EAT: Eviction-Aware Training for Long-Context LLM Inference on Edge Devices

Ke Chen[1] **Xiaohui Song**[1]* **Chi Xie**[1] **ZhiGuang Zhu**[1] **Zixian Li**[1]
Hao-Yun Chen[2] **Peng-Wen Chen**[2] **Chen Chen**[1] **Haonan Lu**[1]
[1]OPPO AI Center [2]MediaTek Inc.
{chenke2,songxiaohui,xiechi,zhuzhiguang}@oppo.com
{lizixian,chenchen4,luhaonan}@oppo.com
{Hao-Yun.Chen,Peng-Wen.Chen}@mediatek.com

Abstract

Deploying large language models on edge devices requires mitigating two major bottlenecks: the computational cost of the model and the memory overhead of the KV cache. While combining model quantization and KV-cache eviction seems like a natural solution, we reveal that this naive pipeline suffers from a critical failure mode. The issue stems from a hidden conflict: modern eviction methods rely on attention scores to identify important tokens, yet quantization inherently adds noise to these exact scores. Beyond reducing numerical precision, quantization can distort eviction decisions, causing critical context to be discarded.

To overcome this non-orthogonal degradation, we propose **Eviction-Aware Training (EAT)**, a training framework that integrates discrete eviction behavior into quantization-aware training. EAT computes eviction from quantized attention, applies the resulting mask in the forward pass, and uses a masked straight-through estimator in the backward pass. The method requires no architectural changes and adds no inference-time parameters.

Systematic evaluations show that EAT mitigates the failure mode of naive quantization-plus-eviction pipelines. Under W4A16 + KV-INT8 with 50% KV eviction, eviction-unaware composition reduces the five-task average from 38.37 to 29.04, whereas EAT recovers it to 39.44 under the same cache budget. Real-world deployment on the MediaTek Dimensity 9500 demonstrates that, under a 50% KV-eviction budget, EAT improves decoding throughput by 64.1% and reduces total memory traffic by 20.4% compared to quantized full-KV inference. Mechanistic analyses suggest that the improvement is associated with reduced attention sink behavior rather than with sharper attention.

1 Introduction

Large language models (LLMs) are rapidly evolving toward long-context capabilities, which are important for tasks such as document understanding, long-horizon dialogue, code analysis, and agentic reasoning [Team et al., 2024, Liu et al., 2025b]. At the same time, growing demands for privacy, low latency, and offline operation are driving long-context inference from the cloud to edge devices. This shift is constrained by a central systems bottleneck: the Key-Value (KV) cache grows linearly with context length, quickly straining the limited memory capacity and bandwidth of edge hardware.

Two families of techniques are widely used to address this bottleneck. **Quantization** reduces the numerical precision of weights and cached states to reduce memory usage and improve compute

*Corresponding author.

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

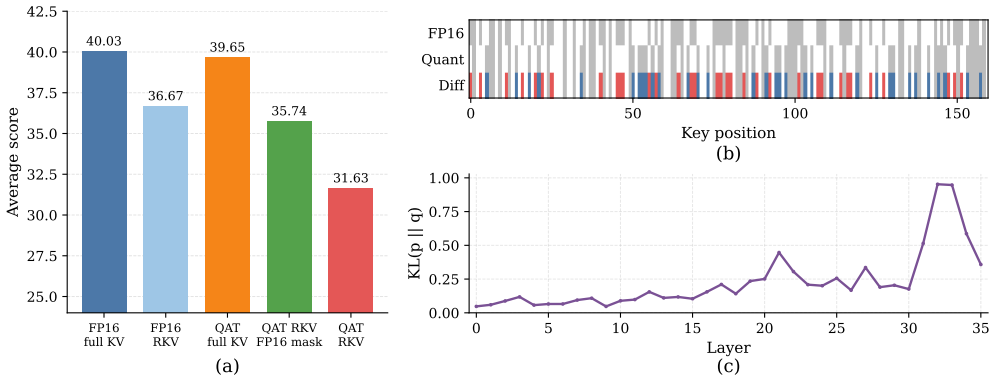

Figure 1: **Quantization and eviction are not orthogonal. (a)** Mean score on BoolQ and LongBench (VCsum, 2WikiMQA). **(b)** First 160 key positions of 50% RKV masks on a VCsum case; FP16/Quant rows (gray/white: retain/evict); Diff highlights disagreements (red: FP16-only retained; blue: quant-only retained; gray/white: agreement). **(c)** Layer-wise KL of quantized vs FP16 attention over full VCsum under full KV.

efficiency [Xiao et al., 2023a, Frantar et al., 2022, Lin et al., 2024]. **KV-cache eviction** limits the cache size by retaining only the tokens judged most salient by importance-based heuristics [Zhang et al., 2023, Li et al., 2024, Xiao et al., 2023b]. Edge deployments typically necessitate both strategies, as quantization alone cannot suppress the linear scaling of the KV cache at long context lengths, and eviction alone leaves the model's static memory and computational costs unoptimized.

We study a practical but underexplored question: how well do quantization and KV eviction compose when deployed together? Prior work has shown that sparsity and quantization can interact non-orthogonally [Zhang et al., 2024, Harma et al., 2025]. We investigate this interaction in attention-driven KV eviction, where quantization noise perturbs the attention statistics used to determine token retention. As a result, it can alter token eviction decisions under the same retention policy. Figure 1 illustrates this effect. In panel (a), we report the mean task score over three representative evaluations: BoolQ and the LongBench tasks VCsum and 2WikiMQA. Across this compact slice of context-grounded QA, summarization, and multi-hop reasoning, quantization alone remains close to the full-KV FP16 baseline, whereas naively adding 50% RKV [Cai et al., 2026] eviction causes a large performance drop. Panel (a) also includes a fixed-mask intervention: holding the quantized model and retention budget constant while reusing the FP16-derived RKV mask raises the score from 31.63 to 35.74. As an offline diagnostic that is not intended for deployment, this counterfactual intervention recovers 81.5% of the FP16-to-quantized gap. This supports retention-mask drift as a major contributor to the degradation. Panels (b) and (c) further show that quantization already changes attention patterns under full KV and can lead to different retention decisions under eviction. Together, these results indicate that the errors introduced by quantization and eviction are coupled rather than simply additive.

To overcome this non-orthogonal degradation, we introduce **Eviction-Aware Training (EAT)**, an architecture-agnostic training framework that simulates discrete cache eviction inside quantization-aware training. During training, EAT computes eviction decisions from quantized attention, masks the corresponding tokens in attention, and uses a straight-through estimator (STE) [Courbariaux et al., 2015] to preserve gradient flow through the discrete mask. This exposes the model to the same sparse and quantized conditions that it will face at deployment, without modifying the model architecture or adding inference-time parameters. Our contributions are as follows:

- **We characterize how quantization interacts with attention-driven KV-cache eviction.** We show that quantization perturbs attention statistics, inducing drift in token-retention decisions and degrading naive quantization-plus-eviction pipelines.
- **We propose Eviction-Aware Training (EAT), a methods-consistent training framework for compressed long-context inference.** EAT incorporates eviction simulation into quantization-aware optimization through attention masking and straight-through gradient

estimation, enabling the model to adapt jointly to quantization noise and cache sparsity during training.

- **We validate EAT across benchmark and deployment settings.** Across two 3B LLM families, two eviction methods, and five evaluations, EAT consistently outperforms naive quantization-plus-eviction composition. Appendix D.1 further shows the same trend on Qwen2.5-7B and Llama3.1-8B with RKV eviction.

