# Beyond: Better-than-Full-Precision KV Caches via Learnable Non-Uniform Quantization as Implicit Regularization

Anonymous ACL submission

## Abstract

Key–value (KV) caching enables efficient autoregressive decoding, but long-context serving is dominated by KV-cache memory. Existing KV-cache quantizers are typically calibrated with local reconstruction or distortion losses, even though generation quality depends on how quantization perturbations propagate through subsequent model computation. We propose **Beyond**, a learnable non-uniform KV quantizer that trains representation levels and decision thresholds directly on language-model loss while keeping the deployment hard quantizer in the forward pass. The backbone remains frozen; optimization uses standard straight-through estimator input gradients and hard-forward boundary pseudo-gradients for thresholds. On Ministral-3-14B-Instruct-2512-BF16, Llama-3.1-8B-Instruct, and Qwen3-30B-A3B-Instruct-2507, 4-bit/group-32 **Beyond** reduces evaluation Pile perplexity relative to full-precision PPL and matches or improves upon full precision across seven downstream benchmarks. On a B300 GPU, our packed-cache decode path serves the learned cache directly and achieves up to 1.43× single-token decode speedup over FlashAttention-4.

## 1 Introduction

In modern decoder-only LLM families such as Qwen3 and Llama 3 (Yang et al., 2025; Grattafiori et al., 2024), key–value (KV) caching is the standard mechanism for efficient autoregressive decoding: past keys and values are stored so each step computes only the new token's states. At long context, this creates a KV-memory bottleneck because cache size scales linearly with sequence length.

KV-memory reduction spans low-rank projection (Chang et al., 2025; Saxena et al., 2024), token eviction/merging (Zhang et al., 2023, 2024b), and quantization (Liu et al., 2024b; Hooper et al., 2024). Low-rank methods reduce the representational dimension of each cached state, and token-level methods can discard, merge, or blur individual context tokens, while quantization preserves the full token history and original KV dimensionality.

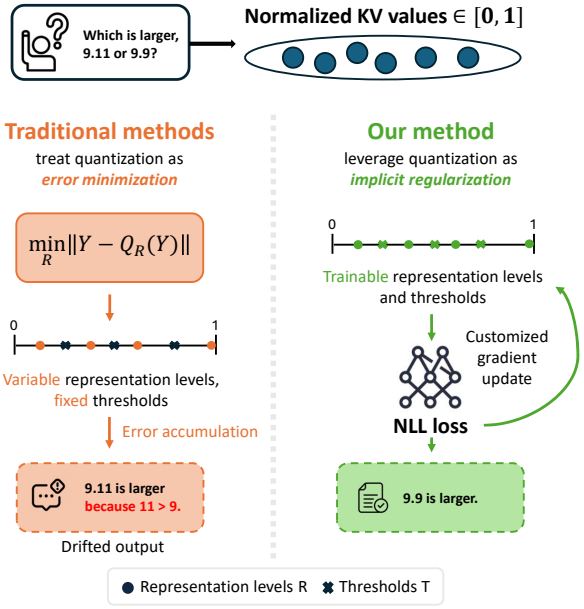


Figure 1: **From reconstruction-error minimization to loss-aware KV-cache quantization.** Conventional KV quantization fits local reconstruction objectives and then freezes the quantizer, while **Beyond** keeps the deployment-time hard quantizer but optimizes representation levels **R** and decision thresholds **T** against the language-model loss.

Most KV quantizers optimize local distortion. Uniform schemes are efficient but use fixed bins that can mismatch heterogeneous KV distributions (Liu et al., 2024b), while non-uniform and vector-compression methods improve the fit with calibrated datasets or online distortion objectives (Hooper et al., 2024; Zandieh et al., 2026). However, local distortion is only a proxy for generation loss: KV perturbations propagate through the subsequent computation before affecting predictions (Fig. 1).

Our token-level diagnostics show that even uniform low-bit KV quantization can lower perplexity

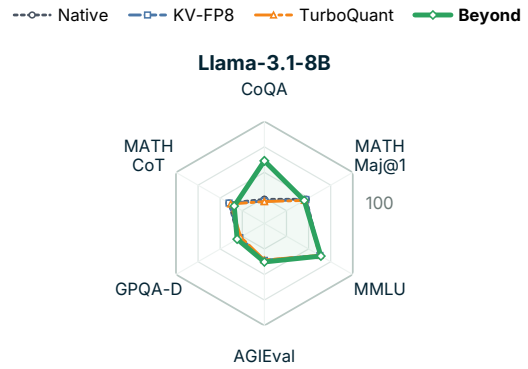


Figure 2: **Llama-3.1-8B-Instruct Six-Benchmark Comparison.** Scores on the six benchmark tasks. Higher is better.

on some positions, suggesting that discretization is not purely destructive. This motivates treating KV quantization as a constrained adaptation problem: ***can a compressed, hard-forward KV cache be trained to reduce language-model loss?*** We propose **Beyond**, which learns non-uniform levels and thresholds with the backbone frozen. The forward pass is the exact hard quantizer used at inference, and the backward pass uses STE-style pseudo-gradients for representation levels and thresholds, turning quantization from a fixed calibration artifact into a lightweight loss-aware adaptation.

Our contributions are:

- **Loss-aware quantization principle.** We show that reconstruction error is a degenerate local objective: downstream loss also depends on perturbation mean and anisotropic curvature.
- **Deployment-exact learning. Beyond** learns non-uniform levels and thresholds with a hard forward pass that matches inference and STE-style backward updates.
- **Custom decode kernel.** We implement a customized FlashAttention-style decoding kernel that consumes the learned packed KV cache directly for compressed-cache inference.

## 2 Preliminaries and Related Works

This section reviews the background needed for KV-cache quantization and related memory-reduction techniques.

### 2.1 Preliminaries

We first review KV caching and then define standard scalar quantization for KV states.

**KV cache for multi-head attention.** In decoder-only LLMs, e.g., LLaMA (Touvron et al., 2023), key/value (KV) caching accelerates autoregressive inference by storing the keys and values produced for previous tokens. Given a prefix $X = [x_1, \ldots, x_t]$, for each layer and attention head $i$, the cache before processing $x_t$ contains $\mathbf{K}_i^{t-1}$ and $\mathbf{V}_i^{t-1}$ for $x_{1:t-1}$.

To process token $x_t$, the model computes the query, key, and value vectors from its hidden state $\mathbf{h}_t$ (we omit the layer index and positional encodings for notational simplicity):

$$\mathbf{q}_i^t = \mathbf{h}_t \mathbf{W}_{Q,i}, \mathbf{k}_i^t = \mathbf{h}_t \mathbf{W}_{K,i}, \mathbf{v}_i^t = \mathbf{h}_t \mathbf{W}_{V,i},$$

where $\mathbf{W}_{Q,i}$, $\mathbf{W}_{K,i}$, and $\mathbf{W}_{V,i}$ are the head-specific projection matrices. In practical LLMs, positional encodings (Vaswani et al., 2017), including RoPE (Su et al., 2024), are applied to the query/key states before the attention scores are formed; they are omitted here only to keep the notation focused on KV caching. The new key and value are appended to the cache, $\mathbf{K}_i^t = \mathrm{Concat}(\mathbf{K}_i^{t-1}, \mathbf{k}_i^t)$ and $\mathbf{V}_i^t = \mathrm{Concat}(\mathbf{V}_i^{t-1}, \mathbf{v}_i^t)$, after which attention is

$$\mathrm{Attn}(\mathbf{q}_i^t, \mathbf{K}_i^t, \mathbf{V}_i^t) = \mathrm{Softmax}\Big(\frac{\mathbf{q}_i^t (\mathbf{K}_i^t)^\top}{\sqrt{d}}\Big)\mathbf{V}_i^t,$$

where $d$ is the per-head hidden dimension shared by queries and keys, and the softmax is applied over cached positions. KV caching therefore avoids recomputing keys and values for the full prefix at every decoding step.

The memory footprint of KV cache grows linearly with the sequence length $S = |X|$, as shown in Formula 1:

$$\mathrm{size}(\mathbf{K}, \mathbf{V}) = 2 \times B \times S \times L \times h \times d \times b_{\mathrm{dtype}}, \quad (1)$$

where $\mathbf{K}$ and $\mathbf{V}$ denote the KV cache across $h$ heads, $B$ is the batch size, $L$ is the number of layers, and $b_{\mathrm{dtype}}$ is the number of bytes per stored element in memory. While providing critical performance gains, the additional memory footprint of KV cache can quickly saturate the limited device memory and become the most stringent bottleneck for LLM inference.

**Hard scalar KV quantization.** For a cached scalar $z$ in a quantization group, scalar quantization commonly begins with groupwise affine normalization:

$$x = \mathrm{clip}\left(\frac{z - z_{\min}}{z_{\max} - z_{\min} + \epsilon}, 0, 1\right).$$

The hard $b$-bit scalar quantizer has $K = 2^b$ bins, representation values $R = \{r_k\}_{k=0}^{K-1}$, and ordered thresholds $T = \{t_k\}_{k=0}^{K}$ with fixed endpoints $t_0 = 0$ and $t_K = 1$. Let $B_k(T) = [t_k, t_{k+1})$ for $k < K-1$ and $B_{K-1}(T) = [t_{K-1}, t_K]$. The normalized hard quantizer is

$$Q(x; R, T) = \sum_{k=0}^{K-1} r_k \, \mathbf{1}\{x \in B_k(T)\}. \quad (2)$$

The corresponding dequantized value is

$$\hat{z} = z_{\min} + (z_{\max} - z_{\min})\, Q(x; R, T).$$

Uniform quantization is the special case where the representation values are evenly spaced:

$$r_k = \frac{k}{K-1}, \qquad k = 0, \ldots, K-1, \quad (3)$$

$$t_k = \frac{r_{k-1} + r_k}{2}, \qquad k = 1, \ldots, K-1. \quad (4)$$

By contrast, non-uniform quantization allows unequal spacings in $R$ and $T$. Prior non-uniform KV quantization methods often choose these parameters through clustering (Hooper et al., 2024).

### 2.2 Related Works

**KV-cache quantization.** KV-cache quantization reduces memory usage by compressing cached keys and values to lower bit-widths. KIVI (Liu et al., 2024b) uses asymmetric uniform quantization, applying per-channel quantization to keys and per-token quantization to values. KVQuant (Hooper et al., 2024) fits per-layer non-uniform quantizers with k-means and a Fisher-weighted objective, while CQQuant (Zhang et al., 2024a) jointly quantizes coupled key/value channels to exploit inter-channel dependence at very low bit-widths. NSNQuant (Son et al., 2025) normalizes KV activations toward a standard normal prior and applies low-bit vector quantization with a reusable codebook. GEAR (Kang et al., 2024) instead augments quantized KV states with residual corrections. TurboQuant (Zandieh et al., 2026) applies an online vector-compression scheme based on randomized rotations, scalar quantization of rotated coordinates, and Quantized JL residual correction. These methods mainly optimize reconstruction, distortion, or calibration criteria, whereas **Beyond** optimizes quantizer parameters against the downstream language-model loss.

