LAMPATTENTION: LOOK-AHEAD MIXED-PRECISION FLASHATTENTION FOR DEDICATED ACCELERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

While most attention logits can be computed in low precision without degrading numerical stability, current attention kernels fail to exploit this phenomenon. We introduce a novel hardware-algorithm co-design in the form of mixed-precision FlashAttention. Our method accumulates key-query products and evaluates their exponentials in 8-bit formats, then adaptively identifies sensitive sub-blocks and recomputes them in 16-bit formats. We propose the specifications for a dedicated accelerator capable of executing this pipeline efficiently. Simulated experiments with Qwen3 and Gemma 3 show that rerouting a selective minority of sub-blocks to high precision is sufficient to recover the baseline model performance.

1 INTRODUCTION

Mixed-precision floating-point arithmetic is a fundamental driver of scale in modern deep learning (Gupta et al., 2015; Micikevicius et al., 2018; Kalamkar et al., 2019; Micikevicius et al., 2022). In practice, this efficiency relies entirely on mixed-precision matrix products: the operands are stored in low precision, whereas the output is accumulated in high precision. As modern AI accelerators natively support these operations, they yield massive throughput gains across both training and inference. Furthermore, the numerical reliability of mixed-precision matrix products is guaranteed by well-established theoretical round-off error bounds (Blanchard et al., 2020; Abdelfattah et al., 2025).

Beyond isolated matrix products, the overall numerical stability of a neural network depends heavily on how local round-off errors propagate through the entire architecture and interact with its layers, especially the highly nonlinear attention layers of transformers (Budzinskiy et al., 2025). A standard, conservative strategy is to accumulate all matrix products and execute all intermediate computations in 32-bit arithmetic before passing the results down the computational pipeline. However, recent theoretical advances (El Arar et al., 2026; Budzinskiy et al., 2026) reveal that this approach is overly restrictive and introduce a dynamic alternative: a *look-ahead mixed-precision* (LAMP) framework. The core idea of LAMP is to maximize the use of low-precision arithmetic by default, using analytical error bounds to selectively trigger high-precision recomputation only where numerical stability is at risk. These works empirically validate LAMP on feedforward activations and transformer softmax functions, showing that large fractions of the directly preceding matrix products can be accumulated in low precision without degrading end-to-end inference accuracy.

A fundamental limitation of these prior theoretical works is that they are blind to physical hardware realities. Their LAMP algorithms fail to account for the structural constraints imposed by practical, hardware-aware kernels such as the industry-standard FlashAttention (Dao et al., 2022; Dao, 2024).

1.1 CONTRIBUTIONS

We address this gap by explicitly adapting the LAMP framework to FlashAttention. Specifically, we develop a hardware-algorithm co-design (Sections 2 and 3):

- *LampAttention*, a LAMP modification of FlashAttention that identifies numerically destabilizing logits in two stages, guided by running maxima and an analytical error bound;
- a specification for a hypothetical dedicated accelerator capable of executing LampAttention efficiently, featuring native 8-bit accumulation and LUT-based evaluation of exponentials.

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

LampAttention is distinctly not a quantization technique. Rather than compressing model weights or the input key-query-value operands, it exclusively minimizes the precision of intermediate calculations inside the attention kernel: the matrix accumulations and the softmax exponentials. Therefore, the algorithm operates independently of, and seamlessly alongside, external optimizations such as operand quantization or rotary positional encodings (Su et al., 2024).

Targeting the massive computational demands of LLM deployment, we evaluate LampAttention for inference across two modern LLM architectures, Qwen3 (Yang et al., 2025) and Gemma 3 (Kamath et al., 2025), measuring its impact on perplexity and downstream tasks (Section 4). Because commercially available hardware lacks native support for 8-bit accumulation, we simulate the arithmetic to guarantee numerical fidelity. Consequently, rather than reporting wall-clock timings, we quantify the volume of high-precision recomputations against task metrics, mapping the efficiency-accuracy trade-off attainable by future accelerators.

1.2 Related work

Transformers. The attention mechanism was originally introduced in Vaswani et al. (2017) in the context of machine translation and forms the basis of all modern transformers. These have achieved widespread success in a variety of natural language tasks, with prominent examples including BERT (Devlin et al., 2019), GPT (Brown et al., 2020), PaLM (Chowdhery et al., 2023), Llama (Touvron et al., 2023), Qwen (Yang et al., 2025), and DeepSeek (Liu et al., 2024). Transformers and related attention-based architectures have also enabled advances in scientific and mathematical applications, including accurate protein structure prediction with AlphaFold 2 (Jumper et al., 2021) and Olympiad-level geometry problem solving with AlphaGeometry (Trinh et al., 2024).

Efficient attention. Despite their success, transformers are bound by a fundamental bottleneck: the compute and memory costs of standard attention scale quadratically with sequence length. The family of FlashAttention algorithms (Dao et al., 2022; Dao, 2024; Shah et al., 2024; Zadouri et al., 2026) achieves linear memory cost by reorganizing the standard formulation in an I/O-aware manner. Sparse attention Beltagy et al. (2020); Kitaev et al. (2020) uses local sliding windows or fixed stride patterns to discard specific token interactions. Linear (Katharopoulos et al., 2020) and log-linear (Guo et al., 2026) variants of attention remove the softmax function. KV-caching (Dai et al., 2019) prevents redundant recomputation of past tokens during generation, while methods such as grouped-query attention (Ainslie et al., 2023) reduce the memory footprint of the KV-cache.

Quantization. To alleviate disk-storage and runtime memory requirements, quantization reduces the precision of model weights and activations. Weight-only quantization (Frantar et al., 2023; Lin et al., 2024) compresses static model parameters. Weight-activation quantization (Dettmers et al., 2022; Xiao et al., 2023) compresses both weights and activations, preparing them to become mixed-precision matrix-product operands. *LampAttention performs neither.* It exclusively controls the precision of calculations themselves.

Mixed-precision arithmetic. Frameworks for mixed-precision deep learning (Micikevicius et al., 2018; Kalamkar et al., 2019) accelerate training and inference by allocating different numerical formats to different components of the computational pipeline. This assignment is fundamentally inter-operational and homogeneous: both operands of a mixed-precision matrix product are quantized to one precision[1] and its entire output is accumulated in another precision; all of the softmax exponentials are computed in the same precision. In contrast, *LampAttention introduces intra-operational, heterogeneous mixed precision*.

2 Look-ahead mixed-precision computation of online softmax

The principal enabling feature of FlashAttention is the *online* computation of the softmax function

$$\varphi(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} \exp(x_i)} \begin{bmatrix} \exp(x_1) & \cdots & \exp(x_n) \end{bmatrix}^{\intercal}, \quad \mathbf{x} \in \mathbb{R}^n,$$

[1] Heterogeneous quantization of operands was proposed in Dettmers et al. (2022).

whereby an intermediate result is updated with new logits $\mathbf{y} \in \mathbb{R}^m$. Let $\varphi_{\mathbf{x}} : \mathbb{R}^m \to \mathbb{R}^{m+n}$ be the corresponding update function. It is computed as follows. The algorithm keeps track of

$$\mu \leftarrow \max_{1 \le i \le n} x_i, \quad \omega \leftarrow \sum_{i=1}^{n} \exp(x_i - \mu), \quad \mathbf{s} \leftarrow \varphi(\mathbf{x}),$$

and updates their running values according to

$$\hat{\mu} \leftarrow \max\left\{\mu, \max_{1 \le i \le m} y_i\right\}, \quad \hat{\alpha} \leftarrow \exp(\mu - \hat{\mu})\omega, \quad \hat{\mathbf{e}} \leftarrow [\exp(y_1 - \hat{\mu}) \quad \cdots \quad \exp(y_m - \hat{\mu})]^\intercal,$$

$$\hat{\omega} \leftarrow \hat{\alpha} + \|\hat{\mathbf{e}}\|_1, \quad \hat{\mathbf{s}} \leftarrow \frac{1}{\hat{\omega}} [\hat{\alpha}\mathbf{s}^\intercal \quad \hat{\mathbf{e}}^\intercal]^\intercal = \varphi_{\mathbf{x}}(\mathbf{y}).$$

2.1 Floating-point normalization

To analyze the effects of rounding errors on the computation of $\varphi_{\mathbf{x}}$, we focus on its compositional structure and introduce the ℓ_1 normalization function $\eta(\mathbf{z}) = \mathbf{z}/\|\mathbf{z}\|_1$, so that

$$\varphi_{\mathbf{x}}(\mathbf{y}) = \eta(\mathbf{z}), \quad \mathbf{z} = [\hat{\alpha}\mathbf{s}^\intercal \quad \hat{\mathbf{e}}^\intercal]^\intercal \in \mathbb{R}^{n+m}.$$

Then the componentwise error of floating-point (FP) evaluation of $\varphi_{\mathbf{x}}$ can be bounded by[2]

$$|\mathsf{fl}(\eta(\mathbf{z})) - \eta(\mathbf{z})| \le |\mathsf{fl}(\eta(\mathbf{z})) - \eta(\mathsf{fl}(\mathbf{z}))| + |\eta(\mathsf{fl}(\mathbf{z})) - \eta(\mathbf{z})|,$$

where the first term is the rounding error of normalization at the evaluated value of $\mathbf{z}$ and the second term is the discrepancy between the exactly normalized exact and evaluated values of $\mathbf{z}$. Denoting by u_η the FP precision used to evaluate η, we get with standard techniques (Higham, 2002)

$$|\mathsf{fl}(\eta(\mathbf{z})) - \eta(\mathbf{z})| \le \gamma_{\eta,m+1}\eta(\mathbf{z}) + (1+\gamma_{\eta,m+1})|\eta(\mathsf{fl}(\mathbf{z})) - \eta(\mathbf{z})|, \quad \gamma_{\eta,m+1} = \frac{(m+1)\mathrm{u}_\eta}{1-(m+1)\mathrm{u}_\eta}. \tag{1}$$

The bound is now determined by how the ℓ_1 normalization function η propagates the rounding errors in the computation of the exponentials and their arguments.

