# ENCRYPTED LATENT KV CACHES FOR FHE-BASED LLM DECODING

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

## ABSTRACT

Fully homomorphic encryption (FHE) enables LLM inference over encrypted inputs without exposing plaintext prompts. Applying FHE to autoregressive LLM decoding, however, requires maintaining and repeatedly accessing a growing encrypted KV cache. KV cache compression is a natural way to reduce such state in plaintext serving, but we show that under RNS-CKKS, plaintext compression ratios do not directly predict encrypted capacity or latency. Page geometry determines the cache page count, while execution layout and scheduling also determine runtime and workspace requirements. We present Encrypted Latent Attention (ELA), to our knowledge the first system to realize and evaluate latent KV cache compression for per-step non-interactive LLM decoding under RNS-CKKS. ELA maps shared latent KV states to encrypted pages without reconstructing full keys or values. Page-bucket rank fitting raises latent rank without increasing page count. Head-Diagonal Packing reduces repeated per-head computation on the shared latent. PageFold aggregates aligned page states before applying shared reduction, masking, and refresh. On an RTX A6000, $8\times$ KV compression expands the measured layer-local resident encrypted cache capacity of a 1.7B model by $7.49\times$. At a logical context length of 8k tokens, ELA achieves an approximately $18.1\times$ speedup over page-wise native full-KV in the measured encrypted attention core. We validate the encrypted implementation against a plaintext reference. These results show that effective encrypted KV compression depends on co-designing latent representations, ciphertext layout, and execution scheduling.

## 1 INTRODUCTION

**Private inference and the encrypted KV cache.** Cloud-hosted LLMs often process sensitive or proprietary prompts. Fully homomorphic encryption (FHE) addresses this problem by enabling the server to compute directly on ciphertexts, without relying on trusted hardware or access to plaintexts (Cheon et al., 2017; 2018a). Recent work has demonstrated transformer inference under CKKS (Moon et al., 2025; Zhang et al., 2025; 2026b; de Castro et al., 2025), but autoregressive decoding introduces a persistent KV cache that grows with the context (Kwon et al., 2023). Under FHE, the KV cache is packed into large RNS-CKKS ciphertexts, while the FHE context, evaluation keys, and transient workspace also consume GPU memory. This reduces the memory available to the cache itself. In our evaluation setting, the layer-local encrypted full-KV cache of a 1.7B model therefore holds only $504$ cached tokens (§6.1). As the context grows, encrypted KV capacity constrains autoregressive FHE decoding.

**Latent compression as the starting point.** Plaintext KV cache compression spans low-rank latents (DeepSeek-AI, 2024; Chang et al., 2025), quantization (Li et al., 2025b), and eviction (Zhang et al., 2023; Xiao et al., 2024). Data-dependent eviction (Zhang et al., 2023) is poorly matched to a fixed data-oblivious setting. Query-dependent selection (Lee et al., 2024; Jie et al., 2025) would require costly homomorphic access or reveal access patterns. Reduced numerical precision alone does not reduce CKKS slot occupancy. Low-rank latent compression instead uses static linear maps and directly reduces the packed cache width. We instantiate ELA with MLA-style latent caches (DeepSeek-AI, 2024; Ji et al., 2025). The absorb formulation folds the key and value up-projections into the query and output projections, so encrypted decode never reconstructs full keys or values. The decoupled RoPE stream confines the positional transformation to a few lanes.



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.

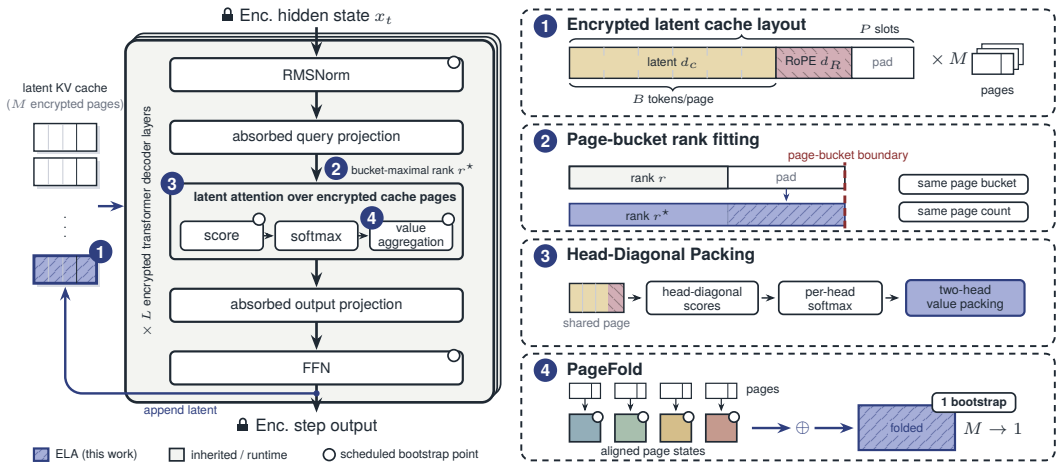


Figure 1: **Overview of ELA.** The left panel locates ELA within the decoding architecture, and the right panels show its four design elements. The GPU measurements isolate the attention core. Two-head value packing is geometry-dependent (§4.3), and PageFold is shown for one output group.

**Why cache compression alone is insufficient.** A smaller stored cache does not automatically execute efficiently under CKKS. First, fixed slot counts and power-of-two padding quantize the cache into page buckets, so the number of tokens per page and the page count for a fixed context length change only at bucket boundaries. Second, the latent is shared across heads, whereas each head applies its own absorbed query map and forms its own attention weights. A direct execution for each head therefore repeats a full-width pass over the shared latent. Third, page-wise execution repeats reduction, masking, and refresh for every cache page. In our ablation, a direct per-head execution of the $8\times$-compressed latent cache is $3.21\times$ slower than native full-KV attention when both use PageFold (§6.3).

**Encrypted Latent Attention.** We present **ELA**, which organizes four design elements around the encrypted cache page (Fig. 1). A *dimension-major (dim-major) page layout* stores the latent and decoupled RoPE state of $B$ consecutive tokens per ciphertext page and fixes the page geometry. *Page-bucket rank fitting* considers the largest latent rank within a page bucket together with the execution layout it admits, improving quality at an unchanged page count. *Head-Diagonal Packing* (HDP) organizes the computation of all heads on the shared latent within a page. It arranges the absorbed queries of different heads in diagonal form for baby-step giant-step (BSGS) evaluation (Halevi & Shoup, 2014; 2018). It maintains a separate softmax denominator for each head and packs two heads per value ciphertext where the geometry permits. *PageFold* accumulates the aligned page states across pages and applies their shared reduction, masking, and refresh once per output group. ELA denotes the integrated system of these four elements, with ELA-$d_r$ its instance at per-group latent rank $r$. Its per-head execution without HDP appears only as an ablation (§6.3).

**Results.** We evaluate ELA on SmolLM models converted with MHA2MLA (Ben Allal et al., 2024; Ji et al., 2025) under RNS-CKKS. At 1.7B and a logical context length of 8k tokens, ELA-$d_8$ executes the encrypted attention core in $18.48\,\mathrm{s}$ per layer, $18.08\times$ faster than native full-KV attention under page-wise execution. This speedup reflects the integrated ELA system rather than compression alone. Under our PageFold schedule, native full-KV has a latency of $37.12\,\mathrm{s}$. ELA-$d_8$ remains $2.01\times$ faster under this matched schedule. Its layer-local resident cache capacity increases from $504$ to $3{,}776$ tokens, a $7.49\times$ gain. Under the same PageFold schedule, HDP improves ELA-$d_8$ by $6.45\times$ over its per-head execution (§6.3).

**Quality and scope.** These MHA2MLA checkpoints serve as the compressed model representations, and ELA designs their encrypted cache layout and execution schedule. Their plaintext quality is evaluated separately at the native 2k context (§5). Our systems measurements cover one encrypted incremental decode step of one layer's attention core over a persistent latent cache. The next token is supplied at the per-token protocol boundary, and no client interaction occurs within a step. Batched prefill is not evaluated.

**Contributions.**

- **Encrypted latent-cache representation and page-bucket rank fitting.** We characterize how fixed CKKS slot capacity and power-of-two page geometry quantize the capacity gains of latent KV compression, showing that plaintext compression ratios do not directly translate into encrypted resident capacity. ELA stores latent cache states as dim-major encrypted pages and uses this geometry to choose the latent rank within each page bucket together with the execution layout it admits, improving quality without increasing page count (§4.1 to 4.2, §6.1).
- **Head-Diagonal Packing for shared-latent attention.** HDP organizes the score, denominator, value, and output computation of all heads on a shared latent page around diagonal encoding and BSGS evaluation, reducing repeated rotations and reductions across heads (§4.3, §6.3).
- **PageFold and evaluation of the integrated system.** PageFold folds aligned page states before their shared reduction, masking, and refresh (§4.4, Algorithm 1). We evaluate the integrated system against native full-KV under page-wise execution and under our PageFold schedule, and we separate the HDP and PageFold effects (§6.2 to 6.3).

## 2 Background and Setting

**RNS-CKKS scheme and homomorphic operations.** CKKS (Cheon et al., 2017), an approximate-arithmetic FHE scheme, encrypts a vector of reals into a single ciphertext and applies SIMD add/multiply across its slots at once. Slot rotations cyclically shift the packed vector. Our implementation uses its RNS variant (Cheon et al., 2018b). Each directly supported rotation index requires a corresponding Galois key, and other offsets can be composed from supported ones at additional rotation cost. Each rescaled multiplication consumes one level of the modulus chain, so the chain length bounds the multiplicative depth. When levels are exhausted, they are restored by bootstrapping (Cheon et al., 2018a). Latency depends on the circuit's multiplicative depth through the required bootstraps. It also depends on the cache layout through the page count, output grouping, reduction and rotation work, and page-dependent *refreshes*, which bootstrap intermediate attention states (§4). The deployed parameters are given in Appendix B.

**Multi-head latent attention (MLA).** MLA (DeepSeek-AI, 2024) caches (per token) a shared low-rank latent $c_i \in \mathbb{R}^{d_c}$ and a small decoupled RoPE key $k_i^R \in \mathbb{R}^{d_R}$ instead of full per-head $K, V$. MLA's absorb formulation rewrites the content score $(W_{UK}c_i)^\top (W_Q x_t) = c_i^\top (W_{UK}^\top W_Q x_t)$, absorbing $W_{UK}$ into the query projection so that $K$ is never formed. Symmetrically, $W_{UV}$ is absorbed into $W_O$, and thus $V$ is never formed. The per-token cache is thus $d_c + d_R$ reals rather than $2n_{\mathrm{kv}}d_{\mathrm{head}}$, where $n_{\mathrm{kv}}$ is the number of key/value heads (groups), $d_c{=}n_{\mathrm{kv}}r$ is the total latent with $r$ the rank of each group, and $d_R{=}n_{\mathrm{kv}}d_{\mathrm{rope}}$ is the total decoupled RoPE width with $d_{\mathrm{rope}}$ the RoPE width per KV head. The model has $H$ query heads, and each attention head has dimension $d_{\mathrm{head}}$. MHA2MLA (Ji et al., 2025) converts a pre-trained MHA/GQA model into this form with post-conversion fine-tuning, providing the latent representation we evaluate under CKKS.

**Encrypted incremental decode setting.** For each step, the server (i) holds an encrypted latent cache $\{(c_i, k_i^R)\}_{i<t}$, (ii) holds the encrypted step input, (iii) computes content and decoupled RoPE attention scores against the cache, a homomorphic softmax, and the value aggregation, and (iv) appends the new latent.

## 3 Related Work

Performing a single-pass FHE transformer inference has become feasible, from secure BERT-style encoders (Moon et al., 2025; Zhang et al., 2025; 2026b) to GPU-accelerated generative inference (de Castro et al., 2025). However, these systems do not address the growing KV cache of autoregressive decode. Cachemir (Yu et al., 2026) places a full KV cache in non-interactive CKKS decode and leaves its compression as future work. To our knowledge, this is the first *evaluation* of latent KV cache compression for non-interactive autoregressive decoding with RNS-CKKS (as detailed in Appendix A). Cachemir optimizes bootstrap *placement* for an uncompressed cache, whereas ELA

(i) changes the encrypted cache representation and (ii) co-designs page geometry, the computation within a page, and refresh scheduling around a latent cache. ELA is thus complementary to bootstrap-placement optimizations.

