LOW-RANK KEY-VALUE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The key-value (KV) cache is a primary memory bottleneck in Transformers. We propose Low-Rank Key-Value (LRKV) attention, which reduces KV-cache memory by exploiting redundancy across attention heads. Each layer pairs a shared KV projection with low-rank, head-specific residuals, interpolating continuously between full sharing and per-head independence. Across pretrained models from 128M to 6.3B parameters, LRKV has the lowest observed held-out loss among standard MHA, MQA/GQA, and MLA at each method's operating point while using 48–67% of MHA's cache, and it remains below full-cache MHA with as little as 8.3% of the cache at 128M, 2.5B, and 6.3B. At 128M and 100B tokens, three-seed means give LRKV lower held-out loss than MLA at every matched cache budget (8.3%, 25%, and 50%). At 2.5B, LRKV reaches each baseline's final quality in 18–30% fewer optimizer steps. Trained at 8K context, 512M LRKV models attain the lowest long-context loss and lead every baseline on RULER at 32K, four times the training length. Its exact decode kernel evaluates the factored representation end to end without storing reconstructed per-head K/V. On a B200 at 2.5B, LRKV serves 8.5–8.9× MHA's resident tokens at every context and decodes at 103–111% of MHA's batch-1 throughput from 1.2B to 30B. After fixed-recipe midtraining, LRKV has the highest observed combined downstream accuracy.

1 INTRODUCTION

Transformers dominate large-scale sequence modeling across language, vision, and multimodal domains (Vaswani et al., 2017; OpenAI, 2023), but their inference cost grows rapidly with model size and context length. A major source of this cost is the key-value (KV) cache, which stores attention keys and values for every layer and scales linearly with sequence length and number of heads. For long-context inference, the cumulative KV footprint can become comparable to parameter memory (Shah et al., 2024; Child et al., 2019). For example, a 2.5B parameter model with 36 layers, 18 heads, and d_h=128 requires $2 \times 36 \times 18 \times 128 \times 8192 \times 2 \approx 2.72$ GB of KV cache per 8K-token request in bfloat16. At batch size 4, this reaches ≈10.9 GB, more than twice the model's 5 GB bfloat16 parameter memory.

Existing KV-efficient designs reduce this footprint by sharing or compressing keys and values. Multi-Query Attention (Shazeer, 2019, MQA) and Grouped-Query Attention (Ainslie et al., 2023, GQA) share KV projections across heads or head groups. GQA remains prominent in recent models such as Gemma 3 and Qwen3 and in the attention blocks of the hybrid Nemotron 3 Nano, Super, and Ultra architectures (Gemma Team, 2025; Yang et al., 2025; NVIDIA, 2025; 2026a;b). However, aggressive sharing constrains head-level diversity, even though attention heads encode complementary syntactic and semantic patterns (Michel et al., 2019). At the same time, empirical and spectral analyses show that head-specific KV projections are highly correlated and occupy overlapping subspaces, suggesting structured redundancy across heads (Yunis et al., 2024). This motivates our central question: *can KV memory be reduced by exploiting cross-head redundancy without collapsing useful head specialization?*

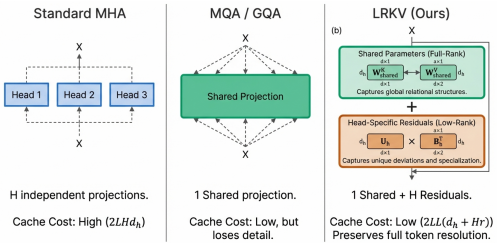


Figure 1: LRKV shares full-rank K/V projections across heads and adds head-specific low-rank residuals, reducing cache from $2LHd_h$ to $2L(d_h + Hr)$ while preserving head diversity.

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.

We propose **Low-Rank Key-Value Attention (LRKV)**, which decomposes each head's K/V projection into a shared full-rank component plus a compact head-specific low-rank residual. LRKV preserves per-head variation unlike MQA/GQA. It also preserves full token resolution and compresses across heads, unlike latent compression methods such as Multi-Latent Attention (DeepSeek-AI et al., 2024, MLA). Unlike LoRA (Hu et al., 2022) or post-hoc factorized QKV methods (Saxena et al., 2024; Chang et al., 2025), LRKV learns this structure during pretraining and directly reduces cached representations.

Contributions. (1) At 128M and 100B tokens, every three-seed mean LRKV point records lower held-out loss than the tested MHA, GQA, and MQA operating points, and lower loss than MLA at exact or near-exact 8.3%, 25%, and 50% cache matches (Table 1). (2) LRKV attains the lowest observed test loss across 128M–6.3B models at each method's standard operating point, leads cache-matched MLA at every tested budget from 8.3% to 66.7%, and with 8.3% of MHA's cache stays below full-cache MHA at 128M, 2.5B, and 6.3B. At 2.5B it reaches baseline quality in 18–30% fewer optimizer steps, a sample-efficiency result rather than a per-step compute claim. (3) LRKV has the strongest combined downstream score across the tested checkpoints, the lowest 8K-context loss and the highest RULER average at 32K among 512M models, and gauge-invariant analysis finds 93.5% normalized centered effective rank against 94.0% for MHA. (4) The factored decode kernel is exact, not an approximation: it is bit-exact against explicit evaluation of the factored form on hidden states, and on a B200 it converts the cache reduction into 8.6× MHA's serving capacity at a configuration whose loss is below full-cache MHA.[1] We next situate LRKV within related work.

2 Related Work

KV-sharing mechanisms. MQA (Shazeer, 2019) and GQA (Ainslie et al., 2023) reduce KV cache by sharing projections across all heads or within head groups, interpolating between maximal compression and full MHA. These methods improve memory efficiency but restrict the representational capacity of individual heads. LRKV retains the cache benefits of sharing while restoring head-specific variation through low-rank residuals.