2.2 Floating-point inputs of normalization

Let us now consider the evaluation of $\mathbf{z}$, taking into account the rounding errors due to FP evaluation of logits $\mathbf{y}$. Specifically, suppose that $\Delta y_j = |\mathsf{fl}(y_j) - y_j| = \mathcal{O}(\mathrm{u}_{y_j})$ for $1 \le j \le m$, and denote $\Delta\hat{\mu} = |\mathsf{fl}(\hat{\mu}) - \hat{\mu}|$. Then standard worst-case rounding error analysis yields

$$|\mathsf{fl}(\hat{\alpha}s_i) - \hat{\alpha}s_i| \le \hat{\alpha}s_i\left[(1+\gamma_{\hat{\alpha},3})\exp\left(\mathrm{u}_{\hat{\alpha}}(\hat{\mu} - \mu + \Delta\hat{\mu}) + \Delta\hat{\mu}\right) - 1\right] = \hat{\alpha}s_i w_0,$$

$$|\mathsf{fl}(\hat{e}_j) - \hat{e}_j| \le \hat{e}_j\left[(1+\mathrm{u}_{\hat{e}_j})\exp\left(\mathrm{u}_{\hat{e}_j}(\hat{\mu} - y_j + \Delta\hat{\mu} + \Delta y_j) + \Delta\hat{\mu} + \Delta y_j\right) - 1\right] = \hat{e}_j w_j.$$

It follows from Taylor's expansion that

$$|\eta(\mathsf{fl}(\mathbf{z})) - \eta(\mathbf{z})| \le |\mathbf{J}_\eta(\mathbf{z})\mathrm{diag}(\mathbf{z})|\hat{\mathbf{w}} + \mathcal{O}(\|\hat{\mathbf{w}}\|_\infty^2), \quad \hat{\mathbf{w}} = [w_0\mathbf{1}_n^\intercal \quad \mathbf{w}^\intercal]^\intercal \in \mathbb{R}^{n+m}, \tag{2}$$

where $\mathbf{J}_\eta(\mathbf{z}) \in \mathbb{R}^{(n+m)\times(n+m)}$ is the Jacobian and $\hat{\mathbf{w}}$ describes the relative rounding errors of $\mathsf{fl}(\mathbf{z})$, where distinct precisions are used for its components.

It remains to bound $\Delta\hat{\mu}$, for which we introduce a reasonable assumption that rounding errors in the evaluation of $\mathbf{y}$ do not switch its maximizer, i.e., $y_c = \max_j y_j$ and $\mathsf{fl}(y_c) = \max_j \mathsf{fl}(y_j)$. Then

$$\Delta\hat{\mu} \le \begin{cases} 0, & \mu \ge \max\{y_c, \mathsf{fl}(y_c)\}, \\ \Delta y_c, & \text{otherwise.} \end{cases}$$

We make another structural decision and assume that $\hat{\alpha}$ is computed in high precision. This choice imposes negligible computational overhead since a single exponential is evaluated, and in the limiting case of $\mathrm{u}_{\hat{\alpha}} = 0$, which we shall adopt, it yields $w_0 = \exp(\Delta\hat{\mu}) - 1$.

[2] We denote by $\mathsf{fl}(\cdot)$ the FP evaluation of the entire expression, e.g., $\mathsf{fl}(\eta(\mathbf{z})) := \mathsf{fl}(\eta(\mathsf{fl}(\mathbf{z})))$.

2.3 PROPAGATION OF MIXED-PRECISION ROUNDING ERRORS

To simplify presentation, and without loss of generality, we let $\mathrm{u}_j = \mathrm{u}_{y_j} = \mathrm{u}_{\hat{e}_j}$ for each $1 \le j \le m$. Suppose that these precisions are divided into two categories:

$$\mathrm{u}_j = \begin{cases} \mathrm{u}, & j \notin \Omega, \\ \epsilon\mathrm{u}, & j \in \Omega, \end{cases} \quad \Omega \subseteq \{1, \dots, m\}, \quad \epsilon \ll 1.$$

That is, the exponential $\hat{e}_j$ and the logit y_j are evaluated in high precision when $j \in \Omega$ and in lower precision when $j \notin \Omega$. Henceforth, we resort to the limiting case of $\epsilon = 0$ to streamline the analysis.

Consider a binary vector $\mathbf{g} \in \{0,1\}^m$ that vanishes exactly on Ω. We can represent $\hat{\mathbf{w}}$ as

$$\hat{\mathbf{w}} = \mathrm{diag}(\mathbf{h})\hat{\mathbf{w}}, \quad \mathbf{h} = \begin{cases} \begin{bmatrix} \mathbf{0}_n^\intercal & \mathbf{g}^\intercal \end{bmatrix}^\intercal, & \mu \ge \max\{y_c, \mathsf{fl}(y_c)\} \text{ or } c \in \Omega, \\ \mathbf{1}_{n+m}, & \text{otherwise.} \end{cases} \tag{3}$$

The binary vector $\mathbf{h} \in \{0,1\}^{n+m}$ specifies how the rounding errors propagate through η.

Proposition 1. *Let* $\mathbf{z} \in \mathbb{R}_+^{n+m}$, $\hat{\mathbf{w}} \in \mathbb{R}^{n+m}$, *and* $\mathbf{h} \in \{0,1\}^{n+m}$. *For every* $1 \le l \le n+m$,

$$|\mathbf{e}_l^\intercal \mathbf{J}_\eta(\mathbf{z})\mathrm{diag}(\mathbf{z})\mathrm{diag}(\mathbf{h})|\hat{\mathbf{w}} \le \eta(\mathbf{z})_l \Big[\mathbf{h}^\intercal \eta(\mathbf{z}) + h_l\Big(1 - 2\eta(\mathbf{z})_l\Big)\Big] \|\hat{\mathbf{w}}\|_\infty.$$

Proof. See Appendix A. □

Let us combine Proposition 1 with error bounds (1) and (2). Summing over $1 \le l \le n+m$,

$$\|\mathsf{fl}(\eta(\mathbf{z})) - \eta(\mathbf{z})\|_1 \le \gamma_{\eta,m+1} + 2(1+\gamma_{\eta,m+1})\Big[\mathbf{h}^\intercal \mathrm{diag}\big(\mathbf{1}_{n+m} - \eta(\mathbf{z})\big)\eta(\mathbf{z})\Big]\|\hat{\mathbf{w}}\|_\infty + \mathcal{O}(\|\hat{\mathbf{w}}\|_\infty^2). \tag{4}$$

This bound controls the probability mass shift caused by the FP evaluation of softmax and is maximized when $\mathbf{h} = \mathbf{1}_{n+m}$, i.e., when the running maximum $\hat{\mu}$ is set to a low-precision logit.

2.4 LOOK-AHEAD DISTRIBUTION OF PRECISIONS

Our error bound (4) represents a trade-off between cheap, low-precision evaluation of softmax and its numerical stability: maximum efficiency is achieved with $\mathbf{g} = \mathbf{1}_m$, whereas $\mathbf{g} = \mathbf{0}_m$ guarantees unconditional numerical stability. This setting naturally lends itself to LAMP analysis (El Arar et al., 2026; Budzinskiy et al., 2026). Specifically, we can optimize

$$\mathbf{1}_m^\intercal \mathbf{g} \to \max \quad \text{s.t.} \quad \mathbf{h}^\intercal \mathrm{diag}\big(\mathbf{1}_{n+m} - \eta(\mathbf{z})\big)\eta(\mathbf{z}) \le \tau, \tag{5}$$

where $0 \le \tau < 1$ is a small threshold and $\mathbf{h}$ is derived from $\mathbf{g}$ via (3). However, we do not have access to the exact values of $\eta(\mathbf{z})$ in practice. Additionally, the explicit storage of the first n softmax probabilities would render any method inefficient in long-context scenarios.

A two-step procedure can overcome these issues. First, we look at the logits $\mathsf{fl}(\mathbf{y})$ evaluated in low precision and identify their maximizer $\mathsf{fl}(y_c)$. If $\mu > \mathsf{fl}(y_c) + |\mu|\delta$ with a safety margin $\delta \ge 0$ then we proceed; otherwise, $\mathsf{fl}(y_c)$ is recomputed in high precision. In either case, we end up in the first branch of (3),[3] and therefore the optimization problem (5) relies entirely on the newly added softmax probabilities. Denote by $\mathbf{p} \in \mathbb{R}^m$ these last m components of $\mathsf{fl}(\eta(\mathbf{z}))$ evaluated according to the precisions chosen for $\mathsf{fl}(\mathbf{y})$. Then we can reformulate (5) equivalently as

$$\mathbf{1}_m^\intercal \mathbf{g} \to \max \quad \text{s.t.} \quad \mathbf{g}^\intercal \mathrm{diag}\big(\mathbf{1}_m - \mathbf{p}\big)\mathbf{p} \le \tau, \quad g_c = 0 \text{ if } \mu \le \mathsf{fl}(y_c) + |\mu|\delta. \tag{6}$$

This is a knapsack problem with uniform values, and hence its optimal solution can be obtained with a greedy algorithm: select the smallest entries of $\mathrm{diag}\big(\mathbf{1}_m - \mathbf{p}\big)\mathbf{p}$ until the threshold is surpassed, subject to the constraint on g_c. The solution $\mathbf{g}$ determines the distribution of precisions across logits, and we reevaluate them, together with their exponentials, in higher precision accordingly. Finally, we recompute the ℓ_1 normalization for its updated inputs.

By restricting the binary vector $\mathbf{h}$ in (3) strictly to the first branch, we ensure that the LAMP evaluation of online softmax can be seamlessly integrated into FlashAttention. Indeed, the method relies only on the running maximum and the running normalization constant, completely avoiding the need to materialize the historical probability distribution.

[3] Technically, this would also require $|\mu|\delta$ to be greater than the largest Δy_j. While it is possible to use worst-case upper bounds on Δy_j, they tend to be overly pessimistic, so we prefer to use a tunable margin δ.