As to plaintext compression families (DeepSeek-AI, 2024; Li et al., 2025b; Zhang et al., 2023; Xiao et al., 2024), data-dependent eviction is poorly matched to the fixed data-oblivious non-interactive CKKS setting considered here. Selecting cache entries for eviction as a function of the encrypted query would require costly oblivious homomorphic selection over the cache; otherwise, the selected indices would reveal query-dependent access patterns. Low-rank latent compression, by contrast, uses static linear maps and avoids this query-dependent selection. Quantization does not reduce CKKS slot occupancy, and position-based eviction removes cached tokens without reducing the encrypted width of each token (Appendix A). While we assume the non-interactive setting, interactive MPC systems can use communication to perform query-aware cache selection (Zeng et al., 2025), which is a different private-inference setting.

ELA is orthogonal to layer-level FHE optimizations for CNN inference (Ju et al., 2024; Nam et al., 2025). HDP builds on the diagonal encoding of homomorphic linear transformations and their baby-step giant-step evaluation (Halevi & Shoup, 2014; 2018), which it applies to the per-head maps over a shared latent.

# 4 ELA DESIGN

ELA combines a dimension-major (dim-major) cache layout, page-bucket rank fitting, HDP, and PageFold (Fig. 1). HDP organizes computation within each page, while PageFold moves reduction, masking, and refresh outside the page loop.

## 4.1 ENCRYPTED LATENT CACHE LAYOUT

ELA stores exactly one MLA state $(c_i, k_i^R)$ of each cached token, the shared latent and the decoupled RoPE key (§2). Full keys and values are never materialized. The absorbed key- and value-side maps are folded into the plaintext model parameters of the query and output projections, adding no ciphertext-ciphertext multiplications and preserving the multiplicative depth of full-KV attention. The cached state is arranged dim-major. Within a page ciphertext, the $B$ token values of each packed dimension are stored contiguously. Thus, a reduction over the latent lanes is a fixed rotation tree, and hence lane-masked rotations keep RoPE confined to its $d_R$ lanes. The cached per-token footprint is $d_c + d_R$ (Appendix D).

A *page* is a single ciphertext holding the packed state of $B$ consecutive tokens. A cache of $t$ tokens then occupies $M$ pages as follows.

$$P = \texttt{nextPow2}(d_c + d_R), \qquad B = S/P, \qquad M = \lceil t/B \rceil,$$

where $\texttt{nextPow2}(x)$ denotes the smallest power of two no smaller than $x$, $P$ is the padded per-token cache size in slots, and $S$ is the number of packed slots in a ciphertext. Since $P$ changes only at power-of-two boundaries, $B$ and $M$ are quantized into page buckets. The resident capacity also depends on the execution workspace, which the page geometry does not fix. The rotation amounts used within a page depend on both $P$ and the model parameters rather than on $t$ (Appendix D).

## 4.2 PAGE-BUCKET RANK FITTING

In plaintext latent compression, the cache footprint grows linearly with the rank. In our page layout, all ranks whose $d_c + d_R$ falls in the same page bucket share $P$, $B$, and the page count $M$ at a given $t$ (Table 13). Page-bucket rank fitting therefore considers the largest rank that fits the bucket,

$$r^\star = \left\lfloor \frac{P - d_R}{n_{\text{kv}}} \right\rfloor,$$

together with the execution layout that this rank admits (§4.3). A rank below $r^\star$ leaves bucket headroom that a higher rank can use to improve quality without increasing the page count. A page bucket fixes the page geometry but not necessarily the execution layout. For example, $r{=}16$ and $r^\star{=}24$ share $P{=}1{,}024$ and $B{=}32$, but only $r{=}16$ satisfies the two-head value-packing condition in §4.3. Section 6.1 evaluates the same-bucket pairs with and without a change of layout.

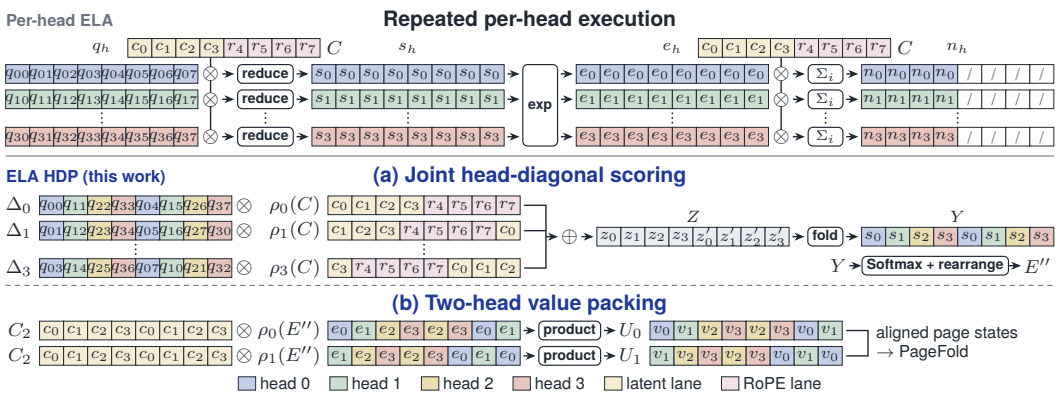


Figure 2: **Head-Diagonal Packing versus per-head execution on one shared latent page** (toy geometry $H{=}H_s{=}4$, $P{=}8$). (a) Diagonal scoring computes all head scores jointly. (b) Each value product carries contributions from two heads, and the aligned products enter PageFold (§4.4).

### 4.3 Head-Diagonal Packing

**Repeated computation in per-head execution.** In the absorb formulation, every head reads the same latent page. Head $h$ applies its own absorbed query vector across the $d_c$ latent lanes and reduces over them. A direct execution therefore performs $H$ full-width passes per page, and its work grows with $Hd_c$ rather than with the stored width $d_c$.

**Joint scoring of all heads.** HDP computes the scores of all heads against a page together. It distributes the heads across the dimension blocks of the same ciphertexts and combines partial scores only across blocks assigned to the same head. Concretely, slot $dB{+}i$ holds dimension $d$ of token $i$, and a rotation $\rho_r$ by $rB$ slots shifts the dimension index by $r$ blocks. HDP assigns block $d$ to head $h(d) = d \bmod H_s$, where the diagonal period $H_s$ is the smallest power of two not below $H$. It writes the absorbed queries into $H_s$ query diagonals $\Delta_r$, and block $d$ of $\Delta_r$ holds the query of head $h(d)$ at dimension $(d{+}r) \bmod P$. All head scores are obtained by summing the diagonal products and applying a coset fold. The sum $Z = \sum_{r<H_s} \Delta_r \odot \rho_r(C)$ leaves in block $d$ the partial sum of head $h(d)$ over $H_s$ consecutive dimensions. Starting from $Y \leftarrow Z$, the fold applies $Y \leftarrow Y + \rho_{H_s 2^k}(Y)$ for $k < \log_2(P/H_s)$, which adds the blocks of equal residue. Block $d$ then holds the complete score $s_{h(d)}[i]$ of head $h(d)$ for every token $i$ of the page (Fig. 2). No per-head mask is needed, and the softmax exponential is evaluated once per page on the whole ciphertext. The RoPE contribution is accumulated with the content score before exponentiation. The products follow the diagonal encoding of homomorphic linear maps (Halevi & Shoup, 2014) and its baby-step giant-step schedule (Halevi & Shoup, 2018). The query diagonals are constructed once per decode step from the encrypted per-head queries. Appendix D.1 gives the full slot mapping, rotation schedule, query-diagonal cost, and output recovery.

**Per-head denominators and two-head value packing.** Each head keeps its own softmax denominator $u_h = \sum_i e_h[i]$ over its exponentiated scores $e_h[i]$, so normalization remains per head. On the value path, the weighted values $e_h[i]\, c[d,i]$ of all heads are formed by block rotations of the weight ciphertext against the latent. For 1.7B-$d_8$, $d_{16}$, and $d_{32}$, $P = 2d_c$, which allows the latent to be duplicated into both halves of the page ($C_2$). The weight ciphertext $E''$ carries head $h$ in the lower half and head $h{+}H_s/2$ in the upper half at the same block positions, so each of the $H_s/2$ products $U_k = C_2 \odot \rho_k(E'')$ holds the weighted values of two heads (Fig. 2). Under PageFold, these products are accumulated across pages in aligned ciphertexts before token reduction and masking. At 1.7B-$d_8$, two-head packing uses 16 numerator accumulators and one denominator accumulator, against 64 accumulators in the per-head implementation (§4.4). After aggregation, the output layout is recovered once. For a single output group, this recovery places $o_h[d]$ at slot $dB{+}h$, and the absorbed output projection is applied unchanged. The persistent cache and page geometry are unchanged, and HDP adds no rotation (Galois) key.

**Conditions.** The diagonal placement requires $H_s \mid P$. The output is packed in groups of $S_K$ heads, with $S_K$ the smallest power of two not below $\min(H, B)$, and head $h = gS_K{+}h'$ of output

group $g$ is placed at slot $dB+h'$. One output group thus holds all heads when $H \leq B = S/P$, a condition stronger than the per-head condition $Hd_c \leq S$. At 1.7B-$d_{32}$ ($B$=16), ELA forms two output groups of 16 heads, while retaining a diagonal period of 32 and sharing 16 numerator accumulators and one denominator accumulator across both groups. The groups are separated in the output packing. Two-head packing further requires $P = 2d_c$ and $H_s \mid d_c$, which 1.7B-$d_8$, $d_{16}$, and $d_{32}$ satisfy. Where $P \neq 2d_c$, at every rank of 135M and 360M and at the bucket-maximal 1.7B $r^\star$=24, HDP uses one head per value product with a mask that zeroes the padding heads of the diagonal period.

### 4.4 PageFold: cross-page aggregation

Across pages, each page $p$ produces a weighted-value state $V_p$ in the same slot coordinates, followed by the same linear reduction and masking $\mathcal{L}$ and by a refresh. Since $\mathcal{L}$ is linear and identical across pages,

$$\sum_{p=1}^{M} \mathcal{L}(V_p) = \mathcal{L}\left(\sum_{p=1}^{M} V_p\right).$$

PageFold accumulates the aligned page states first and applies $\mathcal{L}$ and the refresh once (Algorithm 1). Ciphertext additions consume no multiplicative level. The aligned page states are the unnormalized weighted-value numerators and the corresponding per-head denominators. Score exponentiation remains inside the page loop. Normalization is applied after the global accumulation. Page-specific token-validity masks are applied before accumulation, whereas the folded reduction and masking are shared across pages.

**Algorithm 1** Page-wise execution versus PageFold for one output ciphertext group.

**Input:** aligned weighted-value states $V_1, \ldots, V_M$ of the group, one per cache page, and the shared linear reduction and masking $\mathcal{L}$.

*Page-wise execution*

$A \leftarrow 0$
**for** $p = 1, \ldots, M$ **do**
  $A \leftarrow \textsc{Bootstrap}(A + \mathcal{L}(V_p))$
**end for**
**return** $A$

*PageFold*

$A \leftarrow 0$
**for** $p = 1, \ldots, M$ **do**
  $A \leftarrow A + V_p$
**end for**
$A \leftarrow \textsc{Bootstrap}(\mathcal{L}(A))$
**return** $A$

The execution layout and page geometry fix the number $N_G$ of output ciphertext groups. Algorithm 1 applies to each group, with $N_G$=2 at 1.7B-$d_{32}$ and $N_G$=1 in all other evaluated configurations (§4.3). Page-wise execution thus performs $MN_G$ page-dependent refreshes in this stage, whereas PageFold performs $N_G$. Since $M$ grows with the context while $N_G$ does not, PageFold removes the $M$-dependence of $\mathcal{L}$ and the refresh. At 1.7B-$d_8$, the PageFold attention core invokes three bootstraps per layer (attention input, one refresh per output group, and attention output), against 130 under page-wise execution. At 1.7B-$d_{32}$, the corresponding counts are five and 1,027. The linear identity alone does not guarantee precision, because folding changes the magnitude of the bootstrapping input. The folded state is validated after its refresh for $d_8$ (Appendix C).