Latent and sparse attention. Sparse, kernelized, and hardware-aware attention methods reduce sequence-level computation or memory traffic (Wang et al., 2020; Beltagy et al., 2020; Child et al., 2019; Shah et al., 2024). MLA (DeepSeek-AI et al., 2024) instead compresses each token's cache into a low-dimensional latent; TransMLA converts GQA checkpoints into this form (Meng et al., 2025), and grouped-tied and grouped-latent attention share keys or latents across head groups for decode efficiency (Zadouri et al., 2025). LRKV preserves full token resolution and reduces duplication across heads rather than compressing the hidden-feature dimension.

Low-rank and factorized projections. LoRA (Hu et al., 2022) uses low-rank updates for parameter-efficient fine-tuning, while recent QKV factorizations reduce parameters, computation, or post-hoc storage (Khalaf et al., 2026; Lv et al., 2024). Concurrent factorized KV methods such as MFA/MFA-KR (Hu et al., 2024) and TPA (Zhang et al., 2025) explore related structured compression. LRKV differs by learning a full-rank shared base with head-specific low-rank residuals during pretraining, enabling exact associative decoding and direct KV-cache reduction without post-hoc approximation.

3 Methodology

Transformers represent each token in a sequence through queries, keys, and values that interact via scaled dot-product attention. For an input matrix $\mathbf{X} \in \mathbb{R}^{L\times d}$, the h-th attention head computes $\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^Q$, $\mathbf{K}_h = \mathbf{X}\mathbf{W}_h^K$, and $\mathbf{V}_h = \mathbf{X}\mathbf{W}_h^V$, where $\mathbf{W}_h^{Q,K,V} \in \mathbb{R}^{d\times d_h}$ and $d_h = d/H$ for H heads. The head output is $\mathbf{O}_h = \mathrm{softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^\top}{\sqrt{d_h}}\right)\mathbf{V}_h$, and outputs from all heads are concatenated and projected to form the layer output.

KV caching. During autoregressive decoding, the KV cache stores $\mathbf{K}_h$ and $\mathbf{V}_h$ for all previous tokens and heads, incurring per-layer memory $M_{\text{standard}} = 2LHd_h = 2Ld$. Equivalently, per token the cache stores $2Hd_h$ floating-point values (keys and values across H heads). Over N layers in bfloat16 precision, this is $2N \cdot (2LHd_h)$ bytes of KV storage.

[1] Code and trained models will be released upon acceptance.

Low-Rank KV Attention. To reduce redundant KV storage while preserving head-specific expressivity, we parameterize each head's key/value projection as a shared dense (unconstrained) base plus a head-specific LR residual, while keeping the resulting projection shape $d \times d_h$:

$$\mathbf{W}_h^K = \mathbf{W}_{\text{shared}}^K + \mathbf{U}_h^K \mathbf{B}_h^{K\top}, \quad \mathbf{W}_h^V = \mathbf{W}_{\text{shared}}^V + \mathbf{U}_h^V \mathbf{B}_h^{V\top}, \tag{1}$$

where $\mathbf{W}_{\text{shared}}^{K,V} \in \mathbb{R}^{d \times d_h}$ are dense projection matrices with no rank constraint, $\mathbf{U}_h^{K,V} \in \mathbb{R}^{d \times r}$, $\mathbf{B}_h^{K,V} \in \mathbb{R}^{d_h \times r}$, and $r \ll d_h$. The effective keys and values are $\mathbf{K}_h = \mathbf{X}\mathbf{W}_h^K$ and $\mathbf{V}_h = \mathbf{X}\mathbf{W}_h^V$. When $r = 0$, LRKV reduces to complete KV sharing (MQA-style) within a layer. Increasing r interpolates continuously toward standard MHA.

Training and initialization. All parameters in Equation 1 are optimized jointly during pretraining. Gradients from the shared and residual paths are additive, allowing the model to learn how much per-head variation is required. We initialize $\mathbf{W}_{\text{shared}}^{K,V}$ with standard Kaiming, and the per-head factors $\mathbf{U}_h^{K,V}, \mathbf{B}_h^{K,V}$ with small random values scaled by $1/\sqrt{r}$ so that initial residuals $\mathbf{U}_h\mathbf{B}_h^\top$ have magnitude $\approx 0.1\|\mathbf{W}_{\text{shared}}\|_F$, starting near the shared baseline and learning specialization. Since $\mathbf{K}_h, \mathbf{V}_h$ each sum two independent branches (Equation 2), we further scale every branch's std by $1/\sqrt{2}$ (σ_{shared}=σ_{residual}=$\sigma_{\text{base}}/\sqrt{2}$) to preserve the post-sum variance. During training, we materialize full $\mathbf{K}_h$ and $\mathbf{V}_h$ matrices before attention computation for simplicity. During inference, the factored form realizes the memory savings. This is an exact reparameterization, so training and inference are mathematically equivalent.

LRKV caching scheme. During decoding, the bottleneck is storing $\mathbf{K}_h, \mathbf{V}_h \in \mathbb{R}^{L \times d_h}$ for every head. LRKV instead caches shared features once per layer,

$$\mathbf{K}_{\text{shared}} = \mathbf{X}\mathbf{W}_{\text{shared}}^K, \quad \mathbf{V}_{\text{shared}} = \mathbf{X}\mathbf{W}_{\text{shared}}^V \in \mathbb{R}^{L \times d_h},$$

and compact per-head latents $\mathbf{R}_h^K = \mathbf{X}\mathbf{U}_h^K$ and $\mathbf{R}_h^V = \mathbf{X}\mathbf{U}_h^V \in \mathbb{R}^{L \times r}$. The full per-head features implied by Equation 1 are

$$\mathbf{K}_h = \mathbf{K}_{\text{shared}} + \mathbf{R}_h^K \mathbf{B}_h^{K\top}, \quad \mathbf{V}_h = \mathbf{V}_{\text{shared}} + \mathbf{R}_h^V \mathbf{B}_h^{V\top}. \tag{2}$$

Importantly, the matrices $\mathbf{B}_h^{K,V}$ are model parameters and are *not* cached per token.