3 LOOK-AHEAD MIXED-PRECISION FLASHATTENTION

The framework developed in Section 2 deviates from the practice of FlashAttention in three aspects, two of which stem from the block-based nature of efficient matrix multipliers in modern AI accelerators. First, instead of looking at a single vector of logits $\mathbf{y} \in \mathbb{R}^m$ at a time, FlashAttention processes an entire block of queries simultaneously, leading to a matrix of logits $\mathbf{Y} \in \mathbb{R}^{m \times b_q}$. Second, assuming that the "atomic" matrix product is $(b_k \times b) \times (b \times b_q) \to (b_k \times b_q)$, the number of new keys per query is $m = b_k N_k$ and the (re)computation of logits is strictly blockwise. Third, the output of attention is scaled by the normalization constant only at the very end of the computational pipeline, i.e., FlashAttention keeps the intermediate exponentials unnormalized.

3.1 BLOCK MODIFICATION OF LAMP COMPUTATION OF ONLINE SOFTMAX

Let $\boldsymbol{\mu} \in \mathbb{R}^{b_q}$ be the vector of running maxima for each query and denote by $\mathbf{C} \in \{1, \dots, b_k\}^{N_k \times b_q}$ the matrix of maximum-logit indices for each sub-block of low-precision $\mathsf{fl}(\mathbf{Y})$. Consider

$$T_\theta = \max_{1 \le j \le b_q} \big\{ \mathsf{fl}(Y_{i,j}) - \mu_j \big\}, \quad i = C_{\theta,j} + (\theta - 1) b_k, \quad 1 \le \theta \le N_k.$$

This quantity represents the "threat" caused by the θth sub-block, and we assume that the sub-blocks are sorted so that $T_1 \ge \cdots \ge T_{N_k}$. Let $\hat{\boldsymbol{\mu}} = \boldsymbol{\mu}$ and process the sub-blocks sequentially. If

$$\hat{\mu}_j > \mathsf{fl}(Y_{i,j}) + |\hat{\mu}_j| \delta, \quad i = C_{\theta,j} + (\theta - 1) b_k \quad \text{for all} \quad 1 \le j \le b_q, \tag{7}$$

the θth sub-block of $\mathsf{fl}(\mathbf{Y})$ remains in low precision; otherwise, it is recomputed in high precision and the running maxima $\hat{\boldsymbol{\mu}}$ are updated. Denote by $\Theta \subseteq \{1, \dots, N_k\}$ the set of indices of recomputed sub-blocks. The role of sorting according to T_θ is to attempt to reduce the cardinality of Θ.

Next, we need to modify the LAMP problem (6). For every query $1 \le j \le b_q$, denote by $\hat{\mathbf{e}}_j \in \mathbb{R}^{b_k N_k}$ the vector of unnormalized shifted exponentials and by $\hat{\omega}_j \in \mathbb{R}$ the updated running normalization constant. Let $\mathbf{g} \in \{0, 1\}^{N_k}$ encode the precisions used for the sub-blocks, so that $g_\theta = 0$ whenever the θth sub-block requires high precision. Then we propose to solve

$$\begin{aligned} \mathbf{1}_{N_k}^\intercal \mathbf{g} \to \max \quad \text{s.t.} \quad & \xi_j(\mathbf{g}) \le \tau \hat{\omega}_j^2 \ \text{ for all } \ 1 \le j \le b_q, \quad g_\theta = 0 \ \text{ for } \ \theta \in \Theta, \\ & \xi_j(\mathbf{g}) = (\mathbf{g} \otimes \mathbf{1}_{b_k})^\intercal \mathrm{diag}\big(\hat{\omega}_j \mathbf{1}_{b_k N_k} - \hat{\mathbf{e}}_j\big) \hat{\mathbf{e}}_j. \end{aligned} \tag{8}$$

Here, $\otimes$ is the Kronecker product. Unlike (6), this modified optimization problem is a multidimensional knapsack problem (Kellerer et al., 2004) and, in general, does not admit a greedy solution. To solve it, we can rely on the number of sub-blocks N_k being small and compare all feasible configurations of $\mathbf{g}$ in the descending order of $\mathbf{1}_{N_k}^\intercal \mathbf{g}$. Upon finding an optimal precision distribution, we recompute the identified sub-blocks, their exponentials, and the running normalization constant.

3.2 LAMPATTENTION

We integrate the LAMP framework into the FlashAttention-2 kernel (Dao, 2024), omitting the warpgroup instructions and specialization of FlashAttention-3 (Shah et al., 2024) and the asynchronous computations with dedicated memory levels of FlashAttention-4 (Zadouri et al., 2026). We assume a standard accelerator architecture characterized by (i) a memory hierarchy that distinguishes between HBM, SRAM, and registers; (ii) a SIMT execution model partitioned into thread blocks and further subdivided into warps. The pseudocode of our *LampAttention* algorithm is listed in Algorithm 1.

The mixed-precision sections of Algorithm 1 are on lines 10–20 and 22–26, while the rest is vanilla FlashAttention-2. Note that (causal) masking and RoPE (Su et al., 2024) are seamlessly supported.

3.3 HARDWARE-ALGORITHM CO-DESIGN

In this section, we map the pseudocode of Algorithm 1 to the hardware of a hypothetical dedicated AI accelerator that would be able to execute it efficiently.

Floating-point formats. We propose the 8-bit `e4m3` format for the low-precision accumulation of key-query matrix products. However, this choice poses a strict upper limit of 448 on the maximum

Algorithm 1: LampAttention

Input: $\mathbf{Q}_{full}, \mathbf{K}_{full}, \mathbf{V}_{full} \in \mathbb{R}^{d\times L}$ stored in HBM for a single attention head and a single sequence of L tokens; query block size b_q and key/value block size b_k such that $L = b_q N_q M_q = b_k N_k M_k$; safety margin $\delta > 0$ of condition (7); threshold $0 \le \tau < 1$ of block-LAMP problem (8)

Output: attention $\mathbf{O}_{full} \in \mathbb{R}^{d\times L}$

foreach $\upsilon_q = 1, \dots, M_q$ *in parallel* **do**
  `/* single thread block with its SRAM section */`
  load υ_qth tile $[\mathbf{Q}_1 \ \cdots \ \mathbf{Q}_{N_q}] \in \mathbb{R}^{d\times b_q N_q}$ into SRAM
  initialize output $\mathbf{O} = [\mathbf{O}_1 \ \cdots \ \mathbf{O}_{N_q}] \leftarrow \mathbf{0}_{d\times b_q N_q}$
  initialize running quantities $\boldsymbol{\mu} = [\boldsymbol{\mu}_1^\intercal \ \cdots \ \boldsymbol{\mu}_{N_q}^\intercal]^\intercal \leftarrow (-\infty)\mathbf{1}_{b_q N_q}$ and $\boldsymbol{\omega} = [\boldsymbol{\omega}_1^\intercal \ \cdots \ \boldsymbol{\omega}_{N_q}^\intercal]^\intercal \leftarrow \mathbf{0}_{b_q N_q}$
  for $\upsilon_k = 1, \dots, M_k$ **do**
    load υ_kth tiles $\mathbf{K} = [\mathbf{K}_1 \ \cdots \ \mathbf{K}_{N_k}], \mathbf{V} = [\mathbf{V}_1 \ \cdots \ \mathbf{V}_{N_k}] \in \mathbb{R}^{d\times b_k N_k}$ into SRAM
    initialize $\hat{\boldsymbol{\mu}} \leftarrow \boldsymbol{\mu}$
    foreach $w = 1, \dots, N_q$ *in parallel* **do**
      `/* single warp with its SRAM subsection and registers */`
      accumulate $\mathbf{Y} \leftarrow \frac{1}{\sqrt{d}}\mathbf{K}^\intercal\mathbf{Q}_w \in \mathbb{R}^{b_k N_k \times b_q}$ in low precision
      find maximum-logit indices $\mathbf{C} \in \{1, \dots, b_k\}^{N_k \times b_q}$ for $\mathbf{Y}$ and compute $\mathbf{T} \in \mathbb{R}^{N_k}$
      sort $(\theta_1, \dots, \theta_{N_k})$ so that $T_{\theta_1} \ge \cdots \ge T_{\theta_{N_k}}$ and initialize $\Theta \leftarrow \varnothing$
      for $\theta = \theta_1, \dots, \theta_{N_k}$ **do**
        if *safety-margin condition* (7) *based on* $\hat{\boldsymbol{\mu}}_w$ *fails for* θ*th sub-block of* $\mathbf{Y}$ **then**
          accumulate $\mathbf{Y}_\theta \leftarrow \frac{1}{\sqrt{d}}\mathbf{K}_\theta^\intercal\mathbf{Q}_w \in \mathbb{R}^{b_k \times b_q}$ in high precision and update $\Theta \leftarrow \Theta \cup \{\theta\}$
          update $\hat{\boldsymbol{\mu}}_w \leftarrow \max\{\hat{\boldsymbol{\mu}}_w, \mathrm{colmax}(\mathbf{Y}_\theta)\}$
      for $\theta = 1, \dots, N_k$ **do**
        if $\theta \in \Theta$ **then**
          compute in-place $\hat{\mathbf{E}}_\theta \leftarrow \exp(\mathbf{Y}_\theta - \mathbf{1}_{b_k}\hat{\boldsymbol{\mu}}_w^\intercal)$ in high precision
        else
          compute in-place $\hat{\mathbf{E}}_\theta \leftarrow \exp(\mathbf{Y}_\theta - \mathbf{1}_{b_k}\hat{\boldsymbol{\mu}}_w^\intercal)$ in low precision
      compute $\hat{\boldsymbol{\lambda}} \leftarrow \exp(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)$, and $\hat{\boldsymbol{\alpha}} \leftarrow \mathrm{diag}(\hat{\boldsymbol{\lambda}})\boldsymbol{\omega}_w$, and $\hat{\boldsymbol{\omega}} \leftarrow \hat{\boldsymbol{\alpha}} + [\|\hat{\mathbf{e}}_1\|_1 \ \cdots \ \|\hat{\mathbf{e}}_{b_q}\|_1]^\intercal$
      obtain solution $\mathbf{g} \in \{0,1\}^{N_k}$ of block-LAMP problem (8)
      for $\theta = 1, \dots, N_k$ **do**
        if $g_\theta = 0$ *and* $\theta \notin \Theta$ **then**
          accumulate $\mathbf{Y}_\theta \leftarrow \frac{1}{\sqrt{d}}\mathbf{K}_\theta^\intercal\mathbf{Q}_w \in \mathbb{R}^{b_k \times b_q}$ in high precision
          compute in-place $\hat{\mathbf{E}}_\theta \leftarrow \exp(\mathbf{Y}_\theta - \mathbf{1}_{b_k}\hat{\boldsymbol{\mu}}_w^\intercal)$ in high precision
      update $\boldsymbol{\mu}_w \leftarrow \hat{\boldsymbol{\mu}}_w$ and $\boldsymbol{\omega}_w \leftarrow \hat{\boldsymbol{\alpha}} + [\|\hat{\mathbf{e}}_1\|_1 \ \cdots \ \|\hat{\mathbf{e}}_{b_q}\|_1]^\intercal$
      update $\mathbf{O}_w \leftarrow \mathbf{O}_w\mathrm{diag}(\hat{\boldsymbol{\lambda}}) + \mathbf{V}\hat{\mathbf{E}}$
  scale $\mathbf{O} \leftarrow \mathbf{O}\mathrm{diag}(\boldsymbol{\omega})^{-1}$ and write into υ_qth submatrix of $\mathbf{O}_{full}$ in HBM

