CAN FIRST-ORDER MEMORY REPAIR COMPRESSED ATTENTION?

Anonymous authors
Paper under double-blind review

ABSTRACT

Compressing attention by sharing key and value (KV) heads reduces KV-cache storage, but can hurt predictions. We address this loss of predictive capacity through a repair mechanism called *First-order Memory* (FoM). Specifically, FoM adds a compact recurrent state that summarizes causal features with storage independent of context length. We then study how well FoM repairs compressed attention, and assess whether it can also outperform stateless repair. Across a controlled byte-level study (one UTF-8 byte per token) and studies using Pythia-class models (up to 1B), FoM recovers prediction quality, often even outdoing comparably sized stateless MLP repair. In a held-out 410M MQA study, FoM lowers mean cross-entropy by 0.02235 nats per token over the selected MLP across five new seeds; this strong advantage is, however, not universal, and in some cases a strong MLP nearly matches mean performance. Overall, the experiments are encouraging and suggest that FoM can provide valuable repair, possibly at an additional execution cost.

1 INTRODUCTION

Autoregressive transformers pay for long contexts by storing a KV cache that grows with context length and batch size (Vaswani et al., 2017; Shazeer, 2019), limiting context length and concurrency under a fixed memory budget. Multi-query attention (MQA) cuts this storage by sharing one KV head across query heads, while grouped-query attention (GQA) keeps fewer KV heads and can be obtained by converting multi-head attention (MHA) checkpoints (Ainslie et al., 2023). These methods save KV-cache storage, but they may also lose predictive capacity. One fix is to restore the missing KV heads, but that gives up the memory savings we wanted. We instead ask whether that lost predictive capacity can be repaired while keeping the compressed KV cache.

FoM adds a compact repair state alongside compressed attention (Figure 1). It carries information across positions without rebuilding per-token KV caches removed by head sharing, and its storage does not grow with context length. We ask two questions:

Does FoM recover prediction quality lost through compression?
Can FoM outperform an MLP repair with a comparable parameter count?

The MLP provides the stateless control. Both repairs see contextual inputs, but only FoM carries repair state between positions. Because extra parameters alone may improve predictions, we compare FoM with the MLP in prediction quality, live attention-state storage and execution cost. A smaller KV cache alone does not guarantee lower total inference memory or faster generation.

We test FoM first in a controlled study with a byte-level model, then in three MQA and three GQA Pythia settings with reduced KV caches (Biderman et al., 2023). In held-out 410M MQA, EMA-FoM lowers mean cross-entropy (CE) by 0.02235 relative to the selected MLP across five new seeds. The advantage is not universal. At fixed KV2, a strong MLP nearly matches mean CE, and frozen transfer to PG19 reverses the result before adaptation restores FoM's advantage. A single-seed deployment reconstruction finds execution overhead and no extra batch capacity over the MLP. Together, these results suggest that FoM can recover prediction quality without restoring KV heads, although the gain depends on configuration.

2 REPAIR AFTER HEAD COMPRESSION

With the central questions and empirical picture in place, we now make FoM precise. We first fix notation for head-shared attention, then define the repair.

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.

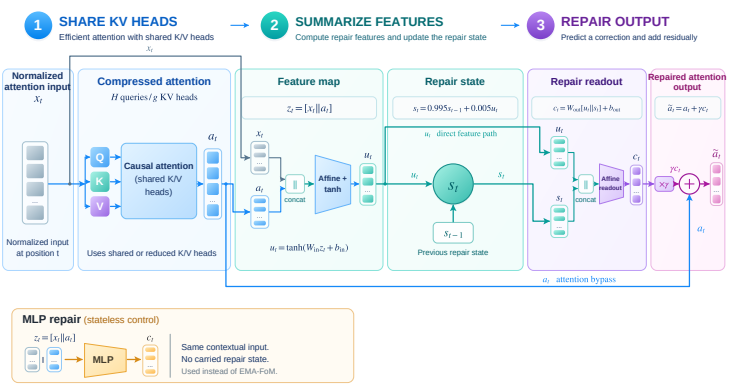


Figure 1: EMA-FoM at one layer and token position t. The feature map is affine followed by $\tanh$. The affine readout takes repair features u_t and repair state s_t. The inset shows the MLP repair used in place of EMA-FoM. Block residuals and feed-forward sublayers are omitted.

Head sharing saves memory by using fewer KV heads. KV2 and KV4 denote layouts with two and four KV heads. In a width-d layer with H query heads of width $d_h = d/H$, let $g \mid H$ be the number of physical KV heads. Conversion averages KV weights and biases within contiguous groups (Ainslie et al., 2023). At fixed batch size, context length and dtype, KV storage is g/H of MHA. FoM asks whether this saving survives repair. KV count and repair placement may vary by layer.

At position t, FoM sees the normalized attention input $x_t \in \mathbb{R}^d$ and compressed attention output $a_t \in \mathbb{R}^d$ after output projection. It forms z_t and adds correction c_t with fixed scale γ.

$$z_t = [x_t \,\|\, a_t], \qquad \widetilde{a}_t = a_t + \gamma c_t. \tag{1}$$

The rest of the block, including Pythia's parallel feed-forward residual, is unchanged. Zero initialization of the repair readout gives $c_t = 0$, so FoM starts from the compressed model. The stateless MLP uses the same interface, $c_t = V\phi(Wz_t + b) + b_o$. Earlier MQA controls use GELU with development-selected rates, while fixed-KV2 searches GELU/tanh. Parameter counts are matched, though activations and tuning grids are not.

EMA-FoM. EMA-FoM carries repair memory through an exponential moving average. "First-order" means each update uses only the previous state. With feature width m and fixed $0 < \rho < 1$,

$$u_t = \tanh(W_{\text{in}} z_t + b_{\text{in}}),$$
$$s_t = \rho s_{t-1} + (1-\rho)u_t, \tag{2}$$
$$c_t = W_{\text{out}}[u_t \,\|\, s_t] + b_{\text{out}}. \tag{3}$$

Here u_t supplies repair features, while s_t carries their summary. We take $W_{\text{in}} \in \mathbb{R}^{m \times 2d}$ and $W_{\text{out}} \in \mathbb{R}^{d \times 2m}$, with $s_0 = 0$ for each independent sequence or evaluation window. Selected configurations use $\rho = 0.995$. The propositions concern this fixed-weight map, not prediction quality or optimization convergence.

First, is this memory causal and bounded?

Proposition 1 (Causality and bounded repair state)**.** *If x_t and a_t depend only on the input prefix through t, then so does $\widetilde{a}_t$. For every $t \geq 1$, $s_t = (1-\rho)\sum_{i=1}^{t} \rho^{t-i} u_i$, $\qquad \|s_t\|_\infty \leq 1 - \rho^t$ (4).*

Proof. Unrolling Equation (2) gives the sum and prefix dependence. Since $\|u_i\|_\infty \leq 1$, the bound follows. Equation (3) preserves causality. □

We next ask what the carried state contributes. We probe this by resetting it before each update.

Proposition 2 (Reset collapse and geometric memory)**.** *Write $W_{\text{out}} = [W_u \quad W_s]$. At fixed repair features and weights, resetting the incoming state before each update gives*

$$c_t^{\text{reset}} = [W_u + (1-\rho)W_s]u_t + b_{\text{out}}, \qquad c_t - c_t^{\text{reset}} = \rho W_s s_{t-1}. \tag{5}$$

With unrestricted affine maps, reset-trained EMA-FoM represents exactly the width-m tanh MLP local function class. Without reset, positive-lag coefficients on repair features are $K_j = (1-\rho)\rho^j W_s$ for $j \geq 1$.

