
| $k$ | number of learned summary slots |
| $d_z$ | latent dimension per summary slot |

The units of the surrogate. If natural logarithms are used when evaluating the KL divergence, the result is in nats. We convert to bits using

$$R_{\mathrm{var}}^{(\mathrm{bits})} = \frac{R_{\mathrm{var}}^{(\mathrm{nats})}}{\ln 2}, \tag{16}$$

and when reporting the surrogate per token of the discarded block we use

$$\overline{R}_{\mathrm{var}} = \frac{R_{\mathrm{var}}^{(\mathrm{bits})}}{M} \quad \text{bits/token.} \tag{17}$$

Tables report $R_{\mathrm{var}}$ in bits over the block, which is $M\overline{R}_{\mathrm{var}}$, and the distortion of (2) in bits per token of $U$.

This appendix derives (6) without assuming Gaussian distributions.

### D.1 Joint and conditional distributions induced by the encoder

Let

$$(X, Y) \sim p_{\mathrm{data}}(x, y). \tag{18}$$

Because the encoder observes only $X$,

$$q_\phi(z \mid x, y) = q_\phi(z \mid x), \tag{19}$$

and the joint distribution induced by the data and the encoder is

$$q_\phi(x, y, z) = p_{\mathrm{data}}(x, y)\, q_\phi(z \mid x). \tag{20}$$

The corresponding conditional latent distribution given the retained side information is

$$q_\phi(z \mid y) = \int p_{\mathrm{data}}(x \mid y)\, q_\phi(z \mid x)\, \mathrm{d}x. \tag{21}$$

For discrete $X$, the integral is replaced by a sum. Although $q_\phi(z \mid y)$ is well defined, it is generally intractable, since evaluating it requires averaging over all $x$ that are compatible with $y$. The learned model $r_\psi(z \mid y)$ is introduced as a tractable approximation.

### D.2 Expansion of the KL surrogate

Starting from

$$R_{\mathrm{var}} = \mathbb{E}_{x,y}\left[D_{\mathrm{KL}}\left(q_\phi(Z \mid x) \| r_\psi(Z \mid y)\right)\right], \tag{22}$$

expand the KL divergence:

$$R_{\mathrm{var}} = \mathbb{E}_{x,y}\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{q_\phi(z \mid x)}{r_\psi(z \mid y)}\right]. \tag{23}$$

Multiply and divide by the encoder-induced conditional distribution $q_\phi(z \mid y)$:

$$\log \frac{q_\phi(z \mid x)}{r_\psi(z \mid y)} = \log \frac{q_\phi(z \mid x)}{q_\phi(z \mid y)} + \log \frac{q_\phi(z \mid y)}{r_\psi(z \mid y)}. \tag{24}$$

Substituting (24) into (23),

$$\begin{aligned} R_{\mathrm{var}} = {} & \mathbb{E}_{x,y,z}\left[\log \frac{q_\phi(z \mid x)}{q_\phi(z \mid y)}\right] \\ & + \mathbb{E}_{x,y,z}\left[\log \frac{q_\phi(z \mid y)}{r_\psi(z \mid y)}\right]. \end{aligned} \tag{25}$$

### D.3 First term: conditional mutual information

By definition,

$$I_q(X; Z \mid Y) = \mathbb{E}_{x,y,z}\left[\log \frac{q_\phi(z \mid x, y)}{q_\phi(z \mid y)}\right]. \tag{26}$$

Using the Markov relation in (19),

$$q_\phi(z \mid x, y) = q_\phi(z \mid x), \tag{27}$$

which gives

$$I_q(X; Z \mid Y) = \mathbb{E}_{x,y,z}\left[\log \frac{q_\phi(z \mid x)}{q_\phi(z \mid y)}\right], \tag{28}$$

and the first term in (25) is exactly $I_q(X; Z \mid Y)$.

### D.4 Second term: variational mismatch

For the second term in (25), the logarithmic factor no longer depends explicitly on $x$. Marginalizing over $x$ gives

$$\begin{aligned}&\mathbb{E}_{x,y,z}\left[\log\frac{q_\phi(z\mid y)}{r_\psi(z\mid y)}\right]\\&\quad=\mathbb{E}_y\mathbb{E}_{z\sim q_\phi(z|y)}\left[\log\frac{q_\phi(z\mid y)}{r_\psi(z\mid y)}\right]\\&\quad=\mathbb{E}_Y\left[D_{\mathrm{KL}}\left(q_\phi(Z\mid Y)\|r_\psi(Z\mid Y)\right)\right].\end{aligned}\tag{29}$$

Combining (28) and (29),

$$R_{\mathrm{var}}=I_q(X;Z\mid Y)+\mathbb{E}_Y\left[D_{\mathrm{KL}}\left(q_\phi(Z\mid Y)\|r_\psi(Z\mid Y)\right)\right].\tag{30}$$

Since the KL divergence is nonnegative,

$$R_{\mathrm{var}}\geq I_q(X;Z\mid Y),\tag{31}$$

with the bound tight for a fixed encoder if

$$r_\psi(z\mid y)=q_\phi(z\mid y).\tag{32}$$

### D.5 Unconditional model and the role of side information

For an unconditional model $r_\psi(Z)$,

$$R_{\mathrm{var}}^{\mathrm{uncond}}=\mathbb{E}_X\left[D_{\mathrm{KL}}\left(q_\phi(Z\mid X)\|r_\psi(Z)\right)\right],\tag{33}$$

and an analogous derivation gives

$$R_{\mathrm{var}}^{\mathrm{uncond}}=I_q(X;Z)+D_{\mathrm{KL}}\left(q_\phi(Z)\|r_\psi(Z)\right).\tag{34}$$

The chain rule for mutual information gives

$$I_q(X;Z)+I_q(Z;Y\mid X)=I_q(Z;Y)+I_q(X;Z\mid Y).\tag{35}$$

Because $Z-X-Y$ implies

$$I_q(Z;Y\mid X)=0,\tag{36}$$

we obtain

$$I_q(X;Z\mid Y)=I_q(X;Z)-I_q(Z;Y)\leq I_q(X;Z).\tag{37}$$

The information-theoretic floor of the conditional surrogate can be lower than that of the unconditional surrogate by $I_q(Z;Y)$. In practice the learned KL surrogate also depends on the corresponding variational mismatch. The observed difference between conditional and unconditional models should not be attributed solely to this information-theoretic term.

### D.6 Closed-form KL for the Gaussian implementation

For completeness, we give the KL expression used by the diagonal-Gaussian implementation. Let

$$q_\phi(z_j\mid X)=\mathcal{N}\left(\mu_{\phi,j},\operatorname{diag}\sigma_{\phi,j}^2\right),\qquad r_\psi(z_j\mid Y)=\mathcal{N}\left(\widetilde{\mu}_{\psi,j},\operatorname{diag}\widetilde{\sigma}_{\psi,j}^2\right).\tag{38}$$

Then

$$\begin{aligned}&D_{\mathrm{KL}}\left(q_\phi(z_j\mid X)\|r_\psi(z_j\mid Y)\right)\\&=\frac{1}{2}\sum_{\ell=1}^{d_z}\left[\log\frac{\widetilde{\sigma}_{\psi,j,\ell}^2}{\sigma_{\phi,j,\ell}^2}+\frac{\sigma_{\phi,j,\ell}^2+\left(\mu_{\phi,j,\ell}-\widetilde{\mu}_{\psi,j,\ell}\right)^2}{\widetilde{\sigma}_{\psi,j,\ell}^2}-1\right].\end{aligned}\tag{39}$$

The encoder and the conditional model factorize over the $k$ slots,

$$q_\phi(Z\mid X)=\prod_{j=1}^{k}q_\phi(z_j\mid X),\qquad r_\psi(Z\mid Y)=\prod_{j=1}^{k}r_\psi(z_j\mid Y),\tag{40}$$

and the divergence is the sum over slots,

$$D_{\mathrm{KL}}\left(q_\phi(Z \mid X) \| r_\psi(Z \mid Y)\right) = \sum_{j=1}^{k} D_{\mathrm{KL}}\left(q_\phi(z_j \mid X) \| r_\psi(z_j \mid Y)\right). \tag{41}$$

The diagonal-Gaussian choice is computationally convenient for two reasons. It gives the analytic KL in (39), and it permits the reparameterized sampling of Section 3.4. It should be read as a practical instantiation of the distribution-agnostic formulation, and not as a claim that Gaussian latent models are optimal for context compression.