## 5 Experimental Setup

Experiments cover SmolLM-135M, SmolLM-360M, and SmolLM-1.7B (Ben Allal et al., 2024), converted using MHA2MLA at KV cache compression ratios of 3.2×, 5.33×, and 8×, named $d_{32}$, $d_{16}$, and $d_8$ after the per-group latent rank. We evaluate plaintext quality and encrypted execution for all three models.

**Plaintext checkpoint quality.** We use released MHA2MLA checkpoints for the scale grid and train a separate matched pair for page-bucket rank fitting. Plaintext quality is evaluated at the native 2k context, and the protocols and results are provided in Appendix F.4.

**Encrypted attention-core latency.** Latency is measured for one decode step of one layer's encrypted attention core at a logical context length of $t$=8,192 tokens on a single RTX A6000,

as the median of three runs. Run ranges and timing details are provided in Appendix F.3. The harness keeps two encrypted cache pages resident and reuses them as the operands of every page iteration of the $M$-page computation. Thus, latency covers the attention circuit of all $M$ pages without host-to-device page transfers, and resident capacity is measured separately. The timed region includes the construction of the query diagonals and the recovery of the output layout, and it excludes the query and output projections, the cache append, and encryption and decryption.

**Resident capacity.** The capacity measurement keeps $M$ distinct encrypted cache pages of one layer resident with the FHE context and all keys. It reports the largest $M$ that executes under PageFold.

**Baselines and configuration.** The main comparison is between ELA (§4) and native full-KV under page-wise execution. As an additional comparison, we apply our PageFold schedule to native full-KV as well (full-KV + PageFold), so that both systems are compared under the same schedule. For the GQA (Ainslie et al., 2023) checkpoints (135M and 360M), native full-KV is *compact GQA*, which keeps the persistent cache in *storage pages* at the native KV width. Inside the timed region, each storage page is unpacked into head-expanded *computation pages* (Appendix F.2). Both representations evaluate the softmax exponential once per computation page, which for ELA is the cache page itself. GPU measurements use patched builds of FIDESlib 2.1.2 (Agulló-Domingo et al., 2025) and OpenFHE 1.5.1 (Al Badawi et al., 2022) in an RNS-CKKS configuration with one bootstrapping operation per refresh and 128-bit classical security (Appendix B).

# 6 Evaluation

## 6.1 Cache capacity and page-bucket rank fitting

Under PageFold, the 1.7B full-KV cache holds 63 pages of $B$=8 tokens, or 504 tokens. ELA-$d_8$ holds 59 pages of $B$=64 tokens, or 3,776 tokens (Fig. 3a and Table 12). The gain decomposes as $8 \times (59/63) = 7.49\times$, the ratio of tokens per page times the ratio of resident pages. At 135M and 360M, the ELA-$d_8$ gains over the compact GQA baselines are $8.39\times$ and $8.80\times$. This bucketed structure also determines which latent ranks can be increased without adding cache pages.

**Rank fitting.** Table 1 separates same-bucket pairs with and without a change of execution layout (§4.2). At 135M and 360M, each released pair shares both the page bucket and the single-head layout. Within each pair, the higher rank lowers WikiText-2 (Merity et al., 2017) perplexity at essentially unchanged latency and resident capacity. These released checkpoints were trained separately by Ji et al. (2025), so this quality difference is not a controlled comparison. The 1.7B pair provides the controlled comparison, because both ranks share the initialization and training protocol. There, increasing the rank from 16 to 24 reduces WikiText-2 perplexity by 6.1% at the same page count. However, $r^\star$=24 has a wider latent and, since $P \neq 2d_c$, cannot use two-head value packing. Together, these differences increase its attention-core latency by 19.1% and reduce its resident capacity by 9.8%.

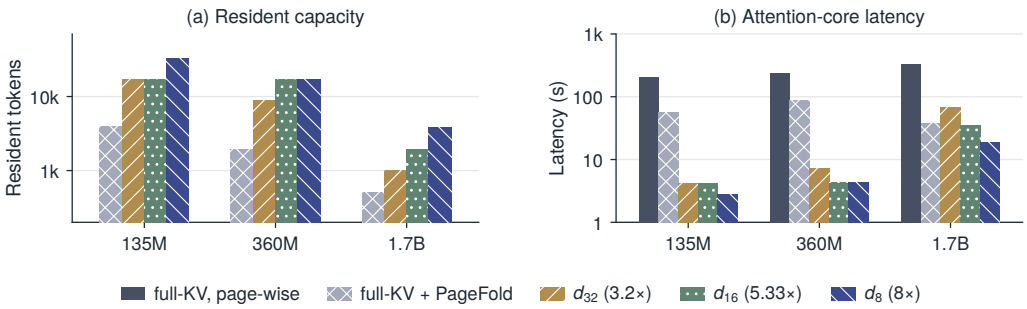


Figure 3: **Capacity and latency across model scales.** (a) Layer-local resident capacity under PageFold, measured with distinct cache pages. (b) Attention-core latency at logical $t$=8,192, with ELA under PageFold. Vertical axes are logarithmic. Full-KV is compact GQA at 135M and 360M.

Table 1: Same-bucket rank pairs under PageFold. Latency (logical $t$=8,192) and capacity (measured separately) use each rank's circuit geometry. Perplexity uses the native 2k context. The 1.7B pair has its own training protocol (Appendix F.4), so its perplexities are comparable only within the pair.

| Model | Rank | $P$ | Value packing | Latency (s) ↓ | Resident tokens ↑ | WikiText-2 PPL ↓ |
|---|---|---|---|---|---|---|
| 135M | $d_{16}$ | 128 | Single-head | 4.10 | 17,152 | 19.84 |
| | $d_{32}$ | | Single-head | 4.10 | 17,152 | 17.81 |
| 360M | $d_8$ | 128 | Single-head | 4.35 | 16,896 | 15.62 |
| | $d_{16}$ | | Single-head | 4.36 | 17,152 | 13.96 |
| 1.7B | $r$=16 | 1,024 | Two-head | 34.44 | 1,952 | 11.29 |
| | $r^\star$=24 | | Single-head | 41.02 | 1,760 | 10.60 |

These configurations mirror compressed MHA2MLA checkpoints, whose conversion method and quality tradeoffs were studied by Ji et al. (2025). At the 1.7B-$d_8$ operating point, our additional plaintext evaluation of the released checkpoint shows a 13.2% WikiText-2 perplexity increase and a 10.7% LongBench (Bai et al., 2024) macro-F1 decrease relative to the base model (Tables 14–15).

### 6.2 Encrypted attention performance

Table 2: System performance at 1.7B and logical $t$=8,192. Latency and resident capacity are measured separately. Parentheses give the change relative to native full-KV (▼ lower, ▲ higher).

| System | Latency (s) ↓ | Resident tokens ↑ |
|---|---|---|
| Native full-KV | 334.13 (1.00×) | 504 (1.00×) |
| **ELA-$d_8$** | **18.48 (▼18.08×)** | **3,776 (▲7.49×)** |
| ELA-$d_{16}$ | 34.44 (▼9.70×) | 1,952 (▲3.87×) |
| ELA-$d_{32}$ | 67.50 (▼4.95×) | 992 (▲1.97×) |
| Full-KV + PageFold | 37.12 (▼9.00×) | 504 (1.00×) |

ELA-$d_8$, $d_{16}$, and $d_{32}$ are 18.08×, 9.70×, and 4.95× faster than native full-KV under page-wise execution (Table 2). Against full-KV + PageFold, ELA-$d_8$ and ELA-$d_{16}$ remain 2.01× and 1.08× faster, whereas ELA-$d_{32}$ is 1.82× slower. ELA-$d_{32}$ performs 24,586 ciphertext-ciphertext multiplications per step, four times as many as ELA-$d_8$, because HDP work grows with the latent width (Table 10). Its two output groups also raise the PageFold bootstrap count from three to five (§4.4). Full-KV + PageFold reduces latency by 9.00× but leaves resident capacity at 1.00×.

For the smaller models, ELA-$d_8$ has latencies of 2.75 s at 135M and 4.35 s at 360M, against 205.53 s and 236.76 s for page-wise compact GQA. It is also 20.33× and 20.08× faster than full-KV + PageFold (Fig. 3b and Table 11).

### 6.3 Ablation of HDP and PageFold

Table 3 ablates HDP and PageFold, and Table 4 reports their operation counts at 1.7B-$d_8$.

**HDP.** Under the same PageFold schedule, HDP improves ELA over its per-head execution by 6.45×, 7.39×, and 8.05× at 1.7B-$d_8$, $d_{16}$, and $d_{32}$, respectively (Table 9). Against full-KV + PageFold, per-head ELA-$d_8$ is 3.21× slower. HDP reverses this ordering and makes ELA-$d_8$ 2.01× faster. These speedups include the construction of the query diagonals and the output recovery. At $d_8$ with PageFold fixed, HDP reduces rotations from 74,281 to 4,420.

At 135M and 360M, with one head per value product, the speedups are 1.67× and 3.03×, respectively. At 135M, rotations fall from 1,922 to 626 while ciphertext-ciphertext multiplications rise from 293 to 517, because the diagonal period of 16 exceeds the 9 heads. Resident capacity falls by 4.5% at 135M and rises by 3.1% at 360M (Table 11).

**PageFold.** With HDP fixed, PageFold improves on page-wise execution by 4.05× to 6.30× across 1.7B-$d_8$, $d_{16}$, and $d_{32}$, and by 3.30× to 4.55× at 135M and 360M. The largest gain occurs at 1.7B-$d_{32}$, which has the largest page count. Relative to shared refresh alone, PageFold provides an

additional $1.76\times$ to $2.09\times$ speedup in the HDP configurations measured under both schedules. At 1.7B-$d_8$, the step from shared refresh to PageFold folds the reduction and masking, which reduces the rotations and ciphertext-plaintext multiplications at the same bootstrap count (Table 4).

**Interaction between layout and schedule.** At 1.7B-$d_8$, HDP provides a $2.88\times$ speedup under page-wise execution and $6.45\times$ under PageFold. Conversely, PageFold provides $1.81\times$ under per-head execution and $4.05\times$ under HDP. The benefit of each component thus depends on the other.

Table 3: Layout and schedule ablation at 1.7B (units as in Table 2, smaller models in Table 11, all schedules in Appendix E). The per-head row is ELA without HDP, the page-wise column is ELA without PageFold, and shared refresh folds only the refresh.

| Representation | Layout | Page-wise | Shared refresh | PageFold |
|---|---|---|---|---|
| Native full-KV | native | 334.13 | 43.78 | 37.12 |
| ELA-$d_8$ | per-head | 215.37 | 181.18 | 119.20 |
| ELA-$d_8$ | HDP | 74.79 | 38.61 | 18.48 |
| ELA-$d_{16}$ | HDP | 144.12 | 71.61 | 34.44 |
| ELA-$d_{32}$ | HDP | 425.07 | 134.10 | 67.50 |

Table 4: Operation counts of the 1.7B-$d_8$ attention core per decode step at $t$=8,192, with ciphertext-ciphertext (ct×ct) and ciphertext-plaintext (ct×pt) multiplications (all configurations in Table 10).

| Configuration | Rotations | ct×ct | ct×pt | Bootstraps |
|---|---|---|---|---|
| Per-head + PageFold | 74,281 | 8,197 | 8,290 | 3 |
| HDP + shared refresh | 29,566 | 6,149 | 9,568 | 3 |
| HDP + PageFold | 4,420 | 6,149 | 1,567 | 3 |

### 6.4 Correctness validation

Every reported ELA configuration and schedule passes checks against an independent plaintext reference at four stages on 15 small encrypted cases, including partial-page boundaries. The two output groups of 1.7B-$d_{32}$ are scored separately. Each reported PageFold configuration also passes a representative case at $t$=8,192. Separately, a chain of two learned SmolLM-135M-$d_8$ layers in per-head form under CPU RNS-CKKS matches its fp64 polynomial-circuit reference with 17.3 bits of precision. These learned-weight validations and the persistent-cache loop are reported in Appendix C.