2 Related Work

Our work intersects with three primary research areas: quantized LLM inference, KV-cache compression for long-context, and quantization-aware training.

2.1 Model and KV Cache Quantization

Quantization lowers memory and compute cost by representing weights, activations, or cached keys and values in low-bit formats. Post-training methods include AWQ and OmniQuant [Lin et al., 2024, Shao et al., 2023]; dedicated KV-cache quantization such as KIVI, KVQuant, QAQ, and KVTuner uses asymmetric, adaptive, or mixed-precision designs [Liu et al., 2024, Hooper et al., 2024, Dong et al., 2024, Li et al., 2025]. Most of this literature assumes unchanged cache-access patterns; we study quantization composed with attention-based eviction, where quantization also perturbs the signal that determines which tokens stay accessible.

2.2 KV Cache Eviction and Sparse Long-Context Inference

A separate line of work reduces KV-cache footprint and long-context attention cost by compressing cache access at inference time, including attention-based eviction, sink- or window-based retention, query-aware retrieval, and head- or layer-adaptive compression [Zhang et al., 2023, Xiao et al., 2023b, Tang et al., 2024, Ge et al., 2023, Cai et al., 2024]. Most of these methods are post-hoc inference procedures applied to pretrained models. More recent training-aware sparse-attention approaches, such as DeepSeek Sparse Attention, learn auxiliary token-selection modules and train the model to operate under sparse attention [Liu et al., 2025a]. In contrast, EAT does not modify the backbone or introduce additional inference-time parameters; it adapts standard models to existing attention-based eviction methods within the same fine-tuning stage used for quantization-aware adaptation. Our claims are therefore scoped to the method family whose selection signal is computed from attention.

2.3 Quantization-Aware Training

Quantization-aware training (QAT) [Jacob et al., 2018, Nagel et al., 2021] reduces the mismatch between training and quantized deployment by simulating low-bit arithmetic during optimization. In the LLM setting, this idea has been extended through calibration-based fine-tuning and learnable transformations for low-bit inference [Esser et al., 2025, Maskey et al., 2026], often with straight-through estimators for discrete quantization operators [Bengio et al., 2013]. However, prior QAT methods primarily adapt models to quantized numerics under dense attention or fixed cache access. They do not explicitly optimize the model under the discrete retention masks induced by an eviction method, nor for the interaction between quantized attention and token selection.

3 Preliminary Analysis

To make the source of the deployment mismatch precise, we first formalize inference under joint quantization and KV-cache eviction, and then derive the deployment-aligned objective optimized by EAT. The key point is that, once retention decisions are computed from quantized attention, quantization changes not only the attention values themselves but also the subset of tokens that remains accessible. Detailed perturbation analysis and stability arguments are deferred to Appendix A.

3.1 Problem Setup

Dense attention as the reference path. Let $\mathbf{Q} \in \mathbb{R}^{m\times d}$, $\mathbf{K} \in \mathbb{R}^{n\times d}$, and $\mathbf{V} \in \mathbb{R}^{n\times d}$. Standard scaled dot-product attention computes

$$\mathbf{O} = \mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\mathbf{S})\mathbf{V} = \mathbf{A}\mathbf{V},$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the attention probability matrix.

Quantized attention as the deployed method input. At deployment, the eviction method operates on the quantized attention path. During QAT, we therefore simulate deployment with fake-quantizers $\mathcal{Q}_a$ and $\mathcal{Q}_{kv}$:

$$\tilde{\mathbf{Q}} = \mathcal{Q}_a(\mathbf{Q}), \qquad \tilde{\mathbf{K}} = \mathcal{Q}_{kv}(\mathbf{K}), \qquad \tilde{\mathbf{V}} = \mathcal{Q}_{kv}(\mathbf{V}).$$

The deployment attention induced by this quantized path is

$$\tilde{\mathbf{A}} = \mathrm{softmax}\left(\frac{\tilde{\mathbf{Q}}\tilde{\mathbf{K}}^\top}{\sqrt{d}}\right),$$

which is the method input used both by eviction at inference and by EAT during training.

Eviction as selection on the attention path. Under eviction, only a retained subset $\mathcal{S} \subset \{1, \ldots, n\}$ remains accessible, giving

$$\mathrm{Attn}_{\text{evict}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}; \mathcal{S}) = \mathrm{softmax}\left(\mathbf{S} + \mathbf{M}_{\mathcal{S}}\right)\mathbf{V},$$

where $\mathbf{M}_{\mathcal{S}}$ masks evicted columns. In attention-based eviction, the retained set is produced by a method π_r under eviction ratio r from token-importance scores computed on the attention matrix. Along the reference path,

$$\mathbf{s} = g(\mathbf{A}), \qquad \mathcal{S}_{\text{fp}} = \pi_r(\mathbf{A}) = \mathrm{TopK}(\mathbf{s}, K),$$

and along the deployed quantized path the same method instead operates on $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{s}} = g(\tilde{\mathbf{A}}), \qquad \mathcal{S}_{\text{q}} = \pi_r(\tilde{\mathbf{A}}) = \mathrm{TopK}(\tilde{\mathbf{s}}, K).$$

3.2 Coupled Failure Mode and Deployment Objective

A common assumption is that quantization and eviction are orthogonal compression axes. However, once the eviction method is driven by attention, quantization can perturb the importance ranking used for token retention and therefore change the retained set itself. This introduces a second source of error beyond fixed-set quantization error: the model may now lose access to different prompt tokens than it would have kept under the full-precision path. Appendix A gives a more explicit perturbation analysis, a margin-based stability argument, and the associated error decomposition.

The preceding observation identifies the missing ingredient in standard dense QAT: it adapts the model to quantized numerics, but not to eviction-induced changes in accessible context. Dense QAT minimizes the cross-entropy risk under fake quantization with a full prompt KV cache,

$$\mathcal{R}_{\text{dense}}(\theta) = \mathbb{E}_{(x,y)}\left[-\log p_\theta^{\text{quant}}(y|x)\right],$$

whereas deployment-aligned EAT minimizes

$$\mathcal{L}^{\text{EAT}}(\theta) = \mathbb{E}_{(x,y),\, r\sim\mathcal{D}_r}\left[-\log p_\theta^{\text{quant+evict}}(y|x;r)\right]. \tag{1}$$

Here θ are backbone parameters; (x, y) are supervised LM pairs; $p_\theta^{\text{quant}}(y|x)$ is the likelihood under fake-quantized dense attention (no prompt eviction); $p_\theta^{\text{quant+evict}}(y|x;r)$ is the likelihood under fake quantization with eviction ratio r, induced by the masked attention construction in Sec. 4; $r \sim \mathcal{D}_r$ randomizes the cache budget (we use $\mathcal{D}_r = \mathcal{U}(r_{\min}, r_{\max})$). Section 4 instantiates Eq. 1.

4 Method: Eviction-Aware Training

4.1 Overview

EAT realizes Eq. 1 by executing the deployment-time eviction rule inside each training-time attention layer. The central design goal is method consistency: the prompt tokens retained during training must be exactly those that would survive under the same quantized inference path, eviction method, and memory budget at deployment. Relative to dense QAT, which only adapts the model to low-precision numerics under full context access, EAT additionally adapts the backbone to the method-induced removal of prompt KV entries.