# E Additional Results on Side Information

This appendix extends the controlled comparison of Section 4.2 in three additional settings. In each setting SideKV and the unconditional prior share the encoder architecture, the frozen LLM, the data and the training, and differ in the rate model alone, $r_\psi(Z \mid Y)$ against $r_\psi(Z)$. Appendix E.1 changes the corpus, Appendix E.2 changes the weight $\beta$ of the distortion term in Equation 13, and Appendix E.3 replaces text with two correlated time series. As in Table 1, a smaller $\rho_R$ and a larger $G_D$ indicate a benefit from side information, and so does a larger $G_L$ for the time series.

## E.1 Additional corpora

Table 6: Effect of side information on four additional corpora. Quantities follow Table 1 for seven frozen LLMs, with both models trained and evaluated on each corpus. Smaller $\rho_R$ and larger $G_D$ indicate greater benefit.

| | | Variational rate | | | Excess distortion (bits/token) | | |
|---|---|---|---|---|---|---|---|
| Predictor | Corpus | $R_{\mathrm{u}}$ | $R_{\mathrm{SI}}$ | $\rho_R \downarrow$ | $\Delta D^{\mathrm{u}}_{XZ}$ | $\Delta D^{\mathrm{SI}}_{XZ}$ | $G_D \uparrow$ |
| Qwen2.5-0.5B | WikiText | 0.52 | 0.44 | 86% | +0.083 | +0.025 | +0.058 |
| | PG19 | 0.27 | 0.18 | 66% | +0.100 | +0.092 | +0.009 |
| | novels | 0.30 | 0.15 | 48% | +0.106 | +0.093 | +0.013 |
| | Python | 0.28 | 0.09 | 30% | +0.243 | +0.233 | +0.010 |
| Qwen2.5-7B | WikiText | 2.45 | 0.26 | 11% | +0.144 | −0.072 | +0.216 |
| | PG19 | 0.60 | 0.16 | 26% | −0.015 | −0.105 | +0.091 |
| | novels | 0.72 | 0.22 | 31% | +0.047 | −0.070 | +0.116 |
| | Python | 0.31 | 0.15 | 48% | +0.161 | +0.099 | +0.061 |
| Qwen2.5-14B | WikiText | 1.84 | 0.14 | 8% | +0.134 | +0.019 | +0.115 |
| | PG19 | 0.21 | 0.11 | 51% | +0.119 | +0.109 | +0.010 |
| | novels | 0.55 | 0.10 | 18% | +0.127 | +0.104 | +0.023 |
| | Python | 0.17 | 0.08 | 48% | +0.158 | +0.138 | +0.020 |
| Llama-3.1-8B | WikiText | 1.00 | 0.26 | 26% | +0.097 | +0.019 | +0.078 |
| | PG19 | 0.80 | 0.26 | 33% | +0.085 | +0.051 | +0.034 |
| | novels | 1.08 | 0.35 | 32% | +0.057 | +0.016 | +0.041 |
| | Python | 0.38 | 0.12 | 31% | +0.183 | +0.169 | +0.014 |
| Gemma-3-1B | WikiText | 7.11 | 1.20 | 17% | +0.138 | −0.150 | +0.288 |
| | PG19 | 4.23 | 0.89 | 21% | +0.086 | −0.147 | +0.234 |
| | novels | 3.73 | 0.76 | 20% | −0.217 | −0.455 | +0.238 |
| | Python | 5.02 | 0.66 | 13% | +0.364 | +0.130 | +0.234 |
| DeepSeek-V2-Lite | WikiText | 0.67 | 0.25 | 38% | +0.041 | −0.033 | +0.075 |
| | PG19 | 0.54 | 0.22 | 41% | −0.084 | −0.133 | +0.049 |
| | novels | 0.72 | 0.19 | 26% | −0.163 | −0.215 | +0.052 |
| | Python | 0.46 | 0.19 | 41% | +0.130 | +0.095 | +0.036 |
| Mistral-7B | WikiText | 0.62 | 0.21 | 33% | +0.132 | +0.056 | +0.076 |
| | PG19 | 0.39 | 0.13 | 33% | +0.099 | +0.089 | +0.010 |
| | novels | 0.44 | 0.09 | 20% | +0.126 | +0.107 | +0.019 |
| | Python | 0.36 | 0.09 | 23% | +0.215 | +0.205 | +0.011 |

We replicate the conditional–unconditional evaluation on WikiText, PG19, five public-domain novels, and Python source code across seven of the thirteen LLMs, which establishes the sensitivity of the results to corpus variations. For each corpus, SideKV and the unconditional prior are trained on that corpus and evaluated on its held-out final tenth, using the layout of Section 4.1, $\beta = T/M$, and the training protocol of Appendix B.

Table 6 reports both the retained-rate ratio $\rho_R$ and the distortion gain $G_D$, following the notation of Table 1. SideKV has the lower rate in all 28 cells, with $\rho_R$ from 8% to 86%. The gain $G_D$ is

positive in all 28 cells. It ranges from $+0.009$ bits/token, on PG19 with Qwen2.5-0.5B, to $+0.288$, on WikiText with Gemma-3-1B.

### E.2 Different rate–distortion weights

We investigate the effect of the rate-distortion trade-off on side information by training SideKV and the unconditional prior on SlimPajama. For the five LLMs evaluated, we tested $\beta \in \{4T/M, T/M, T/(4M)\}$, using $\beta = T/M$ as the baseline. Since $\beta$ multiplies the distortion term in Eq. (13), smaller values place relatively greater emphasis on the rate term. The layout and training otherwise follow Section 4.1 and Appendix B. Table 7 reports the quantities of Table 1 at each setting.

Table 7: Effect of side information under three rate–distortion weights $\beta$. Smaller $\rho_R$ and larger $G_D$ indicate greater benefit from side information.