allowed absolute value of a pre-softmax logit to avoid overflow. To guarantee this bound, we require that a transformer employ headwise QK-Norm with moderate gain weights, e.g., as in Qwen3 (Yang et al., 2025) and Gemma 3 (Kamath et al., 2025) models. For a typical head dimension of $d = 128$, the maximum attainable value of pre-softmax logit is $8\sqrt{2} \times \mathrm{maxgain}^2$. Thus, as long as the $1/\sqrt{d}$ scaling is applied before the matrix product is accumulated, the condition $\mathrm{maxgain} \le 6.29$ prevents overflow during `e4m3` accumulation. To execute LampAttention efficiently, a dedicated accelerator needs to support native packed `e4m3` accumulation, whereby 4 accumulators occupy a single 32-bit register. The swamping due to `e4m3` accumulation is typically benign for Algorithm 1: it primarily affects "unimportant" sub-blocks where the resulting exponential probabilities are vanishingly small, thus contributing no meaningful error to the final attention output. To mitigate its potential adversary effects, stochastic rounding (Croci et al., 2022) could be used instead of round-to-nearest.

Next, the logits are shifted by the running maximum, guaranteeing that the shifted logits are strictly non-positive. To account for the predetermined sign and the doubled absolute-value limit of 896, we propose to compute the shifted logits in-place using an unsigned `ue5m3` format.

The exponentials of the shifted logits cannot exceed one. Therefore, we can use the `ue5m3` format again—but with an adapted dynamic range—to compute the exponentials in-place. Namely, we can choose the exponent bias to map the bit pattern `11111000` to 1.0, which yields the smallest normalized number 2^{-30} and the smallest subnormal number 2^{-33}. Crucially, as both the exponentials and their operands are restricted to 8 bits, the evaluation can be physically implemented as a look-up table (LUT) and executed in one clock cycle, bypassing the multi-cycle special-function unit (SFU).

To carry out high-precision recomputations, we recall that the upper bound on the pre-softmax logits remains fixed. Therefore, we propose using a 16-bit `e4m11` format to accumulate matrix products, maximizing mantissa width, and an unsigned `ue5m11` format to evaluate the shifted exponentials. In this case, the accelerator must support native packed `e4m11` accumulation with two accumulators per register and must route the evaluation of exponentials through the SFU, since a 16-bit LUT would occupy a prohibitive amount of silicon space. In addition, while the running maxima of LampAttention require `e4m11` storage, they can be seamlessly rounded to `e4m3` for the low-precision shifts by truncating the lower mantissa bits.

Finally, to compute the attention output, the accelerator must support native matrix products where one operand is stored in the `ue5m3` or `ue5m11` format with the modified exponent bias.

Register layout. To execute the inner loop over w in Algorithm 1, each warp must have a sufficient number of registers for both the initial 8-bit results and their 16-bit refinements. A "safe" layout, which aligns perfectly with the pseudocode, is to reserve enough registers to store all N_k sub-blocks of $\mathbf{Y}$ in high precision. While this layout accommodates the worst-case scenario, it allows only two 8-bit accumulators to reside in a register, leaving space for potential recomputations, and therefore half of the register space is wasted in the best-case scenario without any recomputations.

In practice, softmax probability distributions tend to be highly concentrated for pre-trained models, and we thus expect the recomputations to be sparse. For a "compact" layout, we propose to separate 8-bit and 16-bit storage in the register space. Namely, sufficient registers are reserved for the entire low-precision $\mathbf{Y}$ with four 8-bit accumulators per register without padding. Furthermore, we require additional register space to store a single high-precision sub-block of $\mathbf{Y}$. In those instances where LampAttention marks two or more sub-blocks for recomputation, the algorithm should branch to a sequential, fully 16-bit update of $\mathbf{O}_w$, reusing the registers for $\lfloor N_k/2 \rfloor + 1$ high-precision sub-blocks.

Sources of speedup. The hardware-algorithm co-design of LampAttention serves to maximize the speedup relative to vanilla FlashAttention-2. Independent of the register layout, the primary latency reduction stems from the exponentials: evaluating an 8-bit exponential in one clock cycle via a LUT is drastically more efficient than routing a high-precision value through a multi-cycle SFU.[4] This operation-level efficiency is present in both the "safe" and "compact" layouts. At the same time, the identification of salient sub-blocks and their recomputation introduces additional latency; while we purposefully omit warp-group specialization in the present paper, it could be used to mask this latency overhead (Shah et al., 2024).

In contrast, the "compact" layout improves computational throughput by reducing register pressure. Packing four 8-bit accumulators into a single 32-bit register reduces the accumulator footprint by 75%. This enables a two-step hardware optimization: first, tile dimensions can be scaled up to the physical SRAM limit to minimize HBM reads; second, the remaining register savings increase warp occupancy to better hide memory-transfer latency. Provided that "disruptive" recomputations of two or more sub-blocks remain infrequent, this dual scaling can potentially yield consistent speedups.

4 Numerical experiments

4.1 Setup and implementation

We implement LampAttention as a Triton kernel with simulated 8-bit and 16-bit arithmetic. Namely, the corresponding numbers are stored in FP32 with mantissas truncated to 3 and 11 bits, respectively.

[4] The evaluation of exponentials is one of the two main bottlenecks addressed in FlashAttention-4 (Zadouri et al., 2026), which bypasses the SFU by proposing a low-degree polynomial approximation mapped to ALUs. In contrast, we resolve this natively without occupying either the SFU or ALU.

The low-precision accumulation of "atomic" matrix products is carried out via recursive summation with FMA, i.e., based on $\mathrm{round}(c + a \cdot b)$, where the scalar multiplication and addition are in FP32. Low-precision exponentials are modeled as correctly rounded, $\mathrm{round}(\exp a)$. The tile sizes are fixed as $b_q N_q = 128$ and $b_k N_k = 64$ with $N_k = 4$ key sub-blocks.

To validate the performance of LampAttention, we evaluate it on models from the Qwen3 (8B, 30B-MoE, 32B) and Gemma 3 (12B, 27B) families. The LampAttention kernel is dynamically injected into the standard HuggingFace implementations at runtime. We compare this computation against two fixed-precision baselines: the vanilla 32-bit baseline and an 8-bit LampAttention baseline restricted from utilizing any 16-bit recomputations. We stress that these bit-widths refer strictly to the attention-logit accumulators and their exponentials, rather than the quantization of model weights. Task metrics are measured using the EleutherAI lm-evaluation-harness.

The code and the raw numerical results (including those not reported in the paper) are provided in the Supplementary Material. The experiments were performed on a cloud-based GPU equipped with 80 GB of VRAM and required approximately 540 GPU-hours.

4.2 Numerical results

Tables 1 and 2 show that the adaptive 16-bit recomputations of LampAttention substantially improve upon the pure 8-bit baseline across both perplexity (C4) and downstream (MMLU) tasks; two more tasks are evaluated in Appendix B. Even with a high threshold of $\tau = 0.5$ in (8), i.e., with a small number of recomputations during the second stage of Algorithm 1, the recovered metrics broadly approach the 32-bit vanilla baselines (although the degree of closeness remains model- and task-dependent). Crucially, the recovery is achieved with only about 20-30% of key tiles being "disruptive" for the "compact" layout, requiring the recomputation of two or more sub-blocks.

Table 1: LampAttention with $\delta = 2^{-8}$ and $\tau = 2^{-1}$ on C4.

	Perplexity (↓)			**16-bit sub-blocks per tile**		
Model name	8-bit	**8/16-bit LAMP**	32-bit	0	1	2+
gemma-3-27b-pt	21.470	**17.150**	16.540	46.30%	20.52%	33.18%
gemma-3-12b-pt	36.740	**21.830**	18.680	50.94%	20.30%	28.76%
Qwen3-32B	28.180	**23.740**	23.800	65.09%	15.43%	19.48%
Qwen3-8B	44.030	**33.780**	33.340	66.19%	14.58%	19.23%
Qwen3-30B-A3B	43.900	**28.270**	27.510	69.75%	13.71%	16.54%

Table 2: LampAttention with $\delta = 2^{-8}$ and $\tau = 2^{-1}$ on 5-shot MMLU.

	Accuracy (↑)			**16-bit sub-blocks per tile**		
Model name	8-bit	**8/16-bit LAMP**	32-bit	0	1	2+
gemma-3-27b-pt	0.6261	**0.7851**	0.7993	48.49%	21.55%	29.96%
gemma-3-12b-pt	0.3618	**0.5899**	0.7654	53.39%	22.19%	24.42%
Qwen3-32B	0.7555	**0.8355**	0.8344	54.52%	19.67%	25.81%
Qwen3-8B	0.5625	**0.7412**	0.7840	54.05%	19.11%	26.84%
Qwen3-30B-A3B	0.5998	**0.7884**	0.8169	56.97%	18.99%	24.04%

Furthermore, Figure 1 depicts the empirical efficiency-accuracy trade-off of LampAttention across varying thresholds. As τ decreases, task performance converges toward the 32-bit vanilla baseline, though non-monotonically for smaller values of τ. This observation indicates that enforcing higher precision on intermediate computations does not unconditionally guarantee superior metrics; in fact, Table 4 in Appendix B illustrates an instance where the 8-bit baseline accuracy exceeds the 32-bit baseline. In contrast, the rate of "disruptive" recomputations increases monotonically as τ tends to zero, reflecting the increasingly stringent constraints of the block-LAMP optimization problem (8).