**Other KV-cache reduction strategies.** Low-rank methods such as Palu (Chang et al., 2025), Eigen Attention (Saxena et al., 2024), and ASVD's KV-cache extension (Yuan et al., 2025) reduce KV channel dimensions. Token eviction methods, including Heavy-Hitter Oracle (Zhang et al., 2023), PyramidKV (Cai et al., 2025), SnapKV (Li et al., 2024), and StreamingLLM (Xiao et al., 2024), shrink the cache by discarding tokens, while cache merging or fusion methods such as CaM (Zhang et al., 2024b), Adaptive KV Merging (Wang et al., 2024), MiniCache (Liu et al., 2024a), and ZS-Merge (Liu et al., 2025) combine redundant cache states. These approaches are largely orthogonal to quantization, but they may remove or blur information that later tokens require.

**Learnable quantization via STE.** Straight-through estimators replace the zero or undefined derivative of discrete operators with surrogate gradients during backpropagation (Bengio et al., 2013). In quantization-aware training, this allows the forward pass to keep a hard quantizer while selected quantizer parameters are learned from task loss. LSQ learns quantizer step sizes for weights and activations (Esser et al., 2020), and PACT learns activation clipping bounds (Choi et al., 2018). DSQ relaxes quantization with differentiable soft assignments that are annealed toward hard decisions (Gong et al., 2019). N2UQ learns non-uniform input thresholds while keeping uniform output levels through a generalized STE for threshold parameters (Liu et al., 2022). In LLM KV-cache activation quantization, however, end-to-end learning that jointly optimizes representation levels and decision thresholds with downstream language-model loss remains limited.

## 3 Methodology

**Beyond** treats KV-cache quantization as loss-aware hard-forward non-uniform quantization: it learns representation levels $R$ and thresholds $T$ by minimizing next-token NLL under a hard quantized forward pass. At inference, the learned quantizers are used in a real-compression path where optimized CUDA kernels execute attention directly over the packed KV cache, keeping the training and inference quantization semantics matched.

### 3.1 Observations and Motivations

Figures 3 and 4 provide the Llama-3.1-8B-Instruct (Grattafiori et al., 2024) diagnostics used to motivate this section: token-level PPL changes under low-bit uniform quantization on a The Pile (Gao

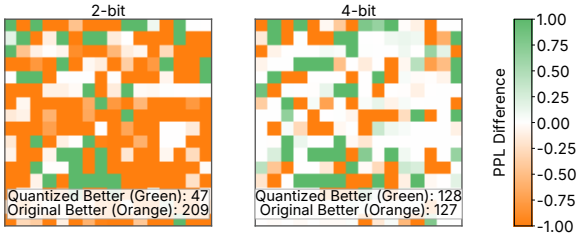


Figure 3: **Token-level PPL differences under uniform KV quantization on a chat-wrapped The Pile sample for Llama-3.1-8B-Instruct, with 2-bit (left) and 4-bit (right).** Heatmaps show $\Delta\text{PPL} = \text{PPL}_{\textit{float16}} - \text{PPL}_{\text{quant}}$ over 256 supervised target tokens. Quantization uses group-32 per-token normalization over the head dimension for both K and V. Green indicates lower token PPL after quantization and orange indicates degradation; a non-trivial subset of tokens improves over the full-precision baseline, especially at 4-bit.

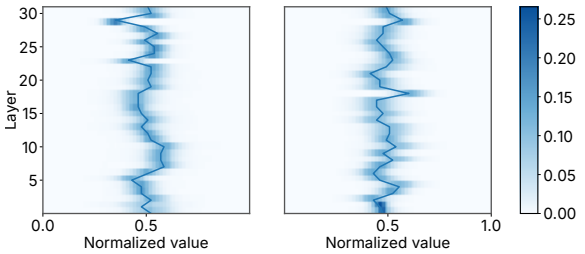


Figure 4: **Per-layer heatmaps of min–max normalized post-RoPE key (Left) and post-projection value (Right) activations in Llama-3.1-8B-Instruct for the same The Pile chat-wrapped sample.** Color indicates empirical density over normalized activation values, and the overlaid curve traces the modal bin for each layer. The concentrated mass, layer-dependent shifts, and K/V differences show why fixed uniform levels can mismatch the activation geometry.

et al., 2020) chat-prefix sample, and layer-wise KV activation heterogeneity. In Fig. 3, uniform 2-bit quantization lowers token PPL on 47 of 256 scored positions (18.4%), while 4-bit quantization improves 128 of 256 positions (50.0%). Thus the quantization perturbation is signed and token-dependent rather than uniformly harmful. Fig. 4 further shows that min–max normalized K/V activations remain concentrated but shift across layers and differ between keys and values. This agrees with prior observations that KV activations place most mass in a narrow range while a small outlier fraction reaches much larger magnitudes (Hooper et al., 2024), and that atypical tokens can disproportionately affect quantization fidelity (Su et al., 2025). The resulting geometry favors layer-aware non-uniform levels over fixed uniform levels; fixed uniform schemes such as KIVI (Liu et al., 2024b) cannot adapt representation density to these shifts.

These observations motivate a loss-aware view of KV quantization. Existing KV-cache compressors fit fixed local surrogates, including sensitivity- or Fisher-weighted fitting, residual correction, normalized codebooks, and online distortion criteria (Hooper et al., 2024; Zhang et al., 2024a; Kang et al., 2024; Son et al., 2025; Zandieh et al., 2026). Downstream NLL can rank these perturbations differently: equally well-calibrated perturbations can align differently with the local loss gradient and curvature. The expansion below formalizes this gap and identifies when it collapses back to reconstruction-style fitting.

**Proposition 3.1** (Local loss-aware quantization principle)**.** *Let $\Psi(z)$ be the token loss induced by cached representation $z$ in a frozen downstream computation, and let a quantizer with parameters $\varphi$ induce perturbation $\delta_\varphi = Q_\varphi(z) - z$. If $\Psi$ has $\beta$-Lipschitz Hessian in operator norm in a neighborhood of $z$, with $g = \nabla\Psi(z)$ and $H = \nabla^2\Psi(z)$, then*

$$\begin{aligned}\mathbb{E}\Psi(z+\delta_\varphi) - \Psi(z) &= \langle g, m_\varphi\rangle \\ &\quad + \tfrac{1}{2}\mathrm{Tr}(HC_\varphi) + \rho_\varphi, \\ |\rho_\varphi| &\le \tfrac{\beta}{6}\mathbb{E}\|\delta_\varphi\|_2^3,\end{aligned}$$

*where $m_\varphi = \mathbb{E}[\delta_\varphi]$ and $C_\varphi = \mathbb{E}[\delta_\varphi\delta_\varphi^\top]$. Thus local loss-aware fitting depends on gradient alignment $\langle g, m_\varphi\rangle$ and Hessian-weighted second moment $\mathrm{Tr}(HC_\varphi)$, not only a calibration distortion. Fisher weighting follows only after dropping the first-order term and substituting a fixed calibration weight for $H$; plain reconstruction further requires isotropic $H$.*

*Proof.* See Appendix F.1.

For KV caches, the perturbation enters through attention. A threshold shift that slightly increases local reconstruction error can still reduce negative log-likelihood if it suppresses a competing continuation preferred by the full-precision model. We therefore treat KV quantization as a loss-aware perturbation rather than as pure numerical error, and instantiate the principle through a local attention decomposition (Propositions 3.2–3.3).

**Proposition 3.2** (Loss-aware perturbation view of KV quantization)**.** *Under the first-order model in Appendix F.1, let $\Delta o = (J_{\mathrm{sm}}(a)\varepsilon)V + p\,\Delta V$ denote the linearized attention-output perturbation and define $o_{\mathrm{lin}} = o + \Delta o$. Let $\mu = \mathbb{E}[\Delta o]$. If the downstream loss $\ell(\cdot)$ is $L$-smooth, then*

$$\mathbb{E}\big[\ell(o_{\mathrm{lin}})\big] \;\le\; \ell(o) \;+\; \big\langle \nabla\ell(o), \mu\big\rangle + \frac{L}{2}\,\mathbb{E}\big\|\Delta o\big\|_2^2,$$

*where* $a = \frac{1}{\sqrt{d}} qK^\top,\ p = \mathrm{Softmax}(a),\ o = pV$. *If* $\langle \nabla \ell(o), \mu \rangle < -\frac{L}{2}\mathbb{E}\|\Delta o\|_2^2$, *the quantized linearized output has strictly lower expected loss than the full-precision output.*

*Proof.* See Appendix F.1.

**Proposition 3.3** (Competing-path margin suppression)**.** *Let* $s_j(o_t) = w_j^\top o_t$ *and* $\pi_t = \mathrm{Softmax}(Wo_t)$. *For a competing token* $j \neq y_t$, *let* $\Delta m_t(j)$ *be the change in its logit margin over the target token after perturbing* $o_t$ *by* $\mu_t$*:*

$$\begin{aligned}\Delta m_t(j) \triangleq &\big[s_j(o_t+\mu_t) - s_{y_t}(o_t+\mu_t)\big] \\ &- \big[s_j(o_t) - s_{y_t}(o_t)\big] \\ &= \big\langle w_j - w_{y_t}, \mu_t \big\rangle.\end{aligned}$$

*The cross-entropy first-order term is the softmax-weighted average of these one-token shifts:*

$$\big\langle \nabla_{o_t} \ell_t(o_t), \mu_t \big\rangle = \sum_{j \neq y_t} \pi_{t,j} \Delta m_t(j).$$

*For a competing continuation* $c_{1:T}$, *use the same teacher-forced states* $o_t$ *from the target prefixes* $y_{<t}$, *and compare the path margins before and after perturbation:*

$$\begin{aligned}M(c,y) &= \sum_t \big(s_{c_t}(o_t) - s_{y_t}(o_t)\big), \\ M_\mu(c,y) &= \sum_t \big(s_{c_t}(o_t+\mu_t) - s_{y_t}(o_t+\mu_t)\big).\end{aligned}$$

*Their difference simply selects the one-token shift on the path:*

$$\begin{aligned}\Delta M(c,y) &\triangleq M_\mu(c,y) - M(c,y) \\ &= \sum_{t=1}^{T} \Delta m_t(c_t) = \sum_{t=1}^{T} \big\langle w_{c_t} - w_{y_t}, \mu_t \big\rangle.\end{aligned}$$

*Thus* $\Delta M(c,y) < 0$ *means the perturbation suppresses path* $c$*; cross-entropy uses the same shifts, averaged over all competing tokens.*

*Proof.* See Appendix F.1.