At a high level, EAT has three components. First, it computes the method input from the same fake-quantized attention path used to simulate deployment. Second, it converts the method output into per-layer column masks so that the forward pass matches physical KV eviction. Third, it applies a masked straight-through estimator so that the backward pass does not propagate gradients through tokens that would be inaccessible at inference. Figure 2 summarizes this pipeline; detailed GQA-aware mask construction and pseudocode are deferred to Appendix B.

4.2 Method-Consistent Mechanism

Deployment-consistent eviction simulation. Consider one attention layer on the quantized deployment path. Let

$$\tilde{\mathbf{S}} = \frac{\tilde{\mathbf{Q}}\tilde{\mathbf{K}}^\top}{\sqrt{d}}, \qquad \tilde{\mathbf{A}} = \mathrm{softmax}(\tilde{\mathbf{S}} + \mathbf{M}_{\mathrm{causal}}).$$

Let $\mathcal{I}_{\mathrm{sup}}$ denote the token positions that participate in the language-model loss, and define the prompt boundary

$$p = \min \mathcal{I}_{\mathrm{sup}}.$$

Only the prompt prefix $[1, p)$ is eligible for eviction, matching the prefill-stage cache-compression regime studied in this work. This definition is agnostic to the particular supervision encoding: in practice, $\mathcal{I}_{\mathrm{sup}}$ may be recovered from a loss mask, labels not equal to an ignore index, or any equivalent implementation-specific convention.

Using the same quantized attention that will be available at deployment, EAT forms a method observation on the prompt region,

$$\Phi(\tilde{\mathbf{A}}, p) = \tilde{\mathbf{A}}_{:,:,\,\mathcal{Q}(p),\,\mathcal{K}(p)},$$

where $\mathcal{Q}(p)$ is a method-dependent query window near the prompt-response boundary and $\mathcal{K}(p) = \{1, \ldots, p\}$ are the candidate prompt keys. The eviction method then returns per-KV-head evicted positions

$$\mathcal{E} = \pi\Big(\Phi(\tilde{\mathbf{A}}, p); r, \eta_\pi\Big),$$

with eviction ratio r and method-specific hyperparameters η_π. This interface deliberately abstracts away the internal scoring rule of the method: RKV-style and SnapKV-style methods differ in how they summarize attention, but both fit the same map from quantized attention statistics to evicted prompt indices. Because eviction removes KV entries rather than query-specific edges, each evicted prompt token corresponds to a masked column in the attention matrix for that layer. In GQA models, this eviction decision is computed at the KV-head level and broadcast to the associated query heads; the explicit construction is given in Appendix B. Defining

$$\mathbf{L} = \tilde{\mathbf{S}} + \mathbf{M}_{\mathrm{causal}} + \mathbf{M}_{\mathcal{E}}, \qquad \mathbf{O}_{\mathrm{EAT}} = \mathrm{softmax}(\mathbf{L})\tilde{\mathbf{V}}, \tag{2}$$

EAT exactly matches the effect of physical cache removal at inference: evicted columns receive zero probability mass and contribute neither value content nor KV-memory traffic. When $r = 0$, the eviction mask vanishes and the formulation reduces to standard dense fake-quantized attention, making EAT a strict generalization of dense QAT.

Optimization through discrete eviction. The only non-smooth component of the training pipeline is the discrete selection performed by π. EAT does not introduce a differentiable surrogate selector

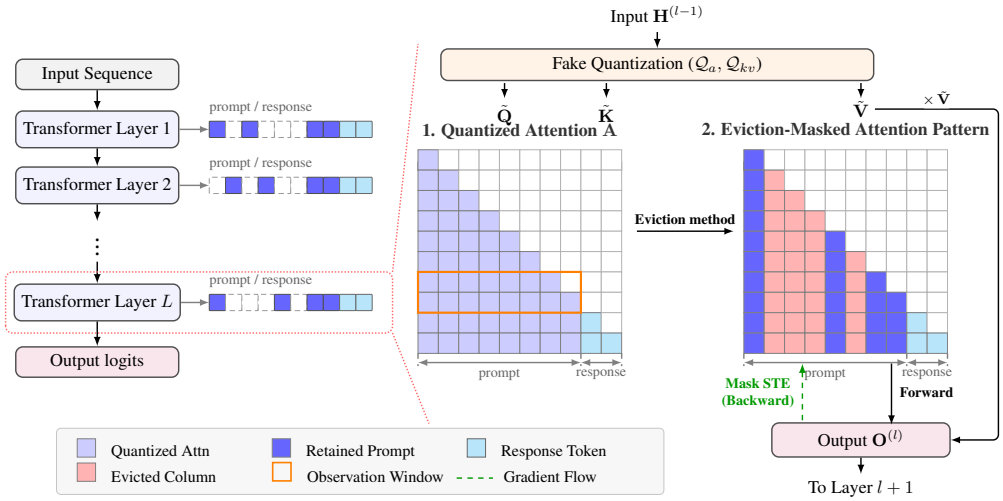

Figure 2: **Eviction-Aware Training (EAT) framework. Left: Layer-by-layer eviction simulation:** each transformer layer recomputes its own retention pattern from local quantized attention statistics, so different layers may preserve different prompt KV entries under the same global eviction ratio, while response tokens remain visible and are not eviction candidates. This makes the training-time sparse structure match the deployment-time behavior of attention-based eviction. **Right: Per-layer EAT mechanism:** fake-quantized $\tilde{\mathbf{Q}}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}}$ yield $\tilde{\mathbf{A}}$; the eviction policy reads the prompt-side slice $\Phi(\tilde{\mathbf{A}}, p)$ and turns per-head eviction choices into a column mask $\mathbf{M}_{\mathcal{E}}$ on attention logits. The forward path follows Eq. 2 under deployment semantics; MaskSTE zeros gradients through evicted columns so the backward path respects the same sparse access pattern without extra inference-time parameters.

and does not learn the method itself. Instead, it keeps π fixed and optimizes the backbone under the sparse regime induced by the chosen method through a masked straight-through estimator:

$$\mathrm{MaskSTE}(\mathbf{L}, \mathbf{M}_{\mathrm{grad}}) = \begin{cases} \mathbf{L}, & \text{forward,} \\ \nabla_{\mathbf{L}} \odot \mathbf{M}_{\mathrm{grad}}, & \text{backward.} \end{cases}$$

Thus the forward computation remains identical to deployment, while the backward pass blocks gradients through logits corresponding to tokens that the deployment method would have removed. This prevents optimization from relying on information that would be inaccessible at inference. In our main recipe, we sample the training eviction ratio from $r \sim \mathcal{D}_r$ to expose the model to a range of high-compression operating budgets. Because all additional logic is confined to the attention forward pass, EAT introduces no auxiliary routing module, no extra inference-time parameters, and no architecture-specific sparse attention operator.

5 Experiments

We evaluate EAT under joint quantization and KV-cache eviction. We first report results at a fixed 50% KV budget, then study the quality-retention trade-off across compression ratios, present on-device results, and finally analyze the attention changes induced by EAT.

5.1 Experimental Protocol

We evaluate EAT on two resource-efficient 3B instruction-tuned LLM families, Qwen2.5-3B-Instruct and Llama3.2-3B-Instruct, under a W4A16 + KV-INT8 setting [Qwen et al., 2025, Grattafiori et al., 2024]. Our main comparison is against post-hoc composition of standard quantization-aware training (QAT) and attention-based eviction, which is the simplest deployment pipeline without further adaptation. We also report the FP model and the quantized full-KV model as references, and later include component ablations to isolate the contribution of masked STE and dynamic ratio sampling beyond standard dense QAT. For eviction, we consider both an RKV-style rank-based method and SnapKV [Li et al., 2024].