Figure 1 also provides a new perspective on the results in Tables 1- 2. We observe that the empirical difference between $\tau = 1$ and $\tau = 0.5$ is marginal in terms of both recomputation rate and accuracy. Because the optimal solution to (8) at $\tau = 1$ strictly omits stage-two recomputations, we can deduce the structural division of labor within the algorithm:

- the first, running-maximum stage of LampAttention is responsible for the "heavy lifting" of the evaluation metrics from the 8-bit baseline toward the 32-bit baseline using a moderate budget of "disruptive" recomputations for the compact register layout;
- the second stage, solving (8), acts as a fine-grained correction mechanism that closes the final metric gap at the cost of additional recomputations.

Finally, Figure 2 visualizes the dynamics of the recomputation cascade as the threshold τ tightens: the mass of zero-recomputation cases systematically transfers into full-recomputation cases.

Additional experimental results are presented in the Appendix.

Our findings highlight the potential utility of incorporating the new mixed-precision paradigm, the LAMP, into FlashAttention. They expose a controllable trade-off between inference efficiency and downstream performance without resorting to the 32-bit arithmetic for the computationally intensive sections of the algorithm.

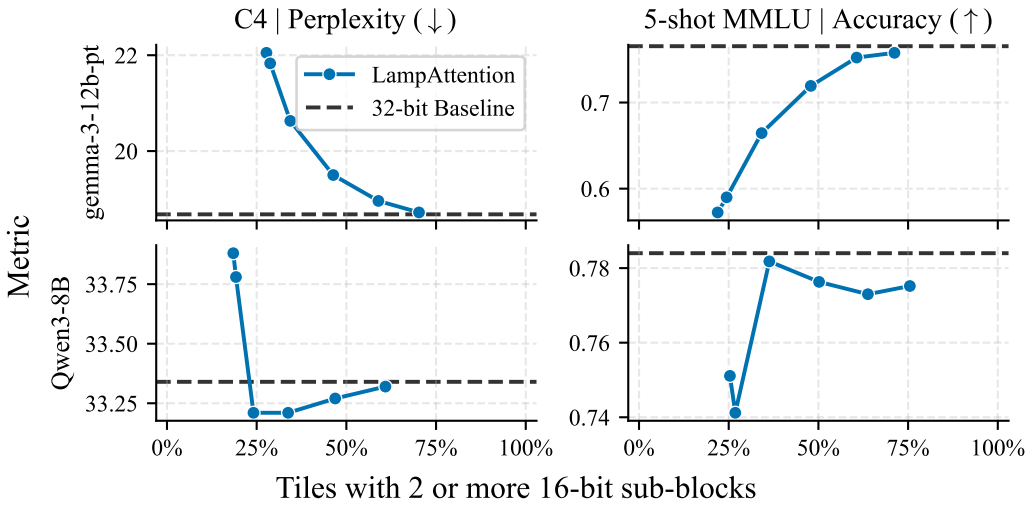

Figure 1: LampAttention with $\delta = 2^{-8}$ and $\tau \in \{2^{-t} : t = 0, \ldots, 5\}$.

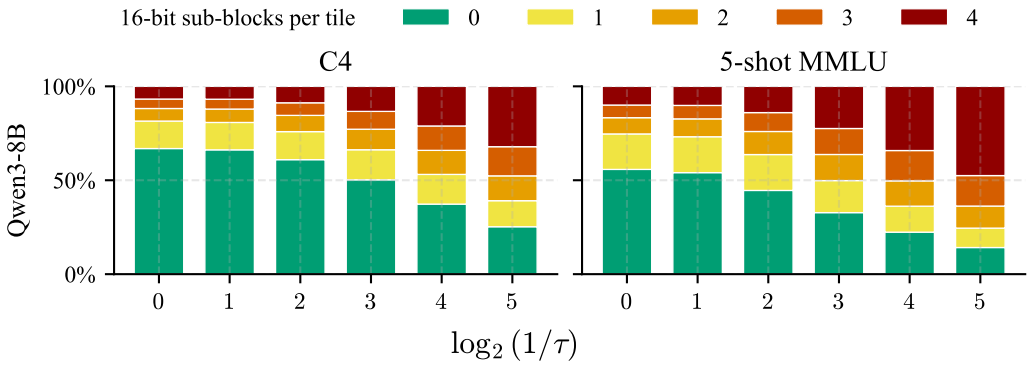

Figure 2: LampAttention with $\delta = 2^{-8}$ and $\tau \in \{2^{-t} : t = 0, \ldots, 5\}$.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of machine learning by improving the efficiency of large language models. Our method has the potential to reduce the energy consumption required to deploy these models, contributing to "Green AI" initiatives.

REPRODUCIBILITY STATEMENT

To ensure full reproducibility of the results, we have included in the Supplementary Material the complete source code, the configurations of the Python environment, the exact evaluation script, and hyperparameters used during the experiments.

AI USE STATEMENT

In this work, we used generative AI tools to assist with software development and to refine the clarity and flow of the text. All core research ideation, theoretical development, experimental design, and data analysis were conducted entirely by the authors. We have reviewed all AI-assisted work and assume full responsibility for the accuracy, integrity, and originality of this work.

REFERENCES

Ahmad Abdelfattah, Jack Dongarra, Massimiliano Fasi, Mantas Mikaitis, and Françoise Tisseur. Analysis of floating-point matrix multiplication computed via integer arithmetic. *arXiv*, art. 2506.11277, 2025. doi: 10.48550/arXiv.2506.11277.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *EMNLP*, pp. 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv*, art. 2004.05150, 2020. doi: 10.48550/arXiv.2004.05150.

Pierre Blanchard, Nicholas J Higham, Florent Lopez, Theo Mary, and Srikara Pranesh. Mixed precision block fused multiply-add: Error analysis and application to gpu tensor cores. *SIAM J Sci Comput*, 42(3):C124–C141, 2020. doi: 10.1137/19M1289546.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, pp. 1877–1901, 2020. doi: 10.48550/arXiv.2005.14165.

Stanislav Budzinskiy, Wenyi Fang, Longbin Zeng, and Philipp Petersen. Numerical stability analysis of large language models. *arXiv*, art. 2503.10251, 2025. doi: 10.48550/arXiv.2503.10251.

Stanislav Budzinskiy, Marian Gloser, Tolunay Yilmaz, Ying Hong Tham, Yuanyi Lin, Wenyi Fang, Fan Wu, and Philipp Petersen. LAMP: Look-ahead mixed-precision inference of large language models. *arXiv*, art. 2601.21623, 2026. doi: 10.48550/arXiv.2601.21623.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *J Mach Learn Res*, 24(240):1–113, 2023. doi: 10.48550/arXiv.2204.02311.

Matteo Croci, Massimiliano Fasi, Nicholas J Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: implementation, error analysis and applications. *R Soc Open Sci*, 9(3):211631, 2022. doi: 10.1098/rsos.211631.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, pp. 2978–2988, 2019. doi: 10.18653/v1/P19-1285.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024. doi: 10.48550/arXiv.2307.08691.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, pp. 16344–16359, 2022. doi: 10.5555/3600270.3601459.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *NeurIPS*, pp. 30318–30332, 2022. doi: 10.48550/arXiv.2208.07339.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, volume 1, pp. 4171–4186, 2019. doi: 10.48550/arXiv.1810.04805.

El-Mehdi El Arar, Silviu-Ioan Filip, Theo Mary, and Elisa Riccietti. Mixed precision accumulation for neural network inference guided by componentwise forward error analysis. *IMA Journal of Numerical Analysis*, art. drag054, 2026. doi: 10.1093/imanum/drag054/8785932.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023. doi: 10.48550/arXiv.2210.17323.

Guo Guo, Songlin Yang, Tarushii Goel, Eric P Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *ICLR*, pp. 91296–91317, 2026. doi: 10.48550/arXiv.2506.04761.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *ICML*, volume 37, pp. 1737–1746, 2015. doi: 10.48550/arXiv.1502.02551.

Nicholas J Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2 edition, 2002. doi: 10.1137/1.9780898718027.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of BFLOAT16 for deep learning training. *arXiv*, art. 1905.12322, 2019. doi: 10.48550/arXiv.1905.12322.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, S Perrin, T Matejovicova, A Ramé, M Rivière, L Rouillard, et al. Gemma 3 technical report. *arXiv*, art. 2503.19786, 2025. doi: 10.48550/arXiv.2503.19786.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, pp. 5156–5165, 2020. doi: 10.48550/arXiv.2006.16236.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004. doi: 10.1007/978-3-540-24777-7.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. doi: 10.48550/arXiv.2001.04451.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *MLSys*, volume 6, pp. 87–100, 2024. doi: 10.48550/arXiv.2306.00978.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 Technical report. *arXiv*, art. 2412.19437, 2024. doi: 10.48550/arXiv.2412.19437.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *ICLR*, 2018. doi: 10.48550/arXiv.1710.03740.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. FP8 formats for deep learning. *arXiv*, art. 2209.05433, 2022. doi: 10.48550/arXiv.2209.05433.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *NeurIPS*, pp. 68658–68685, 2024. doi: 10.5555/3737916.3740109.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv*, art. 2302.13971, 2023. doi: 10.48550/arXiv.2302.13971.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. doi: 10.1038/s41586-023-06747-5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017. doi: 10.48550/arXiv.1706.03762.

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 *ICML*, pp. 38087–38099, 2023. doi: 10.48550/arXiv.2211.10438.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv*, art. 2505.09388, 2025. doi: 10.48550/arXiv.2505.09388.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. FlashAttention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. In *MlSys*, 2026. doi: 10.48550/arXiv.2603.05451.