| | | Variational rate | | | Excess distortion (bits/token) | | |
|---|---|---|---|---|---|---|---|
| Predictor | $\beta$ | $R_{\mathrm{u}}$ | $R_{\mathrm{SI}}$ | $\rho_R \downarrow$ | $\Delta D^{\mathrm{u}}_{XZ}$ | $\Delta D^{\mathrm{SI}}_{XZ}$ | $G_D \uparrow$ |
| Qwen2.5-0.5B | $\frac{4T}{M}$ | 0.88 | 0.33 | 38% | +0.125 | +0.109 | +0.016 |
| | $\frac{T}{M}$ | 0.21 | 0.10 | 48% | +0.125 | +0.109 | +0.016 |
| | $\frac{T}{4M}$ | 0.04 | 0.06 | 150% | +0.126 | +0.110 | +0.016 |
| Qwen2.5-7B | $\frac{4T}{M}$ | 5.05 | 0.53 | 10% | +0.021 | −0.082 | +0.103 |
| | $\frac{T}{M}$ | 1.25 | 0.19 | 15% | +0.028 | −0.078 | +0.106 |
| | $\frac{T}{4M}$ | 0.48 | 0.12 | 25% | +0.032 | −0.074 | +0.106 |
| Qwen2.5-14B | $\frac{4T}{M}$ | 1.17 | 0.26 | 22% | +0.119 | +0.097 | +0.022 |
| | $\frac{T}{M}$ | 0.22 | 0.14 | 64% | +0.119 | +0.097 | +0.022 |
| | $\frac{T}{4M}$ | 0.05 | 0.04 | 80% | +0.120 | +0.104 | +0.016 |
| Llama-3.1-8B | $\frac{4T}{M}$ | 3.27 | 0.65 | 20% | +0.095 | +0.065 | +0.030 |
| | $\frac{T}{M}$ | 0.66 | 0.26 | 39% | +0.097 | +0.068 | +0.029 |
| | $\frac{T}{4M}$ | 0.14 | 0.08 | 57% | +0.099 | +0.070 | +0.029 |
| Mistral-7B | $\frac{4T}{M}$ | 1.80 | 0.19 | 11% | +0.126 | +0.118 | +0.008 |
| | $\frac{T}{M}$ | 0.26 | 0.07 | 27% | +0.128 | +0.118 | +0.009 |
| | $\frac{T}{4M}$ | 0.06 | 0.06 | 100% | +0.130 | +0.121 | +0.009 |

The distortion gain $G_D$ is positive in all fifteen settings. The retained rate $\rho_R$ remains below $100\%$ for all predictors at $\beta = 4T/M$ and at the default setting $\beta = T/M$. When $\beta$ is reduced to $T/(4M)$, placing stronger relative emphasis on the rate term, the rate advantage becomes less uniform: $\rho_R$ reaches $150\%$ on Qwen2.5-0.5B and $100\%$ on Mistral-7B. In both cases, however, the absolute rates are already very small (at most $0.06$ bits over the block), and their differences are not statistically clear. These results support $\beta = T/M$ as a practical default that provides a stable balance between rate reduction and predictive distortion.

### E.3 Correlated time series

Two numerical datasets from the UCI Machine Learning Repository (Trindade, 2015; Cuturi, 2011) serve to confirm the benefits of side information in non-textual domains. For the electricity series we use the first $100$ clients of ElectricityLoadDiagrams20112014. The block $X$ is one client's day of $96$ quarter-hour load readings, and the retained context $Y$ is the same client's previous day. For the traffic series we use the first $200$ sensors of PEMS-SF. The block is one sensor's day of $144$ ten-minute occupancy readings, and $Y$ is the same day at the other sensor most correlated with it over the first $200$ days. Each reading is scaled so that the 99th percentile of its client or sensor becomes $99$, rounded to an integer, clipped to the range $00$ to $99$ and written as two digits. A day then takes $287$ tokens for electricity and $431$ for traffic.

The frozen LLM is Qwen2.5-0.5B. The encoder compresses $X$ into $k{=}16$ slots with $d_z{=}32$, and the conditional rate model $r_\psi(Z \mid Y)$ takes the embeddings of $Y$ as input. In both models, the LLM is given $Y$ followed by the slots and is scored on $X$ itself. The encoder and the conditional rate model have the architecture of Appendix B, and both models are trained for $3000$ steps at a batch of $8$. The

scored tokens are those of $X$, which gives $\beta = T/M = 1$, and the weight takes the values $4T/M$, $T/M$ and $T/(4M)$. Table 8 reports the log-loss on $X$ over 256 held-out days, together with the gain $G_L$.

Table 8: Effect of side information on two correlated time series. Electricity load, with the same client's previous day as the retained context, and traffic occupancy, with the same day at the most correlated other sensor. $L^{\mathrm{u}}$ and $L^{\mathrm{SI}}$ are the prediction log-losses (bits/token) of the unconditional and side-information-aware models, and $G_L \triangleq L^{\mathrm{u}} - L^{\mathrm{SI}}$; larger $G_L$ indicates greater benefit from side information. A smaller $\beta$ weights the rate more.

| Series | $\beta$ | $L^{\mathrm{u}}$ | $L^{\mathrm{SI}}$ | $G_L \uparrow$ |
|---|---|---|---|---|
| electricity | $\frac{4T}{M}$ | 1.290 | 1.241 | +0.049 |
| | $\frac{T}{M}$ | 1.315 | 1.241 | +0.074 |
| | $\frac{T}{4M}$ | 1.322 | 1.243 | +0.079 |
| traffic | $\frac{4T}{M}$ | 1.197 | 1.153 | +0.044 |
| | $\frac{T}{M}$ | 1.233 | 1.151 | +0.082 |
| | $\frac{T}{4M}$ | 1.250 | 1.155 | +0.095 |

The side-information-aware model has the lower log-loss in all six cells. Its log-loss stays between 1.241 and 1.243 bits/token on electricity and between 1.151 and 1.155 on traffic, while that of the unconditional model rises as the rate is weighted more. The gain $G_L$ increases accordingly, from $+0.049$ at $\beta = 4T/M$ to $+0.079$ at $\beta = T/(4M)$ on electricity and from $+0.044$ to $+0.095$ on traffic.

## F Full Comparison Across Thirteen LLMs

Table 9: $\Delta D_{XZ}$ on all thirteen LLMs, in bits per continuation token; lower is better. Bold marks the lowest entry of each class of method.

| | Qwen2.5 | | | Qwen3 | Llama-3.2 | | Llama-3.1 | Gemma-3 | | | OLMoE | DeepSeek | Mistral | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| System | 0.5B | 7B | 14B | 8B | 1B | 3B | 8B | 1B | 4B | 12B | 1B-7B | V2-Lite | 7B | Avg. |
| *Eviction and selection* | | | | | | | | | | | | | | |
| evict. (288) | 0.125 | 0.016 | 0.116 | 0.112 | 0.120 | 0.121 | 0.110 | 0.450 | 0.838 | 2.740 | 0.130 | 0.041 | 0.125 | 0.388 |
| SnapKV | 0.097 | −0.000 | **0.052** | 0.065 | **0.098** | **0.074** | 0.051 | 0.195 | 0.179 | 0.122 | **0.084** | −0.002 | 0.060 | 0.083 |
| KVzip | 0.105 | 0.016 | 0.061 | 0.087 | 0.112 | 0.080 | 0.057 | 0.282 | 0.313 | 0.566 | 0.109 | 0.005 | 0.062 | 0.143 |
| ReST-KV | **0.092** | **−0.007** | 0.054 | **0.059** | 0.099 | 0.078 | **0.050** | **0.183** | **0.127** | **0.106** | **0.084** | **−0.003** | **0.059** | **0.075** |
| *Quantization* | | | | | | | | | | | | | | |
| KIVI (544) | **0.002** | **0.009** | **0.001** | **0.001** | **0.002** | **0.001** | **0.001** | **0.004** | 0.005 | 0.003 | **0.001** | **0.001** | **0.001** | **0.002** |
| TurboQ. (572) | 0.012 | 0.034 | **0.001** | 0.002 | 0.008 | 0.003 | 0.003 | **0.004** | **0.001** | **0.002** | **0.001** | 0.005 | 0.002 | 0.006 |
| *Trained inside the LLM* | | | | | | | | | | | | | | |
| 500xC. | 0.097 | −0.144 | 0.084 | −0.107 | −0.003 | −0.018 | 0.065 | −0.190 | −0.131 | −0.114 | 0.113 | −0.095 | 0.116 | −0.025 |
| *Summary vectors, LLM frozen* | | | | | | | | | | | | | | |
| MeanPool | 0.136 | 2.274 | 0.167 | 0.116 | 0.157 | 0.140 | 0.113 | 0.553 | 0.833 | 2.721 | 0.132 | 0.087 | 0.125 | 0.581 |
| uncond. | 0.125 | 0.028 | 0.119 | 0.068 | 0.068 | 0.068 | 0.097 | 0.209 | **0.106** | **0.078** | 0.131 | 0.004 | 0.128 | 0.095 |
| **SideKV** | **0.109** | **−0.078** | **0.097** | **−0.045** | **0.042** | **0.032** | **0.068** | **−0.031** | 0.111 | 0.145 | **0.117** | **−0.038** | **0.118** | **0.050** |

Table 9 provides the full 13-LLM comparison on the main SlimPajama corpus, underlying the summary results reported in Sections 4.3 and 4.4. It places SideKV alongside eviction and selection methods, 4-bit quantization baselines, predictor-adapted learned compression, and frozen-predictor summary representations under the common evaluation protocol of Section 4.1.