## 7 Discussion and Limitations

A smaller stored cache does not by itself yield a lower encrypted execution cost. For rank fitting, the page bucket determines the storage geometry, whereas the admitted execution layout determines part of the runtime and workspace cost (Table 1).

Our latency measurements isolate the encrypted attention core at the 8k operating point (§5). The timed region excludes the query and output projections, the cache append, and the RMSNorm and FFN bootstraps. Checkpoint-wide encrypted execution, learned multi-layer GPU execution, and the remaining generation pipeline are outside the scope of this work.

## 8 Conclusion

We present ELA, an encrypted execution design that realizes latent KV cache compression for per-step non-interactive FHE-based LLM decoding. Its dim-major cache, page-bucket rank fitting, Head-Diagonal Packing, and PageFold together expand layer-local resident capacity and reduce encrypted attention-core latency. Our results show that encrypted KV compression should be designed around ciphertext execution structure rather than plaintext storage reduction alone.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with method and experiment code development, methodological review by suggesting validation analyses, analysis and presentation of experimental results, literature review, translation, artifact preparation, and manuscript editing. We did not use generative AI tools for any other task that requires disclosure. We reviewed all AI-assisted work. The authors verified and tested AI-assisted code and artifacts, checked suggested related work against the original publications, and validated AI-assisted interpretations against the underlying measurements. All translated text was verified against the intended meaning. The authors take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work studies privacy-preserving inference and uses no human-subject data. Its intended benefit is to reduce the cost of encrypted LLM inference.

### REPRODUCIBILITY STATEMENT

All reported FIDESlib GPU latency and resident-capacity measurements use the single hardware configuration in §5 and Appendix F. Separate CPU validations under Lattigo (Tune Insight SA and EPFL-LDS, 2024) are reported in Appendix C. We provide the checkpoint and matched-pair training details (Appendix F.4), the exact RNS-CKKS parameters and modulus chain (Appendix B), the circuit, depth, and context-independent bootstrap accounting (§4, Appendix B), the cache-layout details and the `nextPow2` requirement (Appendix D), the GQA full-KV capacity layout (Appendix F.2), the schedule definitions, operation counts, and ablations (Appendix E), and the measurement protocol (Appendix F.3). The artifact additionally contains the learned-checkpoint layer exporter, parameter and weight manifests, the fp64 plaintext reference implementing the same polynomial circuit, the encrypted intermediate trace, cache-lane validation, and result hashes. The quality artifact includes the deterministic 32-document PG19 selection, the six-task LongBench manifest, raw per-document PG19 and per-example LongBench outputs sufficient to recompute those reported metrics without inference, aggregate WikiText-2 evaluation records with the corresponding evaluation scripts and configurations, paired-bootstrap results, the quality SHA-256 manifest, and sanitized checkpoint revision and provenance information. Anonymized source code and the artifacts described above will be shared with reviewers and area chairs via a restricted OpenReview comment once the discussion forum opens.

## REFERENCES

Carlos Agulló-Domingo, Óscar Vera-López, Seyda Guzelhan, Lohit Daksha, Aymane El Jerari, Kaustubh Shivdikar, Rashmi Agrawal, David Kaeli, Ajay Joshi, and José L. Abellán. FIDESlib: A fully-fledged open-source FHE library for efficient CKKS on GPUs. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 1–3, 2025.

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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4895–4901, 2023.

Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. OpenFHE: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pp. 53–63, 2022.

Ahmad Al Badawi, Andreea Alexandru, Yuriy Polyakov, and Vinod Vaikuntanathan. SoK: Private LLM inference using approximate homomorphic encryption. Cryptology ePrint Archive, Paper 2026/935, 2026.

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

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. SmolLM - blazingly fast and remarkably powerful. Hugging Face Blog, `https://huggingface.co/blog/smollm`, 2024.

Jean-Philippe Bossuat, Rosario Cammarota, Ilaria Chillotti, Benjamin R. Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, Xianhui Lu, Carsten Maple, Alberto Pedrouzo-Ulloa, Rachel Player, Yuriy Polyakov, Luis Antonio Ruiz Lopez, Yongsoo Song, and Donggeon Yhee. Security guidelines for implementing homomorphic encryption. *IACR Communications in Cryptology*, 1(4), 2025. doi: 10.62056/anxra69p1.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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

Chenghao Chen, Kailun Qin, Xiaolin Zhang, Chi Zhang, and Dawu Gu. Bifrost: Hybrid TEE-FHE inference for privacy-preserving transformer and LLM serving. *arXiv preprint arXiv:2606.17421*, 2026.

Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2019*, pp. 34–54, 2019. doi: 10.1007/978-3-030-17656-3_2.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pp. 409–437, 2017. doi: 10.1007/978-3-319-70694-8_15.

Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2018*, pp. 360–384, 2018a. doi: 10.1007/978-3-319-78381-9_14.

Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In *Selected Areas in Cryptography – SAC 2018*, pp. 347–368, 2018b. doi: 10.1007/978-3-030-10970-7_16.

Seonyoung Cheon, Yongwoo Lee, Dongkwan Kim, Ju Min Lee, Sunchul Jung, Taekyung Kim, Dongyoon Lee, and Hanjun Kim. DaCapo: Automatic bootstrapping management for efficient fully homomorphic encryption. In *USENIX Security Symposium*, pp. 6993–7010, 2024.

Leo de Castro, Daniel Escudero, Adya Agrawal, Antigoni Polychroniadou, and Manuela Veloso. EncryptedLLM: Privacy-preserving large language model inference via GPU-accelerated fully homomorphic encryption. In *International Conference on Machine Learning (ICML)*, 2025.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Austin Ebel, Karthik Garimella, and Brandon Reagen. Orion: A fully homomorphic encryption framework for deep learning. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 734–749, 2025. doi: 10.1145/3676641.3716008.

Shai Halevi and Victor Shoup. Algorithms in HElib. In *Advances in Cryptology – CRYPTO 2014*, pp. 554–571, 2014.

Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in HElib. In *Advances in Cryptology – CRYPTO 2018*, pp. 93–120, 2018.

Siddharth Jayashankar, Joshua Kim, Michael B. Sullivan, Wenting Zheng, and Dimitrios Skarlatos. A scalable multi-GPU framework for encrypted large-model inference. *arXiv preprint arXiv:2512.11269*, 2025.

Tao Ji, Bin Guo, Yuanbin Wu, Qipeng Guo, Lixing Shen, Zhan Chen, Xipeng Qiu, Qi Zhang, and Tao Gui. Towards economical inference: Enabling DeepSeek's multi-head latent attention in any transformer-based LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 33313–33328, 2025.

Shibo Jie, Yehui Tang, Kai Han, Zhi-Hong Deng, and Jing Han. SpeCache: Speculative key-value caching for efficient generation of LLMs. In *International Conference on Machine Learning (ICML)*, 2025.

Jae Hyung Ju, Jaiyoung Park, Jongmin Kim, Minsik Kang, Donghwan Kim, Jung Hee Cheon, and Jung Ho Ahn. NeuJeans: Private neural network inference with joint optimization of convolution and FHE bootstrapping. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 4361–4375, 2024. doi: 10.1145/3658644.3690375.

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

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 155–172, 2024.

Guihong Li, Mehdi Rezagholizadeh, Mingyu Yang, Vikram Appia, and Emad Barsoum. X-EcoMLA: Upcycling pre-trained attention into MLA for efficient and extreme KV compression. In *Conference on Language Modeling (COLM)*, 2025a.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *International Conference on Machine Learning (ICML)*, 2025b.

Fanxu Meng, Pingzhi Tang, Zengwei Yao, Xing Sun, and Muhan Zhang. TransMLA: Migrating GQA models to MLA with full DeepSeek compatibility and speedup. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. THOR: Secure transformer inference with homomorphic encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 3765–3779, 2025. doi: 10.1145/3719027.3765150.

Kevin Nam, Youyeon Joo, Dongju Lee, Seungjin Ha, Hyunyoung Oh, Hyungon Moon, and Yunheung Paek. LOHEN: Layer-wise optimizations for neural network inferences over encrypted data with high performance or accuracy. In *USENIX Security Symposium*, pp. 5583–5600, 2025.

Jaiyoung Park, Sejin Park, Jai Hyun Park, Jung Ho Ahn, Jung Hee Cheon, Guillaume Hanrot, Jung Woo Kim, Minje Park, and Damien Stehlé. Scaling up FHE-based privacy-preserving ML: Higher throughput, longer inputs for LLama-3-8B. *arXiv preprint arXiv:2601.18511*, 2026.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

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

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tune Insight SA and EPFL-LDS. Lattigo v6. Online: `https://github.com/tuneinsight/lattigo`, 2024.

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

Wenshao Yang, Zhenhua Liu, Yihan Zhang, Xin Bie, and Dongdong Yao. SecretVoice1.0: Non-interactive secure transformer inference. *TechRxiv preprint*, 2026. doi: 10.36227/techrxiv.177220349.97070665/v1.

Ye Yu, Yifan Zhou, Yi Chen, Pedro Soto, Wenjie Xiong, and Meng Li. Cachemir: Fully homomorphic encrypted inference of generative large language model with KV cache. *arXiv preprint arXiv:2602.11470*, 2026.

Wenxuan Zeng, Ye Dong, Jinjin Zhou, Jin Tan, Lei Wang, Tao Wei, Runsheng Wang, and Meng Li. MPCache: MPC-friendly KV cache eviction for efficient private LLM inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Hedong Zhang, Neusha Javidnia, Shweta Pardeshi, Qian Lou, and Farinaz Koushanfar. CryptoGen: Secure transformer generation with encrypted KV-cache reuse. *arXiv preprint arXiv:2602.08798*, 2026a.

Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. In *Network and Distributed System Security Symposium (NDSS)*, 2025.

Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok-Yan Lam. MOAI: Module-optimizing architecture for non-interactive secure transformer inference. In *International Conference on Learning Representations (ICLR)*, 2026b.

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

# A Positioning Relative to Closest Related Work

Cachemir (Yu et al., 2026), concurrent with this work, is the closest related system. It contributes packing layouts for the uncompressed K and V caches and a KV-aware bootstrap-placement schedule. ELA packs a compressed latent cache in a dim-major page layout and contributes page-bucket rank fitting, Head-Diagonal Packing, and the PageFold schedule (§4). The directions are complementary, since its schedule could in principle run on the smaller encrypted cache.

SecretVoice (Yang et al., 2026) implements a ciphertext KV cache for its packing strategy in non-interactive CKKS. Its batched bootstrapping refreshes several sparse ciphertexts together, whereas PageFold folds aligned page states before their shared reduction, masking, and refresh (§4.4). Partial encryption (Park et al., 2026), TEE-FHE hybridization (Bifrost, Chen et al., 2026), and multi-GPU execution (Cerium, Jayashankar et al., 2025) scale FHE inference without studying latent KV cache compression, and hybrid schemes that combine HE with secret sharing reuse encrypted KV caches under a different private-inference protocol (Zhang et al., 2026a).

Palu (Chang et al., 2025) is the closest plaintext low-rank analogue to the MLA-style latent used here (DeepSeek-AI, 2024), which we obtain by MHA2MLA conversion (Ji et al., 2025). X-EcoMLA (Li et al., 2025a) and TransMLA (Meng et al., 2025) are related methods that convert pre-trained attention into MLA. Palu also caches a low-rank latent, but it reconstructs keys at decode time because position-dependent RoPE prevents key-side absorption (Chang et al., 2025, Sec. 3.1.2). Under CKKS, that reconstruction would add ciphertext-plaintext linear transforms, rotations, and level consumption at every decode step, whereas the decoupled RoPE stream of MLA confines the positional transformation to the small decoupled key lanes and keeps both content paths absorbed (§4). ELA therefore adopts an MLA-style latent instead of a generic low-rank map. Cross-layer architectures such as CLA (Brandon et al., 2024) and YOCO (Sun et al., 2024) share KV states across layers rather than reduce the encrypted width and are orthogonal to the post-hoc latent compression studied here. Position-based eviction such as StreamingLLM (Xiao et al., 2024) is data-oblivious. It reduces the number of cached tokens and is likewise orthogonal to latent compression.