Table 1: **Task-level results under 50% KV eviction.** All models use W4A16 + KV-INT8 quantization. For EAT, we report only the deployed 50%-eviction setting.

	Qwen2.5-3B						Llama3.2-3B					
Configuration	**BoolQ**	**Gov.**	**PIQA**	**VCsum**	**2Wiki**	**Avg.**	**BoolQ**	**Gov.**	**PIQA**	**VCsum**	**2Wiki**	**Avg.**
FP (full)	83.52	24.67	**48.42**	11.46	25.11	38.63	83.34	27.38	50.33	11.48	**26.99**	39.90
QAT (full)	82.39	24.65	48.26	11.18	**25.39**	38.37	83.39	27.33	50.05	11.09	26.63	39.70
QAT+RKV	69.94	16.91	33.41	6.68	18.26	29.04	74.92	23.82	32.26	7.83	19.61	31.69
QAT+SnapKV	73.27	13.78	34.77	9.42	20.31	30.31	73.85	20.02	25.52	10.95	21.58	30.38
EAT+RKV	82.23	**27.45**	44.83	**17.88**	24.81	**39.44**	**84.01**	**28.07**	51.08	**17.57**	23.91	**40.93**
EAT+SnapKV	**84.31**	22.41	47.33	17.66	24.51	39.24	83.33	26.22	**51.78**	14.50	24.27	40.02

Our evaluation suite contains five tasks spanning context-grounded QA, commonsense reasoning, long-form summarization and multi-hop reasoning: BoolQ, PIQA [Bisk et al., 2020], LongBench (VCsum, GovReport, and 2WikiMQA) [Bai et al., 2024]. We report each benchmark's standard metric and use the unweighted mean across the five tasks only as a compact summary.

For training, we use a weighted supervision mixture comprising task-aligned data from GovReport [Huang et al., 2021], BoolQ [Clark et al., 2019], and VCsum [Wu et al., 2023], together with broader instruction-tuning and capability-preservation data from Alpaca-GPT4 (English and Chinese) [Taori et al., 2023], CodeAlpaca [Chaudhary, 2023], GSM8K [Cobbe et al., 2021], MATH [Hendrycks et al., 2021], and MathQA [Amini et al., 2019]. Task-aligned examples use only the official training splits for GovReport, BoolQ, and VCsum; we do not include any examples from the evaluation benchmarks' held-out test sets in the fine-tuning corpus. FP-full, QAT-full, and EAT use the same fine-tuning data mixture. Full hyperparameters are reported in Appendix C.

5.2 Main Results at the Target Budget

Table 1 reports results at a fixed 50% KV budget. Eviction-unaware composition is fragile: although quantized full-KV models remain close to the FP reference, adding 50% eviction causes a large quality drop for both model families and both methods. EAT recovers most of this loss, and in several settings matches or slightly exceeds the corresponding quantized full-KV reference. These results show that training in the same quantized-and-evicted regime encountered at inference largely removes the degradation of naive composition.

Overall, the task-level breakdown shows no evidence that EAT trades gains on one benchmark for losses on another. Under a fixed 50% cache budget, it improves both model families across all five tasks, which is exactly the regime most relevant to memory-constrained deployment.

The same pattern holds at larger scale under the same W4A16 + KV-INT8 setting and 50% KV budget: EAT improves over QAT+RKV by 8.30 points on Qwen2.5-7B and 7.74 points on Llama3.1-8B. These results provide evidence that the benefit of deployment-consistent eviction-aware training is not limited to the original 3B models; detailed per-task results are reported in Appendix D.1.

5.3 Performance Across KV Compression Ratios

The 50% setting is our anchor point, but deployment budgets vary across devices and prompt lengths. Figure 3 therefore plots performance under a broader range of KV retention ratios on Qwen2.5-3B. EAT shifts the quality-retention frontier upward rather than improving only one matched operating point. As retention becomes more aggressive, the eviction-unaware baselines deteriorate rapidly, especially on GovReport, VCsum, and 2WikiMQA; in contrast, both RKV+EAT and SnapKV+EAT degrade more gradually.

5.4 Component Ablations

We isolate two design choices behind EAT: (i) whether gradients should be masked on evicted prompt positions in the backward pass, and (ii) whether the training eviction ratio should be fixed to the deployment budget or sampled from a range. All runs use the same Qwen2.5-3B-Instruct initialization, fine-tuning data, optimization budget, W4A16 + KV-INT8 deployment setting, and—unless noted—the RKV method. Full bar and line plots are in Appendix D.2 (Figure 5).

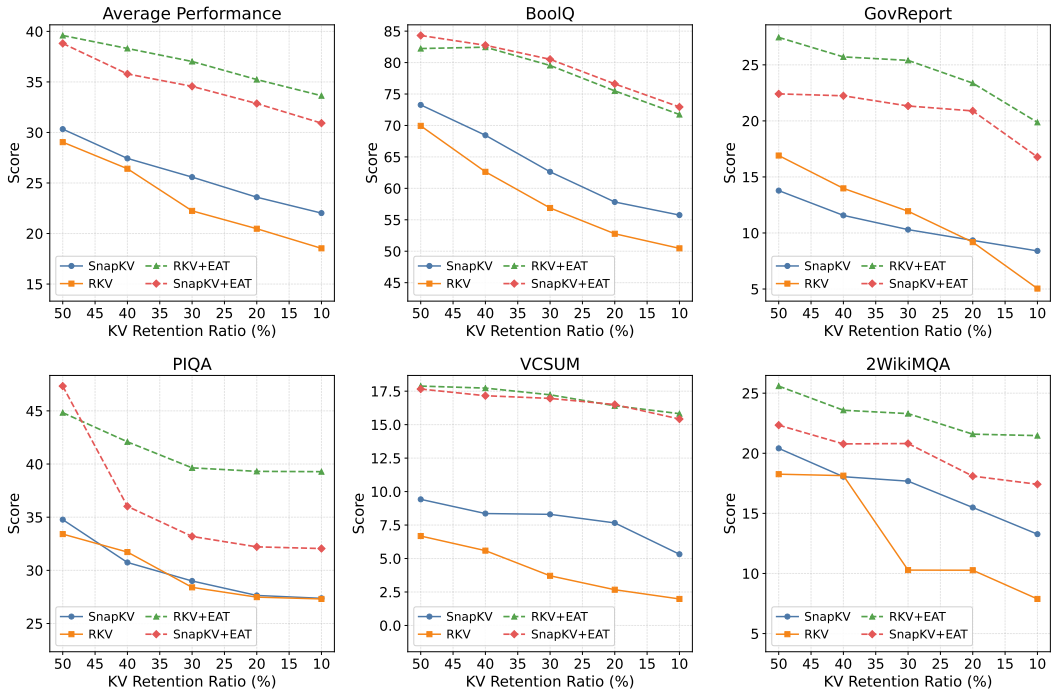

Figure 3: **Performance under different KV retention ratios on Qwen2.5-3B.** We compare eviction-unaware quantization-plus-eviction against EAT across average score and the five individual tasks. Across nearly the entire budget range, EAT shifts the quality-retention curve upward, and the advantage becomes especially pronounced under more aggressive KV compression. This indicates that EAT improves not only one fixed deployment point, but the overall performance of the model to changing KV budgets.