The 4-bit quantization baselines achieve very small excess distortion, but retain all cache entries and therefore correspond to substantially larger cache budgets of more than 540 equivalent slots. Among methods operating at approximately the 320-slot budget, ReST-KV is the strongest selection baseline on average, with $\Delta D_{XZ} = 0.075$ bits/token, whereas SideKV achieves 0.050 bits/token. Within the frozen-predictor summary-vector methods, SideKV substantially improves over MeanPool and also outperforms the unconditional variant on 11 of the 13 LLMs. It further improves over simple eviction on all 13 models.

Negative values of $\Delta D_{XZ}$ indicate that replacing part of the historical context can occasionally yield a lower continuation loss than using the full context. The Gemma-3 models are particularly sensitive to eviction, with the excess distortion increasing to 2.740 bits/token on Gemma-3-12B.

SideKV substantially mitigates this degradation, remaining below 0.15 bits/token across all three Gemma-3 scales.

## G Additional Comparisons with Predictor-Adapted Compressors

Section 4.3 compares SideKV with learned context-compression methods that adapt the predictor or introduce trainable components inside it. This appendix provides further comparisons with SAC, 500xCompressor, and Activation Beacon. Because the released implementations of these methods support different predictors, the comparisons cannot be carried out under a single common model suite: the released SAC adapters are available for Llama-3.2-1B, whereas Activation Beacon is evaluated on its released Qwen2-7B-Instruct checkpoint, while 500xCompressor can be evaluated across the broader set of LLMs considered in this paper. We therefore report the comparisons separately. Simple eviction is included in Table 11 as a reference baseline, since it represents the case where the historical block is discarded without a learned representation and provides a natural reference for quantifying the gain from learned compression.

Table 10: Comparison of SAC and SideKV on Llama-3.2-1B across five corpora. Values are $\Delta D_{XZ}$ in bits per continuation token; lower is better, and bold marks the lowest entry in each column.

| System | SlimPajama | WikiText | PG19 | novels | Python |
|---|---|---|---|---|---|
| *Llama-3.2-1B base* | | | | | |
| SAC 5× | +0.125 | +0.196 | +0.157 | +0.181 | **+0.182** |
| SAC 15× | +0.140 | +0.247 | +0.177 | +0.190 | +0.247 |
| SAC 5×, its layout | +0.226 | +0.360 | +0.271 | +0.334 | +0.412 |
| SAC 15×, its layout | +0.285 | +0.552 | +0.349 | +0.395 | +0.673 |
| **SideKV** | **+0.102** | **+0.060** | **+0.121** | **+0.135** | +0.219 |

We first compare SideKV with SAC across multiple corpora. Table 10 reports results using the released Llama-3.2-1B SAC adapters at 5× and 15× compression (Liu et al., 2026) on SlimPajama, WikiText, PG19, five public-domain novels, and Python source code. In the layout of this paper, across these five corpora, SideKV achieves lower $\Delta D_{XZ}$ than SAC at 15× compression in all cases and than SAC at 5× compression in four of the five, while using fewer cache slots. The largest difference occurs on WikiText, where $\Delta D_{XZ}$ is +0.060 bits/token for SideKV, compared with +0.196 and +0.247 bits/token for SAC at 5× and 15× compression, respectively.

Table 11: Comparison with predictor-adapted compressors. Values are $\Delta D_{XZ}$ in bits/token; lower is better.

| Predictor | eviction | SideKV | 500xC. | Beacon | rec. |
|---|---|---|---|---|---|
| Qwen2.5-0.5B | +0.125 | +0.109 | +0.097 | – | 57% |
| Qwen2.5-7B | +0.016 | −0.078 | −0.144 | – | 59% |
| Qwen2.5-14B | +0.116 | +0.097 | +0.084 | – | 59% |
| Qwen3-8B | +0.112 | −0.045 | −0.107 | – | 71% |
| Llama-3.2-1B | +0.120 | +0.042 | −0.003 | – | 63% |
| Llama-3.2-3B | +0.121 | +0.032 | −0.018 | – | 64% |
| Llama-3.1-8B | +0.110 | +0.068 | +0.065 | – | 95% |
| Gemma-3-1B | +0.450 | −0.031 | −0.190 | – | 75% |
| Gemma-3-4B | +0.838 | +0.111 | −0.131 | – | 75% |
| Gemma-3-12B | +2.740 | +0.145 | −0.114 | – | 91% |
| OLMoE-1B-7B | +0.130 | +0.117 | +0.113 | – | 77% |
| DeepSeek-V2-Lite | +0.041 | −0.038 | −0.095 | – | 58% |
| Mistral-7B | +0.125 | +0.118 | +0.116 | – | 74% |
| Qwen2-7B-Instruct | +0.114 | +0.107 | −0.080 | +0.056 | 4% |

Table 11 provides additional comparisons between SideKV, eviction, and 500xCompressor. 500xCompressor uses the LLM itself as an encoder through a LoRA adapter and transfers the keys and values of its memory tokens at every layer (Li et al., 2025). In contrast, SideKV keeps the pretrained LLM unchanged and passes one 64-dimensional vector per summary slot. Both SideKV and 500xCompressor use 320 cache slots and are trained for 3000 steps under the protocols described in Appendices A and B.

The column "rec." reports the fraction of the improvement from eviction to 500xCompressor that is recovered by SideKV. Across the thirteen LLMs, SideKV recovers 71% of this improvement on average, ranging from 57% on Qwen2.5-0.5B to 95% on Llama-3.1-8B.

The final row reports the corresponding comparison on Qwen2-7B-Instruct, the base model of the released Activation Beacon checkpoint (Zhang et al., 2025). SideKV attains $\Delta D_{XZ} = +0.107$, compared with $+0.114$ for eviction, $+0.056$ for Activation Beacon, and $-0.080$ for 500xCompressor. These results complement the main comparison in Section 4.3 by illustrating the trade-off relative to predictor-adapted compressors while SideKV keeps the pretrained LLM unchanged.

## H Computational Efficiency

Table 3 in Section 4.4 compares the prefill arithmetic and the time to first token of each system. This appendix measures the prefill throughput at batch sizes of 1 to 16 requests (Table 12) and the time and peak memory of scoring a continuation with SideKV and with the full context on all 13 LLMs (Table 13). Both measurements run on one GPU with the LLM in bfloat16.