Proof. Substitute $s_t = \rho s_{t-1} + (1-\rho)u_t$ into the readout. Taking $W_s = 0$ makes the effective matrix arbitrary. Equation (4) gives K_j. □

Under reset, $s_t = 0.005u_t$, so the repair remains a local feature map rather than the wider matched MLP. Evaluation-only reset can change downstream features and does not causally decompose CE gains. Width adds directions but leaves the timescale unchanged. At $\rho = 0.995$, the half-life is $\log(1/2)/\log\rho \approx 138.3$ positions. Contextual features can already contain history, so this is not the model's context limit.

Deployment raises one final question. Does cached execution match a full-sequence pass?

Proposition 3 (Execution equivalence and update sensitivity)**.** *Full-sequence unrolling and cached EMA-FoM execution agree in exact arithmetic for identical repair features, initialization and decay. If* $\widehat{s}_t = \rho\widehat{s}_{t-1} + (1-\rho)u_t + e_t$*, with* $\widehat{s}_0 = s_0$ *and* $\|e_t\|_\infty \le \epsilon$*, then* $\|\widehat{s}_t - s_t\|_\infty \le \epsilon\frac{1-\rho^t}{1-\rho}$ (6).

Proof. Both executions satisfy the same recurrence. Subtraction gives $\widehat{s}_t - s_t = \sum_{i=1}^{t}\rho^{t-i}e_i$. Taking norms gives the bound. □

FoM computes full-sequence states with a rescaled cumulative sum and FP32 accumulation, while cached execution carries m_ℓ FP32 scalars per repaired layer and sequence. At $\rho = 0.995$, worst-case update-error amplification approaches 200 as the sequence grows. This bound does not certify historical implementations or timings. The current implementation regenerates FP32 power tables after device/dtype conversion, though historical timings were not rerun (Appendix C). FoM state is not an addressable KV record and does not recreate per-token KV records for the removed heads.

GRU-FoM. The controlled byte-level experiment also asks whether the fixed EMA update can be replaced by a learned recurrent update. Here each token is a byte, and GRU-FoM uses a single-layer unidirectional `torch.nn.GRU` (Cho et al., 2014) to carry the repair state. With repair features u_t, it sets $s_t = \mathrm{GRU}(u_t, s_{t-1})$ and $c_t = W_{\mathrm{out}}s_t + b_{\mathrm{out}}$. Unlike EMA-FoM, its state update is learned and its correction has no direct feature path. The EMA and GRU studies therefore do not form a matched cross-study ablation. The state starts at zero, and reset sets $s_{t-1} = 0$ before each update.

3 Experimental design

Controls and adaptation. We first test whether FoM recovers prediction quality after head compression while preserving the compressed KV layout. We then ask whether it can outperform an MLP repair with a comparable parameter count. We measure next-position cross-entropy (CE), live attention-state storage and throughput. Distilled methods minimize

$$\mathcal{L} = (1-\alpha)\,\mathrm{CE}(y, p_S) + \alpha\tau^2\,\mathrm{KL}\Big(p_T^{(\tau)}\Big\|p_S^{(\tau)}\Big), \quad \tau = 2, \tag{7}$$

using the same frozen MHA teacher for all distilled controls within each seed (Hinton et al., 2015). Repair comparisons keep the KV layout fixed, while comparisons with more KV heads expose the quality–cache tradeoff. Byte adaptation updates all student parameters. Pythia updates attention projections, repair modules and layer norms, while embeddings, the output head and existing feed-forward weights remain frozen.

Core protocols. We begin with a controlled four-layer UTF-8 WikiText-103 model (Merity et al., 2016), with width 384 and four query heads. Each seed trains MHA for 900 steps, then adapts MQA for 220 steps at batch 48 and length 256. GRU-FoM repairs every layer ($m = 192$, $\gamma = 0.2$), while the MLP has width 384. Seed-0 validation selects the controls. Seeds 1–5 use fresh teachers and paired adaptation streams.

We then move to Pythia. MQA development uses seed 0 and 4,194,304 tokens per method. Selection considers validation quality, matched-MLP and reset controls, and requires upper-layer EMA-FoM to retain at least 70% of MQA full-sequence throughput. GQA uses 16,777,216 tokens per method and seed at 160M, and 4,194,304 at 410M and 1B. All Pythia models adapt on 512-token sequences. Evaluation at 1024 and 2048 tokens remains within the pretrained context range, but extends beyond the adaptation length. Longer contexts therefore mix compression recovery with longer-conditioning robustness. GQA selection uses mean validation CE before held-out testing. The 160M and 410M studies each use five adaptation seeds from a shared checkpoint, while the 1B study uses one seed. Table 1 and Appendix A give the full protocols.

Evaluation and inference. The two model families use different CE units. Pythia reports nats per tokenizer token, while the byte model reports nats per byte. For Pythia, we score 256-token suffixes at contexts 512, 1024 and 2048. MQA development uses 64 windows per context, while GQA tests and MQA confirmation use 128, giving 32,768 targets. Targets are paired across methods and lengths.

Table 1: Selected EMA-FoM configurations for GQA comparisons. Layer indices are zero based. Width m is per repaired layer. All use $\rho = 0.995$ with unchanged query heads.

Model	Layers	Query heads	m	γ	KV layout and repair
160M	12	12	256	0.1	KV2 throughout, repair 6–11
410M	24	16	512	0.2	KV2 throughout, repair 6–23
1B	16	8	1024	0.2	KV4 0–11, KV2 + repair 12–15

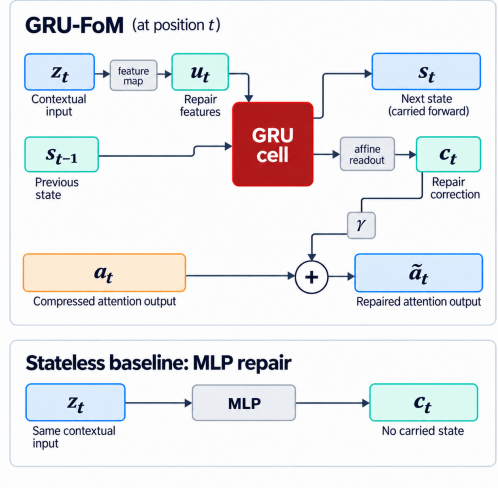


Figure 2: Schematic comparison of GRU-FoM and the stateless MLP repair. GRU-FoM carries repair state across positions, while both repairs use the same contextual interface.

Within each seed, mean CE averages the three contexts before we report the across-seed mean and sample SD. Context lengths are therefore not independent replications.

Because the main multi-seed comparisons use five seeds, exact statistical resolution is limited.

Lemma 4 (Resolution of the exact sign-flip test)**.** *For $n \geq 1$ paired differences, the two-sided exact sign-flip test using the absolute mean, all 2^n sign assignments and inclusive ties must satisfy $p \geq 2^{1-n}$. Equality holds when all differences are nonzero and have the same sign.*

Proof. The observed assignment and its negative give the same absolute mean. If every difference is nonzero and all have the same sign, no other assignment attains it. □

With five pairs, the smallest possible p is 0.0625, so the sign-exchangeable null has no 5% rejection region. We therefore emphasize effect sizes and descriptive intervals rather than adding post-hoc tests or seeds. Apart from the quantization reconstruction (Appendix D), no GPU experiments were rerun during manuscript preparation.