Masked STE in the backward pass. The w/o STE mask baseline applies the same forward eviction simulation as EAT but backpropagates through all prompt positions, including those that would be physically absent from the KV cache at inference. The full EAT model instead uses the masked STE so that no gradient updates tokens that the eviction policy would discard. At a fixed 50% test-time eviction budget, removing the STE mask drops the five-task average from **39.44** to **34.08**. Forward-only exposure creates a training-inference discrepancy: although the forward pass simulates sparsity, the unmasked backward pass allows gradients from discarded positions to guide the weight updates. Masking the backward pass ensures that the optimization process is strictly aligned with the sparse access pattern of inference.

Fixed training ratio vs. dynamic ratio sampling. We compare training with eviction ratio fixed to 50%, 70%, or 90% eviction ratio against dynamic sampling $r \sim \mathcal{U}(0.5, 0.9)$, and evaluate each checkpoint at the same three deployment eviction levels (Figure 5b). Fixed-ratio runs behave as matched specialists: performance peaks when train and test budgets align but drops sharply off-diagonal (Appendix D.2). Dynamic sampling stays within a small margin of each specialist at its sweet spot yet degrades much more gracefully when the deployment budget shifts; we therefore use it as the default when retention is not known a priori across devices or workloads.

5.5 On-Device Results on MTK Dimensity 9500

Eviction improves not only cache capacity but also attention-time memory traffic, which is often the dominant systems bottleneck in long-context edge inference. On MTK Dimensity 9500, we measure end-to-end deployment under 16K-token context decode configuration, comparing quantized full-KV inference to eviction at a 50% retained-KV budget with EAT. Relative to the quantized full-KV baseline, decoding throughput increases by **64.1%** and total memory traffic decreases by about **20.4%**. We report the full normalized breakdown in Table 6 (Appendix D.4). This is single-setting, directional device-side evidence rather than a complete systems study.

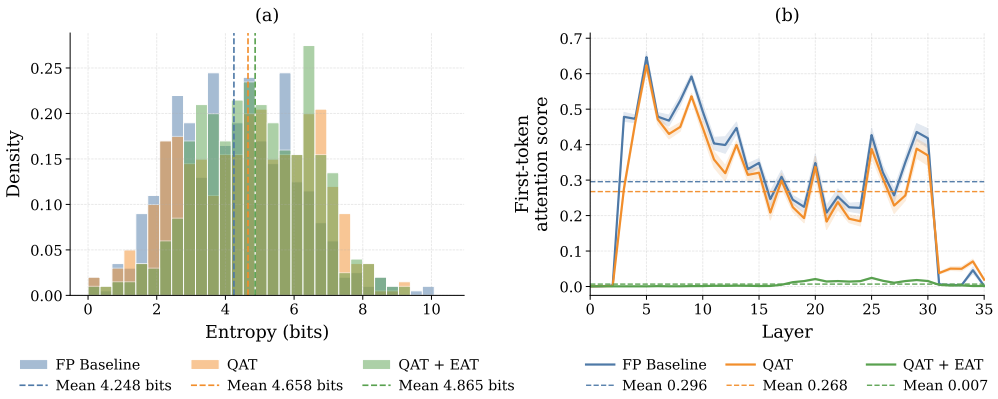

Figure 4: **Mechanistic analysis of attention behavior.** (a): the attention-entropy distribution does not become lower under EAT; if anything, it shifts slightly higher than standard QAT. (b): despite this entropy increase, EAT strongly suppresses first-token attention, reducing the mean sink score from 26.8% to 0.7%. Taken together, these results suggest that improved performance under EAT is associated with reallocation away from sink tokens rather than with globally sharper attention, consistent with prior evidence that sink suppression benefits long-context modeling.

5.6 Mechanistic Analysis: Entropy Increase, Sink Suppression

We next ask why EAT recovers performance. An initially surprising observation is that EAT does not reduce attention entropy. Figure 4 shows that under full attention, the mean attention entropy increases from 4.658 bits for QAT to 4.865 bits for QAT+EAT, even though downstream performance improves under eviction. Taken alone, this result seems to contradict the intuition that performance recovery should come from sharper or more concentrated attention.

We therefore ask what drives this entropy increase. We hypothesize that EAT creates optimization-level pressure on attention under joint quantization noise and eviction-consistent sparsity. With gradients masked on eviction-discarded prompt positions, allocating significant probability mass to non-semantic anchors—patterns often correlated with sink-like early-token concentration—becomes an inefficient use of the limited attention budget for minimizing the training objective. The optimizer may then favor reallocating mass toward retained, informative prompt tokens rather than uniformly sharpening or diffusing the attention distribution. **We treat this as a hypothesis consistent with correlational measurements, not as evidence of a causal mechanism.** We inspect first-token attention as a standard sink proxy: the right panel of Figure 4 shows that QAT+EAT suppresses first-token attention across layers, reducing the mean sink score from 26.8% to 0.7%. Together with panel (a), where entropy shifts upward rather than toward uniformly sharper peaks, the pattern suggests *reallocation* rather than unstructured diffusion. Qualitatively, selective modulation of attention mass is reminiscent of gated-attention perspectives [Qiu et al., 2025], although EAT introduces no explicit gating modules and we do not claim architectural equivalence.

6 Conclusion

When KV-cache eviction is driven by attention, quantization affects both the cached representations and the attention statistics used to select tokens. EAT incorporates the deployment eviction process into QAT, so that the model is trained under the same quantized and sparse cache-access pattern used at inference, without changing the inference graph.

Under matched KV budgets, EAT consistently outperforms applying post-hoc eviction to an eviction-unaware QAT model and maintains stronger performance across retention ratios. In a 16K-context deployment experiment on MediaTek Dimensity 9500, EAT achieves higher decoding throughput and lower memory traffic than quantized full-KV inference at 50% retention. Our analysis indicates that reduced attention-sink behavior is an important contributor to these gains.

Limitations. Despite the performance gains of EAT, several limitations remain. First, the main comparison isolates eviction-unaware composition versus EAT under two attention-score eviction methods; we do not benchmark the full space of KV compression algorithms or training-aware alternatives. Second, although we include 7B and 8B scale-up results, the study covers only two model families and one main quantization regime; it does not establish scaling beyond this range, transfer to other architectures, or behavior under more aggressive low-bit settings. Third, performance still degrades at extreme compression ratios (*e.g.*, beyond 80% eviction), where essential context is irretrievably lost. Fourth, while the training recipe generalizes across RKV and SnapKV in our runs, performance still depends on matching the training-time method to deployment-time behavior. Fifth, our device-side evidence is limited to one SoC and normalized metrics; a fuller systems study should include absolute throughput/latency sweeps, prompt lengths, thermal states, and energy measurements. Finally, the mechanistic evidence in Section 5 is correlational; a stronger account would require direct intervention and retention-stability analyses.

Broader Impacts. The deployment of long-context LLMs is increasingly constrained by the memory and compute limitations of edge devices. By improving the quality of quantized-and-evicted inference, EAT can make on-device long-context processing more practical. Potential positive impacts include broader access to capable models on consumer hardware, reduced dependence on cloud processing for sensitive inputs, and lower energy use for memory-bound workloads. At the same time, improved efficiency may also reduce the cost of deploying capable language models in settings where monitoring and safeguards are weaker. As with other efficiency techniques for LLMs, the broader societal impact of EAT depends primarily on the downstream applications in which it is adopted.

Acknowledgments and Disclosures

This work received no external funding or support from third parties. The authors declare no competing interests.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, 2019.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137, 2024.