Attention computation without explicit reconstruction. Naively materializing $\mathbf{K}_h, \mathbf{V}_h$ for all cached tokens would cost $O(Lrd_h)$ memory and compute. Instead, we exploit associativity to compute attention exactly without forming full per-head KV tensors. For a decoding step with query $\mathbf{q}_h \in \mathbb{R}^{d_h}$,

$$\mathbf{q}_h \mathbf{K}_h^\top = \mathbf{q}_h \mathbf{K}_{\text{shared}}^\top + (\mathbf{q}_h \mathbf{B}_h^K)\,(\mathbf{R}_h^K)^\top, \tag{3}$$

and for attention weights $\mathbf{a}_h \in \mathbb{R}^{1 \times L}$,

$$\mathbf{a}_h \mathbf{V}_h = \mathbf{a}_h \mathbf{V}_{\text{shared}} + (\mathbf{a}_h \mathbf{R}_h^V)\,\mathbf{B}_h^{V\top}. \tag{4}$$

Equations 3 and 4 compute the *exact* attention logits and outputs implied by Equation 2, but avoid explicit reconstruction of $\mathbf{K}_h, \mathbf{V}_h$. This form can be implemented inside fused attention kernels. See Table 8 for a detailed mathematical comparison to existing mechanisms.

Compatibility with Positional Embeddings. Positional embeddings such as RoPE are applied after projection in the full d_h-dimensional key space. Since the RoPE map at each position is linear in the key vector, $\text{RoPE}_t(\mathbf{K}_h) = \text{RoPE}_t(\mathbf{K}_{\text{shared}}) + \text{RoPE}_t(\mathbf{R}_h\mathbf{B}_h^\top)$. Thus LRKV is compatible with full-dimensional RoPE, whereas MLA-style projection absorption requires decoupling RoPE from the compressed K/V pathway (DeepSeek-AI et al., 2024). For query position t, cached position s, and RoPE matrix $\mathbf{R}_u$ at position u, the exact key interaction is $\mathbf{q}_{h,t}^\top \mathbf{R}_t^\top \mathbf{R}_s (\mathbf{k}_{\text{shared},s} + \mathbf{r}_{h,s}^K \mathbf{B}_h^{K\top})$. The optimized kernel caches the shared term in rotated form, then expands and rotates each residual contribution transiently within a fused tile. It never writes or persists a full per-head key tensor. The full-space rotation is essential because, in general, $\text{RoPE}(\mathbf{R}_h\mathbf{B}_h^\top) \neq \text{RoPE}(\mathbf{R}_h)\mathbf{B}_h^\top$. **KV cache memory complexity (during decoding).** Standard attention stores per-head keys and values: $M_{\text{standard}} = 2LHd_h$. LRKV stores shared features plus per-head latents:

$$M_{\text{LRKV}} = \underbrace{2Ld_h}_{\text{shared } (K,V)} + \underbrace{2LHr}_{\text{per-head latents}} = 2L(d_h + Hr).$$

Table 1: **Pretraining performance across model scales.** **Top:** MHA, MQA, and two GQA head configurations at their standard operating points; budgets differ across rows. **Bottom:** LRKV against MLA at matched KV-cache budgets, with the MLA latent width d_c set to each LRKV budget; each pair is one budget tier and the cache columns give the tier's budget at each scale. Config values are listed per scale; the lowest tier also reduces the LRKV shared width d_s below d_h=128. Cache ratios are $(d_s+Hr)/(Hd_h)$ for LRKV and $d_c/(2Hd_h)$ for MLA (Appendix I). Bold marks the best value per column (top) or the better of each matched pair (bottom); the shaded row is the default LRKV operating point. Metrics are on held-out FineWeb-Edu; 128M and 1.2B are means of three seeds (128M LRKV per-configuration CE ranges 0.0015–0.0068, s.d. 0.0008–0.0036; Appendix B), 2.5B and 6.3B single runs. The 128M LRKV default is the r=64 rank-sweep endpoint (Table 13).

		KV cache (% of MHA)				128M		1.2B		2.5B		6.3B	
Model	**Config. (128M/1.2B/2.5B/6.3B)**	128M	1.2B	2.5B	6.3B	CE↓	BPB↓	CE↓	BPB↓	CE↓	BPB↓	CE↓	BPB↓
Baselines at their standard operating points													
Standard MHA	KV heads 6/12/18/32	100	100	100	100	2.903	0.878	2.530	0.765	2.389	0.723	2.319	0.701
GQA	KV heads 3/4/6/2	50	33	33	6	2.907	0.880	2.531	0.765	2.387	0.720	2.354	0.712
GQA	KV heads 2/2/2/4	33	17	11	12.5	2.918	0.883	2.536	0.767	2.397	0.725	2.328	0.705
MQA	KV heads 1	17	8	6	3	2.929	0.886	2.573	0.778	2.408	0.729	2.304	0.697
Cache-matched pairs: MLA latent width d_c set to each LRKV budget													
MLA	d_c=128/256/384/704	8.3	8.3	8.3	8.6	2.901	0.876	2.564	0.775	2.392	0.7240	2.377	0.719
LRKV	r=8, d_s=16/32/48/96	8.3	8.3	8.3	8.6	**2.891**	**0.873**	**2.537**	**0.767**	**2.384**	**0.7215**	**2.299**	**0.695**
MLA	d_c=384/768/544/960	25.0	25.0	11.8	11.7	2.894	0.875	2.560	0.774	2.388	0.7228	2.370	0.717
LRKV	r=11/21/8/11	25.3	24.7	11.8	11.7	**2.889**	**0.871**	**2.542**	**0.768**	**2.368**	**0.7167**	**2.231**	**0.675**
MLA	d_c=768/1536/832/1472	50.0	50.0	18.1	18.0	2.891	0.873	2.552	0.771	2.379	0.7200	2.364	0.715
LRKV	r=43/53/16/19	50.3	49.7	18.1	18.0	**2.880**	**0.864**	**2.517**	**0.761**	**2.372**	**0.7145**	**2.230**	**0.675**
MLA	d_c=1024/1480/2560/4352	66.7	48.2	55.6	53.1	2.898	0.874	2.514	0.760	2.380	0.7192	2.360	0.714
LRKV (default)	r=64/51/64/64	66.7	48.2	55.6	53.1	**2.862**	**0.862**	**2.509**	**0.758**	**2.349**	**0.7110**	**2.288**	**0.692**

Thus the memory ratio is $\frac{M_{\text{LRKV}}}{M_{\text{standard}}} = \frac{d_h+Hr}{Hd_h} = \frac{1}{H} + \frac{r}{d_h}$. Per token, LRKV stores $2d_h + 2Hr$ values versus $2Hd_h$ for standard attention.