**Prefill throughput.** Table 12 evaluates serving throughput under batched prefill ($B \in \{1, 4, 8, 16\}$) on Qwen2.5-7B under the configuration of Section 4.1. Timing runs from input token IDs to a populated cache ready for decoding, capturing all intermediate operations: attention scoring passes for selection baselines, quantization overhead for KIVI, and encoder forward passes for 500xCompressor and SideKV. All methods run with eager attention to expose the attention weights needed for scoring. Each entry reports throughput (requests per second), defined as the batch size divided by the median latency across 12 batches of held-out SlimPajama sequences, measured after one warmup batch with explicit GPU synchronizations before and after each pass.

Table 12: Prefill throughput of Qwen2.5-7B at a 1024-token block on one GPU, in requests per second from the prompt to a cache ready to decode; higher is better.

| System | batch 1 | batch 4 | batch 8 | batch 16 |
|---|---|---|---|---|
| *full context* | 2.18 | 5.26 | 5.13 | 5.14 |
| eviction | 7.49 | 27.78 | 30.20 | 30.13 |
| SnapKV | 1.57 | 4.61 | 4.59 | 4.60 |
| KVzip | 0.86 | 2.45 | 2.45 | 2.47 |
| ReST-KV | 0.51 | 2.31 | 2.99 | 3.61 |
| KIVI | 2.08 | 5.18 | 5.06 | 5.09 |
| MeanPool | 7.06 | 25.72 | 26.38 | 27.09 |
| 500xC. | 1.93 | 5.62 | 5.73 | 5.76 |
| **SideKV** | 6.84 | 24.63 | 25.32 | 26.11 |

At batch size 16, SideKV reaches 26.11 requests/s compared to 5.14 for full context, while baselines processing the full prompt achieve between 2.47 and 5.76 requests/s. For batch sizes $B \geq 4$, full-context throughput plateaus between 5.13 and 5.26 requests/s.

**Scoring a continuation.** Table 13 evaluates latency and peak GPU memory when scoring a 256-token continuation across all thirteen LLMs at block lengths $M \in \{1024, 4096\}$. The forward pass processes the prefix, the block (or $k = 32$ summary slots for SideKV), the local window, and the continuation; SideKV timings include the encoder pass. Latencies are averaged over 20 runs on a single SlimPajama sequence at batch size 1, following 3 warmup passes. Speedup and memory reduction factors are reported relative to full-context processing, with peak memory accounting for LLM parameters.

Table 13: Speedup and peak-memory reduction of SideKV over the full context in one forward pass that scores a 256-token continuation, at two block lengths. Bold marks the mean over the 13 LLMs.

| | block 1024 | | block 4096 | |
|---|---|---|---|---|
| Predictor | speedup | memory | speedup | memory |
| Qwen2.5-0.5B | 3.9× | 1.26× | 31.5× | 4.18× |
| Qwen2.5-7B | 3.7× | 1.05× | 22.9× | 1.51× |
| Qwen2.5-14B | 4.3× | 1.04× | 28.2× | 1.40× |
| Qwen3-8B | 3.8× | 1.06× | 24.7× | 1.57× |
| Llama-3.2-1B | 4.9× | 1.33× | 36.1× | 4.30× |
| Llama-3.2-3B | 3.8× | 1.11× | 26.3× | 2.05× |
| Llama-3.1-8B | 3.8× | 1.06× | 25.7× | 1.58× |
| Gemma-3-1B | 1.5× | 1.23× | 4.7× | 1.93× |
| Gemma-3-4B | 2.0× | 1.06× | 6.8× | 1.29× |
| Gemma-3-12B | 2.3× | 1.03× | 8.4× | 1.25× |
| OLMoE-1B-7B | 1.1× | 1.02× | 2.1× | 1.07× |
| DeepSeek-V2-Lite | 1.4× | 1.02× | 4.6× | 1.18× |
| Mistral-7B | 4.1× | 1.07× | 6.1× | 1.06× |
| mean | **3.1×** | **1.10×** | **17.5×** | **1.87×** |

Speedups increase with block length, averaging 3.1× at $M = 1024$ and 17.5× at $M = 4096$, with a maximum of 36.1× on Llama-3.2-1B. Peak memory reduction shows a similar trend: full context requires 1.10× the memory of SideKV at $M = 1024$ and 1.87× at $M = 4096$ on average, with the largest ratios on Qwen2.5-0.5B (4.18×) and Llama-3.2-1B (4.30×).

# I BROADER LONGBENCH COMPARISON

Section 4.5 evaluates whether the compressed representations remain useful for downstream long-context tasks on LongBench. This appendix broadens that comparison to include eviction, three selection methods, and two four-bit quantization baselines on the same eight LLMs.

Table 14: LongBench score, averaged over sixteen tasks at 50 samples each; higher is better. The last column is the mean over the LLMs, SideKV is shaded, and bold marks the best score within each method category with more than one system.

| | Qwen2.5 | | | Qwen3 | Llama-3.1 | Gemma-3 | | Mistral | |
|---|---|---|---|---|---|---|---|---|---|
| System | 0.5B | 7B | 14B | 8B | 8B | 4B | 12B | 7B | Mean |
| *full context* | 20.1 | 34.9 | 37.6 | 8.1 | 34.6 | 40.2 | 44.8 | 27.2 | 30.9 |
| *Eviction and selection* | | | | | | | | | |
| eviction | 14.3 | 15.2 | 15.9 | 7.1 | 15.7 | 22.6 | 23.5 | **21.9** | 17.0 |
| SnapKV | 13.5 | 16.8 | 18.2 | 7.0 | 19.3 | **25.7** | 27.4 | 20.1 | 18.5 |
| KVzip | 14.1 | 19.3 | 18.1 | 7.0 | 17.2 | 25.0 | **28.5** | 19.6 | 18.6 |
| ReST-KV | **17.4** | **31.2** | **31.1** | **7.4** | **30.0** | 24.7 | 25.0 | 21.3 | **23.5** |
| *Quantization* | | | | | | | | | |
| KIVI | **19.8** | **34.9** | 37.5 | 8.1 | 34.5 | 40.1 | **44.4** | **26.9** | **30.8** |
| TurboQ. | 19.4 | 34.4 | **37.6** | **8.2** | **34.9** | **40.2** | 44.3 | **26.9** | 30.7 |
| *Trained inside the LLM* | | | | | | | | | |
| 500xC. | 10.3 | 16.6 | 15.7 | 8.1 | 14.5 | 12.8 | 17.8 | 22.0 | 14.7 |
| *Summary vectors, LLM frozen* | | | | | | | | | |
| MeanPool | **15.7** | 2.8 | 14.8 | 7.0 | 16.5 | 24.4 | 23.6 | 21.4 | 15.8 |
| uncond. | 15.6 | 14.8 | 15.8 | **7.1** | 16.9 | 22.8 | 23.7 | 21.5 | 17.3 |
| **SideKV** | 14.8 | **17.8** | **16.2** | **7.1** | **22.9** | **24.7** | **25.1** | **22.0** | **18.8** |

Table 14 evaluates the 16 English tasks of LongBench (Bai et al., 2024), covering single- and multi-document question answering, summarization, few-shot learning, synthetic tasks, and code completion, using the first 50 samples of each task. Prompts longer than 6144 tokens are truncated from the middle while retaining the first and last 3072 tokens. The first 32 tokens form the retained prefix and the last 256 tokens the recent window, while the intervening context is divided into 1024-token blocks. SideKV, the unconditional variant, MeanPool, and 500xCompressor replace each block by $k = 32$ summary slots; the selection methods retain 32 tokens per block, eviction discards the blocks, and the quantization baselines retain them at four-bit precision. The learned compressors are trained on SlimPajama and evaluated on LongBench without task-specific fine-tuning. We use

greedy decoding up to the task-specific output length, ranging from 32 to 512 tokens, and evaluate each task using its standard LongBench metric.