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

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

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

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-kv: Redundancy-aware kv cache compression for reasoning models, 2026. URL `https://arxiv.org/abs/2505.24133`.

Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pages 2924–2936, 2019.

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

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. Qaq: Quality adaptive quantization for llm kv cache. *arXiv preprint arXiv:2403.04643*, 2024.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Silq: Simple large language model quantization-aware training, 2025. URL `https://arxiv.org/abs/2507.16933`.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

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

Simla Burcu Harma, Ayan Chakraborty, Elizaveta Kostenok, Danila Mishin, Dongho Ha, Babak Falsafi, Martin Jaggi, Ming Liu, Yunho Oh, Suvinay Subramanian, and Amir Yazdanbakhsh. Effective interplay between sparsity and quantization: From theory to practice, 2025. URL `https://arxiv.org/abs/2405.20935`.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 conference of the north American chapter of the association for computational linguistics: Human language technologies*, pages 1419–1436, 2021.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Wulong Liu, Yiwu Yao, Sinno Jialin Pan, and Mingxuan Yuan. Kvtuner: Sensitivity-aware layer-wise mixed-precision kv cache quantization for efficient and nearly lossless llm inference. *arXiv preprint arXiv:2502.04420*, 2025.

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

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*, 2025b.

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. *arXiv preprint arXiv:2402.02750*, 2024.

Sohir Maskey, Constantin Eichenberg, Johannes Messner, and Douglas Orr. 1-bit wonder: Improving qat performance in the low-bit regime through k-means quantization, 2026. URL `https://arxiv.org/abs/2602.15563`.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Han Wu, Mingjie Zhan, Haochen Tan, Zhaohui Hou, Ding Liang, and Linqi Song. Vcsum: A versatile chinese meeting summarization dataset, 2023. URL `https://arxiv.org/abs/2305.05280`.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023a.

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

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

Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Zhangyang Wang. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 381–394, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/bbb7506579431a85861a05fff048d3e1-Paper-Conference.pdf.

A Additional Analysis of Quantization-Eviction Coupling

This appendix expands the argument in Section 3 and makes explicit how quantization perturbs the eviction method input, how this can change the retained set, and why the resulting mismatch is not captured by fixed-set quantization analysis alone.

Method-input perturbation. Write the fake-quantized states as $\tilde{\mathbf{Q}} = \mathbf{Q} + \Delta\mathbf{Q}$ and $\tilde{\mathbf{K}} = \mathbf{K} + \Delta\mathbf{K}$. The induced logit perturbation is

$$\tilde{S}_{ij} - S_{ij} = \frac{\mathbf{q}_i \Delta\mathbf{k}_j^\top + \Delta\mathbf{q}_i \mathbf{k}_j^\top + \Delta\mathbf{q}_i \Delta\mathbf{k}_j^\top}{\sqrt{d}}.$$

For sufficiently small perturbations, a first-order expansion of the row-wise softmax gives

$$\tilde{\mathbf{A}}_{i,:} \approx \mathbf{A}_{i,:} + J_{\text{softmax}}(\mathbf{S}_{i,:}) \left(\tilde{\mathbf{S}}_{i,:} - \mathbf{S}_{i,:} \right),$$

where $J_{\text{softmax}}(\mathbf{S}_{i,:})$ is the Jacobian of the softmax at row i. Hence quantization perturbs the attention signal seen by the eviction method.

Selection stability is margin-dependent. Whether the retained set changes depends not only on the perturbation size, but also on how close the method is to the selection boundary. Let the reference importance scores be sorted in descending order as $s_{(1)} \geq \cdots \geq s_{(n)}$, and let the method retain the top-K entries. Define the retention margin

$$\gamma = s_{(K)} - s_{(K+1)}.$$

If the score perturbation satisfies $\|\tilde{\mathbf{s}} - \mathbf{s}\|_\infty < \gamma/2$, then the top-$K$ set is unchanged, *i.e.*, $\mathcal{S}_{\text{q}} = \mathcal{S}_{\text{fp}}$. Conversely, when the perturbation reaches or exceeds this margin, the retained set can change. Therefore, the sensitivity of eviction to quantization is governed not only by average numerical error but also by how close tokens are to the retention boundary.

Error decomposition under eviction. Define

$$\mathbf{O}_{\text{fp}}(\mathcal{S}) = \text{Attn}_{\text{evict}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}; \mathcal{S}), \qquad \mathbf{O}_{\text{q}}(\mathcal{S}) = \text{Attn}_{\text{evict}}(\tilde{\mathbf{Q}}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}}; \mathcal{S}).$$

Then, by the triangle inequality,

$$\|\mathbf{O}_{\text{fp}}(\mathcal{S}_{\text{fp}}) - \mathbf{O}_{\text{q}}(\mathcal{S}_{\text{q}})\| \leq \underbrace{\|\mathbf{O}_{\text{fp}}(\mathcal{S}_{\text{fp}}) - \mathbf{O}_{\text{q}}(\mathcal{S}_{\text{fp}})\|}_{\text{fixed-set quantization error}} + \underbrace{\|\mathbf{O}_{\text{q}}(\mathcal{S}_{\text{fp}}) - \mathbf{O}_{\text{q}}(\mathcal{S}_{\text{q}})\|}_{\text{selection-mismatch error}}.$$

The first term measures the numerical effect of quantization when the retained set is held fixed. The second term measures the additional error induced when quantization changes the retained set itself. Under full cache, the second term is zero because all tokens are kept. Under eviction, the second term is non-zero exactly when the method input perturbation changes the top-K decision.

B Additional Method Details

B.1 GQA-Aware Mask Construction

Modern LLMs frequently use grouped-query attention, so eviction must be defined at the KV-head level and then broadcast to the corresponding query heads. Let $\mathcal{E}^{(h)}$ denote the evicted prompt positions for KV head h, and let $\mathcal{G}(h)$ be the set of query heads sharing that KV head. We construct

$$(\mathbf{M}_{\mathcal{E}})_{u,:,j} = \begin{cases} -\infty, & j \in \mathcal{E}^{(h)},\ u \in \mathcal{G}(h), \\ 0, & \text{otherwise}, \end{cases}$$

$$(\mathbf{M}_{\text{grad}})_{u,:,j} = \begin{cases} 0, & j \in \mathcal{E}^{(h)},\ u \in \mathcal{G}(h), \\ 1, & \text{otherwise}. \end{cases}$$

The first mask enforces deployment-consistent token removal in the forward pass, while the second records which logits should stop receiving gradients.