A recent SoK (Al Badawi et al., 2026) of non-interactive CKKS-based LLM inference does not identify encrypted-cache compression among the surveyed frameworks. Bootstrap-placement compilers (Cheon et al., 2024; Ebel et al., 2025) and the layer-level optimizations of §3 remain applicable to ELA's context-independent bootstraps (Appendix B).

# B RNS-CKKS Parameters and Security

Table 5 lists the RNS-CKKS configuration of our GPU executor (FIDESlib 2.1.2 with our patches and OpenFHE 1.5.1 with the FIDESlib-required patch (Al Badawi et al., 2022)). Each refresh is one non-iterative bootstrapping operation. The measurement manifest of the artifact records this configuration together with the binaries and their hashes. The modulus chain has 35 primes in $Q$ (one 60-bit prime, 33 primes of 59 bits, and one prime $q$ with $\log_2 q \approx 19.585$) and 12 primes of 60 bits in the auxiliary modulus $P_{\mathrm{aux}}$, giving $\log_2(QP_{\mathrm{aux}}) = 2{,}746.585$. Key switching is hybrid with $\mathrm{dnum}{=}3$. At this modulus size the ring dimension is $N = 2^{17}$, and we use sparse packing with $2^{15} = 32{,}768$ slots. This $2^{15}$-slot operating point matches recent CKKS transformer systems, which use $N{=}2^{16}$ with $2^{15}$ slots (Zhang et al., 2026b; Yu et al., 2026). With sparse packing, the larger ring dimension leaves the slot count and the page geometry of the evaluated workload unchanged. The security basis is the OpenFHE parameter table for a dense uniform-ternary secret and a discrete-Gaussian error ($\sigma{=}3.19$), whose $N{=}2^{17}$ entry follows Bossuat et al. (2025, Table 5.2). This table gives 128-bit classical security for $\log_2(QP_{\mathrm{aux}}) \le 3{,}523$ at $N = 2^{17}$. The library labels this setting `HEStd_128_classic`. The 776-bit difference between $3{,}523$ and $2{,}746.585$ is headroom in the modulus budget. Our 128-bit classical security claim follows this OpenFHE parameter table.

**Rotation (Galois) keys.** The application rotation keys in Table 5 are fixed by the page geometry and model parameters rather than by $t$. HDP adds no application rotation keys (Appendix D).

**Bootstrapping.** We use CKKS bootstrapping (Cheon et al., 2018a; Chen et al., 2019) with three levels each for CoeffsToSlots and SlotsToCoeffs (OpenFHE level budget $\{3, 3\}$). Under PageFold, the attention core invokes one bootstrap at the attention input, one refresh per output group, and

one bootstrap at the attention output per output group. The 1.7B-$d_{32}$ configuration uses two output groups and invokes five bootstraps, whereas all other evaluated configurations use one group and invoke three. Per layer, native full-KV and ELA invoke the same context-independent bootstraps, one at the attention input inside the timed core, the RMSNorm and FFN bootstraps outside it, and one attention-output bootstrap per output group. The two representations have the same multiplicative depth per step. Under page-wise execution at 1.7B-$d_8$, the core invokes 130 bootstraps (128 page-dependent refreshes plus the input and output bootstraps), and 1,027 at 1.7B-$d_{32}$.

Table 5: RNS-CKKS parameters of our GPU executor (one non-iterative bootstrapping operation per refresh), as recorded in the measurement manifest.

| | |
|---|---|
| ring dimension $N$ | $2^{17} = 131{,}072$ |
| packed slots (sparse) | $2^{15} = 32{,}768$ |
| scale $\Delta$ / first modulus | $2^{59}$ / $2^{60}$ |
| modulus chain $Q$ | $60 + 59\times33 + 19.585$ bits (35 primes) |
| auxiliary chain $P_{\text{aux}}$ | $60\times12$ bits (12 primes) |
| $\log_2(QP_{\text{aux}})$ | 2,746.585 |
| key-switching | hybrid, $\text{dnum}=3$ |
| multiplicative-depth budget | 33 (13 levels after bootstrapping) |
| bootstrap level budget | $\{3, 3\}$ (CoeffsToSlots, SlotsToCoeffs) |
| application rotation keys | 21 at 1.7B-$d_8$, 24 at 135M-$d_8$, 23 at 360M-$d_8$ ($\approx$282 MiB per key), and BSGS uses a subset of them |
| bootstrap-internal keys | generated by the library, not counted among the application keys |
| secret / error | dense uniform ternary / discrete Gaussian, $\sigma$=3.19 |
| security | 128-bit classical per the OpenFHE table (Bossuat et al., 2025), $\log_2(QP_{\text{aux}})$ bound 3,523 at $N$=$2^{17}$ |
| software | FIDESlib 2.1.2 with our patches, OpenFHE 1.5.1 with the FIDESlib-required patch |

## C Validation and Precision

**Validation scope.** Table 6 lists the three validations, their executors, the configurations each covers, and what each does not cover. The GPU-executor checks compare the executor with an independent plaintext reference at four stages (before the refresh, before and after normalization, and at the output), with fixed pass criteria. Every ELA configuration and schedule whose result is reported passes them on fifteen small encrypted cases with partial-page boundaries ($M \leq 3$), scored per output group (30 checks per stage at 1.7B-$d_{32}$). PageFold also passes one representative case at $t$=8,192. The reported small-model $d_{16}$ and $d_{32}$ configurations under page-wise execution and shared refresh also pass one representative case at 8k each. The per-head 1.7B-$d_{16}$, 1.7B-$d_{32}$, and 360M-$d_8$ configurations also pass one at 8k under every reported schedule. The compact GQA baselines pass them on 22 small cases at 135M and 17 at 360M, and these cases also cover storage-page boundaries. The 1.7B native full-KV baseline passes them on the fifteen small cases under all five schedules and at 8k under PageFold. At $d_8$, the post-refresh check of the folded state passes ($15/15$ small cases, $1/1$ at 8k). In every timed run and capacity probe, including the compact GQA baselines, the attention input and output stay finite and within $|x| \leq 10$. The $d_8$ output level (12) meets the minimum input level of the following layer.

**Layout checkers.** Two plaintext checkers cover the two-head packing and the single-head variant of HDP (Appendix D.1). The two-head checker covers the 1.7B-$d_8$ geometry. At $d_8$, it reproduces the reference with a maximum difference of $9\times10^{-16}$. The single-head checker covers the six 135M and 360M configurations at $t$=$B$, $B-1$, and 1 and passes all 18 cases. Its stage differences lie between $10^{-18}$ and $10^{-13}$. The padding heads remain exactly zero, and no rotation falls outside the declared key set. A deliberately incorrect block-to-head map, applied once to each of the six configurations, is detected in all six. At $d_{32}$, a targeted check tests for head aliasing between the two output groups, which would make their aggregates identical. It confirms that the groups are separated in all fifteen cases under every $d_{32}$ schedule (Appendix D.1).

**Precision regime.** Calibrated noise-model sweeps inject noise matched to the $\approx$26-bit precision of bootstrap-free CKKS decode steps into 627k (135M) and 1.12M (360M) argmax decisions. At

this precision, the homomorphic softmax has an effective floor near $2^{-26}$. At each of the four evaluated KV compression ratios ($1\times$, $3.2\times$, $5.33\times$, and $8\times$), $99.9\%$ of the 135M decisions keep a margin between the top-1 and runner-up scores more than 10 bits above the floor. At 360M, a small number of decisions per $10^6$ fall at or below the floor and carry no top-1 guarantee. Among 800 real-ciphertext argmax decisions under Lattigo RNS-CKKS (Tune Insight SA and EPFL-LDS, 2024), the 100 decisions at operating margins all preserve top-1. The other 700 decisions sweep the top-1 margin over seven values from $2^{-5}$ to $2^{-30}$, with 100 decisions at each value. All 500 decisions at margins of $2^{-25}$ or larger preserve top-1. Below the softmax floor, 86 of 100 decisions preserve top-1 at $2^{-28}$ and 63 of 100 at $2^{-30}$. Three assembled bootstrap-free CKKS decode steps ($d_c$=24/48/96 at $t$=64) agree with the plaintext reference to 26.1 to 26.7 bits of precision.

**Persistent-cache loop.** Because the client encrypts each step input anew at the per-token protocol boundary (§1) and each layer's bootstraps refresh the activation ciphertexts within a step, the only ciphertext state that persists across decode steps is the latent cache. A multi-step loop probes this path. Over 100 chained cache-update steps (Lattigo RNS-CKKS, 135M dimensions), the cache is encrypted once before the loop and updated without intermediate cache re-encryption. In this validation loop, softmax weights are formed from decrypted scores at the per-step boundary. Decryption precision begins above 26 bits after bootstrapping and decreases over the 100-step runs. Full-run ordinary least-squares fits give slopes of $-0.027$ and $-0.025$ bits per step for $d_8$ and $d_{32}$, respectively. At $d_8$, the encrypted attention top-1 matches an fp64 reference on all 100 steps with a minimum margin of $2^{-19}$. Top-1 flips occur only in a $d_{32}$ run of the loop and in a long-cache run ($t\approx$800), at margins at or below the $\approx 2^{-20}$ loop floor. The loop bootstraps once per appended cache update, unlike the GPU timing circuit, and it tests for runaway error accumulation over 100 steps, not long-horizon stability.

The CPU validations of this section (precision regime, cache loop, and learned layers) execute the per-head formulation of the ELA layer. They do not exercise the head-diagonal GPU executor, whose checks are described above.

**Learned-layer CKKS validation.** Complete learned-checkpoint ELA layers of SmolLM-135M-$d_8$ ($t$=128, actual learned weights) run under Lattigo RNS-CKKS at $\Delta=2^{50}$ with a $\{60, 50\times 24\}$ modulus chain (multiplicative-depth budget 24) and $\log N$=16 (Table 7). This validates the learned polynomial circuit of layers 15, 28, and 29 under a separate CPU CKKS configuration. The GPU executor uses the parameters in Appendix B. The encrypted path covers the first RMSNorm, the absorbed attention projections, decoupled partial RoPE, the latent and RoPE cache append, content and RoPE scores, the normalized softmax, value aggregation, the absorbed output projection, the second RMSNorm, the gated MLP with a degree-31 polynomial approximation of SiLU, and both residuals. The primary reference is an fp64 implementation of the same polynomial circuit. Bootstraps are applied only to intermediates within their validated input ranges, and an algebraically equivalent normalization by the ratio $u/t$ of the softmax denominator $u$ to the number $t$ of cached tokens keeps intermediate magnitudes within those ranges. This bootstrap schedule is separate from the GPU timing configuration (Appendix B). The least precise intermediate of layer 15 retains 11.4 bits, above the 10-bit acceptance threshold. The encrypted output of layer 15 has a relative deviation of $2.22\times 10^{-2}$ from the fp32 checkpoint. The fp64 circuit reproduces this deviation. Replacing the polynomial approximation of SiLU with exact SiLU in the fp64 circuit reduces the relative deviation to $4.1\times 10^{-6}$. The gap is therefore dominated by the polynomial approximation. For the reported learned-layer validations, score domains were fixed before the runs. The chain 28→29 feeds layer 28's encrypted output to layer 29 without decryption, re-encoding, or repacking. Its 23 bootstraps comprise 10 in layer 28, one at the handoff, and 12 in layer 29. Layer 29 needs two more bootstraps than in its single-layer run because its carried input enters one level lower. It remains a two-layer component validation, not end-to-end encrypted generation.