4 Controlled comparison of GRU-FoM and MLP repair

GRU-FoM improves on distilled MQA at both tested lengths while preserving its compressed KV cache (Table 2). It also beats the matched MLP at both lengths and in every seed, with the larger advantage at 512 bytes. The repairs share the contextual interface and teacher supervision, and their parameter counts differ by only 0.217%. This favors recurrent repair under the tested protocol, although gate structure, optimization and execution cost differ.

Does the model actually use the state it carries? Evaluation-only reset makes GRU-FoM substantially worse at both lengths and in every seed. This shows that the trained model relies on state carry, although it does not compare against a separately trained stateless GRU. At 512 bytes, the MLP is slightly worse than MQA-KD, so that control partly enlarges the GRU-FoM margin. The archive does not establish whether this reflects instability or overfitting. Continued MHA remains stronger at 256 bytes and is close to GRU-FoM at 512, so the result supports compression repair rather than superiority over full attention.

The intended memory saving is preserved. GRU-FoM keeps the KV cache at one quarter of MHA, while its repair state raises the total only slightly at 256 positions. The execution cost is less

Table 2: Controlled test CE in nats per byte, mean ± sample SD over five seeds. MLP repair and GRU-FoM parameter counts differ by 0.217%. Lower is better.

Method	CE at 256 bytes	CE at 512 bytes
MHA teacher	1.4246 ± 0.0035	1.4927 ± 0.0216
MHA continued	$\mathbf{1.3776} \pm 0.0021$	1.4560 ± 0.0219
MQA continued	1.4373 ± 0.0058	1.4905 ± 0.0134
MQA + KD	1.4195 ± 0.0043	1.4654 ± 0.0167
MQA + MLP + KD	1.4185 ± 0.0045	1.4674 ± 0.0179
MQA + GRU-FoM + KD	1.4162 ± 0.0041	$\mathbf{1.4548} \pm 0.0158$
GRU-FoM with evaluation-only reset	1.4456 ± 0.0095	1.4909 ± 0.0157

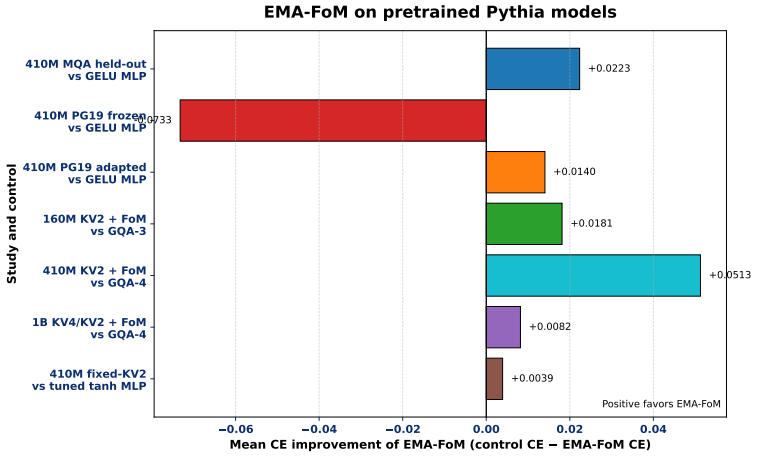


Figure 3: Mean CE comparisons across the pretrained-model studies. Bars show control CE minus EMA-FoM CE, so positive values favor FoM. The named control differs across comparisons.

favorable. An A100 pilot retains about 38% of continued MQA end-to-end full-sequence throughput (Appendix B).

5 EMA-FoM FOR PRETRAINED MODELS

We now ask whether FoM also helps larger pretrained language models. Figure 3 gives the main empirical picture at a glance.

5.1 MQA DEVELOPMENT AND HELD-OUT CONFIRMATION

With one KV head, the 160M, 410M and 1B Pythia models retain $1/12$, $1/16$ and $1/8$ of MHA KV storage. EMA-FoM repairs layers 9–11, 21–23 and 12–15 with widths 128, 256 and 512, adding 0.792%, 0.880% and 1.877% parameters over MQA. All use $\gamma = 0.1$ and $\rho = 0.995$.

Across all three scales, EMA-FoM improves on adapted MQA and the nearly matched MLP (Table 3). Evaluation-only reset weakens FoM but does not remove its 2048-token MLP advantage. The models therefore use carried state, while the residual advantage under reset cautions against attributing the full margin to recurrence. These comparisons were selected during development and are not independent confirmation.

Held-out 410M MQA confirmation. We next test the development result in a fixed held-out comparison. Seeds 101–105 and the primary contrast are fixed before training and scoring, with each method adapted afresh on 4,194,304 tokens per seed. EMA-FoM and the GELU MLP repair the top three layers at widths 256 and 341 with nearly matched parameter counts. Their learning rates are 0.0015 and 0.0005. MLP development searched $\{0.0002, 0.0005\}$ with GELU fixed, a narrower search than the later KV2 grid.

The held-out study supports both findings. EMA-FoM improves on distilled MQA at every context and seed while retaining the same KV layout. It also outperforms the selected MLP throughout, lowering mean CE by 0.02235 (Table 4). The effect is consistent across seeds, with exact sign-flip $p = 0.0625$ and a descriptive bootstrap interval that excludes zero. Scoring excludes legacy targets,

Table 3: Seed-0 MQA (2048 tokens). CE gains are control minus EMA-FoM. Reset penalty measures the evaluation-only reset CE increase. Throughput retention is EMA-FoM/MQA for paired BF16 full-sequence timing (batch 4, length 512), not cached decoding.

Model	EMA-FoM CE	MQA gain	MLP gain	Reset penalty	Throughput retention
160M	5.7188	0.1307	0.0138	0.0063	87.46%
410M	5.0569	0.1113	0.0153	0.0065	95.58%
1B	4.0861	0.0743	0.0187	0.0165	93.76%

Table 4: Held-out 410M MQA CE on fresh WikiText suffixes with distillation (mean $\pm$ sample SD, seeds 101–105). Mean CE averages three contexts per seed.

Method	512 tokens	1024 tokens	2048 tokens	Mean CE
MQA + KD	4.7034 ± 0.0040	4.7804 ± 0.0035	5.1413 ± 0.0063	4.8750 ± 0.0041
MQA + GELU MLP	4.6130 ± 0.0034	4.6876 ± 0.0049	5.0399 ± 0.0058	4.7801 ± 0.0040
MQA + EMA-FoM	$\mathbf{4.5911} \pm 0.0031$	$\mathbf{4.6640} \pm 0.0035$	$\mathbf{5.0183} \pm 0.0044$	$\mathbf{4.7578} \pm 0.0032$

though conditioning contexts may overlap. The earlier 0.024230 gain therefore shows continuity rather than independent confirmation. These checkpoints lack new reset tests and deployment timings.

5.2 PG19 TRANSFER AND ADAPTATION

We next ask whether the 410M result transfers. Frozen WikiText-103 checkpoints are evaluated on PG19, then new EMA-FoM and selected GELU MLP models are adapted there without retuning (Rae et al., 2019). Both stages retain MQA and sparse repair placement, with stage two using 4,194,304 tokens per method and seed. We fix 32 training and 32 disjoint test books, giving 128 paired windows per context.

Frozen transfer reverses the WikiText result. EMA-FoM loses to the MLP at every context and seed and also trails frozen MQA-KD on average. After PG19 adaptation, the comparison reverses again, with EMA-FoM improving mean CE by 0.01404. Both primary MLP contrasts reach the five-seed sign-flip floor of $p = 0.0625$. FoM's advantage is therefore not robust to frozen transfer, but can reappear after in-domain adaptation.