Algorithm 1 EAT training-time forward pass

Require: Initial hidden states $\mathbf{H}^{(1)}$, causal mask $\mathbf{M}_{\text{causal}}$, supervision index set $\mathcal{I}_{\text{sup}}$, eviction-ratio distribution $\mathcal{D}_r$, method π, method hyperparameters η_π, GQA map $\mathcal{G}$
Ensure: Final hidden states $\mathbf{H}^{(L+1)}$ for optimizing Eq. 1

Determine prompt boundary $p \leftarrow \min \mathcal{I}_{\text{sup}}$
Sample training eviction ratio $r \sim \mathcal{D}_r$
for attention layer $l = 1, \ldots, L$ **do**
  Compute fake-quantized projections $(\tilde{\mathbf{Q}}^{(l)}, \tilde{\mathbf{K}}^{(l)}, \tilde{\mathbf{V}}^{(l)}) \leftarrow \text{QuantizedProject}(\mathbf{H}^{(l)})$
  Form quantized logits $\tilde{\mathbf{S}}^{(l)} \leftarrow \tilde{\mathbf{Q}}^{(l)}(\tilde{\mathbf{K}}^{(l)})^\top / \sqrt{d}$
  Compute quantized attention $\tilde{\mathbf{A}}^{(l)} \leftarrow \text{softmax}(\tilde{\mathbf{S}}^{(l)} + \mathbf{M}_{\text{causal}})$
  Extract method observation $\mathbf{T}^{(l)} \leftarrow \Phi(\tilde{\mathbf{A}}^{(l)}, p)$
  Initialize $\mathbf{M}_{\mathcal{E}}^{(l)} \leftarrow \mathbf{0}$ and $\mathbf{M}_{\text{grad}}^{(l)} \leftarrow \mathbf{1}$
  for KV head $h = 1, \ldots, H_{\text{kv}}$ **do**
    Compute evicted prompt positions $\mathcal{E}^{(l,h)} \leftarrow \pi(\mathbf{T}^{(l,h)}; r, \eta_\pi)$
    for query head $u \in \mathcal{G}(h)$ **do**
      Set $(\mathbf{M}_{\mathcal{E}}^{(l)})_{u,:,j} \leftarrow -\infty$ for all $j \in \mathcal{E}^{(l,h)}$
      Set $(\mathbf{M}_{\text{grad}}^{(l)})_{u,:,j} \leftarrow 0$ for all $j \in \mathcal{E}^{(l,h)}$
    end for
  end for
  Form masked logits $\mathbf{L}^{(l)} \leftarrow \tilde{\mathbf{S}}^{(l)} + \mathbf{M}_{\text{causal}} + \mathbf{M}_{\mathcal{E}}^{(l)}$
  Apply masked STE $\hat{\mathbf{L}}^{(l)} \leftarrow \text{MaskSTE}(\mathbf{L}^{(l)}, \mathbf{M}_{\text{grad}}^{(l)})$
  Compute sparse attention output $\mathbf{O}^{(l)} \leftarrow \text{softmax}(\hat{\mathbf{L}}^{(l)})\tilde{\mathbf{V}}^{(l)}$
  Update transformer states $\mathbf{H}^{(l+1)} \leftarrow \text{LayerUpdate}^{(l)}(\mathbf{H}^{(l)}, \mathbf{O}^{(l)})$
end for
Return $\mathbf{H}^{(L+1)}$ and optimize the sequence loss under Eq. 1

B.2 EAT Training Algorithm

Algorithm 1 makes the implementation explicit. For each training sequence, EAT first determines the prompt boundary and sets the eviction ratio to the target training budget. It then applies the same sequence of operations inside every attention layer: fake quantization, method observation, discrete eviction, GQA-aware mask construction, masked attention, and the standard transformer layer update.

C Implementation Details

C.1 Gradient-Mask Implementation

The masked-gradient approximation can be implemented with a custom PyTorch autograd function:

```
class GradMask(torch.autograd.Function):
    @staticmethod
    def forward(ctx, x, mask):
        ctx.save_for_backward(mask)
        return x  # Identity in forward

    @staticmethod
    def backward(ctx, grad_output):
        mask, = ctx.saved_tensors
        # Only pass gradients for non-evicted positions
        return grad_output * mask, None
```

C.2 Eviction Algorithm Implementation

For SnapKV-style eviction, we implement:

```
def compute_eviction_mask(attn_weights, labels,
                          eviction_ratio, window_size):
    # Find prompt boundary
    prompt_end = (labels != -100).float().argmax(dim=-1)

    # Extract observation window
    obs_start = max(0, prompt_end - window_size)
    attn_obs = attn_weights[:, :, obs_start:prompt_end,
                            :prompt_end]

    # Compute importance scores
    scores = attn_obs.mean(dim=2)  # Average over queries
    scores = F.max_pool1d(scores, kernel_size=7,
                          stride=1, padding=2)

    # Select tokens to evict (lowest scores)
    keep_count = int(prompt_end * (1 - eviction_ratio))
    _, indices = scores.topk(keep_count, dim=-1)

    # Create masks
    evict_mask = torch.full_like(scores, float('-inf'))
    evict_mask.scatter_(-1, indices, 0)

    grad_mask = torch.zeros_like(scores)
    grad_mask.scatter_(-1, indices, 1)

    return evict_mask, grad_mask
```

C.3 Training Data Mixture

Our fine-tuning set is a weighted mixture rather than a single benchmark corpus. It contains a *core* subset aligned with the target deployment setting, namely long-context supervision emphasizing summarization and context-grounded QA under long prompts, and a supplemental pool of general-purpose instruction-tuning data, including generic instruction-following, code, and mathematical reasoning examples. Core examples drawn from GovReport, BoolQ, and VCsum use only those datasets' designated **training** portions; no instances from the evaluation test lists are included.

To bias adaptation toward the deployment-relevant regime, we assign the core subset a higher sampling weight in the data loader ($6\times$ relative to the supplemental pool). This increases exposure to the data most sensitive to prompt-KV compression while retaining broad supervision for instruction-following behavior. In the main text, we therefore describe the training corpus as a weighted mixture with upsampled core data rather than as a single public benchmark.

C.4 Training Hyperparameters

Here we list the hyperparameters for EAT in detail. The top block includes the hyperparameters for the basic training, like the learning rate, the scheduler, the warmup steps, the batch size, etc. The middle block covers the hyperparameters used specifically in the eviction process, like the eviction ratio, the size of the observation window, and the pooling kernel. The last block covers the GPU information and the environment for distributed training.

C.5 Additional Per-Task Analysis

We also carried out additional detailed analysis of EAT on each specific task. For this, Qwen2.5-3B is adopted and RKV with 50% eviction is used. For each task, EAT obtain a gain range from 6.55% to 12.29%, which is quite an obvious margin. On average, EAT boosts the performance by 10.40%.

Table 2: **Training hyperparameters for EAT.**

Parameter	Value
Base learning rate	1e-5
LR scheduler	Cosine decay to 0
Warmup steps	64
Batch size (per GPU)	1
Gradient accumulation	2
Max sequence length	13312
Weight decay	1e-4
Eviction ratio (min)	0.5
Eviction ratio (max)	0.9
Observation window	32
Pooling kernel	7
GPUs	32× NVIDIA A800
Distributed training	DeepSpeed ZeRO-2

Table 3: **Detailed per-task analysis** of Qwen2.5-3B under 50% RKV eviction.

Task	Task Type	Baseline 50% Evict	EAT 50% Evict	Gain
BoolQ	Yes/No QA	69.94	82.23	+12.29
GovReport	Long Doc Sum	16.91	27.45	+10.54
PIQA	Physical Reasoning	33.41	44.83	+11.42
VCsum	Video Summary	6.68	17.88	+11.20
2WikiMQA	Multi-hop QA	18.26	24.81	+6.55
Average	–	29.04	39.44	**+10.40**

D Additional Experimental Results

D.1 Scaling to Larger Models

Table 4 reports the per-task results for Qwen2.5-7B and Llama3.1-8B under the same W4A16 + KV-INT8 setting and 50% KV budget used in the main experiments. We use RKV throughout this scale-up study to keep the eviction policy fixed while varying model scale.

D.2 Component Ablations

Figure 5 complements Section 5.4. Panel (a) compares w/o STE mask and full EAT under 50% RKV eviction on Qwen2.5-3B; panel (b) compares fixed-ratio training (50%, 70%, or 90% evicted KV throughout training) against dynamic ratio sampling $r \sim \mathcal{U}(0.5, 0.9)$, evaluated at three deployment budgets.