Among the frozen-predictor summary methods, SideKV achieves the highest mean LongBench score, 18.8, compared with 17.3 for the unconditional variant and 15.8 for MeanPool. It also matches or exceeds simple eviction on all eight LLMs, and its mean score is higher than those of SnapKV and KVzip, which obtain 18.5 and 18.6, respectively. ReST-KV performs better among the selection methods, with a mean score of 23.5. The four-bit quantization baselines remain closest to full-context performance, but operate with substantially larger effective cache budgets and require additional quantization and dequantization operations.

# J Robustness Across Data and Context Configurations

Having established the main comparison on SlimPajama in Table 9, this appendix examines how robust those results are to changes in the data and context configuration. We vary one component of the main experimental setting at a time. Tables 15 and 16 extend the comparison to alternative corpora, while Tables 17–19 vary the retained window size $w$, the discarded-block length $M$, and the cache budget allocated to eviction and selection methods. Unless stated otherwise, $s$=32, $M$=1024, $w$=256, the remaining settings follow the main configuration, and each entry reports the excess distortion $\Delta D_{XZ}$ relative to the corresponding full-context reference under the same corpus and layout. Implementation and training details follow Appendices A and B.

## J.1 WikiText

Table 15: Comparison on WikiText across 7 LLMs. Lower $\Delta D_{XZ}$ is better. Bold marks the lowest value in each row among methods using at most 320 cache slots, and the final row reports the mean for each method.

| | $\Delta D_{XZ} \downarrow$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Eviction and selection | | | | Quantization | | Summary vectors | | |
| Predictor | evict. | SnapKV | KVzip | ReST-KV | KIVI | TurboQ. | MeanPool | uncond. | SideKV |
| slots | 288 | 320 | 320 | 320 | 544 | 572 | 320 | 320 | 320 |
| Qwen2.5-0.5B | +0.223 | +0.220 | +0.227 | +0.184 | +0.003 | +0.016 | +0.223 | +0.083 | **+0.025** |
| Qwen2.5-7B | +0.190 | +0.102 | +0.105 | +0.078 | +0.008 | +0.032 | +2.388 | +0.144 | **−0.072** |
| Qwen2.5-14B | +0.214 | +0.140 | +0.093 | +0.090 | +0.001 | +0.002 | +0.302 | +0.134 | **+0.019** |
| Llama-3.1-8B | +0.170 | +0.118 | +0.081 | +0.077 | +0.001 | +0.004 | +0.163 | +0.097 | **+0.019** |
| Gemma-3-1B | +0.520 | +0.261 | +0.336 | +0.237 | +0.004 | +0.006 | +0.713 | +0.138 | **−0.150** |
| DeepSeek-V2-Lite | +0.146 | +0.106 | +0.047 | +0.053 | +0.001 | +0.005 | +0.238 | +0.041 | **−0.033** |
| Mistral-7B | +0.165 | +0.121 | +0.069 | +0.074 | +0.001 | +0.002 | +0.167 | +0.132 | **+0.056** |
| mean | +0.233 | +0.153 | +0.137 | +0.113 | +0.003 | +0.010 | +0.599 | +0.110 | **−0.019** |

Table 15 evaluates the compression methods on WikiText across seven LLMs spanning five model families. Among methods using at most 320 cache slots, SideKV achieves the lowest excess distortion on all seven LLMs, with a mean $\Delta D_{XZ}$ of $-0.019$ bits/token, compared with $+0.110$ for the unconditional variant and $+0.113$ for ReST-KV. SideKV also improves over simple eviction on every model, indicating that the advantage of side-information-aware compression observed on SlimPajama extends to WikiText. The four-bit quantization methods achieve lower absolute distortion, but retain substantially larger effective cache budgets and introduce additional quantization and dequantization overhead.

### J.2 THREE FURTHER CORPORA

Table 16: Comparison across PG19, five public-domain novels, and Python source code for seven LLMs under the default configuration. Lower $\Delta D_{XZ}$ is better. Bold marks the lowest value of each method group in each row.

| | | $\Delta D_{XZ} \downarrow$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Eviction and selection | | | | Quantization | | Summary vectors | | |
| Setting | predictor | eviction | SnapKV | KVzip | ReST-KV | KIVI | TurboQ. | MeanPool | uncond. | SideKV |
| PG19 | Qwen2.5-0.5B | +0.116 | **+0.089** | +0.102 | **+0.089** | **+0.002** | +0.012 | +0.129 | +0.100 | **+0.092** |
| | Qwen2.5-7B | −0.022 | −0.025 | −0.002 | **−0.030** | **+0.012** | +0.041 | +2.955 | −0.015 | **−0.105** |
| | Qwen2.5-14B | +0.122 | **+0.048** | +0.063 | +0.049 | **+0.001** | **+0.001** | +0.208 | +0.119 | **+0.109** |
| | Llama-3.1-8B | +0.127 | **+0.041** | +0.056 | **+0.041** | **+0.001** | +0.004 | +0.130 | +0.085 | **+0.051** |
| | Gemma-3-1B | +0.483 | **+0.139** | +0.272 | **+0.139** | **+0.003** | +0.006 | +0.665 | +0.086 | **−0.147** |
| | DeepSeek-V2-Lite | −0.006 | **−0.055** | −0.043 | **−0.055** | **+0.002** | +0.003 | +0.146 | −0.084 | **−0.133** |
| | Mistral-7B | +0.093 | +0.044 | +0.044 | **+0.040** | **+0.001** | **+0.001** | +0.093 | +0.099 | **+0.089** |
| novels | Qwen2.5-0.5B | +0.128 | +0.125 | +0.121 | **+0.103** | **+0.002** | +0.013 | +0.146 | +0.106 | **+0.093** |
| | Qwen2.5-7B | +0.038 | −0.001 | +0.023 | **−0.012** | **+0.011** | +0.045 | +2.984 | +0.047 | **−0.070** |
| | Qwen2.5-14B | +0.142 | +0.082 | +0.069 | **+0.056** | **+0.001** | +0.002 | +0.219 | +0.127 | **+0.104** |
| | Llama-3.1-8B | +0.149 | +0.073 | +0.058 | **+0.050** | **+0.001** | +0.006 | +0.158 | +0.057 | **+0.016** |
| | Gemma-3-1B | +0.374 | +0.156 | +0.224 | **+0.138** | **+0.003** | +0.005 | +0.582 | −0.217 | **−0.455** |
| | DeepSeek-V2-Lite | −0.020 | −0.009 | −0.036 | **−0.045** | **+0.002** | +0.004 | +0.212 | −0.163 | **−0.215** |
| | Mistral-7B | +0.127 | +0.071 | +0.053 | **+0.048** | **+0.001** | **+0.001** | +0.131 | +0.126 | **+0.107** |
| Python | Qwen2.5-0.5B | +0.245 | +0.221 | +0.221 | **+0.213** | **+0.003** | +0.010 | +0.246 | +0.243 | **+0.233** |
| | Qwen2.5-7B | +0.160 | +0.116 | +0.120 | **+0.106** | **+0.005** | +0.016 | +1.366 | +0.161 | **+0.099** |
| | Qwen2.5-14B | +0.158 | +0.104 | **+0.101** | +0.103 | **+0.000** | +0.001 | +0.166 | +0.158 | **+0.138** |
| | Llama-3.1-8B | +0.193 | +0.126 | **+0.119** | +0.128 | **+0.001** | **+0.001** | +0.199 | +0.183 | **+0.169** |
| | Gemma-3-1B | +0.461 | +0.377 | +0.415 | **+0.351** | +0.004 | **+0.003** | +0.488 | +0.364 | **+0.130** |
| | DeepSeek-V2-Lite | +0.157 | +0.101 | **+0.085** | +0.100 | **+0.002** | +0.003 | +0.183 | +0.130 | **+0.095** |
| | Mistral-7B | +0.214 | +0.142 | **+0.124** | +0.134 | **+0.001** | **+0.001** | +0.213 | +0.215 | **+0.205** |

We extend the comparison to PG19 (Rae et al., 2020), five public-domain novels, and Python source code using the same seven LLMs and the default configuration. Table 16 reports the resulting 21 LLM–corpus settings. SideKV achieves lower excess distortion than both simple eviction and the unconditional variant in all 21 settings, and produces the lowest $\Delta D_{XZ}$ among methods using at most 320 cache slots in 10 of them. Selection-based methods remain competitive in several settings: ReST-KV or KVzip gives the lowest distortion particularly on Python source code and for the larger Qwen2.5-14B and Mistral-7B models.