Compute, bandwidth and parameters. Per decoding step, standard attention costs $O(Ld_h)$ per head. The distributed forms in Equation 3–Equation 4 add $O(Lr + rd_h)$, a relative overhead of $\approx r/d_h$ for $L \gg 1$. Decoding is typically bandwidth- rather than compute-bound (Shah et al., 2024). Because the shared $L \times d_h$ tensors are reused across heads while each per-head latent is only $L \times r$, LRKV transfers $2L(d_h + Hr)$ bytes against MHA's $2LHd_h$, matching the cache reduction ratio. The kernel evaluates these forms directly on the factored cache (subsection I.5). Per-layer K/V parameters are $P_{\text{LRKV}} = 2dd_h + 2Hr(d + d_h)$ versus $2Hdd_h$ for MHA, and may be lower or higher depending on (H, r, d_h). Ranks are fixed per scale to trace the memory–quality frontier, not tuned post hoc; the 128M sweep improves monotonically with r (Appendix F). LRKV surpasses full-cache MHA with 8.3% cache at 128M, 2.5B, and 6.3B, including settings with fewer K/V parameters than MHA (Figure 7 and Appendix F). **Rank as a control for diversity.** The residual rank r controls a continuous spectrum between fully shared KVs (MQA-style, $r = 0$) and fully independent per-head projections (standard MHA, large r). The shared base $\mathbf{W}^{K,V}_{\text{shared}}$ captures globally useful features, while LR residuals enable head-specific variation within a constrained budget. See Appendix D for discussion of spectral structure and low-rank factorization.

4 Experiments

We evaluate LRKV through pretraining experiments across four model sizes (128M, 1.2B, 2.5B, 6.3B parameters), comparing against standard MHA and representative KV-efficient baselines. We measure pretraining loss, training efficiency, downstream task performance, and operator-space head geometry. **Experimental setup.** We pretrain decoder-only Transformers at four scales (128M, 1.2B, 2.5B, 6.3B) on FineWeb-Edu (Penedo et al., 2024) for 100B tokens (128M) and 50B tokens (others), comparing LRKV against Standard MHA, GQA, MQA, and MLA. Scale labels denote the Standard-MHA backbone size. Non-attention components and all non-learning-rate training choices are fixed, while attention-specific K/V parameter counts vary as implied by each parameterization. For every model scale and attention type, we select the learning rate with a 5B-token grid search before full training. Models use 2048-token context, Muon+AdamW optimizers (Jordan et al., 2024), and 8×H200 GPUs in bfloat16. Midtraining uses 568K examples from SmolTalk (Ben Allal et al., 2025) and only the official training or auxiliary splits of MMLU (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021); evaluation uses disjoint held-out benchmark splits.

4.1 PRETRAINING RESULTS

Final pretraining performance. Table 1 shows that LRKV at its default operating point (shaded row) has the lowest observed test loss at all four scales, against MHA, GQA, and MQA at their standard operating points and MLA at every matched budget. At 128M, the updated r=64 run reaches 2.862 CE (0.862 BPB) with 66.7% of MHA's cache, versus 2.903/0.878 for MHA. At 6.3B scale, LRKV reaches 2.288 CE (0.692 BPB), below MHA (2.319 CE), MQA (2.304 CE), GQA (2.328 CE), and MLA (2.377 CE). At 1.2B scale, LRKV reaches 2.509 CE (0.758 BPB) with 48% of MHA's cache, and at 2.5B it reaches 0.711 BPB with 55.6%.

The lower block of Table 1 supplies the equal-memory control: with d_c set to each LRKV budget, LRKV is lower in CE and BPB than MLA at every matched tier, at 8.3%, 25.3%-vs-25.0%, and 50.3%-vs-50.0% at 128M and at 8.3%, 11.8%, 18.1%, and 55.6% at 2.5B (Figures 3 and 7).

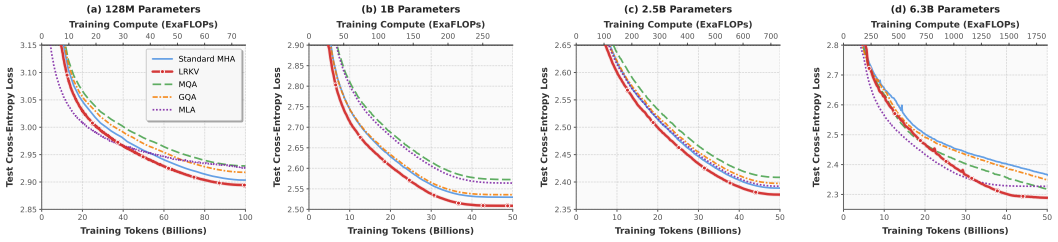


Figure 2: **Cross-scale pretraining curves with dual-axis compute metrics.** Test cross-entropy loss across four model scales (128M, 1.2B, 2.5B, 6.3B) plotted against training tokens (bottom x-axis) and cumulative training compute in ExaFLOPs (top x-axis). These are the archived per-step runs. The updated 128M scalar endpoints in Table 13 are not retrofitted because their per-step histories are unavailable.