The gains are broad rather than isolated. In particular, EAT improves both long-document summarization tasks (GovReport and VCsum) by more than ten points, while also producing substantial gains on reasoning-heavy benchmarks such as BoolQ and 2WikiMQA.

D.3 Effect of Model Family and Method

EAT provides consistent improvements across both model families and both eviction methods, with the largest gains observed for the RKV setting in our 50% budget experiments.

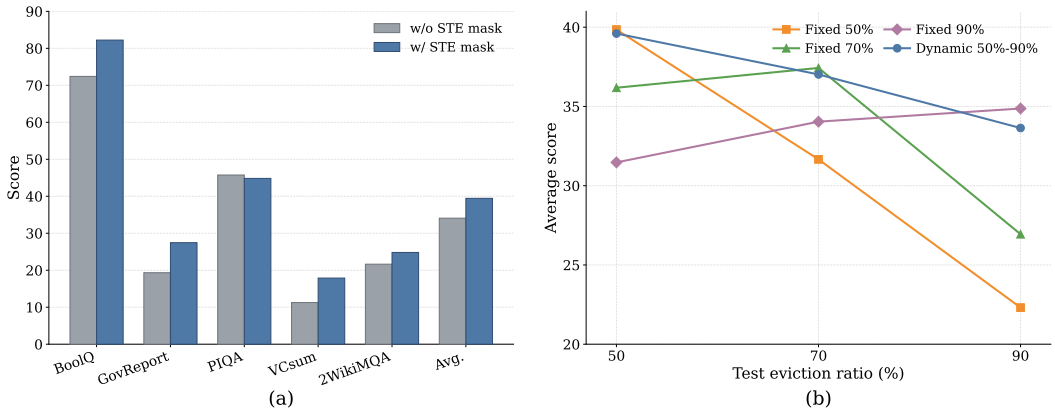

Figure 5: **Component ablations (detailed). (a)** Masked-STE ablation under 50% KV eviction on Qwen2.5-3B with RKV; both variants share the same forward eviction simulation and differ only in whether gradients are masked on evicted prompt positions. **(b)** Training-ratio ablation: fixed-ratio training vs. dynamic sampling from 50% to 90% evicted KV, evaluated at 50%, 70%, and 90% eviction ratio.

Table 4: **Per-task results on larger models under the 50% KV budget.** All scores use the same five-task evaluation suite as the main results.

Model / Method	BoolQ	GovReport	PIQA	VCsum	2WikiMQA	Avg.
Qwen2.5-7B QAT full	85.12	28.41	54.86	13.72	31.05	42.63
Qwen2.5-7B QAT+RKV	75.63	20.37	43.18	9.24	24.28	34.54
Qwen2.5-7B EAT+RKV	85.18	28.32	53.64	17.83	29.23	**42.84**
Llama3.1-8B QAT full	86.24	30.12	56.41	14.08	33.26	**44.02**
Llama3.1-8B QAT+RKV	77.46	21.82	41.28	10.63	25.71	35.38
Llama3.1-8B EAT+RKV	85.34	29.07	54.21	16.38	30.60	43.12

D.4 On-Device Deployment on MTK Dimensity 9500

Table 6 summarizes on-device metrics relative to the quantized full-KV baseline under one shared long-prompt decode configuration. Decode throughput and memory traffic totals are normalized so that **1.00**× denotes the full-KV reference; higher decode throughput is better, while lower memory traffic is better. The eviction row uses SnapKV at a 50% retained-KV budget with EAT, matching the deployment-time cache budget discussed in the main text.

E Assets, Licenses, and URLs

Table 7 summarizes the external datasets used in this work and their usage in this paper. Table 8 summarizes the pretrained base models used in our experiments. We cite the original sources and follow the corresponding licenses and terms of use.

Table 5: **Results across model families and eviction methods** at 50% KV eviction.

Model	Method	Evict.-unaware	EAT
Qwen2.5-3B	RKV	29.04	**39.44** (+10.40)
Qwen2.5-3B	SnapKV	30.31	**39.24** (+8.93)
Llama3.2-3B	RKV	31.69	**40.93** (+9.24)
Llama3.2-3B	SnapKV	30.38	**40.02** (+9.64)

Table 6: **On-device results on MTK Dimensity 9500.** Metrics are normalized to the quantized full-KV baseline under one shared long-prompt decode setting. Decode throughput improves by 64.1% (1.641×); total memory traffic decreases by 20.4% (0.796×).

Configuration	Decode Throughput	Memory Traffic
Quantized Full KV	1.00×	1.00×
SnapKV + EAT + 50% KV	1.641×	0.796×

Table 7: **Datasets used in this work, with usage, license, and source URL.**

Dataset	Usage in this paper	License	URL
BoolQ	Eval + Train	CC BY-SA 3.0	https://github.com/google-research-datasets/boolean-questions
GovReport	Train	CC BY 4.0	https://gov-report-data.github.io/
PIQA	Eval	AFL-3.0	https://yonatanbisk.com/piqa/
VCSum	Train	MIT	https://github.com/hahahawu/VCSum
LongBench (wrapper)	Eval source (GovReport,VCsum, 2WikiMQA)	MIT	https://github.com/THUDM/LongBench
Alpaca-GPT4 (EN/ZH)	Train	CC BY-NC 4.0	https://huggingface.co/datasets/vicgalle/alpaca-gpt4
CodeAlpaca-20k	Train	CC BY-NC 4.0 (data)	https://github.com/sahil280114/codealpaca
GSM8K	Train	MIT	https://huggingface.co/datasets/openai/gsm8k
MATH	Train	MIT	https://github.com/hendrycks/math
MathQA	Train	Apache-2.0	https://math-qa.github.io/

Table 8: **Pretrained models used in this work.**

Model	License	URL
Qwen2.5-3B-Instruct	Qwen Research License Agreement	https://huggingface.co/Qwen/Qwen2.5-3B-Instruct
Llama-3.2-3B-Instruct	Llama 3.2 Community License Agreement	https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct
Qwen2.5-7B-Instruct	Apache-2.0	https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
Llama-3.1-8B-Instruct	Llama 3.1 Community License Agreement	https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes].

   Justification: The main claims made in the abstract and the introduction (Section 1) matches the contribution elaborated in the method (Section 4) and verified in the experiments (Section 5).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes].

   Justification: The limitations of this work are discussed in the conclusion (Section 6).

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A].

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes].

   Justification: Details are disclosed in the Experiments (Section 5) and Appendix B and C.

   Guidelines:

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

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

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

Answer: [No].

Justification: We provide core implementation code in Appendix C. Due to corporate proprietary policies, the complete deployment repository is not publicly released.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes].

Justification: The experimental settings and details are available in Section 5 and Appendix C.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No].

Justification: Following standard evaluation protocols in the LLM literature (e.g., LongBench), all results are evaluated under deterministic greedy decoding across multi-task benchmark suites. Multi-run error bars were omitted due to the prohibitive computational cost of repeatedly fine-tuning and running inference across 13K+ context lengths for 3B, 7B, and 8B models.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes].

   Justification: The information regarding experimental compute resources is provided in Appendix C.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes].

   Justification: We have reviewed and made sure the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes].

    Justification: The broader impacts of this paper are discussed in Section 6.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A].

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes].

    Justification: Licenses for existing datasets and models are provided in Appendix E above.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A].

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

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