### J.3 RETAINED WINDOWS

Table 17: Comparison across five retained-window sizes $w \in \{32, 64, 128, 256, 512\}$ for Qwen2.5-0.5B, Qwen2.5-7B, and Qwen2.5-14B. Each entry reports $\Delta D_{XZ}$ relative to the full-context reference with the same window size; lower is better. Bold indicates the lowest distortion among methods using at most 320 cache slots.

| | | $\Delta D_{XZ} \downarrow$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Eviction and selection | | | | Quantization | | Summary vectors | | |
| Setting | predictor | eviction | SnapKV | KVzip | ReST-KV | KIVI | TurboQ. | MeanPool | uncond. | SideKV |
| $w$=32 | 0.5B | +0.362 | +0.247 | +0.274 | **+0.233** | **+0.005** | +0.025 | +0.379 | +0.358 | **+0.329** |
| | 7B | +0.282 | +0.137 | +0.164 | **+0.116** | **+0.009** | +0.032 | +4.587 | +0.288 | **+0.164** |
| | 14B | +0.355 | +0.152 | +0.157 | **+0.141** | **+0.001** | +0.002 | +0.472 | +0.352 | **+0.305** |
| $w$=64 | 0.5B | +0.274 | +0.180 | +0.209 | **+0.173** | **+0.003** | +0.017 | +0.290 | +0.272 | **+0.244** |
| | 7B | +0.188 | +0.078 | +0.107 | **+0.062** | **+0.011** | +0.035 | +4.018 | +0.198 | **+0.063** |
| | 14B | +0.273 | +0.107 | +0.115 | **+0.102** | **+0.001** | **+0.001** | +0.366 | +0.273 | **+0.223** |
| $w$=128 | 0.5B | +0.216 | +0.156 | +0.176 | **+0.152** | **+0.004** | +0.015 | +0.230 | +0.215 | **+0.195** |
| | 7B | +0.117 | +0.050 | +0.074 | **+0.039** | **+0.009** | +0.034 | +2.984 | +0.128 | **+0.013** |
| | 14B | +0.209 | +0.095 | +0.106 | **+0.092** | **+0.001** | **+0.001** | +0.281 | +0.212 | **+0.177** |
| $w$=256 | 0.5B | +0.125 | +0.097 | +0.105 | **+0.092** | **+0.002** | +0.012 | +0.136 | +0.125 | **+0.109** |
| | 7B | +0.016 | −0.000 | +0.016 | **−0.007** | **+0.009** | +0.034 | +2.274 | +0.028 | **−0.078** |
| | 14B | +0.116 | **+0.052** | +0.061 | +0.054 | **+0.001** | **+0.001** | +0.167 | +0.119 | **+0.097** |
| $w$=512 | 0.5B | +0.058 | **+0.044** | +0.049 | +0.046 | **+0.003** | +0.010 | +0.067 | +0.059 | **+0.045** |
| | 7B | **−0.020** | −0.005 | −0.001 | −0.009 | **+0.005** | +0.028 | +1.700 | −0.009 | **−0.115** |
| | 14B | +0.060 | **+0.028** | +0.030 | +0.029 | **+0.000** | +0.001 | +0.100 | +0.063 | **+0.045** |

The retained window $W$ controls how much recent context remains directly available to the predictor and therefore changes the amount of side information available when the historical block is

compressed. Table 17 examines whether the advantage of SideKV persists as this retained context is varied, with each result measured relative to the full-context reference using the same window size. SideKV achieves lower $\Delta D_{XZ}$ than both simple eviction and the unconditional variant in all fifteen settings. Among eviction, selection, and frozen-predictor summary methods, it attains the lowest distortion on Qwen2.5-7B at $w \in \{128, 256, 512\}$, and the second-lowest at $w = 64$ as well as on Qwen2.5-0.5B at $w = 512$. ReST-KV is the strongest selection baseline overall, giving the lowest distortion in nine of the fifteen settings, six of which occur at the shorter windows $w \leq 64$.

### J.4 Longer blocks

Table 18: Comparison across longer discarded-block lengths $M$ on Qwen2.5. Lower $\Delta D_{XZ}$ is better.

| | | $\Delta D_{XZ} \downarrow$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Eviction and selection | | | | Quantization | | Summary vectors | | |
| Setting | predictor | eviction | SnapKV | KVzip | ReST-KV | KIVI | TurboQ. | MeanPool | uncond. | SideKV |
| $M$=2048 | Qwen2.5-0.5B | +0.164 | **+0.133** | +0.151 | **+0.133** | **+0.004** | +0.023 | +0.174 | +0.163 | **+0.145** |
| | Qwen2.5-14B | +0.156 | **+0.079** | +0.096 | +0.081 | **+0.001** | **+0.001** | +0.251 | +0.160 | **+0.134** |
| | Qwen2.5-7B | +0.134 | **+0.090** | +0.107 | +0.091 | **+0.006** | +0.019 | +3.270 | +0.144 | **+0.047** |
| $M$=4096 | Qwen2.5-0.5B | +0.688 | **+0.143** | +0.167 | +0.144 | **+0.006** | +0.060 | +0.700 | +0.688 | **+0.669** |
| | Qwen2.5-14B | +0.158 | **+0.090** | – | **+0.090** | **+0.001** | +0.002 | +0.348 | +0.161 | **+0.131** |
| | Qwen2.5-7B | +0.125 | **+0.073** | +0.102 | +0.076 | **+0.009** | +0.037 | +3.727 | +0.138 | **+0.021** |
| $M$=8192 | Qwen2.5-7B | +1.004 | **+0.074** | +0.108 | **+0.074** | **+0.012** | +0.065 | +4.871 | +1.013 | **+0.906** |

Eliminating the need for retraining, we evaluate the adaptability of SideKV to extended contexts by applying encoders trained at $M = 1024$ to longer discarded blocks $M \in \{2048, 4096, 8192\}$. Table 18 summarizes the resulting comparison. The prefix, retained window, and cache budgets otherwise follow the main configuration, and each result is measured relative to the corresponding full-context reference. SideKV achieves lower excess distortion than both simple eviction and the unconditional variant at every tested block length. On Qwen2.5-7B, it also gives the lowest $\Delta D_{XZ}$ among methods using at most 320 cache slots at $M = 2048$ and $M = 4096$. At $M = 8192$, however, the distortion of SideKV increases substantially, while the selection methods remain more robust, indicating that generalization to much longer discarded blocks remains challenging without retraining.