Stage two lacks adapted plain-MQA and reset controls, so absolute adaptation gains cannot be assigned specifically to FoM and post-adaptation state carry is not separately established. Testing reaches 2048 tokens rather than whole books, and possible PG19 overlap with Pythia pretraining is not ruled out.

5.3 GQA-FACING FOLLOW-UPS AND FIXED-KV2 CONTROLS

The GQA comparisons ask whether FoM can trade KV heads for compact state. Across scales, the selected configurations retain two-thirds, one-half and seven-eighths of their GQA references' KV storage and slightly improve mean CE, but the context pattern is not uniform (Table 6). At 160M, FoM wins at 512 and 1024 tokens but loses at 2048. At 410M and 1B, the pattern reverses.

At 410M, the 2048-token gain offsets small losses at shorter contexts and improves mean CE across five seeds. Evaluation-only reset substantially worsens mean CE, showing reliance on carried state. Uniform KV2 fails 1B validation, so the pilot keeps KV4 in early layers and repairs KV2 only in the final four. Its single-seed advantage is again confined to 2048 tokens. These results show a quality–cache tradeoff rather than a context-independent advantage.

Retrospective 410M repair comparison. The GQA comparison does not tell us whether recurrence beats stateless capacity at fixed KV layout. We therefore add five-seed KV2 controls with no repair, a tuned MLP, and EMA-FoM trained with state reset. Historical EMA-FoM and GQA-4 scores are reused under the same protocol. Validation selects tanh at learning rate 0.0015 from six activation/rate candidates. The width-683 MLP closely matches the interface, repaired layers and parameter count. Both repairs substantially improve on unrepaired KV2, supporting the repair proof of concept, while the tuned MLP recovers most of EMA-FoM's gain. It wins at 512 and 1024 tokens, while EMA-FoM wins at 2048 in every seed. Their mean difference is small, with three FoM wins and two losses, a descriptive interval crossing zero and exact $p = 0.4375$. The data support neither superiority nor equivalence in mean CE.

Table 5: PG19 subset CE on the same 32 test books (mean ± sample SD across seeds), with MQA and distillation throughout.

Method	512 tokens	1024 tokens	2048 tokens	Mean CE
Frozen WikiText-adapted checkpoints, seeds 101–105				
MQA + KD	6.2608 ± 0.0149	6.3038 ± 0.0155	6.4481 ± 0.0160	6.3375 ± 0.0154
MQA + GELU MLP	$\mathbf{6.2140} \pm 0.0184$	$\mathbf{6.2470} \pm 0.0174$	$\mathbf{6.3731} \pm 0.0167$	$\mathbf{6.2781} \pm 0.0175$
MQA + EMA-FoM	6.2815 ± 0.0131	6.3236 ± 0.0153	6.4488 ± 0.0149	6.3513 ± 0.0144
PG19-adapted models on PG19, seeds 201–205				
MQA + GELU MLP	4.5862 ± 0.0025	4.6587 ± 0.0053	5.2319 ± 0.0071	4.8256 ± 0.0041
MQA + EMA-FoM	$\mathbf{4.5733} \pm 0.0031$	$\mathbf{4.6421} \pm 0.0046$	$\mathbf{5.2192} \pm 0.0074$	$\mathbf{4.8116} \pm 0.0037$

Table 6: GQA test CE, mean ± sample SD across five seeds (one for the 1B pilot). Mean CE averages contexts within seeds.

Method	512 tokens	1024 tokens	2048 tokens	Mean CE
Pythia-160M, five adaptation seeds				
MQA	4.7824 ± 0.0022	4.8840 ± 0.0038	5.3959 ± 0.0063	5.0208 ± 0.0026
GQA-3	4.3063 ± 0.0098	4.4154 ± 0.0090	$\mathbf{5.0849} \pm 0.0089$	4.6022 ± 0.0072
KV2 + EMA-FoM	$\mathbf{4.2796} \pm 0.0074$	$\mathbf{4.3474} \pm 0.0121$	5.1252 ± 0.0058	$\mathbf{4.5841} \pm 0.0073$
Pythia-410M, five adaptation seeds				
MQA	4.6908 ± 0.0034	4.7714 ± 0.0049	5.1504 ± 0.0035	4.8709 ± 0.0039
GQA-4	$\mathbf{3.8852} \pm 0.0028$	$\mathbf{3.9830} \pm 0.0028$	4.4817 ± 0.0020	4.1166 ± 0.0024
KV2 + EMA-FoM	3.9405 ± 0.0072	3.9974 ± 0.0072	$\mathbf{4.2582} \pm 0.0061$	$\mathbf{4.0654} \pm 0.0068$
KV2, no repair	4.3676 ± 0.0030	4.4453 ± 0.0029	4.8177 ± 0.0022	4.5435 ± 0.0025
KV2 + tanh MLP	3.9294 ± 0.0061	3.9852 ± 0.0068	4.2932 ± 0.0099	4.0693 ± 0.0067
KV2 + reset-trained EMA-FoM	3.9382 ± 0.0060	3.9962 ± 0.0063	4.3065 ± 0.0044	4.0803 ± 0.0054
Pythia-1B, one adaptation seed				
MQA	3.7230	3.7921	4.0725	3.8625
GQA-4	**2.6989**	**2.7200**	3.0073	2.8088
KV4/KV2 + EMA-FoM	2.7030	2.7238	**2.9749**	**2.8006**

Stateful EMA-FoM also beats the reset-trained control on mean CE, with the clearest advantage at 2048 tokens. This pattern holds across all five seeds and supports training with state carry beyond evaluation-only distribution shift. It does not make the wider MLP comparison causal. The MQA and KV2 studies also differ in head count, repair capacity and MLP activation, so they do not isolate how compression level changes FoM's relative advantage.

Finally, a five-seed full-sequence INT8 KV quantize–dequantize simulation preserves 99.6% of EMA-FoM's BF16 advantage over the matched MLP at 2048, with five wins. The MLP remains better at 512 and 1024 tokens (Appendix D).

6 Live attention-state storage and execution cost

We now weigh FoM's prediction gains against storage and execution cost. Because FoM changes both the KV layout and the per-sequence repair state, we first need a model-wide accounting of the live attention state. Let B, T, L denote batch size, cached positions and layers, with repaired layers $\mathcal{L}_{\text{rep}}$. KV and repair-state scalars use b_{KV} and b_s bytes ($b_s = 4$ for FP32).

Proposition 5 (Live-state and parameter accounting)**.** *Dense KV caches and one EMA-FoM state per repaired layer and sequence require* $M_{\text{attn}}^{\text{total}} = 2BTb_{\text{KV}} \sum_{\ell=1}^{L} g_\ell d_{h,\ell} + Bb_s \sum_{\ell \in \mathcal{L}_{\text{rep}}} m_\ell$ *(8). At width* d*, EMA-FoM uses* $4dm + m + d$ *affine-map parameters, versus* $3dh + h + d$ *for a width-*h *MLP. KV storage grows linearly with* T*, while repair-state storage stays fixed.*

Proof. Sum storage across layers. EMA-FoM's maps have $2dm + m$ and $2dm + d$ parameters, versus $2dh + h$ and $dh + d$ for the MLP. □

The proposition separates the two live-state costs. The KV cache grows with context length, while the repair state does not. Table 7 applies this accounting to the selected configurations.

Live state is only part of the memory story because FoM also adds weights. The added weights are paid once, while live-state savings accumulate with each sequence in the batch. We therefore ask when the state savings outweigh the fixed parameter cost.

Table 7: Calculated live attention-state storage versus each reference (batch 1, BF16 KV, FP32 state), excluding weights, temporaries and unused capacity. These are not measured peaks.