Table 6: Scope of the three validations.

| Validation | Backend | Covered | Not covered |
|---|---|---|---|
| (i) precision regime | Lattigo, CPU, per-head | argmax margins at 135M and 360M<br>800 real-ciphertext decisions<br>100-step cache loop | GPU executor<br>learned multi-layer execution |
| (ii) learned-weight layers | Lattigo, CPU, per-head | 135M-$d_8$ layers 15, 28, 29<br>chain 28→29 | HDP GPU executor<br>checkpoint-wide execution |
| (iii) GPU executor checks | FIDESlib, GPU | 1.7B native full-KV and $d_8$/$d_{16}$/$d_{32}$ under five schedules<br>1.7B per-head $d_8$/$d_{16}$/$d_{32}$ under five schedules ($d_{32}$ per output group)<br>1.7B $r^\star$=24 under PageFold<br>135M and 360M $d_8$/$d_{16}$/$d_{32}$ and per-head $d_8$ at both scales under three schedules<br>compact GQA baselines under three schedules<br>8k under PageFold everywhere<br>8k at 1.7B-$d_{32}$ under residual-slot packing and shared refresh + mask<br>8k at small-model $d_{16}$/$d_{32}$ under their other reported schedules<br>8k at per-head 1.7B-$d_{16}$/$d_{32}$ and 360M-$d_8$ under every reported schedule<br>two-head (1.7B-$d_8$) and single-head plaintext checkers | 8k under the other schedules<br>residual-slot packing and shared refresh + mask at 135M/360M<br>model quality, full generation |

Table 7: Learned-weight CKKS validations (Lattigo RNS-CKKS, SmolLM-135M-$d_8$, $t$=128). Precision is measured against the fp64 circuit reference. Score domains, fixed before the runs, are $[-38, 0]$ for single layers and $[-26, 0]$ and $[-21, 0]$ in the chain.

| Check | Layer 15 | Layer 28 | Layer 29 | Chain 28→29 |
|---|---|---|---|---|
| output precision (bits) | 17.7 | 20.1 | 20.2 | 17.3 |
| softmax argmax (heads) | 9/9 | 9/9 | 9/9 | 9/9 per layer |
| probability sum | 0.999999 | 0.999999 | 0.999999 | 0.999999 per layer |
| score intervals in domain | yes | yes | yes | yes |
| cache-append lanes (bits) | 31–35 | >30 | >30 | 17.6–19.9 |
| bootstraps | 10 | 10 | 10 | 23 |

## D Latent-Cache Layout and Rotation Keys

We pack each latent in dim-major order. In the GPU executor, the content-score reduction spans the padded page width $P$ and uses $\log_2 P$ rotate-and-sum steps. The CPU validation reference instead zero-pads the $d_c$ content lanes to $P_{d_c}$=`nextPow2`$(d_c)$ and reduces them in $\log_2 P_{d_c}$ steps. These rotations and additions consume no multiplicative level. The logical cached per-token payload remains the latent plus the $d_R$ decoupled RoPE lanes, $d_c+d_R$, stored in $P$=`nextPow2`$(d_c+d_R)$ slots with page capacity $B=S/P$ (§4.1). With this layout, the rotation amounts of the attention computation within a page are fixed by the packed-slot geometry and model parameters and do not depend

on $t$. Extending the context therefore adds pages but no rotation keys for the attention computation.[1] From its packing description, we infer that the attention computation of the concurrent full-KV system (Yu et al., 2026) also uses rotation amounts that do not depend on $t$. The dim-major cache is the persistent storage layout, whereas the head-diagonal arrangement of §4.3 is an execution layout formed at run time. HDP therefore uses the diagonal encoding without storing the cache diagonally, and the append cost of the persistent cache is unchanged. The application rotation keys number 21 at 1.7B-$d_8$, 24 at 135M-$d_8$, and 23 at 360M-$d_8$, and the per-head and HDP executors use the same sets. HDP performs scoring, value aggregation, the shared reduction and masking of PageFold, and the construction of the query diagonals with these keys. Its BSGS evaluation uses a subset of them and introduces no additional keys. Bootstrap-internal automorphism keys are generated by the library and are not counted.

### D.1 HDP Slot Mapping

This subsection records the slot mapping of the HDP executor, on which §4.3 and Fig. 2 are based. HDP is an execution layout over the persistent dim-major pages above and is independent of the PageFold schedule. Slot $j = dB + i$ holds dimension $d < P$ of token $i < B$. The diagonal period $H_s$ is the smallest power of two not below $H$ (32 at every 1.7B rank and 16 at 135M and 360M). The output packing width $S_K$ is the smallest power of two not below $\min(H, B)$ (32 at 1.7B-$d_8$/$d_{16}$ and 16 at 1.7B-$d_{32}$, 135M, and 360M). Block $d$ belongs to head $h(d) = d \bmod H_s$, and $\rho_r$ rotates by $rB$ slots ($r$ blocks).

**Cache, query diagonals, and scores.** A page stores $C[dB + i] = c[d,\, mB + i]$, with the latent in blocks $d < d_c$ and the per-group RoPE keys in the remaining blocks, and it is unchanged by HDP. The query diagonals are $\Delta_r[d, i] = q_{h(d)}[(d + r) \bmod P]$ for $r < H_s$, replicated over the tokens $i$. The diagonal products sum to $Z = \sum_{r<H_s} \Delta_r \odot \rho_r(C)$. A coset fold then starts from $Y \leftarrow Z$ and applies $Y \leftarrow Y + \rho_{H_s \cdot 2^k}(Y)$ for $k < \log_2(P/H_s)$, after which $Y[d, i] = s_{h(d)}[i]$ in every block. No per-head mask is applied, and the exponential uses the same affine map and Chebyshev approximation as the per-head executor on all blocks. The RoPE blocks of the page are untouched. Since the query of head $h$ has RoPE entries only in the RoPE blocks of its own KV group and the diagonal placement is a cyclic shift of block indices, the RoPE products are summed into the same score and pass through the same exponential.

**BSGS schedule and query diagonals.** With $H_s = n_1 n_2$ ($n_1$=8, $n_2$=4 for $H_s$=32 and $4 \times 4$ for $H_s$=16), $Z = \sum_{b<n_2} \rho_{n_1 b}\big(\sum_{a<n_1} D'_{a,b} \odot \rho_a(C)\big)$ with $D'_{a,b} = \rho_{-n_1 b}(\Delta_{a+n_1 b})$. Per page this costs $n_1 - 1$ baby-step rotations, $\sum_{b\geq 1} \mathrm{popcount}(n_1 b)$ giant-step rotations, and $\log_2(P/H_s)$ fold rotations, which is $7 + 4 + 4 = 15$ at 1.7B-$d_8$. The $n_1$ baby-step results stay live during the giant steps. The diagonals are built from the encrypted per-head queries $Q_h$ (at level 13) as $K_r = \sum_h M_{(h+r) \bmod H_s} \odot Q_h$ and $D'_{a,b} = \rho_a(K_{a+n_1 b})$, where $M_h$ masks the blocks of residue $h$. Because $\rho_{-r}(M_h) = M_{(h+r) \bmod H_s}$, only $H_s$ distinct masks exist. At 1.7B-$d_8$ this costs 1,024 ciphertext-plaintext multiplications, 992 additions, 48 rotations, and one level, once per decode step and independent of the page count. No plaintext query and no precomputed diagonal query are used.

**Weights and two-head value packing.** With the exponentiated scores $E$ and the token-validity mask $T$, the weights are $E_v = E \odot T$. Two-head packing forms $E'' = (M_{\mathrm{lo}} T) \odot E + (M_{\mathrm{hi}} T) \odot \rho_{H_s/2}(E)$ and the products $U_k = C_2 \odot \rho_k(E'')$ for $k < H_s/2$, where $C_2$ is the latent duplicated into both halves of the page. Without it, $U_r = C \odot \rho_r(E_v)$ for $r < H_s$. Two-head packing requires $P = 2d_c$ (the latent and its duplicate each fill one half of the page), $d_c \equiv 0 \pmod{H_s}$ (the head cosets are preserved across the duplication), and a rotation by $d_c B$ slots from the application keys or their composition. The output is packed separately in groups of $S_K$ heads (below), so $B \geq H_s$ is not required. At 1.7B, the ranks $d_8$, $d_{16}$, and $d_{32}$ satisfy these conditions with $H_s$=32. They use $H_s/2 = 16$ value products, which $d_{32}$ shares across its two output groups. At 135M and 360M,

[1] The application key counts below are those requested by the attention computation of the GPU executor, which does not execute the persistent-cache append. The CPU persistent-cache loop implements its append shifts with one rotation key per cached column ($-1, \ldots, -t_{\max}$). These shifts could instead be composed from power-of-two rotation keys. The learned-weight validation uses a fixed set of $\pm 2^j$ rotation keys that does not depend on $t$.

every rank violates $P = 2d_c$ (for example $d_c$=24 and $P$=64 at 135M-$d_8$, and $d_c$=40 and $P$=128 at 360M-$d_8$) and runs the single-head variant. The diagonal scores and BSGS are unchanged ($H_s$=16, $n_1$=$n_2$=4), and each value product carries one head ($H_s$ products per page). One mask applied to the weights zeroes both the padding residues $h \geq H$ of the diagonal period and the padding tokens of the last partial page. The padding lanes of the accumulators are therefore exactly zero and enter neither the denominators nor the output. The bucket-maximal $r^\star$=24 at 1.7B ($d_c$=768, $P$=1,024 $\neq 2d_c$) runs the single-head variant with $H_s$=32 in one output group.

**Accumulators, output groups, and refresh points.** Before the refresh, the summed page states are packed in the skewed layout $G(d, s) = n_{h(d+s)}[d]$ for $d < d_c$, $s < H$, where $n_h[d] = \sum_i e_h[i]\, c[d, i]$, and the denominators $u_h = \sum_i e_h[i]$ are placed as $U_{\mathrm{den}}(d, s) = u_{h(d+s)}$ by a mask, a coset fold, and a broadcast. At 1.7B-$d_8$ under PageFold, the accumulator count falls from 64 (32 heads and 32 denominators) under the per-head executor to 17 under HDP, because two heads share each value product and the denominators share one accumulator. The output-group count (1), the refresh count (1), and the bootstrap count (3) are unchanged. The page-alignment operations fall from 8,192 inside the products to 256 explicit ones. At 1.7B-$d_{32}$ the two output groups of 16 heads share the 16 numerator accumulators and the denominator accumulator. They are separated in the output packing and then refreshed per group (Appendix B). At 135M and 360M the single-head variant keeps $H_s$ value accumulators in one group. Under page-wise execution, residual-slot packing, and shared refresh, the refresh policy is that of the per-head executor, and each page's value products are summed over tokens and packed into the skewed layout. Page-wise execution adds the page state to the running sum and refreshes in place ($A \leftarrow \textsc{Bootstrap}(A + \mathcal{L}(V_p))$, Algorithm 1). Residual-slot packing gathers $G_{\mathrm{res}}$ pages in slot phases and refreshes them once. Shared refresh accumulates the page sums and refreshes once after the loop.

**Output recovery.** For one output group, the ciphertext $Z_s$ in the skewed layout is converted to the output layout $O(d, h) = o_h[d]$ at slot $dB + h$ ($d < d_c$, $h < H$, other slots zero) by

$$O = \sum_{a<H_s} \left[M_a^{\mathrm{lo}} \odot \rho_{-a}(Z_s) + M_a^{\mathrm{hi}} \odot \rho_{H_s-a}(Z_s)\right],$$

which costs $2H_s - 1 = 63$ rotations and 63 ciphertext-plaintext multiplications at one level at $d_8$ and $d_{16}$. With several output groups, head $h = gS_K + h'$ is placed at slot $dB + h'$ of the output ciphertext of group $g$, and at 1.7B-$d_{32}$ each group costs $2S_K - 1 = 31$ rotations and 31 multiplications. This level replaces the level of the previous output mask, so the output dimensions and group structure are unchanged, and the output still meets the minimum level of 10 required by the following layer.