A Proof of Proposition 1

The ℓ_1 normalization function η is differentiable when every entry of $\mathbf{z}$ is nonzero; in our case, they are positive. With $\boldsymbol{\eta} = \eta(\mathbf{z})$, direct calculation yields

$$\mathbf{A} = \mathbf{J}_\eta(\mathbf{z})\mathrm{diag}(\mathbf{z})\mathrm{diag}(\mathbf{h}) = (\mathbf{I} - \boldsymbol{\eta}\mathbf{1}^\intercal)\mathrm{diag}(\boldsymbol{\eta})\mathrm{diag}(\mathbf{h}).$$

Since $\eta_i \in (0, 1)$ for each i, the diagonal entries of $\mathbf{A}$ are nonnegative, and hence

$$|\mathbf{e}_l^\intercal \mathbf{A}|\hat{\mathbf{w}} \leq \mathbf{e}_l^\intercal \mathbf{A}\mathbf{1}\|\hat{\mathbf{w}}\|_\infty = \left[\eta_l \sum_{i \neq l} \eta_i h_i + (1 - \eta_l)\eta_l h_l\right]\|\hat{\mathbf{w}}\|_\infty = \eta_l \left[\mathbf{h}^\intercal \boldsymbol{\eta} + h_l(1 - 2\eta_l)\right]\|\hat{\mathbf{w}}\|_\infty.$$

B Additional benchmarks

In addition to the C4 and MMLU tasks presented in the main text, we have evaluated the performance of LampAttention on Wikitext and ARC-Challenge; see Tables 3 and 4. Notably, the 8-bit accuracy baseline of Qwen3-32B on ARC-Challenge is higher than the corresponding 32-bit baseline, indicating that there is no strict causal relationship between higher-precision intermediate calculations and higher final accuracy. Figure 3 is an extension of Figure 2 that contains all models and tasks.

Table 3: LampAttention with $\delta = 2^{-8}$ and $\tau = 2^{-1}$ on Wikitext.

	Perplexity (↓)			**16-bit sub-blocks per tile**		
Model name	8-bit	**8/16-bit LAMP**	32-bit	0	1	2+
gemma-3-27b-pt	8.6290	**6.2100**	5.9000	47.21%	19.86%	32.93%
gemma-3-12b-pt	15.940	**8.6370**	7.3790	52.86%	19.10%	28.04%
Qwen3-32B	12.170	**9.4290**	9.3610	76.50%	11.55%	11.95%
Qwen3-8B	17.480	**12.510**	12.300	76.44%	11.01%	12.55%
Qwen3-30B-A3B	18.780	**11.260**	10.890	81.82%	9.02%	9.16%

Table 4: LampAttention with $\delta = 2^{-8}$ and $\tau = 2^{-1}$ on 0-shot ARC-Challenge.

	Accuracy (↑)			**16-bit sub-blocks per tile**		
Model name	8-bit	**8/16-bit LAMP**	32-bit	0	1	2+
gemma-3-27b-pt	0.5977	**0.6484**	0.6562	48.47%	30.41%	21.12%
gemma-3-12b-pt	0.4570	**0.6191**	0.6387	48.71%	32.21%	19.08%
Qwen3-32B	0.6191	**0.6016**	0.6094	48.54%	31.22%	20.24%
Qwen3-8B	0.4766	**0.5352**	0.5605	48.24%	28.71%	23.05%
Qwen3-30B-A3B	0.4512	**0.5176**	0.5762	48.28%	28.94%	22.78%

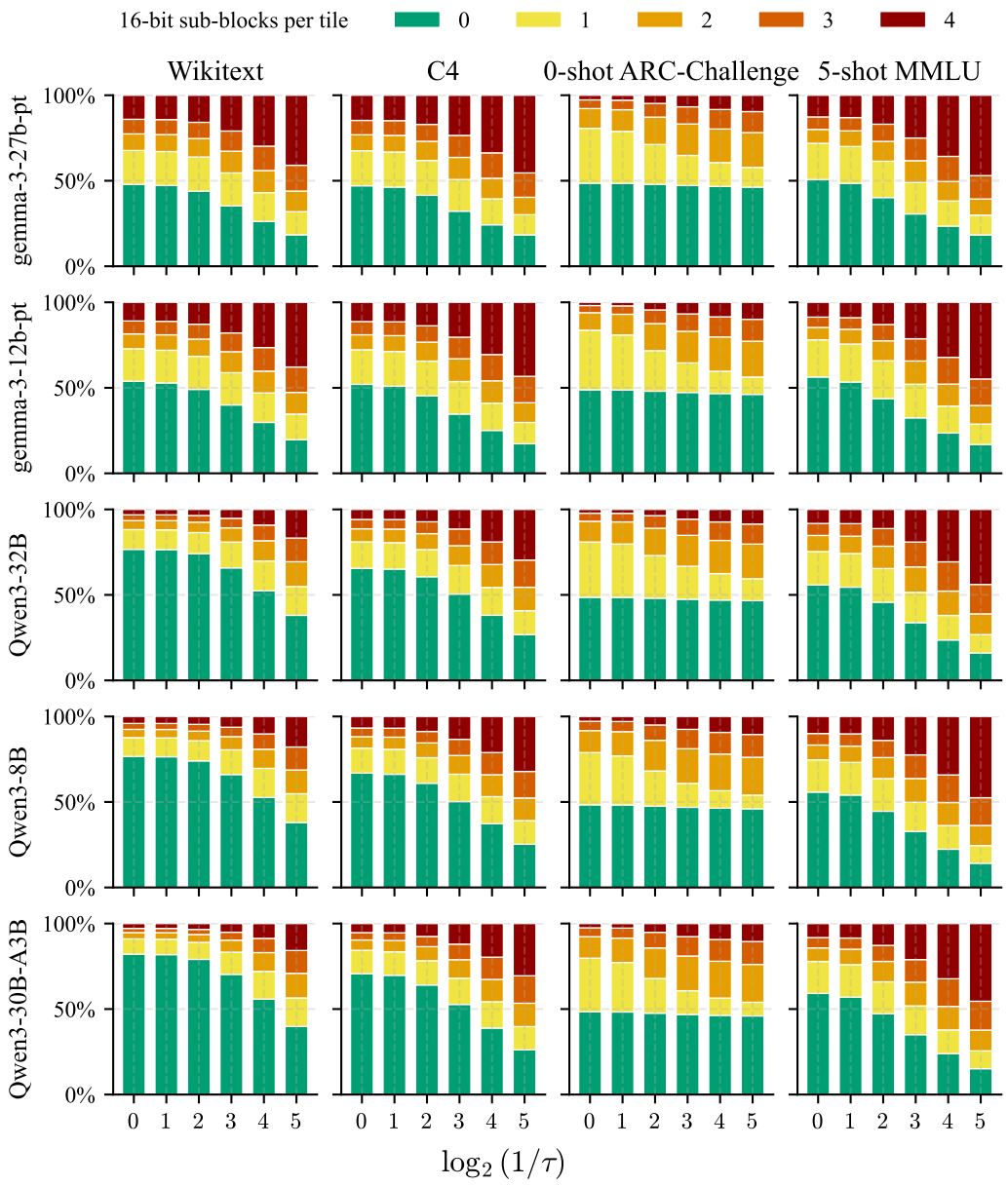

Figure 3: LampAttention with $\delta = 2^{-8}$ and $\tau \in \{2^{-t} : t = 0, \ldots, 5\}$.

C IMPACT OF SAFETY MARGIN

Here, we compare the performance of LampAttention with two different values of the safety margin δ from the first stage of the algorithm. As Figures 4 and 5 show, larger δ results in more "disruptive" recomputations being initiated during the first stage (which $\tau = 1$ isolates). At the same time, the two trade-off curves exhibit similar convergent behavior as the threshold τ tightens.