Configuration	Reference	Context (tokens)	KV cache	KV cache + repair state
160M MQA + EMA-FoM	MHA	512	8.3333%	8.3415%
410M MQA + EMA-FoM	MHA	512	6.2500%	6.2561%
1B MQA + EMA-FoM	MHA	512	12.5000%	12.5122%
160M KV2 + EMA-FoM	GQA-3	1536	66.6667%	66.7101%
410M KV2 + EMA-FoM	GQA-4	1536	50.0000%	50.0977%
1B KV4/KV2 + EMA-FoM	GQA-4	2048	87.5000%	87.5122%†

† Four width-1024 FP32 repair states require 16 KiB. The historical report gives 64 KiB (87.5488%), with the extra allocation unverified (Appendix B).

Table 8: Single-seed 410M on A100-80GB (1536 prompt tokens, 128 outputs), using the largest tested batches below 8 GiB measured peak allocation. Throughput is aggregate output tokens/s.

Method	Batch	Peak GiB	Cached decoding throughput	End-to-end generation throughput
KV2 + EMA-FoM	**140**	7.945	6,230	4,610
KV2 + tanh MLP	**140**	7.937	**7,734**	**5,252**
GQA-4	103	7.972	6,897	5,230

Corollary 6 (Weight–state break-even)**.** *Let Δ_W denote FoM-minus-reference weight bytes and $\Delta_S(T)$ reference-minus-FoM live attention-state bytes per sequence. With weights counted once, FoM's weight-plus-state storage is smaller exactly when $B\Delta_S(T) > \Delta_W$* (9)*. For $\Delta_W \geq 0$ and $\Delta_S(T) > 0$, the minimum integer batch is $\lfloor \Delta_W/\Delta_S(T) \rfloor + 1$.*

Proof. Less storage means $\Delta_W - B\Delta_S(T) < 0$. □

For 410M KV2 + EMA-FoM relative to GQA-4, the extra BF16 weights cost about 60.0 MiB, while each sequence saves about 18.0 MiB of live attention state at 1536 tokens. The savings therefore outweigh the added weights from batch size $B \geq 4$ onward (Appendix B). This is an analytical break-even point, not a measured peak-memory guarantee.

Memory savings do not automatically imply faster execution. The MQA development timings measure full-sequence forward passes rather than a common deployment benchmark, and they use different A100 variants across model scales. Separate 160M GQA measurements show that FoM retains most, but not all, of GQA-3 throughput in both full-sequence and cached-decoding tests. The 1B KV4/KV2 configuration has no deployment timings.

Single-seed 410M deployment reconstruction. We next ask what this tradeoff looks like in an autoregressive deployment setting. A seed-3 reconstruction freshly adapts EMA-FoM and GQA-4 on 4,194,304 tokens each and reuses the archived seed-3 tanh MLP checkpoint. It adds neither an independent seed nor five-seed timing evidence. Its quality pattern matches the earlier KV2 result. FoM improves over the MLP at 2048 tokens, loses at the two shorter contexts and is nearly tied on mean CE.

The timing results separate memory capacity from execution speed. BF16 measurements use 512/1536-token prompts and 128 generated tokens, with five equal-batch and three near-budget repeats. At batch 1 and prompt length 1536, FoM raises prefill latency by 20.1% over the MLP and 44.4% over GQA-4. Cached decoding is also slower, and the same ordering persists at batch 16.

At the allocation frontier, head sharing lets both KV2 repairs fit about 36% more sequences than GQA-4 (Table 8). FoM itself adds no batch capacity over the MLP and is slower than both controls in cached decoding. Its end-to-end throughput also trails GQA-4. The larger batch therefore comes from KV-head reduction rather than recurrence. Each next tested batch exceeds the 8 GiB criterion, and this retrospective comparison does not predict performance on a physical 8 GiB GPU.

Archived checks support the cached implementation, though they were not independently rerun. Full and cached execution agree closely in FP32, and BF16 discrepancies remain small. The checks cover nonzero repairs, future perturbations and state reset. Chunked-prefill correctness uses math SDPA, while timings use FlashAttention. Missing deployment source leaves repair fusion and dispatch overhead unverified. Without matched quality measurements at the timed 1536+128 horizon, these timings establish neither a quality advantage nor Pareto dominance.

7 Related work

FoM sits between methods that compress attention and methods that supplement it with compact memory. MQA and GQA reduce KV storage by sharing heads (Shazeer, 2019; Ainslie et al., 2023), while Palu uses low-rank KV projections (Chang et al., 2024). DeepSeek-V2 introduces latent attention (MLA), and MHA2MLA converts pretrained attention to latent KV representations (DeepSeek-AI, 2024; Ji et al., 2025). Other methods compress the cache itself. Dynamic Memory Compression learns KV append/merge decisions (Nawrot et al., 2024), while KIVI quantizes KV (Liu et al., 2024). Appendix E gives broader context on recurrent attention and related architectures.

The closest recurrent precedents repair other forms of information loss. LESS adds fixed-size low-rank recurrent state to an eviction-based cache (Dong et al., 2024a), while KVLinC uses recurrent corrections for key quantization errors (Saxena and Roy, 2025). Infini-attention combines local softmax attention with compressive memory (Munkhdalai et al., 2024), and Attamba compresses token chunks into KV representations with state-space models (Akhauri et al., 2024). Trellis replaces KV caching with learned recurrence (Karami et al., 2025), while Key-Value Means uses fixed or growing block-recurrent state (Goldstein et al., 2026).

LoLCATs combines attention transfer and low-rank adaptation with linear/local softmax attention (Zhang et al., 2025). FoM instead preserves softmax attention and context positions after head reduction, repairing from contextual features. Our MLP comparisons keep the KV layout fixed so that recurrent and stateless repair can be compared in quality, storage and execution cost. Recurrence and EMA are established, and these experiments do not show superiority over these alternatives or models pretrained with compressed attention.

8 Discussion and limitations

The experiments support FoM as a proof of concept for recovering prediction quality while retaining compressed KV caches. The gains over MLP repair are less dramatic, but they recur in the controlled byte study, held-out MQA confirmation and PG19 adaptation. At fixed KV2, FoM shows its clearest advantage at the longest tested context, while mean performance remains close. Together with the reset controls, these results suggest that carried repair state can make a useful contribution beyond stateless repair, although activation and optimization differences still limit how precisely that contribution can be isolated. Cross-entropy on prose also does not directly test recall, copying or retrieval, and alternative compression methods are not evaluated. Frozen transfer to PG19 reverses the WikiText gains for reasons not yet established. Proposition 1 bounds the state rather than prediction error. For pretrained EMA-FoM, learned decay and input-dependent gating or reset remain unexplored, as does the interaction between decay, adaptation length and context.

The evidence is also uneven across scales. The 160M and 1B MQA results are developmental, while the 1B GQA study uses a single seed. The 160M and 1B GQA studies lack matched MLP repairs, and the 410M KV2 comparison is retrospective. Repair width, placement, KV-head count and training budget also vary together, limiting causal attribution. More broadly, the pretrained-model studies cover one model family with frozen feed-forward weights, contexts up to 2048 tokens and 32 PG19 test books. Deployment uses one seed and does not jointly measure quality and speed at 2048 tokens. Held-out MQA and PG19 also lack deployment timings. EMA-FoM adds prefill and cached-decoding overhead without improving batch capacity over the matched MLP. Missing weights and exports, together with unresolved 1B allocation details, further limit reproduction (Appendices B and C).