Cross-scale training efficiency and compute normalization. Figure 2 presents test performance across all four model scales, measured in cross-entropy loss with dual-axis compute metrics (training tokens and cumulative FLOPs). The archived curves show LRKV with the lowest test loss in the 128M and 2.5B run sets and competitive trajectories at 1.2B and 6.3B. The upper axis reports model-derived cumulative training FLOPs. At 2.5B, LRKV's ordering is unchanged under this normalization. LRKV reaches equivalent baseline performance in 18–30% fewer optimizer steps (23.6% average across all methods), while no baseline reaches LRKV's final performance within the full compute budget. At the largest 6.3B scale, LRKV achieves 2.288 CE versus Standard MHA's 2.319 CE (1.3% improvement). At 1.2B every method is averaged over three runs. The 2.5B and 6.3B results use one run per configuration.

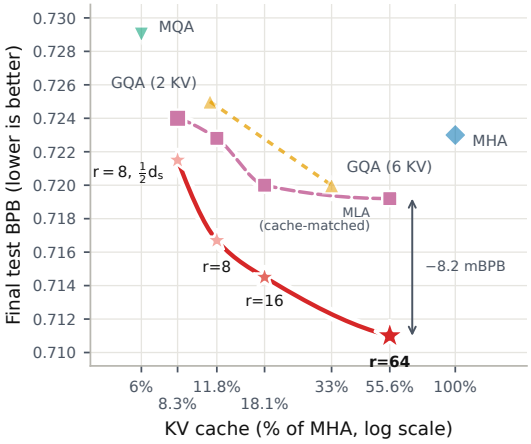
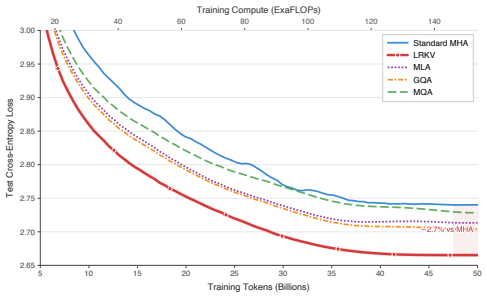


Figure 3: **Memory–quality tradeoff at 2.5B scale.** Final test BPB against KV cache; gridlines mark the tested budgets. Solid: LRKV rank scan (r=8 with reduced shared width, r=8, r=16, r=64). Dashed: MLA with d_c matched to each LRKV budget; dotted: GQA with 2 and 6 KV heads. LRKV is below MLA at every budget, by 2.5–8.2 mBPB, and reaches each baseline's final BPB in 18.7–29.7% fewer steps.

Training efficiency analysis. LRKV also reaches each baseline's final validation loss in fewer optimizer steps at 2.5B scale (Figure 3). All four tested operating points lie below MHA. A reduced-shared-width r=8 variant uses 8.3% cache and reaches BPB 0.7215, matching MLA's cache while improving on its 0.724. The full-width r=8 configuration uses 11.8% cache and reaches 0.7167, r=16 uses 18.1% and reaches 0.7145, and the default r=64 uses 55.6% and reaches 0.711. The step savings (mean **23.6%**) follow the same 5B-token learning-rate selection protocol, and every method materializes full K/V during training, so this is a sample-efficiency result. LRKV also improves on full-parameter MHA with fewer K/V parameters at every scale (Table 12), so its gain is not explained by parameter allocation.

Matched-cache comparison. With d_c set to each LRKV budget, KV memory is held fixed and only



Test CE, trained at 8K	MHA	MQA	GQA	MLA	LRKV
Evaluated at 16K ↓	2.87	2.92	2.85	2.86	**2.79**
Evaluated at 32K ↓	3.20	3.32	3.23	3.24	**3.12**
RULER task (32K)	**MHA**	**MQA**	**GQA**	**MLA**	**LRKV**
Single-needle retrieval	40.4	32.6	38.4	38.9	**41.7**
Multi-needle retrieval	36.8	27.4	33.1	33.7	**38.3**
Variable tracking	33.0	24.9	30.5	31.2	**34.6**
Frequent-word extraction	43.4	37.3	41.1	41.6	**44.3**
Question answering	40.3	28.5	36.6	37.2	**41.2**
Average (displayed groups)	38.8	30.1	35.9	36.5	**40.0**

Figure 4: **Long-context results for 512M models pretrained at 8K. Left:** test cross-entropy at 8192-token sequences over 50B tokens; final losses are 2.67 (LRKV), 2.70 (GQA), 2.71 (MLA), 2.73 (MQA), and 2.74 (MHA), so LRKV attains the lowest loss with 48.2% of MHA's KV cache. **Right, top:** test cross-entropy of the same checkpoints evaluated at 16K and 32K without further training. **Right, bottom:** RULER at 32K context, four times the training context, without long-context fine-tuning. RULER entries are accuracies in % by string match; multi-answer tasks are scored by the fraction of expected items recovered, and the final row is the unweighted average over task groups. Shaded bold cells mark the best method per row.

the parameterization varies. In the 100B-token 128M sweep LRKV leads MLA at every target, with ΔCE of -0.010 at 8.3%, -0.005 at 25%, and -0.011 at 50%; each entry is a mean of three seeds, and every gap exceeds the corresponding three-seed range (Appendix B). The 8.3% pair reduces the shared width to d_s=16, since with six heads a full-width shared branch alone costs 16.7% of MHA's cache, so the lead survives that aggressive regime. At 2.5B the scan extends the ordering to four budgets with single runs: LRKV is below MLA by 2.5, 6.1, 5.5, and 8.2 mBPB at 8.3%, 11.8%, 18.1%, and 55.6% (Figure 3), and the same block shows LRKV below MLA at every tier at 1.2B and 6.3B (Figure 9). Every LRKV point also has lower CE and BPB than the fixed MHA, GQA, and MQA operating points; Appendix B gives the full cache accounting.