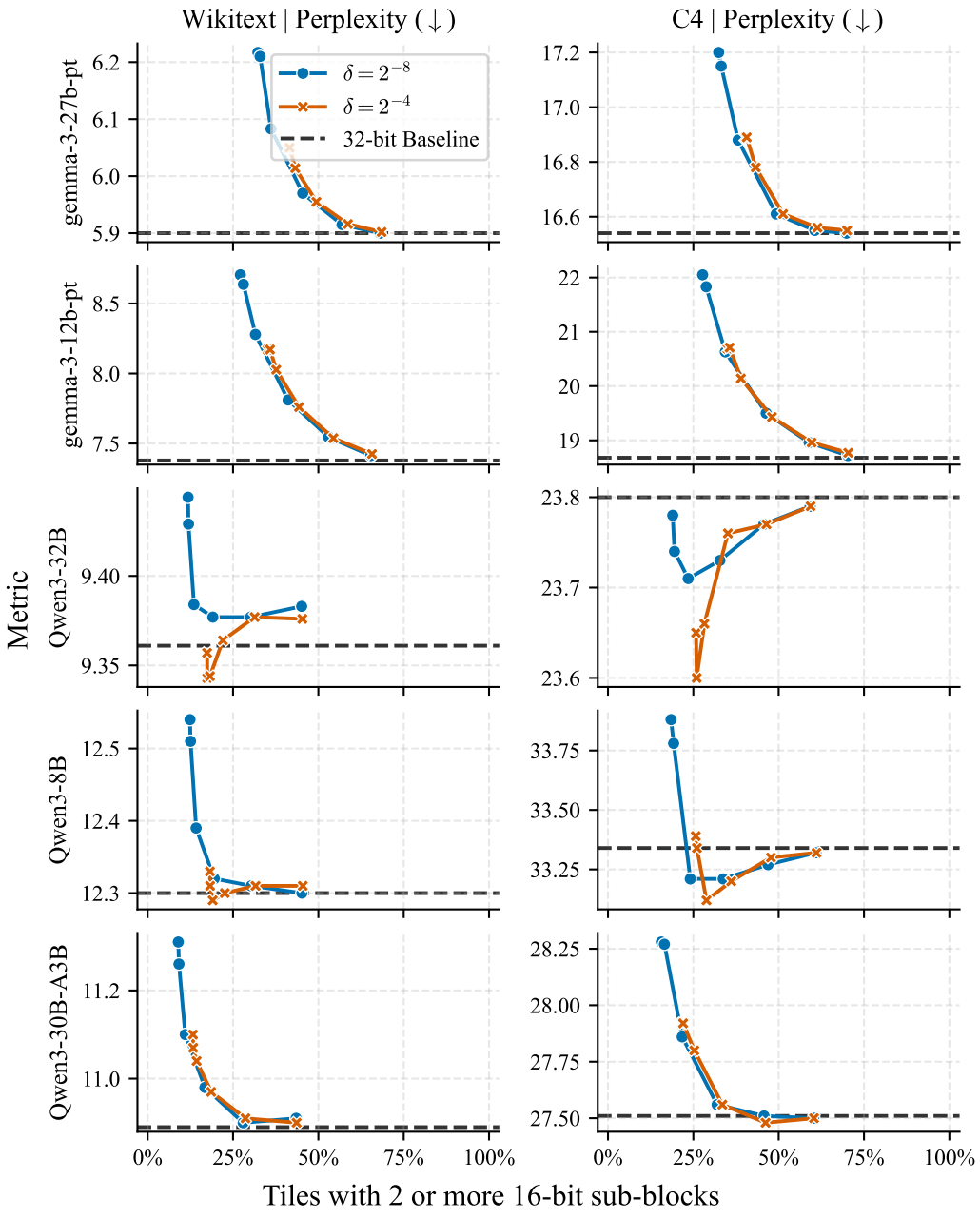

Figure 4: LampAttention with $\delta \in \{2^{-8}, 2^{-4}\}$ and $\tau \in \{2^{-t} : t = 0, \ldots, 5\}$.

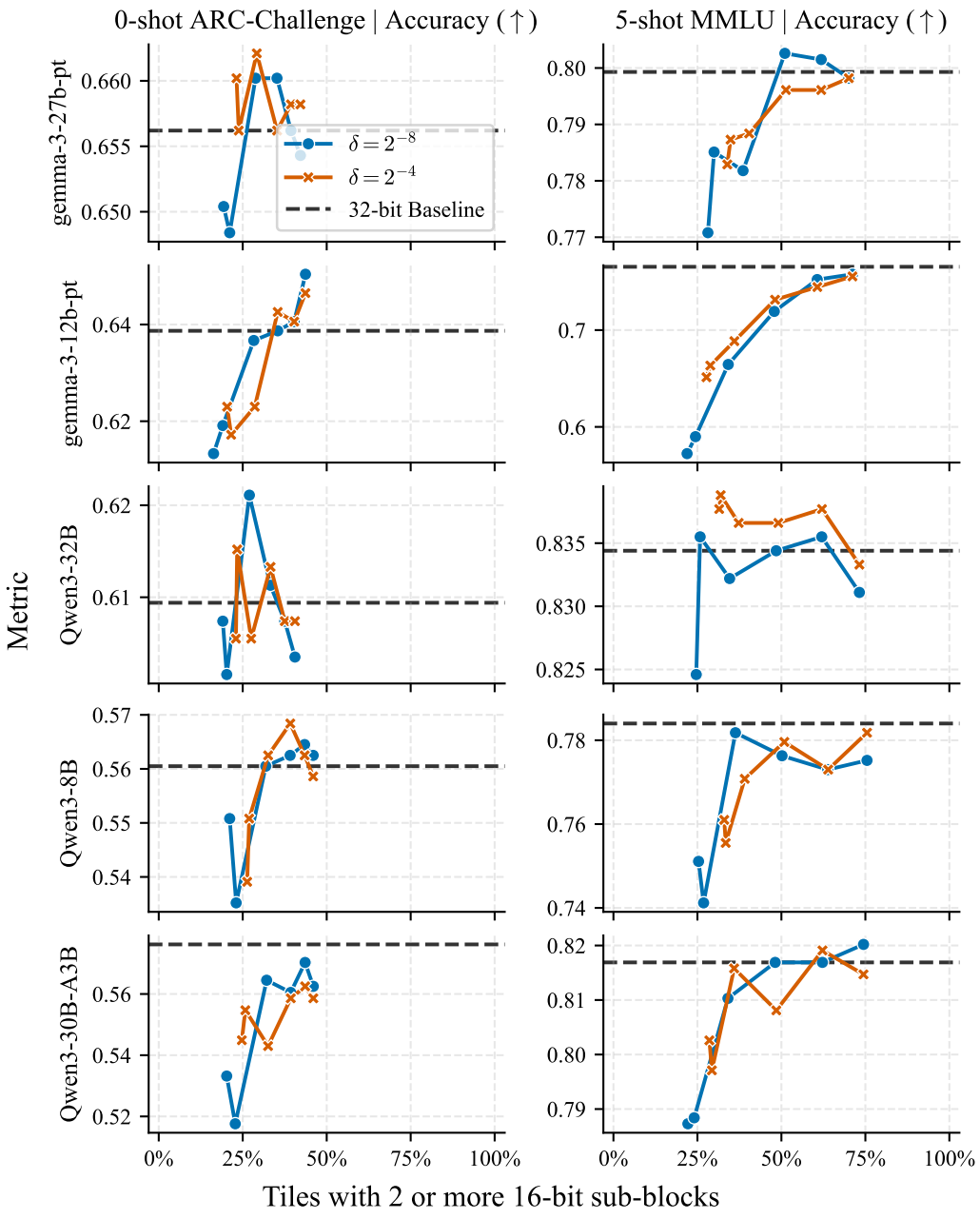

Figure 5: LampAttention with $\delta \in \{2^{-8}, 2^{-4}\}$ and $\tau \in \{2^{-t} \ : \ t = 0, \ldots, 5\}$.

D IMPACT OF FIRST-STAGE RECOMPUTATION

While setting $\tau = 1$ effectively disables the second recomputation stage in Algorithm 1, we can choose to disable the first stage instead (even though this violates the theoretical assumptions underlying the derivation of the block-LAMP problem (8)). Figures 6 and 7 show the corresponding trade-off curves. Taken together with the results in Appendix C, they suggest that the specific mechanism triggering a recomputation is secondary; whether driven by the first-stage running maximum or the second-stage threshold, both phases identify fundamentally "important" sub-blocks.

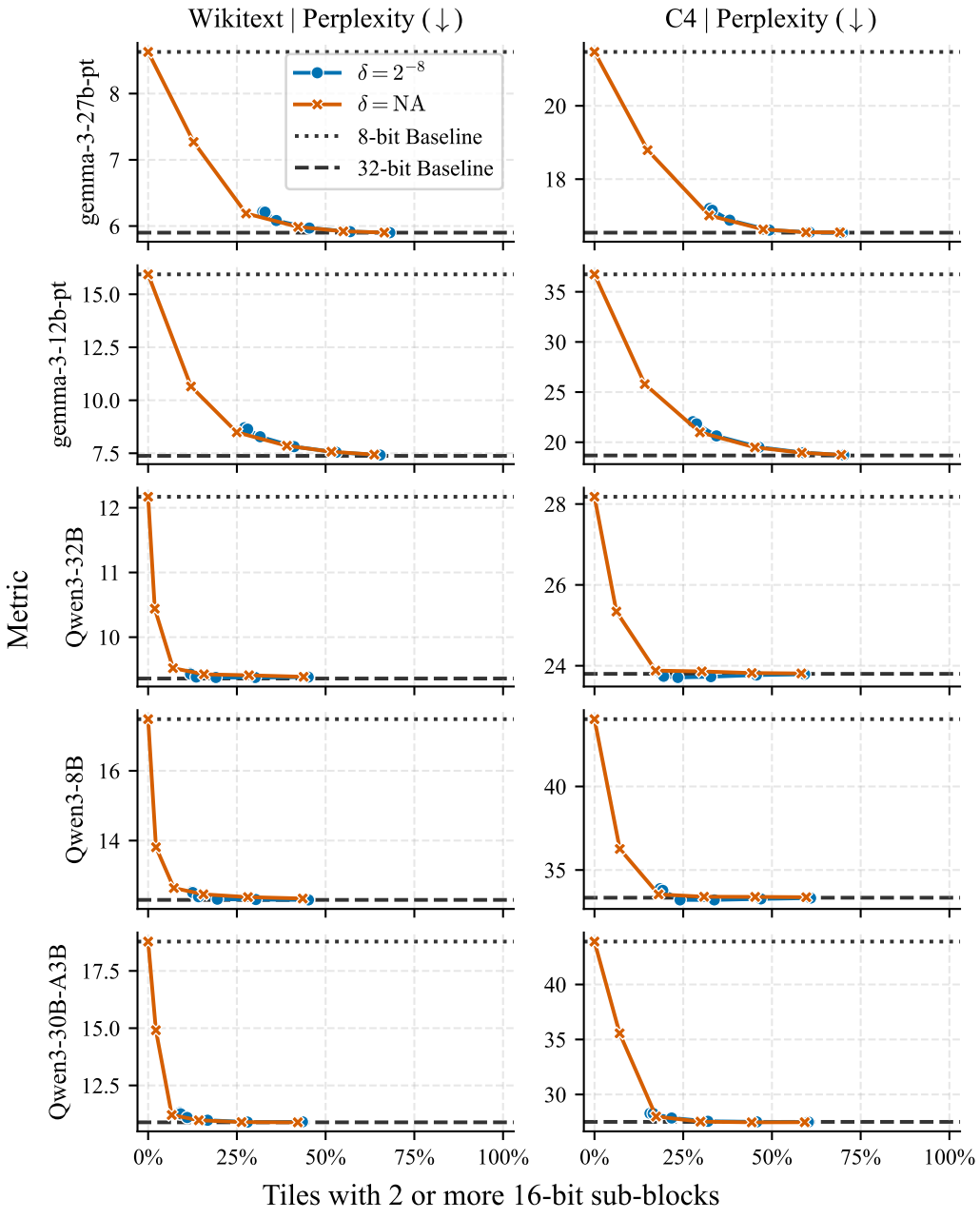

Figure 6: LampAttention with $\delta = 2^{-8}$ or disabled first stage, and $\tau \in \{2^{-t} : t = 0, \ldots, 5\}$.