9 Conclusion

FoM provides an encouraging proof of concept for recovering prediction quality after head compression without restoring the removed KV heads. EMA-FoM achieves this with a bounded repair state whose storage does not grow with context length. Under reset, it reduces to the same local function class as a width-m tanh MLP. In the controlled byte study, GRU-FoM beats a nearly matched MLP at both tested lengths across five seeds. In larger pretrained models, held-out 410M MQA and PG19-adapted EMA-FoM improve mean CE over their selected MLP repairs by 0.02235 and 0.01404. These repeated gains over comparably sized MLP repairs suggest that carried repair state can contribute beyond stateless repair. The advantage is not universal. A strong MLP nearly matches FoM at fixed KV2, and frozen PG19 transfer reverses the result. Both KV2 repairs fit larger batches than GQA-4, but EMA-FoM adds execution overhead without increasing batch capacity over the MLP.

REPRODUCIBILITY STATEMENT

Sections 2–6 specify the method and report all study families. Appendix A provides the experimental protocols, Appendix B details the storage accounting, and Appendix C records the evidence supporting each result and what remains missing. Available seed and window records were used to verify means and paired differences. Appendix D reports a new Colab A100 run from the recovered implementation. No other GPU experiments were repeated during manuscript preparation.

AI USE STATEMENT

GPT-5.5 Pro and 5.6 Sol supported hypothesis refinement, experimental code development, literature lookup, drafting and figure preparation. The authors remain responsible for the text, references, code and scientific claims.

ETHICS STATEMENT

This work uses public model checkpoints, WikiText-103 and a fixed PG19 subset. It involves no new data collection from human participants. Prediction quality and resource measurements do not establish that safety or fairness is preserved under compression. Separate evaluation is needed before drawing such conclusions for deployment.

REFERENCES

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*, pages 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Yash Akhauri, Safeen Huda, and Mohamed S. Abdelfattah. Attamba: Attending to multi-token states. *arXiv preprint arXiv:2411.17685*, 2024. URL https://arxiv.org/abs/2411.17685.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*, 2023. URL https://arxiv.org/abs/2304.01373.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2207.06881.

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: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024. URL https://arxiv.org/abs/2407.21118.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014. URL https://arxiv.org/abs/1406.1078.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://arxiv.org/abs/1901.02860.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024. URL https://arxiv.org/abs/2405.04434.

Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with LESS: Synthesizing recurrence with KV cache compression for efficient LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11437–11452. PMLR, 2024a. URL https://proceedings.mlr.press/v235/dong24f.html.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models. arXiv preprint arXiv:2411.13676, 2024b. URL https://arxiv.org/abs/2411.13676.

Daniel Goldstein, Navneel Singhal, and Eugene Cheah. Key-value means: Transformers with expandable block-recurrent compressed memory. *arXiv preprint arXiv:2605.09877*, 2026. URL https://arxiv.org/abs/2605.09877.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL https://arxiv.org/abs/2312.00752.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

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. *arXiv preprint arXiv:2502.14837*, 2025. URL https://arxiv.org/abs/2502.14837.

Mahdi Karami, Ali Behrouz, Praneeth Kacham, and Vahab Mirrokni. Trellis: Learning to compress key-value memory in attention models. In *Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2512.23852.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/2006.16236.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. URL https://arxiv.org/abs/2510.26692.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.02750.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: Moving average equipped gated attention. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.10655.

Xuezhe Ma, Xiaomeng Yang, Wenhan Xiong, Beidi Chen, Lili Yu, Hao Zhang, Jonathan May, Luke Zettlemoyer, Omer Levy, and Chunting Zhou. Megalodon: Efficient LLM pretraining and inference with unlimited context length. *arXiv preprint arXiv:2404.08801*, 2024. URL https://arxiv.org/abs/2404.08801.

Xuezhe Ma, Shicheng Wen, Linghao Jin, Bilge Acun, Ruihang Lai, Bohan Hou, Will Lin, Hao Zhang, Songlin Yang, Ryan Lee, Mengxi Wu, Jonathan May, Luke Zettlemoyer, and Carole-Jean Wu. Gecko: An efficient neural architecture inherently processing sequences with arbitrary lengths. *arXiv preprint arXiv:2601.06463*, 2026. URL https://arxiv.org/abs/2601.06463.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. URL https://arxiv.org/abs/1609.07843.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with Infini-attention. arXiv preprint arXiv:2404.07143, 2024. URL https://arxiv.org/abs/2404.07143.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 37396–37412. PMLR, 2024. URL https://arxiv.org/abs/2403.09636.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019. URL https://arxiv.org/abs/1911.05507.

Utkarsh Saxena and Kaushik Roy. KVLinC: KV cache quantization with Hadamard rotation and linear correction. *arXiv preprint arXiv:2510.05373*, 2025. doi: 10.48550/arXiv.2510.05373. URL https://arxiv.org/abs/2510.05373.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019. URL https://arxiv.org/abs/1911.02150.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1706.03762.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. LoLCATs: On low-rank linearizing of large language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10254.

A ADDITIONAL PROTOCOL DETAILS

Training settings and their sources. To compare the studies, we first ask which training settings the records establish. Tables 9 and 10 draw on supplied implementations and locked run records, leaving missing selections unresolved. The Pythia schedule is explicit in training code and the locked 410M MQA configuration. Report-only runs, including the original 160M GQA run, lack fully recoverable settings. Listed library defaults come from constructors rather than optimizer-state exports. These records do not certify a new execution.

Table 9: Documented optimizer and schedule settings. Pythia values describe the supplied shared runtime and locked 410M runs, without establishing settings for every report-only run.

Setting	Controlled byte model	Pythia adaptation runtime
Optimizer	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.999)$, library defaults	$(0.9, 0.95)$, explicit
ϵ	10^{-8}, library default	10^{-8}, library default
Weight decay	0.01	0.01
Gradient norm clip	1.0	1.0
Microbatch / length	48 / 256	4 / 512
Gradient accumulation	1	4
Positions per update	12,288	8,192
Training updates	900 teacher, 220 student	512 at 4,194,304 tokens
Learning-rate schedule	Constant, no warmup	Linear warmup over 5% of updates, then cosine to 10% of initial rate
Warmup at 512 updates	Not applicable	26 updates after rounding
Base learning rate	Teacher 4×10^{-4}, student 2×10^{-4}	3×10^{-5}
FoM learning rate	10^{-3}	1.5×10^{-3}
FoM distillation	$\alpha = 0.5$, $\tau = 2$	$\alpha = 0.3$, $\tau = 2$
Quality precision	FP32 in the supplied byte driver	BF16 autocast with FP32 CE/KL in supplied runtime

Pythia uses the same warmup and decay schedule across parameter groups. For update k of N, indexed from zero, set $w = \max(1, \mathrm{round}(0.05N))$. The learning-rate multiplier is $(k+1)/w$ for $k < w$ and

$$0.1 + 0.45 \left[1 + \cos\left(\pi \frac{k - w}{\max(1, N - w - 1)} \right) \right]$$

thereafter. The byte driver keeps its learning rate constant. Pythia freezes existing feed-forward weights, embeddings and the output head, while the byte study adapts all student parameters. Both keep the teacher frozen during student adaptation.

Table 10: Recorded control searches and selections. Learning rates refer to repair parameters. CE summaries do not substitute for missing historical selections.