**Conditions.** One output group holds all heads when $S_K \geq H$, that is, when $H \leq B = S/P$, and the diagonal placement requires $H_s \mid P$. The per-head condition $Hd_c \leq S$ is an output-placement condition ($H \leq S/d_c$). Because $P \geq d_c$ gives $S/P \leq S/d_c$, the single-group condition $H \leq S/P$ implies $H \leq S/d_c$ and is the stronger condition. A geometry with $P > d_c$ can therefore satisfy $Hd_c \leq S$ without admitting one group. At 1.7B-$d_{32}$, $Hd_c$ = 32,768 = $S$ holds at the boundary, but $P$ = 2,048 gives $B = 16 < H = 32$, so the executor forms $N_G = 2$ output groups of $S_K = 16$ heads while keeping the diagonal period $H_s = 32$ of the full head set. By design the per-head executor also uses two output groups there. This configuration is included in the layout and schedule comparison of Appendix E. The evaluated configurations are therefore one group with two-head packing ($d_8$, $d_{16}$), two groups with two-head packing ($d_{32}$), and one group with the single-head variant (the 1.7B $r^\star$=24 case and all ranks at 135M and 360M). They need at most two groups ($B \geq H/2$). A general rule for more groups is beyond the scope of this paper.

## E Schedules: Definitions and Full Ablation

**Schedule definitions.** For a fixed page geometry and execution layout, the executed schedule is one of the five in Table 8, applied independently to each output ciphertext group. The schedules differ only in whether each of the reduction, masking, and refresh runs per page, once per $G_{\mathrm{res}}$ pages, or once after the aligned page states are summed. All are algebraically equivalent before CKKS approximation, by the linearity identity of §4.4. Residual-slot packing refreshes once per $G_{\mathrm{res}}$ pages and coincides with page-wise execution when $G_{\mathrm{res}}$=1. It serves as a control configuration.

The page-wise order $A \leftarrow \textsc{Bootstrap}(A + \mathcal{L}(V_p))$ of Algorithm 1 is the order implemented by our GPU executor. All latency measurements follow the timing protocol of §5, whose details and stability notes are in Appendix F.3.

Table 8: Schedules compared in this appendix, for one output ciphertext group of $M$ pages. "Per page" runs inside the page loop, and "after fold" runs once on the summed page states. The artifact configuration labels are `OFF`, `P`, `S0`, `S0M`, and `S0RM`, respectively.

| Schedule | Reduction | Masking | Refresh | Refreshes/group |
|---|---|---|---|---|
| Page-wise | per page | per page | per page (running sum) | $M$ |
| Residual-slot packing | per page | per page | per $G_{\mathrm{res}}$ pages | $\lceil M/G_{\mathrm{res}} \rceil$ |
| Shared refresh | per page | per page | after fold | 1 |
| Shared refresh + mask | per page | after fold | after fold | 1 |
| PageFold | after fold | after fold | after fold | 1 |

**Layout × schedule.** Table 9 collects the measurements at 1.7B and $t$=8,192 for the execution layouts of §6.3, all taken under the same protocol (Appendix F.3). Residual-slot packing gathers $G_{\mathrm{res}}{=}B/S_K$ page states in slot phases before each refresh. At $d_8$, $G_{\mathrm{res}}{=}2$ ($B{=}64$, $S_K{=}32$), and the per-head layout also refreshes once per two pages. At $d_{16}$ and $d_{32}$, $S_K{=}B$ and hence $G_{\mathrm{res}}{=}1$. Residual-slot packing then degenerates to page-wise execution with one refresh per page, which the near-equal times reflect. Native full-KV gathers $G_{\mathrm{res}}{=}8$ pages per refresh, so its 1,024 pages take 128 refreshes.

Table 9: Execution layout × schedule at 1.7B and $t$=8,192 on the GPU executor (seconds per decode step of one layer's attention core, median of three runs, protocol in Appendix F.3 and per-cell ranges in the artifact).

| Representation | Layout | Page-wise | Residual-slot | Shared refresh | Shared refresh + mask | PageFold |
|---|---|---|---|---|---|---|
| native full-KV | native | 334.13 | 81.58 | 43.78 | 42.03 | 37.12 |
| ELA-$d_8$ | per-head | 215.37 | 197.91 | 181.18 | 157.91 | 119.20 |
| ELA-$d_8$ | HDP | 74.79 | 56.85 | 38.61 | 28.36 | 18.48 |
| ELA-$d_{16}$ | per-head | 434.17 | 435.72 | 365.38 | 319.17 | 254.45 |
| ELA-$d_{16}$ | HDP | 144.12 | 144.34 | 71.61 | 51.02 | 34.44 |
| ELA-$d_{32}$ | per-head (two groups) | 1,014.62 | 1,011.79 | 720.97 | 646.36 | 543.11 |
| ELA-$d_{32}$ | HDP (two groups) | 425.07 | 424.74 | 134.10 | 93.94 | 67.50 |

**Sources of the PageFold gain.** Within the HDP layout, folding removes work repeated for every page in addition to the refreshes. Shared refresh already folds the refresh, so the step from shared refresh to PageFold leaves the refresh count unchanged. This step reflects the folding of the reduction and masking, which Table 10 shows as fewer rotations and ciphertext-plaintext multiplications. The measured operation counts agree with those predicted from the schedule. In Table 10, $r^\star{=}24$ shares the page bucket of $d_{16}$ but has a wider latent and runs without two-head packing. Its operation counts do not separate the effects of these two differences.

**Per-head scoring workspace.** Under per-head execution, every head reads the whole shared latent, so the total logical scoring width is $H(d_c{+}d_{\mathrm{rope}})$ lanes against $Hd_{\mathrm{head}}$ for full-KV, a ratio of $(d_c{+}d_{\mathrm{rope}})/d_{\mathrm{head}}$ in which $H$ cancels. Because $d_c = r\,n_{\mathrm{kv}}$ grows with $n_{\mathrm{kv}}$ (3, 5, and 32 at 135M, 360M, and 1.7B), the ratio rises with scale and exceeds one at every 1.7B rank and is about $4\times$ for the $8\times$-compressed cache. The ratio counts logical scoring lanes, not GPU memory, ciphertexts, or time. HDP reduces the repeated rotations and reductions across heads (§4.3).

**Exponential evaluation.** Both representations evaluate the softmax exponential once per computation page under the same per-page policy. The number of evaluations therefore follows the computation page count. At $t$=8,192 it is 128, 256, 512, and 1,024 at 1.7B for $d_8$, $d_{16}$, $d_{32}$, and native full-KV. For ELA-$d_8$, $d_{16}$, and $d_{32}$ it is 16, 32, and 32 at 135M and 32, 32, and 64 at 360M. The compact GQA baselines evaluate it 512 times at both scales, once per computation page of 16 tokens rather than once per storage page.

Table 10: Operation counts per layer and decode step at $t$=8,192, with latency. HDP counts include query-diagonal construction and output recovery. At 135M and 360M the diagonal period (16) exceeds the head count (9 and 15), so ct×ct multiplications increase while rotations fall.

| Model | Configuration | Rotations | ct×ct | ct×pt | Bootstraps | Latency (s) |
|---|---|---|---|---|---|---|
| 1.7B | ELA-$d_8$, per-head, PageFold | 74,281 | 8,197 | 8,290 | 3 | 119.20 |
| 1.7B | ELA-$d_8$, HDP, shared refresh | 29,566 | 6,149 | 9,568 | 3 | 38.61 |
| 1.7B | ELA-$d_8$, HDP, PageFold | 4,420 | 6,149 | 1,567 | 3 | 18.48 |
| 1.7B | ELA-$d_{16}$, HDP, PageFold | 8,756 | 12,293 | 1,951 | 3 | 34.44 |
| 1.7B | $r^\star$=24, HDP, PageFold | 12,403 | 16,389 | 1,665 | 3 | 41.02 |
| 1.7B | ELA-$d_{32}$, HDP, PageFold | 17,696 | 24,586 | 2,752 | 5 | 67.50 |
| 135M | ELA-$d_8$, per-head, PageFold | 1,922 | 293 | 317 | 3 | 4.60 |
| 135M | ELA-$d_8$, HDP, PageFold | 626 | 517 | 227 | 3 | 2.75 |
| 360M | ELA-$d_8$, per-head, PageFold | 7,023 | 965 | 1,007 | 3 | 13.19 |
| 360M | ELA-$d_8$, HDP, PageFold | 1,026 | 1,029 | 367 | 3 | 4.35 |

**Memory.** Every schedule accumulates page-local states into a fixed set of ciphertexts, so none retains intermediate states for all $M$ pages. The reported memory metric is the GPU allocator-pool high-water mark. Under HDP at $d_8$, page-wise execution and PageFold reach the same mark of 41,993 MiB, 1,024 MiB below that of the per-head layout under PageFold. As a pool metric it does not resolve the live memory of individual stages or schedules, and we do not attribute the difference to individual causes.

# F Additional Measurements

## F.1 Smaller Models

Table 11 collects the 135M and 360M system measurements, with the unit and protocol of the 1.7B results.

Table 11: Smaller models at $t$=8,192, with attention-core latency in seconds and layer-local resident capacity under PageFold. Baseline page counts are native-width storage pages (§5).

| Model | Configuration | Page-wise | Shared refresh | PageFold | Pages / tokens |
|---|---|---|---|---|---|
| 135M | full-KV (compact GQA) | 205.53 | 60.03 | 55.97 | 61 / 3,904 |
| 135M | ELA-$d_8$, per-head | 11.24 | 7.01 | 4.60 | 67 / 34,304 |
| 135M | ELA-$d_8$, HDP | 9.09 | 4.85 | 2.75 | 64 / 32,768 |
| 135M | ELA-$d_{16}$, HDP | 16.82 | 8.02 | 4.10 | 67 / 17,152 |
| 135M | ELA-$d_{32}$, HDP | 16.84 | 8.02 | 4.10 | 67 / 17,152 |
| 360M | full-KV (compact GQA) | 236.76 | 91.44 | 87.31 | 60 / 1,920 |
| 360M | ELA-$d_8$, per-head | 29.89 | 21.07 | 13.19 | 64 / 16,384 |
| 360M | ELA-$d_8$, HDP | 17.11 | 8.29 | 4.35 | 66 / 16,896 |
| 360M | ELA-$d_{16}$, HDP | 17.11 | 8.28 | 4.36 | 67 / 17,152 |
| 360M | ELA-$d_{32}$, HDP | 32.33 | 14.31 | 7.11 | 68 / 8,704 |

## F.2 Capacity and Rank Fitting Tables

Capacity boundaries are measured with the FHE context and all required keys resident before cache allocation. Each boundary is found in a fresh process by loading $M$ distinct persistent encrypted cache pages and increasing $M$ until the attention layer no longer executes with everything resident in GPU memory (§5). The boundary is the largest $M$ that executes, and $M$+1 fails with an out-of-memory error. The main sweeps use PageFold for both representations, and Table 12 adds the page-wise and shared-refresh boundaries where measured. The measurement is layer-local. One layer's cache pages, the FHE context, all required keys, and the transient execution workspace must fit simultaneously on the GPU, so the boundary does not represent whole-model resident capacity, and model weights and RMSNorm/FFN plaintexts are outside this accounting. The 135M and 360M full-KV baselines are GQA models ($H$=9, $n_{\mathrm{kv}}$=3 and $H$=15, $n_{\mathrm{kv}}$=5, $d_{\mathrm{head}}$=64) with a compact persistent cache. Their storage pages hold only the native $n_{\mathrm{kv}}$ keys and values (384 and 640 values per token), so $P$=512 and $P$=1,024, respectively, and each storage page holds $B$=64 or $B$=32 tokens. Inside the timed core each storage page is unpacked into 4 or 2 head-expanded computation

pages of 16 tokens ($P$=2,048), so $t$=8,192 spans 128 or 256 storage pages but 512 computation pages. The capacity ratios of ELA-$d_8$ over these baselines decompose as $8 \times (64/61) = 8.39\times$ at 135M and $8 \times (66/60) = 8.80\times$ at 360M, the compression ratio in tokens per page times the ratio of executable page counts. At $d_8$, PageFold trades one page, about $1.7\%$ of the resident capacity, for its $2.09\times$ latency gain over shared refresh, and at $d_{16}$ the measured capacity is unchanged. These observations cover the two measured configurations. Under the same PageFold schedule, the per-head layout of ELA-$d_8$ reached 55, 67, and 64 pages at 1.7B, 135M, and 360M, against 59, 64, and 66 pages under HDP. HDP keeps $H_s$ query diagonals resident while reducing the value accumulators from $2H$ to $H_s/2 + 1$ with two-head packing or $H_s + 1$ without it. These ciphertext counts alone do not predict the measured capacity boundary, which depends on the full live workspace. Table 13 lists the page geometry of the measured configurations.