Long Context Pretraining. We extend evaluation to 8192-token sequences using 512M parameter models trained for 50B tokens (153.6 ExaFLOPs). Figure 4 shows that LRKV achieves 2.67 test loss with 48.2% of MHA's cache, outperforming MHA (2.74) by 2.7% and also surpassing GQA (2.70), MLA (2.71), and MQA (2.73). A controlled retrieval probe on the same checkpoints (Table 15, Appendix F) shows LRKV matching MHA at mid-context depth and improving recent-record retrieval (97.7% vs. 85.9%). Evaluated at 16K and 32K without further training, LRKV keeps the lowest loss. On RULER at 32K, four times the training context, the same 8K-trained models average 40.0 (LRKV) against 38.8 (MHA), 36.5 (MLA), 35.9 (GQA) and 30.1 (MQA), and LRKV leads every task category (Figure 4, right).

Autoregressive decoding efficiency. We benchmark batch-1 decode across eleven rounded systems-shape labels from 128M to 30B (Figure 5; exact tuples in Table 20). Weight streaming from HBM makes the weight-to-KV-cache bandwidth ratio the key efficiency lever.

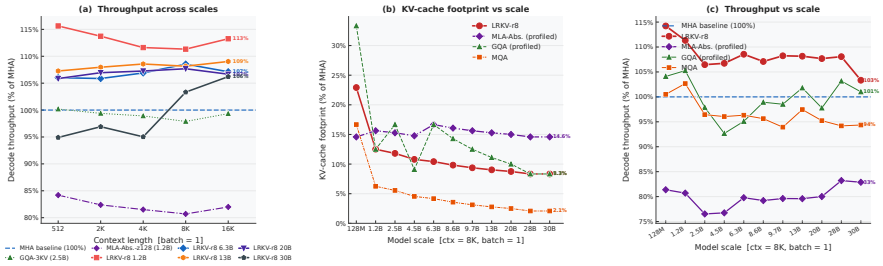


Figure 5: **Decode efficiency of LRKV and competing attention architectures (batch size 1) on a B200 with the optimized LRKV kernel and fused baselines. (a)** Raw decode throughput relative to MHA as a function of context length for five representative LRKV model scales, with GQA-3KV at 2.5B and absorbed MLA-z128 at 1.2B as measured references. **(b)** Factored or absorbed KV-cache footprint as a percentage of MHA at context length 8,192 across all eleven scales. The GQA and MLA references use the profiled configuration available at each scale. **(c)** Raw decode throughput relative to MHA at context length 8,192, batch size=1, for all eleven scales (128M–30B). No calibration multiplier is applied.

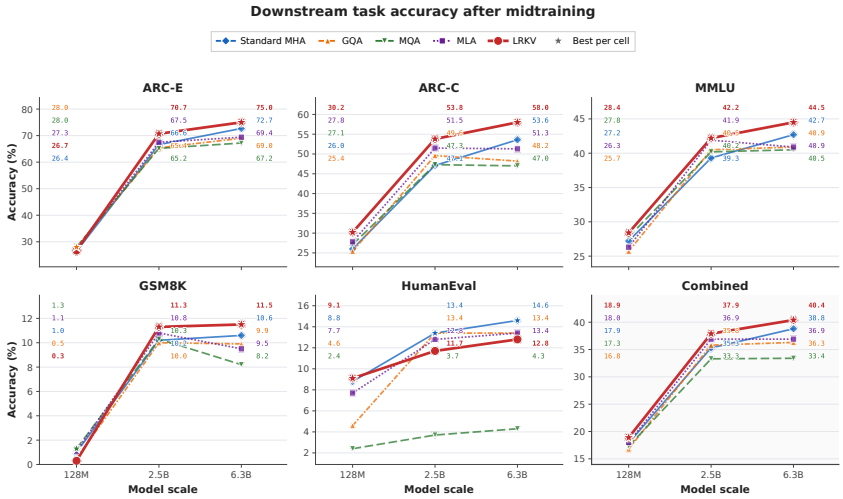


Figure 6: **Downstream task performance after midtraining (128M, 2.5B, 6.3B scales).** Per-benchmark accuracy as a function of model scale for Standard MHA, GQA, MQA, MLA, and LRKV on ARC-Easy, ARC-Challenge, MMLU, GSM8K, HumanEval, and combined average. Inline value clusters next to each marker preserve all individual scores (sorted descending; LRKV bolded). LRKV achieves the highest combined accuracy across all three scales (18.9%, 37.9%, 40.4%). LRKV is best in 11 of the 15 per-benchmark cells; the exceptions are ARC-Easy and GSM8K at 128M, where every method is near chance, and HumanEval at 2.5B and 6.3B, where MHA is higher. Full numerical values are reproduced in Table 2 (Appendix A).

End-to-end factored decoding. The benchmarked LRKV path stores shared K/V features and per-head rank-r latents. Its non-materializing kernel evaluates Equation 3 and Equation 4 without allocating full per-head K/V tensors, realizing the cache and memory-traffic reductions of section 3. The exact path caches the shared term after RoPE and evaluates the rotated residual contribution inside the kernel. Appendix I.5 specifies the operator sequence and benchmark protocol.

LRKV throughput. The batch-1 scale sweep (B200, 8K context) puts LRKV $r{=}8$ at 103–111% of MHA throughput from 1.2B through 30B, and at 107–174% at batch 8. At 2.5B on a B200 with the optimized kernel, the batch-1 step is launch-bound and LRKV $r{=}8$ matches MHA (51.7 vs 49.5 tok/s at 8K); at batch 8 it reaches 422.7 tok/s against 397.0 for MHA and 392.6 for GQA, and at batch 64 2672.8 against 949.5 and 2092.7. Thus LRKV creates memory and batching headroom at medium scale and, with a bandwidth-efficient kernel, converts it into throughput.