Study / control	Recorded choices	Retained selection
Byte MQA-KD	$\alpha \in \{0.1, 0.25, 0.5\}$	Original selected α not bundled
Byte GELU MLP	Rates $\{0.0003, 0.0005, 0.001\}$ crossed with $\alpha \in \{0.1, 0.25, 0.5\}$	Original selected rate and α not bundled
410M MQA GELU MLP	Rates $\{0.0002, 0.0005\}$ in official development run	0.0005, retained in confirmation and PG19
410M KV2 MLP	GELU/tanh crossed with rates $\{0.0005, 0.0015, 0.003\}$	tanh and 0.0015
Held-out MQA / PG19	Original selected configurations, no new grid	EMA-FoM 0.0015, GELU MLP 0.0005

The byte control search minimizes mean validation CE at 256 and 512 bytes. Their grid and code are available, but the original `selected_controls.json` is missing, so final settings remain unknown. The 410M MQA records retain both tested rates and the selected 0.0005. Compared with the six-choice KV2 grid, this narrower search establishes neither optimal tuning nor an optimal activation.

Byte data and uncertainty. The byte model uses 256 vocabulary values, with each WikiText split capped at 8,000,000 characters before UTF-8 encoding. The teacher and student budgets are 11,059,200 and 2,703,360 processed byte positions, not distinct corpus bytes. Evaluation uses 50 fixed batches, with batch 48 at 256 bytes and batch 24 at 512.

The archived GRU-FoM-minus-MLP intervals favor GRU-FoM at both lengths. They are $[-0.002900, -0.001884]$ and $[-0.014625, -0.010463]$, respectively. Against continued MHA at 512 bytes, the interval $[-0.024982, 0.022922]$ crosses zero. Missing per-seed observations prevent independent regeneration of these intervals or explanation of the MLP's weaker 512-byte result.

Pythia budgets and evaluation. Reported budgets cover adaptation, not pilots or search. Larger MQA models are adapted afresh from the original checkpoint after pilots informed by validation. Original GQA budgets are 16,777,216 tokens per method and seed at 160M and 4,194,304 at 410M and 1B. The latter reports record $\alpha = 0.3$ with backbone and repair rates 3×10^{-5} and 1.5×10^{-3}. The 160M report specifies matched distillation settings and budgets, but some optimizer details remain unresolved. Held-out 410M run records establish a 4,194,304-token budget despite an inherited `production_tokens` field. All Pythia adaptation windows have length 512. Evaluation at 1024 and 2048 therefore tests longer conditioning than adaptation, but stays within the pretrained checkpoint's 2048-token range.

PG19 and statistical scope. PG19 evaluation pairs the same target suffixes across stages, methods and lengths. Evaluation windows stay within books, although training can cross concatenated book boundaries. Seed intervals condition on the sampled books and windows, so they exclude variability from corpus selection and pretraining.

The sign-flip test assumes sign exchangeability under its null. With five pairs, its smallest two-sided value is $p = 0.0625$, above 0.05. No outcome can therefore yield rejection at 5% under this test. We retain the prespecified five-seed panel and descriptive effects without adding seeds or changing tests after seeing the results.

B Storage accounting and auxiliary throughput

Architectural storage and historical allocation. Equation (8) connects the architecture to the storage ratios in Table 7 by summing across layers. For example, 160M MQA has 12 layers of width 768 and 12 query heads. At $B = 1, T = 512$, its KV cache uses 1,572,864 bytes and three width-128 FP32 repair states add 1,536 bytes. Together they use 8.3415% of the 18,874,368 MHA KV bytes.

The 1B mixed layout exposes a difference between architectural storage and the historical report. At $B = 1, T = 2048$, 12 KV4 layers and four KV2 layers of head width 256 use 112 MiB of KV storage. Four width-1024 FP32 repair states add 16,384 bytes, or 16 KiB. The total is 112.015625 MiB, or 87.5122% of the 128 MiB GQA-4 reference.

The historical report gives 64 KiB and 87.5488%. Sixty-four KiB would equal 16 width-1024 FP32 vectors, but this does not establish an actual allocation. Without complete source and an allocation trace, excess buffering and reporting error remain unresolved alternatives. The dagger distinguishes the architectural calculation from the historical allocation report.

410M weight–state break-even. When do state savings offset the extra weights? Let $P_F - P_G = 31{,}478{,}784$ be the parameter difference between KV2 + EMA-FoM and GQA-4. At two bytes per weight, this adds $62{,}957{,}568$ bytes, or 60.0410 MiB. Removing two KV heads per layer from 24 layers with $d_h = 64$ saves $12{,}288BT$ KV bytes, while 18 width-512 FP32 repair states require $36{,}864B$ bytes. EMA-FoM therefore uses less storage for weights and live attention state when

$$B\,(12{,}288T - 36{,}864) > 62{,}957{,}568.$$

At $T = 1536$, each sequence saves 18,837,504 bytes, or 17.9648 MiB. The threshold is therefore $B > 3.342$, or integer $B \geq 4$. At batch 1, EMA-FoM has a larger footprint despite its smaller live attention state. The calculation excludes workspace, allocator reservation and extra cache capacity, so it is not a measured peak-memory threshold.

Derived mean per-stream token interval. Table 11 translates reported aggregate throughput r into a derived mean per-stream token interval of $1000B/r$ ms. This calculation adds neither a timing run nor a latency distribution. Reported prefill measures model execution and excludes queuing and other service delays before the first token.

Table 11: Derived mean per-stream token interval in ms/token for a 1536-token prompt. Fixed batches use reported equal-batch rates, while allocation-frontier batches vary by method. Values are reciprocal-rate calculations without new latency measurements.

Method	Batch 1	Batch 16	Allocation frontier
KV2 + EMA-FoM	22.57	22.89	22.47 ($B = 140$)
KV2 + tanh MLP	17.89	19.00	18.10 ($B = 140$)
GQA-4	15.02	15.56	14.93 ($B = 103$)

At batch 1, EMA-FoM has 33.5% lower cached decoding throughput and 44.4% higher prefill latency than GQA-4. The throughput deficit is 32.0% at batch 16. At the reported allocation frontier, mean per-stream throughput falls from $6897/103 \approx 66.96$ to $6230/140 = 44.50$ tokens/s. These measurements combine recurrence and implementation costs. The generic repair source uses separate PyTorch projections and elementwise updates. Missing source prevents checking whether the timed reconstruction used custom fusion. Fused updates and readout remain a possible engineering direction, with no benefit established here.

What the byte-model timers measure. The A100 BF16 pilot archive reports end-to-end and model-only full-sequence throughput in `systems_seed0_bf16.csv`. Table 12 compares GRU-FoM with continued MQA from that same file. End-to-end timing includes CPU batch construction, host-to-device transfer and the forward pass. The GRU-FoM storage comparison also depends on precision. Live attention-state element ratios equal byte ratios only when KV-cache and repair-state dtypes match.

Table 12: Archived A100 BF16 byte-model pilot throughput in processed byte positions/s. Retention divides GRU-FoM by continued MQA within the same timer boundary. The values describe pilot runs rather than final production checkpoints.

Timer boundary	GRU-FoM	Continued MQA	Throughput retention
End-to-end full-sequence	878,798.7	2,325,539	37.79%
Model-only full-sequence	989,938	3,448,876	28.70%

The model-only timer excludes batch construction and transfers, but retains host dispatch and synchronization overhead. The supplied `fullseq_benchmark` measures synchronized wall-clock throughput on preloaded device batches after three warmup forwards. The 28.70% retention is the ratio of archived model-only rates, not a subtraction of end-to-end full-sequence throughputs.

Neither timer isolates kernel time or measures cached decoding throughput. The supplied routine supports a CUDA-event diagnostic, `fullseq_gpu_toks_s`, but the pilot CSV contains neither that field nor individual repeat timings. The report describes the GRU-FoM run as a trained joint pilot with the same architecture, not the final five-seed quality checkpoints. These records provide no new benchmark or final-checkpoint remeasurement.