Table 12: Layer-local resident capacity $B \times M_{\max}$ on the GPU executor (§6.1), with $B$ tokens per page and $M_{\max}$ resident pages. For the compact GQA full-KV baselines, $B$ and $M_{\max}$ count storage pages at the native KV width.

| Model | Config | KV reduction | Schedule | $B$ | $M_{\max}$ | Tokens |
|---|---|---|---|---|---|---|
| 1.7B | native full-KV | 1× | page-wise | 8 | 63 | 504 |
| 1.7B | native full-KV | 1× | PageFold | 8 | 63 | 504 |
| 1.7B | ELA-$d_8$ | 8× | PageFold | 64 | 59 | 3,776 |
| 1.7B | ELA-$d_8$ | 8× | shared refresh | 64 | 60 | 3,840 |
| 1.7B | ELA-$d_{16}$ | 5.33× | PageFold | 32 | 61 | 1,952 |
| 1.7B | ELA-$d_{16}$ | 5.33× | shared refresh | 32 | 61 | 1,952 |
| 1.7B | $r^\star$=24 | 4× | PageFold | 32 | 55 | 1,760 |
| 1.7B | ELA-$d_{32}$ | 3.2× | PageFold | 16 | 62 | 992 |
| 135M | full-KV (compact GQA) | 1× | PageFold | 64 | 61 | 3,904 |
| 135M | ELA-$d_8$ | 8× | PageFold | 512 | 64 | 32,768 |
| 135M | ELA-$d_{16}$ | 5.33× | PageFold | 256 | 67 | 17,152 |
| 135M | ELA-$d_{32}$ | 3.2× | PageFold | 256 | 67 | 17,152 |
| 360M | full-KV (compact GQA) | 1× | PageFold | 32 | 60 | 1,920 |
| 360M | ELA-$d_8$ | 8× | PageFold | 256 | 66 | 16,896 |
| 360M | ELA-$d_{16}$ | 5.33× | PageFold | 256 | 67 | 17,152 |
| 360M | ELA-$d_{32}$ | 3.2× | PageFold | 128 | 68 | 8,704 |

Table 13: Page geometry of the measured ELA configurations at $t$=8,192 (§4.1, §4.2), with $P$=`nextPow2`$(d_c{+}d_R)$, $B$=$S/P$, $M$=$\lceil t/B \rceil$, and $N_G$ output groups. The last column reports the HDP value products (§4.3). Configurations in the same bucket share $P$, $B$, and $M$.

| Model | Config | $d_c{+}d_R$ | $P$ | $B$ | $M$ | $N_G$ | Value products |
|---|---|---|---|---|---|---|---|
| 1.7B | ELA-$d_8$ | 512 | 512 | 64 | 128 | 1 | two-head |
| 1.7B | ELA-$d_{16}$ | 768 | 1,024 | 32 | 256 | 1 | two-head |
| 1.7B | $r^\star$=24 | 1,024 | 1,024 | 32 | 256 | 1 | single-head |
| 1.7B | ELA-$d_{32}$ | 1,280 | 2,048 | 16 | 512 | 2 | two-head |
| 135M | ELA-$d_8$ | 48 | 64 | 512 | 16 | 1 | single-head |
| 135M | ELA-$d_{16}$ | 72 | 128 | 256 | 32 | 1 | single-head |
| 135M | ELA-$d_{32}$ | 120 | 128 | 256 | 32 | 1 | single-head |
| 360M | ELA-$d_8$ | 80 | 128 | 256 | 32 | 1 | single-head |
| 360M | ELA-$d_{16}$ | 120 | 128 | 256 | 32 | 1 | single-head |
| 360M | ELA-$d_{32}$ | 200 | 256 | 128 | 64 | 1 | single-head |

### F.3 Latency Protocol and Stability

**Timing scope.** We measure the per-layer encrypted attention core at $t$=8,192 on our GPU executor (§6.2, §6.3). Each latency run uses one RTX A6000 and 16 OpenMP threads with pinned CPU affinity. The artifact records the GPU assignment and CPU affinity of each run. Each configuration runs in three fresh processes with one warm-up and one timed execution. The reported value is the median of the three runs, with the observed minimum and maximum where reported. The range is not a confidence interval. The unit is the time of one decode step of one layer's attention core. The harness keeps two encrypted cache pages resident on the GPU and reuses them as the operands of every page iteration. The arithmetic, rotations, page loop, and refresh schedule are those of the $M$-page computation. Encryption, decryption, and the output checks are outside the timed region.

Weights are synthetic plaintexts, and RMSNorm and the FFN, which lie outside the timed core, are placeholders. Because the executed CKKS circuit does not depend on the weight or input values, operation counts, depths, rotations, and the bootstrap schedule are exact for the attention circuit. Token outputs are not model outputs. The latency entries were measured with several builds of the same GPU executor. The artifact manifest records the name and hash of each build. It also records a comparison of the builds on shared entries (execution path, operation counts, and level schedule). Correctness is validated separately in Appendix C. The timed region includes the construction of the query diagonals and the recovery of the output layout. The query and output projections, cache append, LM head, and sampling are excluded for every configuration.

**Stability.** All latency entries in Tables 1, 2, 3, 9, and 11 are three-run medians under the same protocol, and the observed minimum and maximum of every entry are recorded in the artifact CSV. The widest observed range at 1.7B is $0.4\%$ of the median (ELA-$d_8$ under shared refresh, $38.54$ to $38.69$ s), and the widest overall is $1.2\%$ (135M ELA-$d_8$ per-head under PageFold, $4.59$ to $4.65$ s). The main results span $334.11$ to $334.16$ s for page-wise full-KV and $18.48$ to $18.50$ s for ELA-$d_8$.

### F.4 Expanded Quality Detail

**Checkpoint and matched-pair training.** We use the author-released MHA2MLA conversions of SmolLM v1 (Ji et al., 2025), with revisions pinned in the artifact. The conversion uses a joint-SVD shared latent and an 8-dimensional partial-RoPE stream per KV head, giving $d_c = r\, n_{\mathrm{kv}}$ (§2). For the controlled within-bucket comparison in §6.1, we initialize $r{=}16$ and $r^\star{=}24$ from the same released 1.7B base checkpoint using the same joint-SVD procedure. We train both on the same FineWeb-Edu subset (Penedo et al., 2024) with a $1{,}500$-step cosine schedule (300-step warmup, effective batch 256) and compare them at step $1{,}000$, the latest checkpoint available for both runs. Only the latent rank differs. Because this training recipe differs from that of the released checkpoints, we compare perplexity only within the pair.

Table 14: Scale grid, our additional plaintext evaluation of the released MHA2MLA checkpoints. WikiText-2 and PG19 perplexity against the unconverted fp32 base, with $\Delta\%$ from unrounded values. The protocol is in Appendix F.4.

| Model | Config | Ratio | $\mathrm{PPL}_{\mathrm{wiki}}$ | $\Delta\%_{\mathrm{wiki}}$ | $\mathrm{PPL}_{\mathrm{pg19}}$ | $\Delta\%_{\mathrm{pg19}}$ |
|---|---|---|---|---|---|---|
| 135M | base | 1.0× | 15.94 | 0.0 | 23.73 | 0.0 |
| | $d_{32}$ | 3.2× | 17.81 | +11.7 | 26.58 | +12.0 |
| | $d_{16}$ | 5.33× | 19.84 | +24.5 | 29.38 | +23.8 |
| | $d_8$ | 8.0× | 23.48 | +47.3 | 33.86 | +42.7 |
| 360M | base | 1.0× | 11.93 | 0.0 | 17.99 | 0.0 |
| | $d_{32}$ | 3.2× | 12.84 | +7.7 | 19.46 | +8.2 |
| | $d_{16}$ | 5.33× | 13.96 | +17.0 | 20.84 | +15.8 |
| | $d_8$ | 8.0× | 15.62 | +30.9 | 23.07 | +28.2 |
| 1.7B | base | 1.0× | 8.73 | 0.0 | 13.28 | 0.0 |
| | $d_{32}$ | 3.2× | 9.14 | +4.7 | 14.08 | +6.0 |
| | $d_{16}$ | 5.33× | 9.41 | +7.8 | 14.45 | +8.9 |
| | $d_8$ | 8.0× | 9.88 | +13.2 | 15.01 | +13.0 |

**Expanded quality evaluation (detail).** WikiText-2 (Merity et al., 2017) perplexity uses a sliding window with context $2{,}048$, stride $1{,}024$, and a 60k-token cap. PG19 (Rae et al., 2020) perplexity uses a fixed 32-document evaluation with two non-overlapping $2{,}048$-token windows per document ($131{,}008$ scored tokens per model), and all $\Delta\%$ values are computed from unrounded perplexities. The six English LongBench (Bai et al., 2024) QA tasks of Table 15 are NarrativeQA, Qasper, MultiFieldQA-en, HotpotQA, 2WikiMQA, and MuSiQue. Each model is evaluated on all $1{,}150$ examples of the six tasks and scored as equal-weight macro QA-F1 under greedy decoding. All intervals are paired $95\%$ bootstrap intervals over evaluation samples ($10{,}000$ resamples, seed 20260717) and do not measure training-seed variance. A separate training-seed check retrains the 1.7B-$d_8$ configuration with the matched-pair protocol above under three training seeds and a fixed data order. The across-seed WikiText-2 dispersion is about $0.06\%$ ($n{=}3$, range $0.10\%$), far below the $6.1\%$ within-pair difference of §6.1 and the $+13.2\%$ $d_8$ penalty at that scale (Table 14). These retrained checkpoints are not comparable in absolute terms to the released check-

points. For the expanded PG19 evaluation, the paired $95\%$ intervals of the relative perplexity change exclude zero for all nine compressed configurations. The $d_8$ intervals are $[+40.8\%, +44.7\%]$, $[+27.1\%, +29.4\%]$, and $[+12.5\%, +13.6\%]$ at 135M, 360M, and 1.7B. As a budget robustness check, we reran all twelve checkpoints on the same fixed prompt sets with the official task-specific budgets ($128/128/64/32/32/32$ tokens). The within-family ordering of macro-F1 is unchanged. The three tasks whose official budget is 32 tokens coincide exactly with the 32-token-budget runs. Macro-F1 differs by at most $0.42$ points, confined to the three longer-budget tasks. This is a single greedy rerun without paired intervals.

Table 15: Downstream accuracy, our additional evaluation of the released MHA2MLA checkpoints. Six-task LongBench QA macro-F1 under the deployed 32-token generation budget (not leaderboard-comparable), with $\Delta\%$ against the same-scale base and paired $95\%$ evaluation-sample intervals. Tasks and protocol in Appendix F.4.

| Model | Config | Macro F1 | $\Delta\%$ | 95% CI ($\Delta\%$) |
|---|---|---|---|---|
| 135M | base | 11.74 | 0.0 | — |
| | $d_{32}$ | 10.98 | $-6.4$ | $[-15.1, +3.6]$ |
| | $d_{16}$ | 10.17 | $-13.3$ | $[-22.0, -3.9]$ |
| | $d_8$ | 9.07 | $-22.8$ | $[-31.5, -13.1]$ |
| 360M | base | 13.78 | 0.0 | — |
| | $d_{32}$ | 12.84 | $-6.8$ | $[-14.5, +1.3]$ |
| | $d_{16}$ | 11.19 | $-18.7$ | $[-25.6, -11.5]$ |
| | $d_8$ | 9.45 | $-31.4$ | $[-38.0, -24.3]$ |
| 1.7B | base | 16.38 | 0.0 | — |
| | $d_{32}$ | 16.25 | $-0.8$ | $[-7.5, +6.3]$ |
| | $d_{16}$ | 15.77 | $-3.7$ | $[-10.7, +4.0]$ |
| | $d_8$ | 14.63 | $-10.7$ | $[-17.7, -3.1]$ |