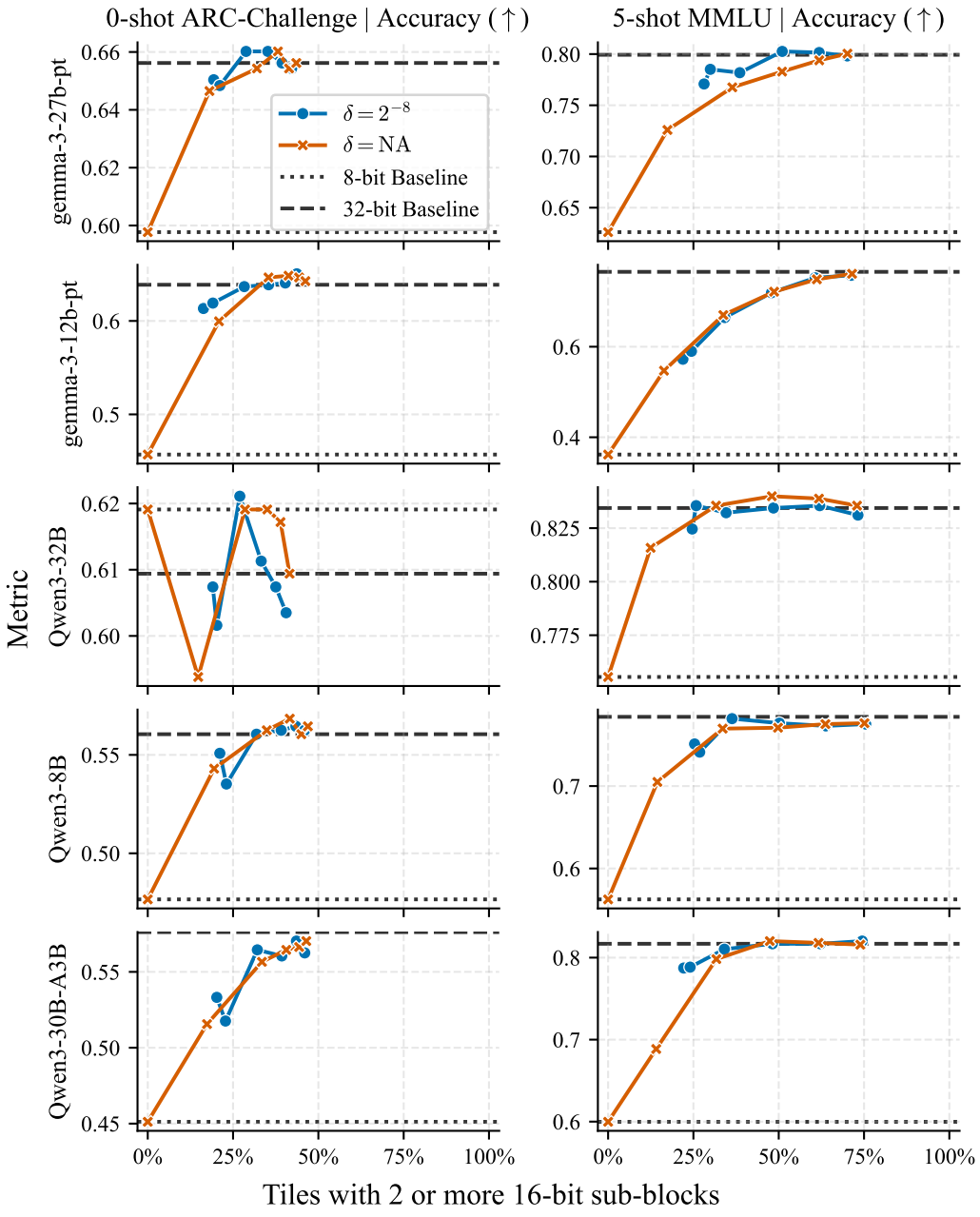

Figure 7: LampAttention with $\delta = 2^{-8}$ or disabled first stage, and $\tau \in \{2^{-t} : t = 0, \ldots, 5\}$.

E IMPACT OF IN-CONTEXT LEARNING

We assess how task familiarity affects the performance of LampAttention by comparing 0-shot and 25-shot evaluations on ARC-Challenge. Tables 5 and 6 (a reproduction of Table 4 for convenience) demonstrate a dual benefit: the few-shot setting naturally increases the accuracy while simultaneously expanding the share of tiles processed entirely in 8-bit. Interestingly, Figure 8 reveals that the distribution of recomputed sub-blocks per tile evolves distinctively between the two task instances.

Table 5: LampAttention with $\delta = 2^{-8}$ and $\tau = 2^{-1}$ on 25-shot ARC-Challenge.

	Accuracy (↑)			**16-bit sub-blocks per tile**		
Model name	8-bit	**8/16-bit LAMP**	32-bit	0	1	2+
gemma-3-27b-pt	0.6504	**0.6953**	0.7070	56.64%	19.37%	23.99%
gemma-3-12b-pt	0.4609	**0.6289**	0.6777	62.56%	19.31%	18.13%
Qwen3-32B	0.6758	**0.7305**	0.7324	55.23%	19.55%	25.22%
Qwen3-8B	0.5488	**0.6660**	0.6660	54.10%	18.53%	27.37%
Qwen3-30B-A3B	0.5879	**0.7031**	0.6953	57.72%	19.01%	23.27%

Table 6: LampAttention with $\delta = 2^{-8}$ and $\tau = 2^{-1}$ on 0-shot ARC-Challenge.

	Accuracy (↑)			**16-bit sub-blocks per tile**		
Model name	8-bit	**8/16-bit LAMP**	32-bit	0	1	2+
gemma-3-27b-pt	0.5977	**0.6484**	0.6562	48.47%	30.41%	21.12%
gemma-3-12b-pt	0.4570	**0.6191**	0.6387	48.71%	32.21%	19.08%
Qwen3-32B	0.6191	**0.6016**	0.6094	48.54%	31.22%	20.24%
Qwen3-8B	0.4766	**0.5352**	0.5605	48.24%	28.71%	23.05%
Qwen3-30B-A3B	0.4512	**0.5176**	0.5762	48.28%	28.94%	22.78%

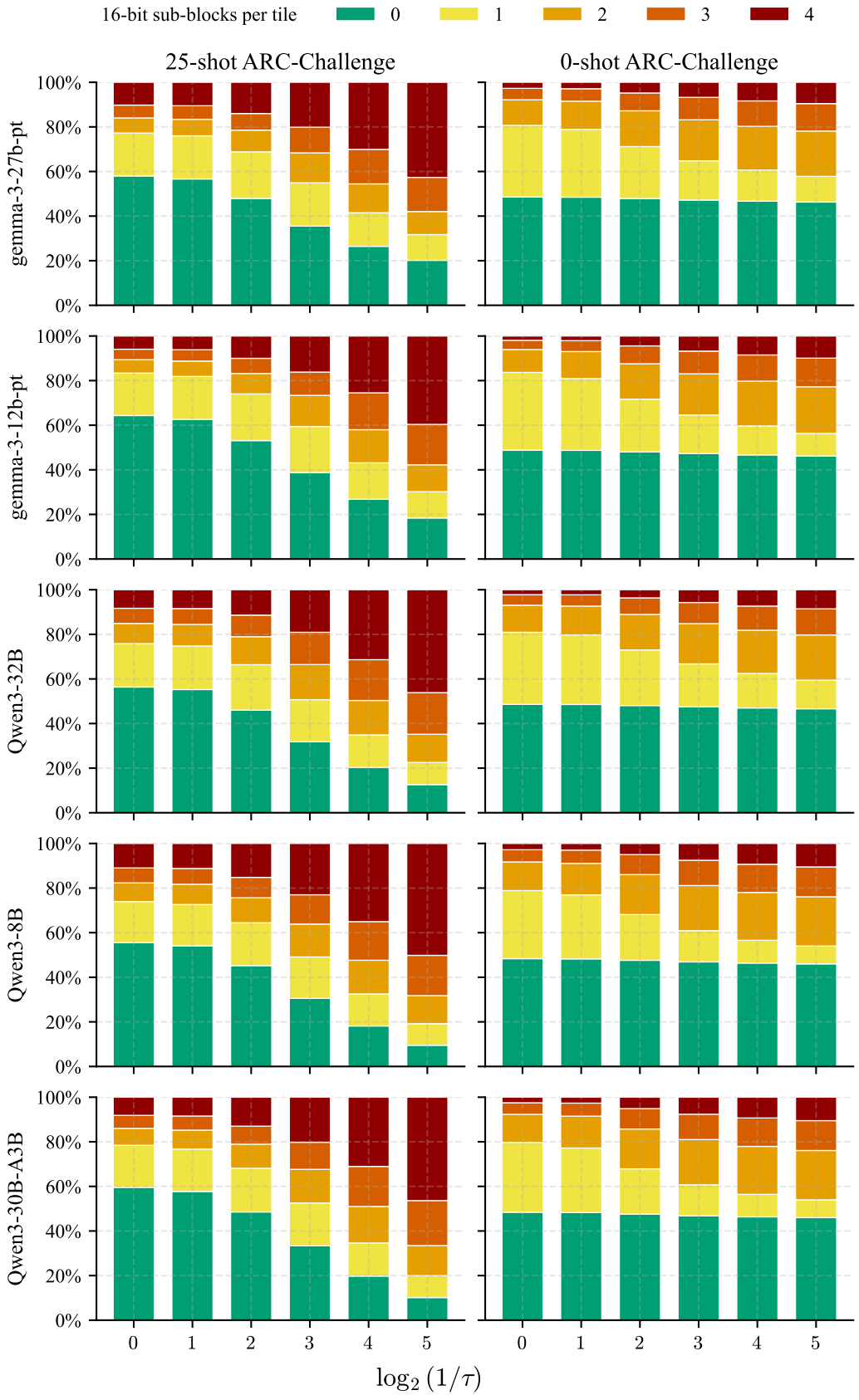

Figure 8: LampAttention with $\delta = 2^{-8}$ and $\tau \in \{2^{-t} : t = 0, \ldots, 5\}$.