A separate tanh RNN provides summary-level results, not a replacement for the GRU-FoM comparison. It repairs the top two layers with $m = 192$ and 591,744 repair parameters. Throughput is 1,788,000 positions/s against its own MQA reference of 2,332,705, retaining 76.65%. CE is 1.418010 at 256 bytes and 1.460146 at 512, with four and five wins against MQA.

C EVIDENCE AND REPRODUCTION GAPS

Available records determine which results can be checked. Original GRU-FoM records retain aggregates and paired statistics, but not per-seed observations or the final control-selection export. MQA development provides a 160M summary JSON and per-window JSONs for 410M/1B. Raw GQA seed scores support 160M/410M statistics, while 1B has only a single-seed summary. The fixed-KV2 extension includes all five methods' per-seed scores and reuses historical reference rows. The summaries permit numerical consistency checks. The reported 410M GQA reset average is $(0.108365 + 0.168552 + 0.300054)/3 \approx 0.192324$. For 1B, the shorter-context losses to GQA-4 are 0.004040 and 0.003806, with a 0.032396 gain at 2048.

Held-out MQA retains the frozen plan, offsets, logs and per-window scores. Its manifest identifies 15 checkpoint files by hash and size, but their weights are missing. PG19 retains the plan, corpus lock and per-seed/window scores, with seed-bootstrap intervals reported as Monte Carlo outputs. Its runner, weights, corpus/token arrays and metadata needed to resolve its run lock are absent, so generic runtime code cannot reproduce the workflow. Neither a frozen plan nor a matching internal digest establishes an independently time-stamped preregistration.

Reproduction requires every changed tensor because adaptation updated attention and normalization parameters alongside repair weights. The base revision must be pinned and activation and reset settings restored. Deployment results come from the supplied README, but its separately described checkpoint and timing archive was not provided for verification. Historical final weights and the selected 1B GQA runtime remain incomplete. We do not claim to have recovered or newly released these missing exports.

EMA precision and the implementation fix. The fix addresses a numerical mismatch between single-token and full-sequence EMA updates. Earlier screening and benchmarking paths cast the model and its EMA power tables to BF16, rounding the first decay power from 0.995 to 0.99609375. Single-token carry then followed a different numerical recurrence from the separately rounded full-sequence powers, despite FP32 accumulation. After device or dtype conversion, the distributed runtime now regenerates both power tables in FP32 from the original scalar decay. Held-out MQA and quantization quality drivers already kept FP32 buffers under BF16 autocast and avoided this conversion, so the issue does not directly invalidate their full-sequence CE records. Missing PG19 and 410M deployment implementations prevent the same check. Historical timings were not rerun with the corrected runtime.

The bundled sources rebuild the manuscript, but missing artifacts still limit full experimental reproduction. Primary archives must be preserved separately. The CE gains remaining after evaluation-only reset, weight–state break-even and derived mean per-stream token intervals follow from existing records. The byte-model timing table likewise calculates retention from existing pilot measurements within each timer boundary. Apart from the quantization study in Appendix D, this revision includes no new training runs, language-model quality evaluations or GPU benchmarks. Future work includes targeted retrieval tests and broader MQA baseline tuning. Matched adaptation-length studies, PG19 controls and repeated deployment profiling also remain open.

D PYTHIA-410M KV QUANTIZATION SIMULATION

Protocol. A fresh paired reconstruction tests whether FoM's relative 2048-token advantage survives KV quantization. Pythia-410M uses KV2 in all 24 layers. Layers 6–23 receive either post-attention EMA-FoM ($m = 512$, $\rho = 0.995$, $\gamma = 0.2$) or a nearly parameter-matched width-683 tanh MLP. Each method adapted on 4,194,304 WikiText-103 tokens at seeds 1–5. Evaluation scored the final 256 tokens of 128 fixed held-out test windows at contexts 512, 1024 and 2048.

Full-sequence evaluation symmetrically quantized and reconstructed each K/V vector across its 64 channels. This was done separately per token and KV head using FP32 scales and BF16 reconstructed K/V. INT8 was confirmatory and INT4 exploratory. Evaluation retained FP32 parameter storage and EMA power tables under BF16 autocast without integer-quantizing model weights.

Prespecified criteria check quality and theoretical storage savings. INT8 must raise mean CE by at most 0.01 per method and retain at least 80% of the BF16 2048-token margin. EMA-FoM must win in at least four seeds at 2048 and have a negative mean CE difference from the MLP. The theoretical live-KV reduction must reach at least 45%. All criteria are met under the corrected FP32-scale accounting below. This is not a significance test.

Table 13: Paired EMA-FoM-minus-MLP CE differences over five seeds under KV quantization simulation. The mean weights the three context lengths equally. Negative values favor EMA-FoM, and INT4 is exploratory.

Cache	Mean CE difference	512	1024	2048	2048 wins
BF16	-0.002708	$+0.013092$	$+0.014219$	-0.035435	5/5
INT8	-0.002542	$+0.013253$	$+0.014405$	-0.035284	5/5
INT4	-0.003969	$+0.011271$	$+0.013866$	-0.037044	5/5

INT8 preserves 99.6% of the BF16 2048-token margin while barely changing mean CE. The increases are $+0.000255$ for EMA-FoM and $+0.000089$ for the MLP. EMA-FoM wins at 2048 in every seed, while the MLP remains better at 512 and 1024 in every seed. Averaged across lengths, INT8 favors EMA-FoM in four of five seeds. Exploratory INT4 also preserves the relative 2048-token advantage, but both methods lose more prediction quality. Mean CE rises relative to BF16 by $+0.049364$ for EMA-FoM and $+0.050626$ for the MLP. The gains remain context-dependent.

The storage saving is hypothetical, not a measured allocation. At batch 1 and a 1536-token prompt, packed INT8 K/V with one FP32 scale per vector requires 9.5625 MiB against 18.000 MiB for BF16 K/V, a 46.875% reduction. Matching the executed quantizer's scales supersedes the archived BF16-scale counts of 9.28125 MiB and 48.4375% reduction. The supplementary README, current assessment and accounting helper now use FP32-scale counts. The original protocol and final result JSON retain their historical values, with corrected storage fields in `ACCOUNTING_CORRECTION.json`.

Full-sequence quantize–dequantize establishes neither packed-cache execution nor cached-decoding throughput. BF16-stored scales and quantized model weights were not tested, and the study does not establish general recovery of prediction quality.

E Additional related-work context

Compact memory takes several forms. Transformer-XL, Compressive Transformer and recurrent memory tokens retain context (Dai et al., 2019; Rae et al., 2019; Bulatov et al., 2022). Linear attention permits recurrent evaluation (Katharopoulos et al., 2020), while Mamba learns selective state-space dynamics (Gu and Dao, 2023). Mega combines EMA and gated attention, with extensions in Megalodon and Gecko (Ma et al., 2023; 2024; 2026). Hymba mixes attention and state-space heads with KV sharing and partial sliding windows (Dong et al., 2024b). Kimi Linear interleaves recurrent Kimi Delta Attention and MLA (Kimi Team, 2025).

KVLinC applies recurrent correction to quantization errors. It combines Hadamard rotation of values with learned corrections for key quantization errors (Saxena and Roy, 2025). Query features read its recurrent states to adjust the attention numerator and normalization. State storage is independent of sequence length. FoM instead targets KV-head sharing, adding a correction from contextual features and repair state after attention output projection.