Together, the propositions give an operational criterion: choose representation levels and thresholds by their effect on downstream NLL, rather than by a separate calibration surrogate.

Accordingly, we learn the quantization levels $R$ and thresholds $T$ by minimizing the language-model negative log-likelihood under KV cache quantization:

$$\begin{aligned}(R^*, T^*) = \arg\min_{R,T} \sum_{X \in \mathcal{X}} \sum_{t=1}^{|X|-1} \\ -\log p_\theta(x_{t+1} \mid x_{\le t}; Q_{R,T})\,.\end{aligned} \tag{5}$$

where $\mathcal{X}$ is the training sequence stream, $\theta$ denotes the base model parameters, and $p_\theta(\cdot \mid x_{\le t}; Q_{R,T})$ is the next-token distribution when the cached K/V states produced while processing $x_{\le t}$ are quantized by the $b$-bit scalar quantizer $Q_{R,T}$. The quantizer is parameterized by representation values $R$ and ordered decision thresholds $T$.

### 3.2 Hard-Forward STE for Learnable Non-Uniform KV Quantization

To enable gradient-based updates of both representation levels and decision thresholds, **Beyond** uses a deployment-exact hard forward pass together with a standard STE input rule and explicit hard-forward pseudo-gradients for the learnable quantizer parameters.

**Hard forward quantization.** For each normalized K/V group, the forward pass simply applies the scalar quantizer in Eq. (2): $\hat{x} = Q(x; R, T)$. This is the same hard operator used at deployment. Because the endpoints remain fixed at $t_0 = 0$ and $t_K = 1$, optimization updates only the levels $R$ and the interior thresholds $\{t_k\}_{k=1}^{K-1}$.

**Non-differentiability and standard STE input backward pass.** The hard quantizer in Eq. (2) is piecewise constant in $x$ and changes assignments only when a threshold crosses a sample. Consequently, the classical derivative with respect to $x$ is zero almost everywhere, and the classical derivative with respect to a threshold is either zero away from assignment changes or undefined at assignment changes, making it unsuitable for gradient-based optimization. Quantization-aware training therefore often uses a straight-through estimator (STE), which keeps the hard forward value while substituting a surrogate derivative in the backward pass (Bengio et al., 2013; Hubara et al., 2018). We use the stop-gradient form

$$\tilde{x} = x + \mathrm{sg}(Q(x; R, T) - x)\,, \tag{6}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. The forward value is exactly $\hat{x} = Q(x; R, T)$, matching deployment, while the backward pass treats the quantizer as the identity map:

$$\frac{\partial L}{\partial x} \approx \frac{\partial L}{\partial \hat{x}}. \tag{7}$$

This standard STE input rule introduces no auxiliary gates, temperatures, or annealing schedules; training and inference use the same hard bin assignment.

**Updates for quantization levels $\mathbf{R} = \{r_k\}_{k=0}^{K-1}$.** Under the hard forward quantizer (2), each level only affects samples assigned to bin $k$, i.e., $\partial\hat{x}/\partial r_k = \mathbb{I}[x \in \text{bin}_k]$. Thus, for a minibatch $\mathcal{B}$,

$$\frac{\partial L}{\partial r_k} = \sum_{x\in\mathcal{B}} \mathbb{I}[x \in \text{bin}_k] \frac{\partial L}{\partial \hat{x}}.$$

**Updates for thresholds $\mathbf{T} = \{t_k\}_{k=1}^{K-1}$ with half-wave rectification.** An interior threshold $t_k$ only changes assignments for samples near the boundary between adjacent reconstruction levels $r_{k-1}$ and $r_k$. Locally, the $K$-level quantizer can be approximated by a 2-level switch:

$$\hat{x} = r_{k-1} + \Delta r_k \, H(x - t_k), \qquad \Delta r_k \triangleq r_k - r_{k-1},$$

where $H(\cdot)$ is the Heaviside step function. Since $H'(u) = \delta(u)$ in the distributional sense,

$$\frac{\partial \hat{x}}{\partial t_k} = -\Delta r_k \, \delta(x - t_k).$$

The impulse $\delta(x - t_k)$ cannot be used directly in backpropagation, so we approximate its boundary effect by aggregating gradients in a narrow window of width $\epsilon_{\text{bw}}$ around $t_k$. For a minibatch $\mathcal{B}$, define

$$\mathcal{W}_L = \{x \in \mathcal{B} : t_k - \epsilon_{\text{bw}} \le x < t_k\},$$
$$\mathcal{W}_R = \{x \in \mathcal{B} : t_k \le x \le t_k + \epsilon_{\text{bw}}\}.$$

and the corresponding aggregated boundary signals

$$G_L = -\Delta r_k \sum_{x\in\mathcal{W}_L} \frac{\partial L}{\partial \hat{x}},$$
$$G_R = -\Delta r_k \sum_{x\in\mathcal{W}_R} \frac{\partial L}{\partial \hat{x}}.$$

The two sides represent directional candidates rather than additive effects. A small left shift ($t_k \downarrow$) moves samples in $\mathcal{W}_L$ from $r_{k-1}$ to $r_k$ and is loss-decreasing iff $G_L > 0$; a small right shift ($t_k \uparrow$) moves samples in $\mathcal{W}_R$ from $r_k$ to $r_{k-1}$ and is loss-decreasing iff $G_R < 0$. We therefore suppress the opposite, loss-increasing directions with half-wave rectification:

$$\widehat{\nabla}_{t_k} L = [G_L]_+ + [G_R]_-, \tag{8}$$

where $[u]_+ = \max(u, 0)$ and $[u]_- = \min(u, 0)$. Under gradient descent, $[G_L]_+$ pushes $t_k$ left and $[G_R]_-$ pushes it right, while noisy or disagreeing boundary evidence is discarded. After each optimizer step, we project $\{t_k\}_{k=1}^{K-1}$ back to the ordered interval $t_1 < \cdots < t_{K-1}$.

**Optimizer and theoretical scope.** We update only the quantizer parameters $\varphi \triangleq (R, T)$ with Adam (Kingma and Ba, 2017); the backbone weights remain frozen. The level and threshold updates are hard-forward pseudo-gradients driven by downstream NLL, not by a separate Euclidean reconstruction objective. Appendix F.3 gives a conditional projected-AMSGrad stationarity check: if the pseudo-gradient is treated as a bounded-mismatch stochastic gradient oracle for a smooth reference objective, then the standard projected-AMSGrad bound applies.

### 3.3 Fused Decode Kernels

To realize the HBM I/O benefit of KV-cache quantization, decode attention must load and consume the compressed K/V representation directly rather than reconstructing a dense cache in HBM. **Beyond** therefore keeps prefill on the standard dense-attention path while producing the compressed cache used by subsequent decode. During prefill, full-precision K/V states are mapped to learned representation levels; a packed copy is stored in the cache, and an uncompressed level-mapped copy is passed to the usual prefill attention. During decoding, a FlashAttention-style kernel implemented in NVIDIA's CuTe DSL reads packed `int32` code words, expands them through cached two- or four-code lookup tables, and writes the resulting full-precision fragments directly into the fused attention kernel's shared-memory operand layout (Dao et al., 2022; Zadouri et al., 2026; NVIDIA, 2026).

Conceptually, the kernel computes

$$\mathbf{o} = \text{softmax}\left(\frac{\mathbf{q}\,\mathcal{D}(\hat{\mathbf{K}})^\top}{\sqrt{d_h}}\right)\mathcal{D}(\hat{\mathbf{V}}),$$

but performs $\mathcal{D}(\cdot)$, score computation, online softmax, and value accumulation within the same tiled attention pipeline. Packed K/V codes are held only as register values before being expanded into shared-memory K/V operand fragments, so no dense K/V tensor is materialized in HBM.

## 4 Experimental Results

### 4.1 Setup

**Backbone LLMs.** We evaluate three instruction-tuned backbones supported by our training and inference pipeline: Ministral-3-14B-Instruct-2512-BF16 (Liu et al., 2026), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025). We use

Table 4: **LongBench-v1 per-task results.**

| Model | Method | NQA | Qasper | MF-en | MF-zh | Hotpot | 2Wiki | Musique | DuR. | Gov. | QMSum | MNews | VCSUM | TREC | Trivia | SAM | LSHT | PCount | PR-en | PR-zh | LCC | Repo | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | Native | 30.3 | 44.9 | 56.3 | 60.6 | 58.2 | 49.7 | **32.1** | 33.3 | 34.6 | 25.3 | 26.7 | 17.5 | **70.0** | **92.2** | 22.3 | 0.5 | **10.6** | **100.0** | 96.0 | 24.9 | 25.2 | 43.4 |
| | KV-FP8 | 31.0 | 44.0 | **56.8** | **60.7** | 57.3 | **50.1** | 30.0 | 33.3 | 34.4 | 25.3 | 26.8 | 17.2 | 69.0 | 89.8 | 20.2 | 0.5 | 7.2 | 99.5 | 96.5 | 24.4 | 23.9 | 42.8 |
| | TurboQuant | 30.9 | 39.9 | 48.3 | 53.0 | **59.0** | 49.3 | 31.9 | **33.9** | 34.9 | 25.4 | 27.0 | **17.6** | 68.5 | 91.6 | 22.0 | 0.5 | 9.7 | 99.5 | **97.5** | 21.8 | 24.7 | 42.2 |
| | **Beyond** | **31.5** | **47.4** | 56.2 | 56.9 | 53.2 | 47.4 | 30.9 | 33.4 | **36.7** | **26.1** | **28.0** | 15.8 | 66.0 | 90.3 | **41.5** | **24.6** | 4.5 | 97.5 | 75.7 | **42.3** | **36.9** | **44.9** |
| Ministral-3-14B-Instruct-2512-BF16 | Native | 32.4 | 47.0 | 48.4 | 61.8 | **65.3** | 56.6 | 45.4 | 18.6 | 22.0 | 17.2 | 19.4 | 13.3 | 68.7 | 92.3 | 36.2 | 3.5 | 8.5 | **100.0** | 99.5 | 58.7 | 37.9 | 45.4 |
| | KV-FP8 | 33.1 | 47.6 | 48.3 | 61.8 | 65.0 | 55.8 | **46.6** | 19.0 | 22.2 | 17.3 | 19.6 | 13.1 | **69.4** | 92.6 | 36.7 | 3.5 | 8.5 | **100.0** | 99.5 | 58.5 | 38.0 | 45.5 |
| | TurboQuant | 33.0 | 46.8 | 48.6 | 62.2 | 64.7 | 55.3 | 45.4 | 19.0 | 21.8 | 17.4 | 19.3 | 13.1 | 69.2 | **93.2** | 36.3 | 2.2 | 7.0 | **100.0** | 99.5 | 58.1 | 37.9 | 45.2 |
| | **Beyond** | **33.7** | **49.1** | **51.1** | **62.4** | 65.2 | **57.7** | 45.5 | **23.0** | **23.1** | **19.2** | **20.4** | **13.8** | 69.2 | 92.5 | **42.5** | **7.0** | **9.0** | **100.0** | **100.0** | **66.3** | **45.1** | **47.4** |
| Qwen3-30B-A3B-Instruct-2507 | Native | 31.0 | **43.0** | 51.1 | **67.3** | 63.5 | **55.9** | 36.6 | 24.3 | 31.4 | 22.1 | 23.5 | **11.3** | 80.0 | **91.3** | 39.1 | 49.5 | 11.0 | **100.0** | **100.0** | 44.0 | 42.7 | 48.5 |
| | KV-FP8 | **31.5** | 42.9 | 52.0 | 66.6 | **63.8** | 54.9 | 36.1 | 24.1 | 31.5 | 22.0 | 23.3 | 11.1 | **81.0** | **91.3** | 39.4 | 49.0 | 10.5 | **100.0** | **100.0** | 39.9 | 37.5 | 48.0 |
| | TurboQuant | 31.3 | 35.8 | 44.6 | 55.7 | 60.3 | 51.9 | **37.4** | 24.4 | 31.0 | 21.8 | 23.6 | 11.1 | 72.0 | 89.1 | 38.8 | 49.0 | **13.5** | **100.0** | 85.1 | 40.1 | 44.1 | 45.7 |
| | **Beyond** | 27.9 | 40.8 | **54.4** | 63.1 | 59.0 | 52.3 | 29.5 | **25.8** | **32.2** | **23.2** | **24.3** | 10.9 | 80.0 | 90.6 | **43.6** | **50.5** | 12.5 | **100.0** | **100.0** | **66.6** | **67.0** | **50.2** |