### J.5 Controlling for Cache Budget

Table 19: Comparison with eviction under matched and enlarged cache budgets. Lower $\Delta D_{XZ}$ is better; bold marks the lowest value in each row before rounding.

| | Eviction and selection | | | | | | Summary vectors |
|---|---|---|---|---|---|---|---|
| Predictor | evict. 288 | evict. 320 | evict. 352 | SnapKV 352 | KVzip 352 | ReST-KV 352 | SideKV 320 |
| Qwen2.5-0.5B | +0.125 | +0.132 | +0.104 | +0.102 | +0.115 | **+0.102** | +0.109 |
| Qwen2.5-7B | +0.016 | +0.026 | +0.005 | +0.011 | +0.023 | +0.002 | **−0.078** |
| Qwen2.5-14B | +0.116 | +0.121 | +0.100 | **+0.059** | +0.067 | +0.059 | +0.097 |
| Qwen3-8B | +0.112 | +0.120 | +0.098 | +0.072 | +0.094 | +0.067 | **−0.045** |
| Llama-3.2-1B | +0.120 | +0.114 | +0.107 | +0.088 | +0.105 | +0.091 | **+0.042** |
| Llama-3.2-3B | +0.121 | +0.119 | +0.113 | +0.070 | +0.081 | +0.074 | **+0.032** |
| Llama-3.1-8B | +0.110 | +0.107 | +0.100 | **+0.050** | +0.059 | +0.051 | +0.068 |
| Gemma-3-1B | +0.450 | +0.403 | +0.328 | +0.159 | +0.248 | +0.153 | **−0.031** |
| Gemma-3-4B | +0.838 | +0.808 | +0.626 | +0.121 | +0.288 | +0.116 | **+0.111** |
| Gemma-3-12B | +2.740 | +2.393 | +2.378 | +0.097 | +0.489 | **+0.092** | +0.145 |
| OLMoE-1B-7B | +0.130 | +0.119 | +0.102 | **+0.079** | +0.107 | +0.082 | +0.117 |
| DeepSeek-V2-Lite | +0.041 | +0.022 | +0.028 | −0.008 | −0.007 | −0.008 | **−0.038** |
| Mistral-7B | +0.125 | +0.105 | +0.110 | **+0.049** | +0.052 | +0.050 | +0.118 |

In the main configuration, simple eviction retains 288 cache slots (32 prefix tokens and a window of $W = 256$), whereas SideKV uses 320 slots after adding $k = 32$ summary vectors. To verify that the improvement of SideKV is not simply due to this larger cache budget, Table 19 increases the retained window for eviction so that it uses either the same number of slots as SideKV (320) or a larger budget (352). We also evaluate the selection baselines at 352 slots.

At the matched 320-slot budget, SideKV achieves lower $\Delta D_{XZ}$ than eviction on 12 of the 13 LLMs. Even when eviction is given 352 slots, SideKV remains lower on 10 of the 13 models. These results

indicate that the improvement over simple eviction cannot be explained solely by the additional 32 cache slots used by the SideKV summaries.

## K Reuse of One Compressed Representation

A core requirement for compressed representations is computational reuse across subsequent generations. Accordingly, we evaluate the temporal longevity of a single discarded block's representation as the prediction distance increases. Following the main configuration, the block is compressed once and then reused to score eight successive 128-token continuations, spanning 1024 tokens beyond the retained window $W$. Each continuation is evaluated independently at its corresponding sequence position using the same fixed compressed cache; earlier continuations are not added back into the context. Selection methods and quantization baselines are evaluated through their prefilled caches against the corresponding full-context reference.

Table 20: Reuse of a single compressed representation over eight successive 128-token continuations. Values are log-loss differences from the corresponding full-context reference, averaged over 13 LLMs and 96 held-out spans. Drift is the difference between the eighth and first continuations; lower is better.

| System | mean log-loss ↓ | drift ↓ |
|---|---|---|
| eviction | +0.392 | −0.033 |
| SnapKV | +0.093 | −0.041 |
| KVzip | +0.164 | +0.018 |
| ReST-KV | +0.071 | −0.023 |
| MeanPool | +0.625 | −0.037 |
| uncond. | +0.325 | −0.119 |
| **SideKV** | **+0.061** | **−0.251** |

Table 20 summarizes two quantities averaged over all 13 LLMs: the mean excess log-loss across the eight continuations and the drift, defined as $\Delta D_8 - \Delta D_1$. A negative drift means that the gap to the full-context reference becomes smaller at later continuation positions. Among eviction, selection, and summary-vector methods, SideKV achieves the lowest mean excess log-loss at $+0.061$ bits/token, compared with $+0.071$ for ReST-KV and $+0.093$ for SnapKV. It also exhibits the most negative drift, $-0.251$, indicating that its relative gap to the full-context reference does not increase as the same compressed representation is reused further from the original block.

Table 21: Mean $\Delta D_{XZ}$ over eight successive continuations across the 13 LLMs. Lower is better. Bold marks the lowest value among the ranked budgeted methods. "uncond." denotes SideKV with an unconditional prior.

| Predictor | eviction | SnapKV | KVzip | ReST-KV | KIVI | TurboQ. | MeanPool | uncond. | SideKV |
|---|---|---|---|---|---|---|---|---|---|
| Qwen2.5-0.5B | +0.080 | +0.072 | +0.071 | +0.063 | +0.001 | – | +0.078 | +0.063 | **−0.004** |
| Qwen2.5-7B | +0.063 | −0.040 | −0.017 | −0.025 | +0.007 | – | +2.841 | +0.075 | **−0.068** |
| Qwen2.5-14B | +0.055 | +0.062 | +0.047 | +0.042 | +0.000 | +0.001 | +0.060 | +0.056 | **−0.091** |
| Qwen3-8B | +0.043 | +0.083 | +0.064 | +0.042 | +0.000 | – | +0.043 | −0.050 | **−0.205** |
| Llama-3.2-1B | +0.086 | +0.109 | +0.099 | +0.090 | +0.002 | +0.011 | +0.132 | +0.002 | **−0.087** |
| Llama-3.2-3B | +0.088 | +0.092 | +0.068 | +0.075 | +0.001 | +0.003 | +0.102 | −0.012 | **−0.102** |
| Llama-3.1-8B | +0.064 | +0.075 | +0.055 | +0.056 | +0.001 | +0.003 | +0.071 | −0.027 | **−0.098** |
| Gemma-3-1B | +0.511 | +0.196 | +0.328 | +0.176 | +0.002 | +0.008 | +0.624 | +0.224 | **−0.047** |
| Gemma-3-4B | +1.034 | +0.179 | +0.423 | +0.131 | +0.002 | +0.001 | +1.079 | +0.620 | **+0.053** |
| Gemma-3-12B | +2.912 | +0.205 | +0.774 | +0.141 | +0.000 | −0.001 | +2.911 | +3.358 | **+1.701** |
| OLMoE-1B-7B | +0.083 | +0.104 | +0.171 | +0.098 | +0.000 | +0.001 | +0.079 | +0.011 | **−0.022** |
| DeepSeek-V2-Lite | −0.008 | −0.017 | −0.021 | −0.036 | +0.002 | +0.019 | +0.028 | −0.134 | **−0.219** |
| Mistral-7B | +0.089 | +0.088 | +0.067 | +0.064 | +0.000 | +0.001 | +0.081 | +0.041 | **−0.019** |

Table 21 reports the mean $\Delta D_{XZ}$ over the eight continuations on each of the 13 LLMs. SideKV has the lowest value among the ranked budgeted methods on all 13, and its mean is negative on 11 of them, all but Gemma-3-4B and Gemma-3-12B. On the Gemma-3 models, the methods scored through the prefilled cache (Appendix A) are not ranked, since eviction there loses less through the prefilled cache than with a re-encoded prefix. A dash marks TurboQuant on Qwen2.5-0.5B, Qwen2.5-7B and Qwen3-8B, where it diverges in this study.