Serving capacity. The kernel converts cache compression into deployable capacity. On one B200 (2.5B, bfloat16, medians of three runs), LRKV ($r{=}8$) serves batch 566 at 8K context (4.6M resident tokens, 3698.5 tok/s) where MHA stops at batch 66 and GQA at 199; at 128K it holds 35 sequences against 4 for MHA, and for MHA, GQA, MQA, and LRKV measured capacity matches the analytic cache size to within one sequence (Appendix B). Across 8K–128K contexts LRKV $r{=}8$ holds 4.6M resident tokens, 8.5–8.9$\times$ MHA and 2.8–2.9$\times$ GQA, and at its boundary it decodes faster than GQA at GQA's boundary (3698.5 vs 2729.5 tok/s at 8K). MQA holds more tokens still, but at the quality cost in Table 1; LRKV is the operating point that keeps quality.

4.2 Downstream Task Performance

Using the same supervised midtraining recipe for every architecture, we evaluate five held-out benchmarks (Figure 6; all scores in Appendix A). HumanEval at 2.5B and 6.3B are the cells where MHA is higher (13.4% versus 11.7% and 14.6% versus 12.8%), which we treat as a benchmark-level exception; at 128M, ARC-Easy and GSM8K are within noise of chance for every method.

Scale-consistent gains in combined accuracy. LRKV achieves the highest combined accuracy at all three scales, reaching 18.9% at 128M, 37.9% at 2.5B, and 40.4% at 6.3B. At 2.5B scale, LRKV outperforms Standard MHA (35.3%), GQA (35.8%), MQA (33.3%), and MLA (36.9%). The largest gains are on ARC-Easy (+4.1pp over MHA), ARC-Challenge (+6.7pp), MMLU (+2.9pp), and GSM8K (+1.1pp). At 6.3B scale, LRKV achieves 40.4% combined versus MHA's 38.8%, with the largest improvements on ARC-Challenge (58.0% vs 53.6%) and MMLU (44.5% vs 42.7%).

The midtraining recipe, data, schedule, and budget are identical across variants with no per-architecture tuning. The downstream models retain their pretraining operating points, while the 128M sweep separately controls cache budget. At 2.5B, LRKV reaches 37.9% combined accuracy against full-cache MHA's 35.3% while reducing KV memory by 44.4%. These are observed checkpoint scores without repeat-run uncertainty estimates.

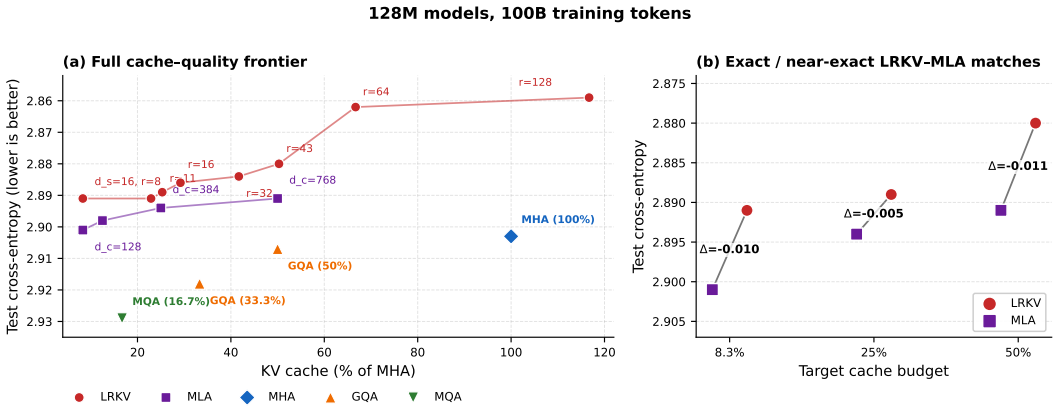


Figure 7: **Cache–quality frontier (128M, 100B tokens). (a)** All observed LRKV ranks, MLA latent dimensions, and fixed MHA/GQA/MQA baselines. The d_s=16, r=8 LRKV point is a reduced-shared-width variant. The remaining LRKV points retain the default full-width shared branch. **(b)** Direct LRKV–MLA comparisons at exact (8.3%) or near-exact (within 0.3pp at 25% and 50%) cache. Δ is LRKV CE minus MLA CE, so all three negative values favour LRKV. MHA, MQA, and GQA remain in panel (a) because the sweep does not provide comparably close LRKV matches at their discrete cache budgets.

MQA is consistently weakest on code generation (HumanEval: 2.4% at 128M, 3.7% at 2.5B, and 4.3% at 6.3B versus 11.7–14.6% for the other methods at 2.5B and 6.3B, and 4.6–9.1% at 128M). LRKV avoids this drop while retaining compact per-head residuals.

Pretraining quality predicts downstream performance. Across the five 2.5B operating points, lower pretraining BPB tracks higher combined accuracy (R^2=0.701, n=5). We use this as a descriptive consistency check rather than causal evidence; see Appendix G.

4.3 Mechanistic Diagnostics

We analyze why LRKV preserves modeling quality despite reducing KV-cache memory. The central hypothesis is that head-specific K/V projections contain structured redundancy: they share a common component, but still require compact head-specific variation. LRKV implements this directly through $\mathbf{W}_h = \mathbf{W}_{\text{shared}} + \mathbf{U}_h\mathbf{B}_h^\top$. We examine three questions: whether rank controls quality, whether LRKV preserves operator-space head diversity, and whether the learned decomposition uses the shared and residual pathways meaningfully.

Analysis setup. We analyze final pretrained checkpoints using gauge-invariant bilinear forms $A_h = W_h^Q (W_h^K)^\top$, which are invariant to per-head rotations. Head diversity is the effective rank of the resulting Gram matrix in uncentered and PCA-centered forms. Centering removes the shared mean direction and isolates specialization (full method in Appendix E).

Rank selection determines capacity and performance. Figure 7 shows monotone improvement across the default full-width sweep from r=8 through r=128, approaching saturation at high ranks. Even r=8 (22.9% cache) is below MHA in CE and BPB, and r=64 reaches 2.862/0.862 at 66.7% cache. The matched points resolve LRKV–MLA directly: LRKV leads at 8.3%, 25%, and 50% after 100B tokens (Appendix B). The 8.3% result uses d_s=16, r=8 because a default shared branch alone has a $1/H = 16.7\%$ cache floor.