Table 1: Default setup.

| Setting | Default |
|---|---|
| Models | Ministral-3-14B-Instruct-2512-BF16; Llama-3.1-8B-Instruct; Qwen3-30B-A3B-Instruct-2507 |
| Bits / group size | 4-bit codes / 32 values |
| Initialization | KV per-token uniform quantization (Eqs. 3–4) |
| Trainable params. | K/V quantizer $R$, $T$ only |
| Quantizer granularity | Layer × K/V projection × KV head × 32-dim group |
| K/V placement | K post-RoPE; V post-projection |
| Dataset | The Pile |
| Train seq. length | 8192 tokens |
| Held-out chunks | 150 val. / 150 eval |
| Batch size | 32 |
| Opt./LR ($R$, $T$) | Adam, $7\times10^{-4}$ cosine decay |
| Boundary window $\epsilon_{\text{bw}}$ | 0.009 |
| Train budget | 1 epoch, max 500 opt. steps |
| Early stopping | 3 consecutive val. increases |

Table 2: **The Pile evaluation perplexity.** Lower is better. "Init" is the 4-bit/group-32 quantizer before training, and "Trained" is the final checkpoint. $\Delta$ is the relative PPL reduction from Init to Trained. "Train Time" is wall-clock time from the first training step through final evaluation.

| Model | Native | Init | Trained | $\Delta$ | Steps | Train Time |
|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | 5.50 | 5.58 | **5.38** | −3.61% | 300 | 43m |
| Ministral-3-14B-Instruct-2512-BF16 | 6.77 | 6.78 | **6.58** | −2.93% | 210 | 47m |
| Qwen3-30B-A3B-Instruct-2507 | 5.31 | 5.54 | **4.94** | −10.74% | 220 | 3h44m |

instruct models because they better match production serving workloads, where KV-cache memory is dominated by interactive chat, tool-use, and long-context instruction-following requests.

**Baseline Methods.** We use three reference points: native full precision, KV-FP8, and TurboQuant-KV4 (Zandieh et al., 2026). Full precision measures whether loss-aware quantizer learning can improve over the uncompressed model, while the two compressed references provide strong deployed KV-compression baselines. Methods that are consistently dominated by these references are omitted from the comparison.

**Evaluating Tasks.** We report **The Pile** (Gao et al., 2020) PPL (Jelinek et al., 1977) and seven downstream benchmarks with the lm-evaluation-harness suite (Biderman et al., 2024): LongBench-v1 (Bai et al., 2024), CoQA (Reddy et al., 2019), MATH Maj@1 (Hendrycks et al., 2021b), MMLU (Hendrycks et al., 2021a), AGIEval (Zhong et al., 2024), GPQA Diamond (Rein et al., 2024), and MATH CoT (Hendrycks et al., 2021b). Appendix D additionally reports RULER NIAH (Hsieh et al., 2024).

**Implementation Details.** Table 1 summarizes the default training and quantization setup. The learned quantization configurations are shared across tokens but indexed at the Table 1 granularity; Appendix E gives exact parameter counts, metadata size, and cache bit-width calculations. The trained quantizer used for downstream benchmarks follows the same hard quantization semantics as the packed-cache inference path. We implement this inference path as a custom vLLM attention backend (Kwon et al., 2023), enabling production-style serving with vLLM scheduling and paged KV-cache management. Appendix A details the chat-template wrapping recipe. Hardware uses two NVIDIA B300 GPUs (Aubrey and Stam, 2025); Table 2 reports wall-clock time from the first training step through final evaluation.

### 4.2 Accuracy and Decode Efficiency

**PPL Evaluation.** Table 2 reports The Pile evaluation perplexity. From the KV per-token uniform initialization, quantizer-only training lowers PPL for all three models. The largest reduction is on Qwen3-30B-A3B-Instruct-2507, from 5.54 to 4.94, while Llama-3.1-8B-Instruct and Ministral-3-14B-Instruct-2512-BF16 also improve. These gains show that downstream loss can favor discretizations beyond reconstruction-error minimization. Figure 5 sweeps Llama-3.1-8B-Instruct over effective cache bit-widths, including 4-bit groups of 32, 64, and 128. Among the 4-bit settings, group-32 gives the lowest PPL (5.38) and the strongest compression, so we use 4-bit/group-32 by default.

**Downstream benchmarks.** Table 4 reports all 21 LongBench-v1 tasks. Against native full precision, KV-FP8 (8.00 effective bits), and TurboQuant-KV4 (≈4.19 effective bits), **Beyond** (5.00 effec-

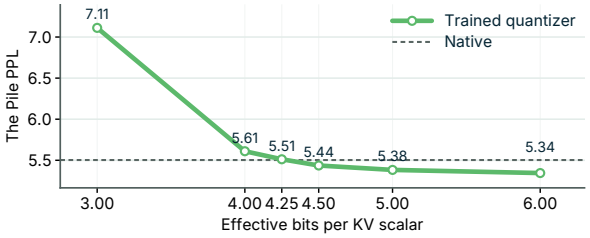


Figure 5: **Effective bit-width versus PPL for Llama-3.1-8B-Instruct.** Points include trained 2-, 3-, 4-, and 5-code-bit/group-32 quantizers and additional 4-code-bit/group-64 and group-128 results, all plotted by effective cache bit-width on the held-out Pile evaluation split. The dashed line is native full-precision PPL. Lower is better.

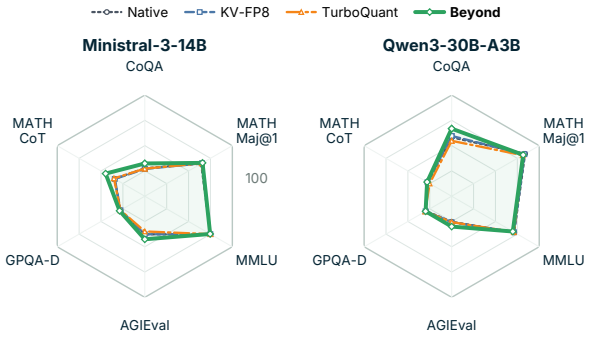


Figure 6: **Six-Benchmark Comparison for Ministral and Qwen.** Scores on the six benchmark tasks. Higher is better.

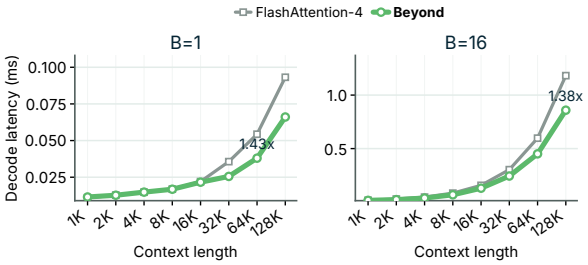


Figure 7: **B300 decode latency for FlashAttention-4 and Beyond.** Latencies are single-token decode times from the same B300 benchmark run and the same FlashAttention-4-derived **Beyond** kernel code path. Lower is better; annotations show the largest FlashAttention-4/**Beyond** speedup in each panel.

Table 3: **Ablation study on Llama-3.1-8B-Instruct.** All variants use the 4-bit/group-32 setup and the same held-out Pile evaluation setup as Table 2. $\Delta$ is the absolute PPL increase relative to **Beyond**.

| Variant | PPL | $\Delta$ |
|---|---|---|
| **Beyond** | 5.38 | 0.00 |
| w/o level learning (freeze $R$) | 5.43 | +0.05 |
| w/o threshold learning (freeze $T$) | 5.40 | +0.02 |
| w/o half-wave rectification | 5.39 | +0.01 |

tive bits) achieves the best Avg21 for all three models. Figures 2 and 6 show the same trend on CoQA, MATH Maj@1, MMLU, AGIEval, GPQA Diamond, and MATH CoT, with detailed scores in Appendix D. Since the quantizer is trained only with continuation-style NLL on the predominantly English Pile corpus, we should not expect every unchanged downstream metric to improve. Weakly aligned slices such as Llama PCount and PR-zh can drop because counting, exact-match retrieval, and Chinese inputs amplify small generation changes. By contrast, gains are consistent on closer-aligned LongBench code-completion tasks (LCC and RepoBench-P) and summarization tasks such as GovReport, QMSum, and MultiNews. **Beyond** also gives the best AGIEval score for all three models, suggesting broader model-capability gains and aligning with the view that loss-aware KV quantization can regularize the model rather than merely compressing the cache.

**Decode speed.** Figure 7 reports B300 single-token decode latency for FlashAttention-4 (Zadouri et al., 2026) and **Beyond** at batch sizes 1 and 16 over 1K–128K contexts. As context length grows, **Beyond** increasingly benefits from reduced HBM I/O and reaches lower decode latency at long sequence lengths.

**Ablation Study.** Table 3 isolates level learning, threshold learning, and half-wave rectification for threshold updates. Removing any one component slightly increases PPL, so each contributes to the final trained quantizer.

## 5 Conclusion

We presented **Beyond**, a loss-aware non-uniform KV-cache quantizer that learns representation levels and thresholds from language-model loss while keeping the forward pass identical to the hard quantizer used at inference. Across three instruction-tuned LLMs, 4-bit/group-32 quantizer-only training improves held-out Pile perplexity over the initialized quantizer, yields competitive LongBench and downstream benchmark results, and can be served through a packed-cache B300 decode path. Taken together, the analysis and experiments support a deployment-aligned view of KV quantization: beyond reducing memory traffic, a compressed cache can act as a lightweight adaptation surface when its discrete parameters are optimized for generation quality.

## Limitations

**Beyond** studies scalar non-uniform KV-cache quantization with frozen backbone weights. Our experiments cover three instruction-tuned model families, a shared 4-bit/group-32 training setup, and an initial Llama-3.1-8B-Instruct effective-bit-width sweep. These results are intended to characterize the proposed loss-aware quantizer within this controlled setting. Extending the study to more architectures, bit-widths, grouping choices, and serving configurations would further clarify how the observed accuracy and decode-efficiency trends transfer across deployments. As with any inference-time compression method, practitioners should validate task quality, latency, memory use, and reliability under their target workloads.

## Ethical Considerations

This paper presents **Beyond**, a KV-cache quantization method for large language model inference. The intended use is resource-efficient long-context serving: reducing KV-cache memory can lower hardware pressure and improve the practicality of deployment. The method does not modify the underlying model weights or training data, so it should be evaluated under the same responsible-deployment procedures as the base model. In particular, application owners should assess output quality, robustness, privacy, and any domain-specific safety requirements after compression.

## References


Kyle Aubrey and Nick Stam. 2025. Inside NVIDIA blackwell ultra: The chip powering the AI factory era. NVIDIA Technical Blog. Accessed: 2026-05-26.

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

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *Preprint*, arXiv:1308.3432.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. Lessons from the trenches on reproducible evaluation of language models. *Preprint*, arXiv:2405.14782.

Christopher M. Bishop. 1995. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. In *COLM 2025*. OpenReview.net.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. 2025. Palu: KV-cache compression with low-rank projection. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *Preprint*, arXiv:1805.06085.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. 2020. Learned step size quantization. In *8th International Conference on Learning Representations*. OpenReview.net.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4852–4861.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt.

2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

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

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

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2018. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30.

Frederick Jelinek, Robert L. Mercer, Lalit R. Bahl, and James K. Baker. 1977. Perplexity: A measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. GEAR: An efficient error reduction framework for KV cache compression in LLM inference. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pages 305–321. PMLR.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

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

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. 2024a. Minicache: Kv cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 139997–140031. Curran Associates, Inc.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, and 1 others. 2026. Ministral 3. *Preprint*, arXiv:2601.08584.

Xin Liu, Xudong Wang, Pei Liu, and Guoming Tang. 2025. Zsmerge: Zero-shot kv cache compression for memory-efficient long-context llms. *Preprint*, arXiv:2503.10714.

Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P. Xing, and Zhiqiang Shen. 2022. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, pages 4942–4952. IEEE.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32332–32344. PMLR.

NVIDIA. 2026. CuTe DSL: Introduction. NVIDIA CUTLASS Documentation. Accessed: 2026-05-26.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. 2024. Eigen attention: Attention in low-rank space for KV cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15332–15344, Miami, Florida, USA. Association for Computational Linguistics.

Donghyun Son, Euntae Choi, and Sungjoo Yoo. 2025. Nsnquant: A double normalization approach for calibration-free low-bit vector quantization of kv cache. *Preprint*, arXiv:2505.18231.

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

Yi Su, Yuechi Zhou, Quantong Qiu, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2025. Accurate KV cache quantization with outlier tokens tracing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12895–12915, Vienna, Austria. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

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

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. 2024. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *Preprint*, arXiv:2407.08454.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. 2025. Asvd: Activation-aware singular value decomposition for compressing large language models. *Preprint*, arXiv:2312.05821.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. 2026. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. *Preprint*, arXiv:2603.05451.

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

Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. 2024a. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. In *Advances in Neural Information Processing Systems*, volume 37, pages 3304–3331. Curran Associates, Inc.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. 2024b. CaM: Cache merging for memory-efficient LLMs inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 58840–58850. PMLR.

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. 2023. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710. Curran Associates, Inc.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314, Mexico City, Mexico. Association for Computational Linguistics.

Table 4: **Boundary-window sensitivity on Llama-3.1-8B-Instruct.** PPL is reported on the same Pile evaluation split as Table 2. Lower is better.

| $\epsilon_{\text{bw}}$ | **Eval. PPL** | **Stopped at** |
|---|---|---|
| 0.007 | 5.40 | step 150 |
| 0.009 | 5.38 | step 300 |
| 0.011 | 5.39 | step 360 |

## A Chat-Prefix Wrapping for Pile Training

For the instruct-model runs, plain Pile documents are formatted as assistant continuations under the model's own tokenizer chat template. The fixed user message is:

```
Continue the following text.
```

Let $p$ denote this prompt. We render this one-turn prompt with an assistant-generation prefix using the tokenizer template of each base model, and then append the raw Pile document text as the target continuation. Equivalently, the input sequence has the form

$$\text{ChatPrefix}_\theta(p) \parallel x,$$

where $x$ is the Pile document text and $\text{ChatPrefix}_\theta$ denotes the model-specific chat-template prefix before assistant content. The prefix tokens are included in the context but are masked out of the loss; only the appended document tokens contribute to the next-token NLL used to train $R$ and $T$. The same wrapping recipe is used for the training stream and the held-out Pile validation/evaluation chunks.

## B Training Dynamics

Figure 8 plots the optimization dynamics for the three 4-bit/group-32 runs reported in Table 2. The curves show the validation loss recorded periodically during training.

## C Boundary-Window Sensitivity

Table 4 reports additional Llama-3.1-8B-Instruct runs that vary the boundary window $\epsilon_{\text{bw}}$ used for threshold pseudo-gradients. Both runs use the same 4-bit/group-32 setup and Pile evaluation protocol as Table 2. The results are close over the tested range, while the default $\epsilon_{\text{bw}} = 0.009$ reaches 5.38 PPL.

## D Detailed Benchmark Results

Table 5 reports exact aggregate scores for CoQA, MATH Maj@1, MMLU, AGIEval, GPQA-D, and MATH CoT. Figure 9 separately visualizes RULER NIAH (Hsieh et al., 2024) for Native and **Beyond** across all three models.

## E Quantizer Granularity and Storage

**Beyond** stores one learned non-uniform scalar quantizer table for each layer, projection type $a \in \{K, V\}$, KV head, and head-dimension group of size $G = 32$. The table is shared across sequence positions and contains $K = 2^b = 16$ representation levels $R$ and $K - 1 = 15$ thresholds $T$. All three evaluated configurations have $d_h = 128$. For a model with $L$ layers, $H_{\text{kv}}$ KV heads, and head dimension $d_h$, the number of learned tables is

$$N_{\text{tab}} = 2LH_{\text{kv}}\frac{d_h}{G},$$

where the factor of 2 accounts for K and V. Thus

$$\begin{aligned} |R| &= N_{\text{tab}}K, \\ |T| &= N_{\text{tab}}(K-1), \\ |\varphi| &= N_{\text{tab}}(2K-1). \end{aligned}$$

In the decode path, thresholds remain FP32 and are used when newly produced K/V states are quantized into the cache; the current CuTe attention kernel itself expands packed codes using a full-precision reconstruction-level LUT. This LUT is a runtime decode convenience and does not change the persistent $R, T$ storage counted in Table 6.

The packed cache stores $b = 4$ code bits per KV scalar plus two group statistics, min and scale, for every group of $G = 32$ scalars. With 16-bit statistics, the cache storage per KV scalar is

$$b_{\text{cache}} = \frac{Gb + 2 \cdot 16}{G} = 4 + \frac{32}{32} = 5.0 \text{ bits}.$$

Equivalently, the packed cache has a $16/5 = 3.2\times$ storage reduction relative to a 16-bit dense KV cache, before allocator/page padding. If the persistent FP32 quantizer tables are amortized over a batch of $B$ sequences with cache length $S$, the amortized bit-width is

$$\begin{aligned} b_{\text{amort}}(B, S) &= b_{\text{cache}} + \frac{32|\varphi|}{2BLSH_{\text{kv}}d_h} \\ &= 5 + \frac{31}{BS} \text{ bits per KV scalar}, \end{aligned}$$

where the final equality uses $b = 4$, $K = 16$, and $G = 32$. For a single sequence, this gives 5.0038 bits at $S = 8192$ and 5.0009 bits at $S = 32768$; larger batches amortize the learned tables further.

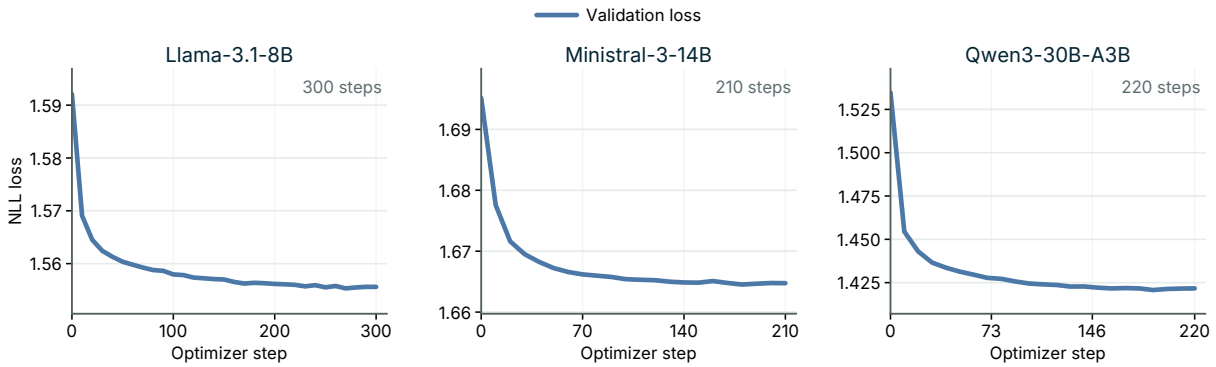


Figure 8: **Training dynamics for the three evaluated models.** Curves show validation NLL evaluated every 10 optimizer steps.

Table 5: **Detailed non-LongBench-v1 Benchmark Scores.** Scores are percentages where higher is better. Within each model block, the best score is bolded and the second-best score is underlined.

| Model | Method | CoQA | MATH Maj@1 | MMLU | AGIEval | GPQA-D | MATH CoT |
|---|---|---|---|---|---|---|---|
| Llama-3.1-8B | Native | 23.6 | **47.4** | 63.0 | 36.3 | 29.3 | **40.0** |
| | KV-FP8 | 21.9 | 46.9 | 62.8 | 36.2 | 27.8 | **40.0** |
| | TurboQuant | 21.4 | 45.1 | 62.6 | 35.9 | 26.8 | 38.2 |
| | **Beyond** | **61.3** | 45.1 | **63.9** | **37.7** | **30.8** | 34.1 |
| Ministral-3-14B | Native | 27.6 | 64.4 | **75.0** | 37.7 | 27.8 | 34.5 |
| | KV-FP8 | 26.9 | 65.4 | 74.9 | 37.7 | 27.8 | 34.2 |
| | TurboQuant | 27.3 | 65.1 | **75.0** | 34.8 | 27.8 | 35.3 |
| | **Beyond** | **32.4** | **66.0** | 74.7 | **42.6** | **28.8** | **44.8** |
| Qwen3-30B-A3B | Native | 58.3 | **83.9** | **71.4** | 25.3 | 29.3 | 27.2 |
| | KV-FP8 | 59.6 | 83.1 | 71.2 | 25.7 | 29.8 | 27.7 |
| | TurboQuant | 54.8 | 81.1 | 71.0 | 25.6 | **30.3** | 25.2 |
| | **Beyond** | **66.8** | 81.7 | 69.8 | **30.0** | 29.8 | **28.0** |

Table 6: **Learned quantizer parameter counts and persistent table memory.** Counts are from the final 4-bit/group-32 quantizer configs. Memory uses the runtime qtable representation: the learned $R$ and $T$ tensors are cached as FP32 on device. This excludes per-token cache codes and min/scale statistics, which are counted in the cache bit-width calculation below.

| Model | $L$ | $H_{\text{kv}}$ | Tables/proj. | $\lvert R\rvert$ | $\lvert T\rvert$ | $\lvert\varphi\rvert$ | FP32 mem. |
|---|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | 32 | 8 | 32 | 32,768 | 30,720 | 63,488 | 253,952 B (248 KiB) |
| Ministral-3-14B-Instruct | 40 | 8 | 32 | 40,960 | 38,400 | 79,360 | 317,440 B (310 KiB) |
| Qwen3-30B-A3B-Instruct | 48 | 4 | 16 | 24,576 | 23,040 | 47,616 | 190,464 B (186 KiB) |

For the KV-FP8 baseline, the run uses vLLM `kv_cache_dtype=fp8`; the cache payload stores one 8-bit FP8 value per K/V scalar. If we also count the two optional FP32 K/V scale scalars per layer, the amortized bit-width is

$$b^{\text{FP8}}_{\text{amort}}(B,S) = 8 + \frac{2L \cdot 32}{2BLSH_{\text{kv}}d_h} = 8 + \frac{32}{BSH_{\text{kv}}d_h}.$$

This is measured in bits per KV scalar; the additive term is negligible for the evaluated context lengths.

# F Mathematical Details and Proofs

## F.1 KV Quantization as Implicit Regularization

This subsection proves the perturbation bounds used in Sec. 3.2. The argument is intentionally local: it identifies the loss terms that any downstream-aware quantizer must control, rather than proving that every quantizer improves every model.

*Proof of Proposition 3.1.* Fix a perturbation $d = \delta_\varphi$ and restrict the loss to the line segment from $z$ to $z+d$ by defining $f(t) = \Psi(z+td)$ for $t \in [0,1]$. Then $f'(0) = \langle g, d\rangle$ and $f''(t) = d^\top \nabla^2 \Psi(z+td)d$. Applying the second-order Taylor formula with integral remainder to this one-dimensional function

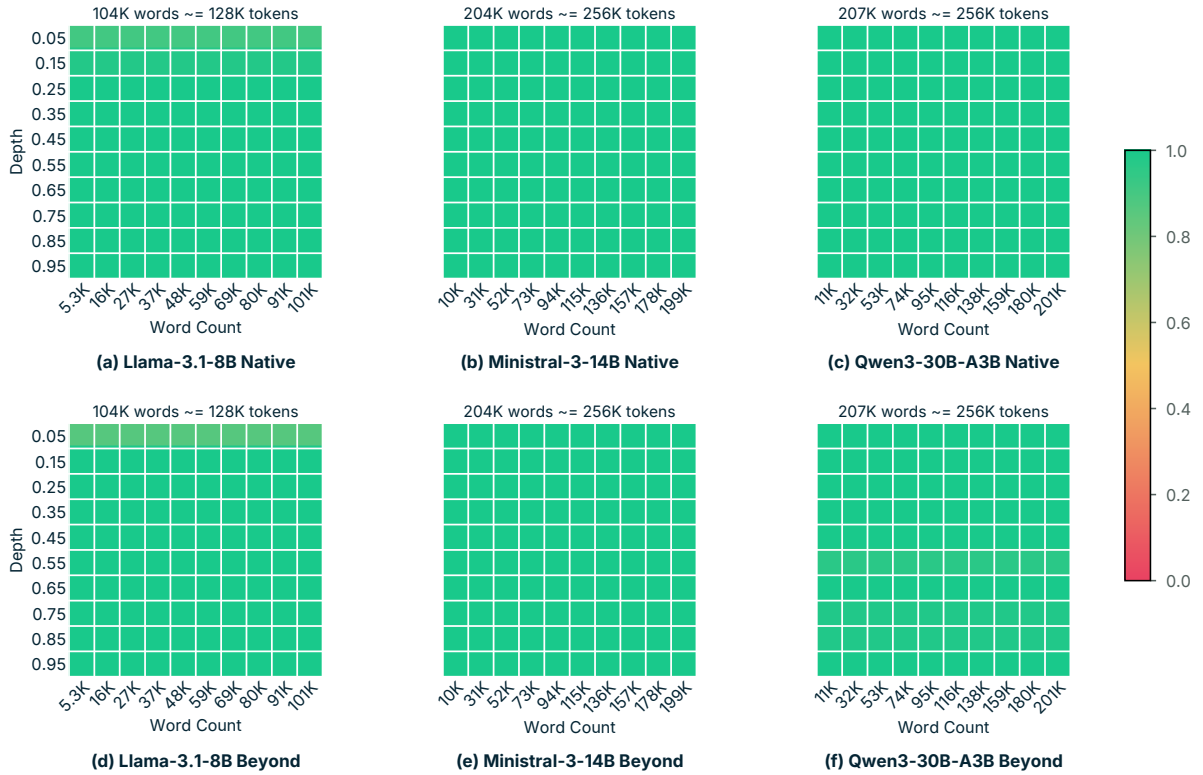


Figure 9: **RULER NIAH heatmaps over depth and word count.** Each panel uses the 500 max-context NIAH samples, bins them by needle depth, and tiles the resulting depth-bin retrieval score across word-count bins to preserve the full KIVI-style heatmap layout.

at $t = 0$ gives

$$f(1) = f(0) + f'(0) + \frac{1}{2}f''(0) + \int_0^1 (1-t)\big(f''(t) - f''(0)\big)\, dt.$$

Here the integral remainder is the part not captured by the Hessian at the base point $z$: it measures how $f''(t)$, equivalently the Hessian of $\Psi$ along the segment from $z$ to $z+d$, changes relative to $f''(0)$. Substituting back $f(1) = \Psi(z+d)$, $f(0) = \Psi(z)$, $f''(0) = d^\top H d$, and $H = \nabla^2\Psi(z)$ yields

$$\Psi(z+d) = \Psi(z) + \langle g, d\rangle + \frac{1}{2}d^\top H d + r(d),$$

where

$$r(d) = \int_0^1 (1-t)\, d^\top \big(\nabla^2\Psi(z+td) - H\big)\, d\, dt.$$

The Lipschitz-Hessian condition $\|\nabla^2\Psi(z+u) - \nabla^2\Psi(z)\|_{\mathrm{op}} \le \beta\|u\|_2$ gives, with $\Delta_t = \nabla^2\Psi(z+td) - H$,

$$\begin{aligned} |r(d)| &\le \int_0^1 (1-t)\|d\|_2^2\|\Delta_t\|_{\mathrm{op}}\, dt \\ &\le \int_0^1 (1-t)\|d\|_2^2 \beta t\|d\|_2\, dt \\ &= \beta\|d\|_2^3 \int_0^1 t(1-t)\, dt = \frac{\beta}{6}\|d\|_2^3. \end{aligned}$$

The first inequality uses $|d^\top A d| \le \|d\|_2\,\|Ad\|_2 \le \|A\|_{\mathrm{op}}\|d\|_2^2$ with $A = \Delta_t$. The second uses the Lipschitz condition with $u = td$, giving $\|\Delta_t\|_{\mathrm{op}} \le \beta t\|d\|_2$. The factor $1/6$ is the scalar integral $\int_0^1 t(1-t)\, dt = 1/6$. Taking expectation yields

$$\mathbb{E}\Psi(z+\delta_\varphi) = \Psi(z) + \langle g, \mathbb{E}\delta_\varphi\rangle + \frac{1}{2}\mathrm{Tr}\Big(H\,\mathbb{E}[\delta_\varphi\delta_\varphi^\top]\Big) + \rho_\varphi,$$

with $|\rho_\varphi| \le \frac{\beta}{6}\mathbb{E}\|\delta_\varphi\|_2^3$. When $g = 0$ and $H = \lambda I$, the only $\varphi$-dependent second-order term is $\frac{\lambda}{2}\mathbb{E}\|\delta_\varphi\|_2^2$. Otherwise, the first-order term and anisotropic curvature term change the local ordering of quantizers, so reconstruction error is not the general downstream objective. □

**Setup.** For one attention head, define

$$a = \frac{1}{\sqrt{d}} qK^\top \in \mathbb{R}^S,$$
$$p = \mathrm{Softmax}(a),$$
$$o = pV.$$

Quantization induces perturbations $\hat{K} = K + \Delta K$ and $\hat{V} = V + \Delta V$. The key perturbation creates logit noise

$$\varepsilon \triangleq \frac{1}{\sqrt{d}} q(\Delta K)^\top,$$
$$a' = a + \varepsilon,$$
$$p' = \mathrm{Softmax}(a + \varepsilon),$$
$$o' = p'\hat{V}.$$

The mean perturbation $\mu = \mathbb{E}[\Delta o]$ may be nonzero for a learned quantizer; this is the degree of freedom that lets loss-aware quantization act differently from reconstruction-only, approximately zero-mean noise.

**Assumption F.1** (Small moment-controlled KV perturbations)**.** Assume $\Delta K$ and $\Delta V$ have bounded second moments and are small enough for the first-order attention expansion to dominate the second-order remainder. Assume further that either $\Delta K$ and $\Delta V$ are independent, or their cross-covariance terms are negligible at first order.

**Lemma F.2** (Softmax first-order expansion)**.** *Let $p = \mathrm{Softmax}(a)$. For small $\varepsilon$,*

$$p' = p + J_{\mathrm{sm}}(a)\,\varepsilon + O(\|\varepsilon\|^2),$$
$$J_{\mathrm{sm}}(a) = \mathrm{Diag}(p) - pp^\top.$$

*Proof.* Taylor expand the smooth map Softmax around $a$ and use its standard Jacobian. □

**Lemma F.3** (Linearized attention output perturbation)**.** *Let*

$$\Delta o \triangleq \big(J_{\mathrm{sm}}(a)\varepsilon\big)V + p\Delta V.$$

*Then, up to second-order terms,*

$$o' = o + \Delta o + O(\|\varepsilon\|^2 + \|\varepsilon\|\,\|\Delta V\|).$$

*Proof.* Expand

$$o' = p'(V + \Delta V)$$
$$= pV + (p' - p)V + p\Delta V$$
$$+ (p' - p)\Delta V.$$

Using Lemma F.2, the first-order terms are exactly $(J_{\mathrm{sm}}(a)\varepsilon)V + p\Delta V$, while $(p' - p)\Delta V$ and the Taylor remainder of $p'$ are second order. □

*Proof of Proposition 3.2.* Let $o_{\mathrm{lin}} = o + \Delta o$ be the linearized attention output from Lemma F.3. By $L$-smoothness of $\ell$,

$$\ell(o_{\mathrm{lin}}) \le \ell(o) + \langle \nabla\ell(o), \Delta o\rangle + \frac{L}{2}\|\Delta o\|_2^2.$$

Taking expectation gives

$$\mathbb{E}[\ell(o_{\mathrm{lin}})] \le \ell(o) + \big\langle \nabla\ell(o), \mathbb{E}[\Delta o]\big\rangle + \frac{L}{2}\mathbb{E}\|\Delta o\|_2^2,$$

which proves the bound with $\mu = \mathbb{E}[\Delta o]$. If $\langle \nabla\ell(o), \mu\rangle < -\frac{L}{2}\mathbb{E}\|\Delta o\|_2^2$, then the right-hand side is strictly smaller than $\ell(o)$, so the expected quantized loss is lower than the full-precision local loss. □

**Corollary F.4** (Zero-mean quantization gives a second-order regularizer)**.** *Under Assumption F.1, if additionally $\mathbb{E}[\Delta K] = 0$ and $\mathbb{E}[\Delta V] = 0$, then $\mu = 0$ in the first-order model, and Proposition 3.2 reduces to*

$$\mathbb{E}[\ell(o_{\mathrm{lin}})] \le \ell(o) + \frac{L}{2}\mathbb{E}\|\Delta o\|_2^2.$$

*Thus zero-mean quantization behaves locally like an input-dependent second-order penalty.*

*Proof.* Since $\mathbb{E}[\Delta K] = 0$ and $\mathbb{E}[\Delta V] = 0$, $\mathbb{E}[\varepsilon] = 0$ and $\mathbb{E}[\Delta o] = (J_{\mathrm{sm}}(a)\mathbb{E}[\varepsilon])V + p\mathbb{E}[\Delta V] = 0$ at first order. Substituting $\mu = 0$ into Proposition 3.2 gives the result. □

*Proof of Proposition 3.3.* Let $w_j^\top$ denote the $j$-th row of $W$ and $\pi_t = \mathrm{Softmax}(Wo_t)$. The cross-entropy loss at position $t$ is

$$\ell_t(o_t) = -\log \pi_{t,y_t}$$
$$= -w_{y_t}^\top o_t + \log \sum_k \exp(w_k^\top o_t).$$

Therefore

$$\nabla_{o_t}\ell_t(o_t) = -w_{y_t} + \frac{\sum_j \exp(w_j^\top o_t) w_j}{\sum_k \exp(w_k^\top o_t)}$$
$$= \sum_j \pi_{t,j} w_j - w_{y_t}.$$

Since $\sum_j \pi_{t,j} = 1$,

$$\nabla_{o_t}\ell_t(o_t) = \sum_j \pi_{t,j}(w_j - w_{y_t})$$
$$= \sum_{j \ne y_t} \pi_{t,j}(w_j - w_{y_t}).$$

Taking the inner product with $\mu_t$ gives the linear term in the loss change,

$$\ell_t(o_t+\mu_t)-\ell_t(o_t) = \langle \nabla_{o_t}\ell_t(o_t), \mu_t\rangle + O(\|\mu_t\|_2^2),$$

where

$$\langle \nabla_{o_t}\ell_t(o_t), \mu_t\rangle = \sum_{j\neq y_t} \pi_{t,j}\langle w_j - w_{y_t}, \mu_t\rangle.$$

Now define the one-step margin shift for any competing token $j \neq y_t$:

$$\Delta m_t(j) \triangleq \big[s_j(o_t+\mu_t) - s_{y_t}(o_t+\mu_t)\big] - \big[s_j(o_t) - s_{y_t}(o_t)\big].$$

Because $s_j(o) = w_j^\top o$,

$$\Delta m_t(j) = \langle w_j - w_{y_t}, \mu_t\rangle.$$

Substituting this identity into the loss expression yields

$$\langle \nabla_{o_t}\ell_t(o_t), \mu_t\rangle = \sum_{j\neq y_t} \pi_{t,j}\Delta m_t(j).$$

Thus cross-entropy averages the one-step competing-margin shifts, weighted by the model probabilities.

For a fixed competing continuation $c_{1:T}$, keep the teacher-forced states $o_t$ fixed from the target prefixes $y_{<t}$ and define

$$M(c,y) = \sum_{t=1}^{T} \big(s_{c_t}(o_t) - s_{y_t}(o_t)\big),$$
$$M_\mu(c,y) = \sum_{t=1}^{T} \big(s_{c_t}(o_t+\mu_t) - s_{y_t}(o_t+\mu_t)\big).$$

Then the path-level margin shift is

$$\begin{aligned}\Delta M(c,y) &\triangleq M_\mu(c,y) - M(c,y)\\ &= \sum_{t=1}^{T} \Delta m_t(c_t)\\ &= \sum_{t=1}^{T} \langle w_{c_t} - w_{y_t}, \mu_t\rangle.\end{aligned}$$

If $\Delta M(c,y) < 0$, the perturbation reduces the margin of path $c$ against the target path. The preceding identity shows that the cross-entropy term optimizes the same one-step shifts, but averaged over all competing tokens rather than selected along a fixed path. □

**Corollary F.5** (Value quantization induces an attention-concentration penalty)**.** *Assume row-wise independent value noise* $\Delta V = [\Delta v_1^\top; \dots; \Delta v_S^\top]$ *with* $\mathbb{E}[\Delta v_i] = 0$ *and* $\mathrm{Cov}(\Delta v_i) = \sigma_V^2 I$*. Then*

$$\mathbb{E}\|p\Delta V\|^2 = d_v\, \sigma_V^2\, \|p\|_2^2.$$

*Since* $\|p\|_2^2$ *is minimized by uniform attention and maximized by one-hot attention, value perturbations contribute a larger second-order penalty for overly concentrated attention.*

*Proof.* $p\Delta V = \sum_i p_i \Delta v_i$. By independence and isotropic covariance, $\mathrm{Cov}(p\Delta V) = \sum_i p_i^2 \sigma_V^2 I = \sigma_V^2 \|p\|_2^2 I$. Taking the trace yields the result. □

**Interpretation.** The zero-mean special case explains the regularization effect: quantization adds a structured second-order penalty whose strength depends on the attention distribution and quantization scale. The learned non-uniform case can go further. By moving $(R,T)$ with the downstream loss, the quantizer can introduce a small systematic perturbation $\mu$ that reduces high-probability competing margins. When this negative first-order effect dominates the positive curvature penalty in Proposition 3.2, the compressed KV cache can obtain lower local loss than the full-precision cache. This is the mathematical mechanism behind treating KV quantization as implicit regularization rather than only as reconstruction error, consistent with classic noise-as-regularization intuition (Bishop, 1995).

### F.2 Hard Objective and Deployment Consistency

This subsection separates two facts that are often conflated in STE-based quantization: the hard empirical objective is not classically differentiable in thresholds, but the forward computation remains exactly the deployed quantizer.

**Feasible set.** Let $\varphi = (R,T)$, with $R \in \mathcal{R}$ for a compact convex set $\mathcal{R} \subset \mathbb{R}^K$ and

$$0 \le t_1 < \cdots < t_{K-1} \le 1.$$

During optimization, this strict ordering is maintained by projection with a small margin $\delta_{\mathrm{thr}} > 0$. For the projected-optimizer statement below, define the closed feasible set

$$\begin{aligned}\Phi = \{(R,T) : R \in \mathcal{R},\ & 0 \le t_1, \quad t_k + \delta_{\mathrm{thr}} \le t_{k+1},\\ & 1 \le k < K-1,\\ & t_{K-1} \le 1\}.\end{aligned}$$

**Hard empirical objective.** For a finite calibration set $\{x_i\}_{i=1}^n$ and per-sample downstream losses $L_i$, define

$$F_n(\varphi) = \frac{1}{n}\sum_{i=1}^{n} L_i(Q(x_i;\varphi)).$$

**Proposition F.6** (Piecewise-constant threshold objective)**.** *Fix $R$ and a finite set $\{x_i\}_{i=1}^n$. On any open region of threshold space where no threshold crosses any sample, i.e., $t_k \neq x_i$ for all $i, k$, the bin assignments are constant and $F_n$ is constant as a function of $T$. Consequently, the classical derivative of $F_n$ with respect to the thresholds is zero almost everywhere.*

*Proof.* Inside such a region, every indicator $\mathbb{I}\{x_i \in B_k(T)\}$ is fixed. Thus each $Q(x_i; R, T)$ is fixed, and so every term $L_i(Q(x_i; R, T))$ is fixed. The only possible changes occur on the finite collection of hyperplanes $t_k = x_i$, which has Lebesgue measure zero. □

**Proposition F.7** (No forward train–deploy gap)**.** *For any parameter iterate $\varphi_t = (R_t, T_t)$ produced by hard-forward STE training, the forward value used during training equals the value used at inference:*

$$\tilde{x}_{\text{forward}} = Q(x; R_t, T_t).$$

*Proof.* From Eq. (6),

$$\tilde{x} = x + \text{sg}(Q(x; R_t, T_t) - x).$$

The stop-gradient operator is the identity in the forward pass, so $\tilde{x} = x + Q(x; R_t, T_t) - x = Q(x; R_t, T_t)$. It only changes the backward derivative. □

**Pseudo-gradient oracle.** The hard-forward estimator is therefore best viewed as an inexact first-order oracle for quantizer parameters. The population update direction can be written abstractly as

$$g_{\text{PG}}(\varphi) = \mathbb{E}\Big[J_{\text{PG},\varphi}(x;\varphi)^\top \nabla_u L(u)\big|_{u=Q(x;\varphi)}\Big], \tag{9}$$

where $J_{\text{PG},\varphi}$ is the pseudo-Jacobian induced by the input, level, and threshold update rules in Sec. 3.2. No smooth sigmoid, softmax relaxation, temperature, or annealing schedule is part of this oracle.

### F.3 Optimizer Sanity Check for an Inexact STE Oracle

We update only $\varphi = (R, T)$ and freeze the backbone weights. The hard empirical objective in Appendix F.2 is piecewise constant in thresholds, so this subsection is not a convergence theory for the classical derivative of that hard objective. It is a conditional optimization sanity check: if the hard-forward backward rule is treated as a stochastic first-order oracle for a smooth reference objective $f$, with a uniformly bounded oracle mismatch, then the standard projected AMSGrad stationarity bound applies. This separates the deployable hard forward pass from the reference first-order geometry used only for optimization.

**Gradient mapping.** For a differentiable reference objective $f$ on the closed convex set $\Phi$, define

$$\mathcal{G}_\eta(\varphi) = \frac{1}{\eta}\left(\varphi - \Pi_\Phi(\varphi - \eta\nabla f(\varphi))\right),$$

where $\Pi_\Phi$ is Euclidean projection. If the projection is inactive, $\mathcal{G}_\eta(\varphi) = \nabla f(\varphi)$. Thus $\|\mathcal{G}_\eta(\varphi)\|$ is the constrained analogue of a gradient norm.

**Projected AMSGrad update.** Given a mini-batch hard-forward pseudo-gradient estimator $g_t$ at $\varphi_t$, projected AMSGrad uses

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\, g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\,(g_t \odot g_t), \\ \hat{v}_t &= \max\{\hat{v}_{t-1},\, v_t\} \qquad \text{(elementwise)}, \\ \varphi_{t+1} &= \Pi_\Phi\left(\varphi_t - \eta_t \frac{m_t}{\sqrt{\hat{v}_t} + \epsilon_{\text{adam}}}\right), \end{aligned} \tag{10}$$

where $\odot$ is the Hadamard (elementwise) product, $\sqrt{\cdot}$ and all divisions are elementwise, and $\epsilon_{\text{adam}} > 0$ is the standard numerical-stability constant. Bias correction is omitted only to simplify notation; it changes constants through a deterministic rescaling of $\eta_t$. Adam uses the same $m_t$ and $v_t$ recursions (Kingma and Ba, 2017). AMSGrad replaces $v_t$ by the monotone envelope $\hat{v}_t$, which is the ingredient used in the convergence proof (Reddi et al., 2018).

**Assumption F.8** (Smooth reference objective and inexact STE oracle)**.** Assume:

(i) $\Phi$ is closed, convex, and bounded, and $f$ is $L_f$-smooth on $\Phi$, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq L_f\|x - y\|$ for all $x, y \in \Phi$.

(ii) $f$ is bounded below on $\Phi$ by $f^{\text{inf}}$.

(iii) With respect to the filtration $\mathcal{F}_t$ generated by the previous minibatches and iterates,

$$b_t \triangleq \mathbb{E}[g_t \mid \mathcal{F}_t] - \nabla f(\varphi_t)$$

is the oracle mismatch and is uniformly bounded by $\|b_t\| \le \varepsilon_{\mathrm{PG}}$ almost surely.

(iv) Stochastic pseudo-gradients are uniformly bounded in $\ell_\infty$: $\|g_t\|_\infty \le G$ almost surely for all $t$.

(v) $\eta_t = \eta/\sqrt{t}$ for some $\eta > 0$, and $\beta_1, \beta_2 \in (0,1)$ with $\beta_1^2 < \beta_2$.

**Lemma F.9** (One-step descent with oracle mismatch)**.** *Under Assumption F.8, there exist constants $a_0, a_1, a_2 > 0$, independent of $t$ and $T$, such that the projected AMSGrad iterate satisfies*

$$\mathbb{E}_t[f(\varphi_{t+1})] \le f(\varphi_t) - a_0\eta_t\|\mathcal{G}_{\eta_t}(\varphi_t)\|^2 + a_1\eta_t\big|\langle \mathcal{G}_{\eta_t}(\varphi_t), b_t\rangle\big| + a_2\eta_t^2,$$

*where $\mathbb{E}_t[\cdot] = \mathbb{E}[\cdot \mid \mathcal{F}_t]$.*

*Proof.* Let $\bar{g}_t = \mathbb{E}[g_t \mid \mathcal{F}_t] = \nabla f(\varphi_t) + b_t$ and write the martingale noise as $\xi_t = g_t - \bar{g}_t$, so $\mathbb{E}_t[\xi_t] = 0$. The bounded-gradient assumption implies $\epsilon_{\mathrm{adam}} \le \sqrt{\hat{v}_{t,i}} + \epsilon_{\mathrm{adam}} \le G + \epsilon_{\mathrm{adam}}$ for every coordinate $i$. Hence the diagonal AMSGrad preconditioner is uniformly positive definite and uniformly bounded; all norm changes between the preconditioned update and the Euclidean gradient mapping are controlled by constants depending only on $G, \epsilon_{\mathrm{adam}}$, and the dimension of $\varphi$.

By $L_f$-smoothness,

$$f(\varphi_{t+1}) \le f(\varphi_t) + \langle \nabla f(\varphi_t), \varphi_{t+1} - \varphi_t\rangle + \frac{L_f}{2}\|\varphi_{t+1} - \varphi_t\|^2.$$

The projection optimality condition for $\varphi_{t+1} = \Pi_\Phi(\varphi_t - \eta_t D_t^{-1} m_t)$, where $D_t = \mathrm{Diag}(\sqrt{\hat{v}_t} + \epsilon_{\mathrm{adam}})$, gives the usual projected-descent inequality: the component of the step aligned with the reference gradient contributes a negative term proportional to $\eta_t\|\mathcal{G}_{\eta_t}(\varphi_t)\|^2$. The replacement of $\nabla f(\varphi_t)$ by the conditional mean $\bar{g}_t = \nabla f(\varphi_t) + b_t$ leaves the additional inner product $\eta_t\langle \mathcal{G}_{\eta_t}(\varphi_t), b_t\rangle$. Finally, the martingale noise $\xi_t$, first-moment recursion, and bounded preconditioner contribute only second-order terms after conditional expectation; the AMSGrad monotonicity of $\hat{v}_t$ and the condition $\beta_1^2 < \beta_2$ control the accumulated momentum/preconditioner variation by a constant multiple of $\eta_t^2$. Collecting these standard AMSGrad terms yields the stated inequality with constants $a_0, a_1, a_2$ independent of $t$ and $T$. □

**Theorem F.10** (Projected AMSGrad stationarity for the reference objective)**.** *Let $f$ be the smooth reference objective in Assumption F.8. Then projected AMSGrad with $\eta_t = \eta/\sqrt{t}$ satisfies*

$$\min_{1\le t\le T} \mathbb{E}\big\|\mathcal{G}_{\eta_t}(\varphi_t)\big\|_2^2 \le \frac{C_1 + C_2(1+\log T)}{\sqrt{T}} + C_3\,\varepsilon_{\mathrm{PG}}^2,$$

*for constants $C_1, C_2, C_3 > 0$ independent of $T$. Thus the projected gradient mapping of the reference objective decays at the standard nonconvex projected-AMSGrad rate up to the STE oracle mismatch.*

*Proof of Theorem F.10.* Let $H_t = \mathcal{G}_{\eta_t}(\varphi_t)$. Taking total expectation in Lemma F.9, summing from $t = 1$ to $T$, and using $f(\varphi_{T+1}) \ge f^{\inf}$ gives the following bound, where

$$A_0 = a_0^{-1}\big(f(\varphi_1) - f^{\inf}\big), \qquad A_1 = a_0^{-1}a_2, \qquad A_2 = a_0^{-1}a_1.$$

$$\sum_{t=1}^T \eta_t\,\mathbb{E}\|H_t\|^2 \le A_0 + A_1\sum_{t=1}^T \eta_t^2 + A_2\sum_{t=1}^T \eta_t\,\mathbb{E}\big|\langle H_t, b_t\rangle\big|. \tag{11}$$

The first term is the finite initial optimality gap, the second term is the usual stochastic/adaptive-stepsize accumulation, and the third term is the only term introduced by the hard-forward oracle mismatch. By Cauchy–Schwarz and Young's inequality, for any $\rho > 0$,

$$\big|\langle H_t, b_t\rangle\big| \le \frac{\rho}{2}\,\|H_t\|^2 + \frac{1}{2\rho}\,\|b_t\|^2 \le \frac{\rho}{2}\,\|H_t\|^2 + \frac{1}{2\rho}\,\varepsilon_{\mathrm{PG}}^2.$$

Choosing $\rho = 1/A_2$ and substituting this bound into Eq. (11) gives one half of $\sum_t \eta_t\mathbb{E}\|H_t\|^2$ back on the right-hand side. Moving it to the left yields

$$\sum_{t=1}^T \eta_t\,\mathbb{E}\|H_t\|^2 \le 2A_0 + 2A_1\sum_{t=1}^T \eta_t^2 + A_2^2\,\varepsilon_{\mathrm{PG}}^2\sum_{t=1}^T \eta_t. \tag{12}$$

Since all $\eta_t$ are positive, the smallest stationarity measure among the first $T$ iterates is bounded by the stepsize-weighted average:

$$\min_{1\le t\le T} \mathbb{E}\|\mathcal{G}_{\eta_t}(\varphi_t)\|^2 \;\le\; \frac{\sum_{t=1}^{T} \eta_t\, \mathbb{E}\|H_t\|^2}{\sum_{t=1}^{T} \eta_t}.$$

For $\eta_t = \eta/\sqrt{t}$, $\sum_{t=1}^{T} \eta_t \ge \eta\sqrt{T}$ and $\sum_{t=1}^{T} \eta_t^2 \le \eta^2(1+\log T)$. Combining these inequalities with Eq. (12) yields

$$\min_{1\le t\le T} \mathbb{E}\|\mathcal{G}_{\eta_t}(\varphi_t)\|^2 \le \frac{2A_0 + 2A_1\,\eta^2(1+\log T)}{\eta\sqrt{T}} + A_2^2\,\varepsilon_{\mathrm{PG}}^2.$$

Setting $C_1 \triangleq 2A_0/\eta$, $C_2 \triangleq 2A_1\eta$, and $C_3 \triangleq A_2^2$ gives the claimed bound. □

### F.4 Scope of the Mathematical Claims

The appendix makes three limited claims. First, **Beyond** has no forward train–deploy mismatch: training and inference evaluate the same hard quantizer. Second, because the hard empirical objective is piecewise constant in thresholds, threshold learning relies on an explicitly biased boundary pseudo-gradient. Third, the Adam/AMSGrad statement is a first-order stationarity guarantee for a smooth reference objective under a bounded-oracle-mismatch STE update; it is not a global-optimality guarantee for the piecewise-constant hard objective. These are the assumptions under which the standard input STE and hard-forward threshold update in the main paper should be interpreted.