LRKV retains operator-space head diversity. The top row of Figure 8 shows LRKV head similarity closely matching Standard MHA. At 2.5B, its gauge-invariant operator-space effective rank is 98.3% versus 98.9% for MHA, while MQA and GQA reach 86.2% and 95.4%. Thus, at 55.6% cache, LRKV retains nearly all of MHA's measured diversity.

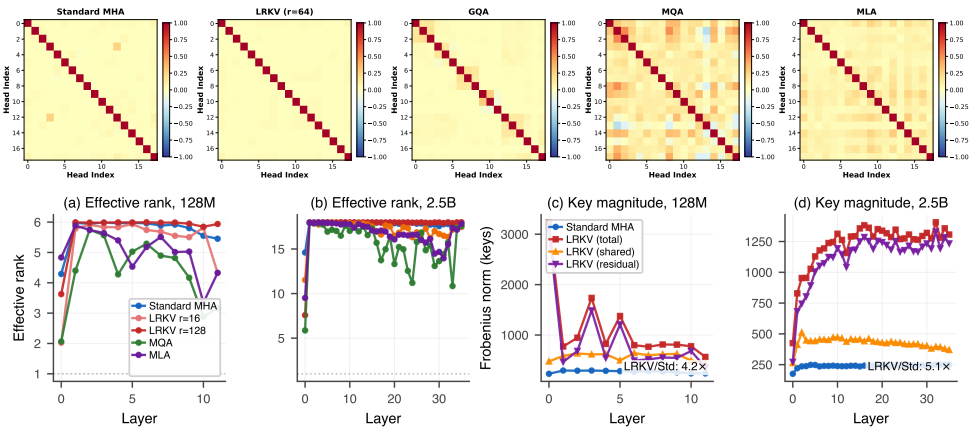


Figure 8: **MHA-like operator diversity and an active shared-plus-residual decomposition. Top:** gauge-invariant pairwise head similarities at 2.5B. LRKV closely matches MHA's dark off-diagonal structure, unlike the stronger correlations induced by more restrictive sharing. **Bottom (a,b):** layerwise effective rank at 128M and 2.5B; the 128M panel contains measured checkpoints only, while LRKV reaches 98.3% versus MHA's 98.9% at 2.5B using 55.6% of its KV cache. **Bottom (c,d):** key-projection norms at 128M and 2.5B show that both the shared and residual pathways remain active. Post-projection RMSNorm attenuates the resulting scale difference before attention.

PCA centering separates shared and residual operator variation. Uncentered effective rank includes the shared mean direction across heads, while PCA centering removes it. After centering, LRKV reaches 93.5% versus 94.0% for Standard MHA, a gap of 0.5pp. MQA rises from 86.2% uncentered to 91.0% centered, a pattern consistent with query-side differentiation around shared K/V. The bottom row of Figure 8 further shows that both LRKV branches carry substantial norm rather than one collapsing. Post-projection RMSNorm attenuates sensitivity to the absolute scale gap against MHA, so we treat these norms as a decomposition diagnostic rather than a quality metric.

Interpreting the diversity metric. The metric shows that an additive shared-plus-residual parameterization has different operator geometry from a latent bottleneck. We do not use it to rank quality across methods, since the 2.5B LRKV and MLA checkpoints use different cache budgets (55.6% and 8.3%); the matched 128M frontier supplies that comparison.

The deployed kernel is bit-exact against the module's explicit einsum evaluation and deviates from an fp32 reference by at most 7×10^{-4}. At 2.5B, batch 8, and 8K context on a B200 it reduces the KV cache from 20.29 to 2.40 GB and peak memory from 27.19 to 9.20 GB; at 128K context it serves 35 sequences where MHA serves 4.

5 Conclusion

Low-Rank Key-Value Attention turns cross-head redundancy into a continuous cache-quality control by pairing a shared dense K/V path with compact per-head residuals, retaining every token and exact full-dimensional RoPE. It records the lowest observed loss across four scales at the standard operating points and leads cache-matched MLA at every tested budget. At 128M, three-seed means give LRKV lower loss than matched-budget MLA at 8.3%, 25%, and 50%, and the 8.3%-cache configuration stays below full-cache MHA at 128M, 2.5B, and 6.3B; at 2.5B it leads cache-matched MLA at every budget, reaches baseline quality in 18–30% fewer steps (23.6% on average), and preserves MHA-like head diversity; the same cache-matched lead holds at 1.2B and 6.3B. After common midtraining it records the strongest combined downstream score, and its 8K-trained model leads RULER at 32K (40.0 vs. 38.8 for MHA). Its exact kernel expands context and batch capacity under fixed memory: at 2.5B and 8K context on one B200 it serves batch 566 where MHA stops at 66 and fused GQA at 199 (8.5–8.9× MHA's resident tokens at every context to 128K), and it runs at 103–111% of MHA batch-1 throughput from 1.2B to 30B, linking the quality frontier to an end-to-end deployment path. Every decode number derives from logged runs released with the code, and the measured capacity boundaries of MHA, GQA, MQA, and LRKV match the analytic cache formula to within one sequence at every context, so the memory accounting in Appendix I predicts deployment directly.

AI USE STATEMENT

AI was not used to originate the LRKV parameterization, design the experiments, conduct the literature survey, or generate experimental observations. In the writing, AI was used to catch grammatical errors and to flag passages that needed more attention, such as claims lacking precise detail or quantitative references to the reported results, which the authors then revised. The authors ran every experiment, reviewed all AI-assisted code and text, executed and verified the revised code and figures, and traced every number to an archived data source, figure input, or author-supplied result record. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses existing public datasets and benchmarks for pretraining, midtraining, and evaluation, and it does not collect new human-subject data. The contribution concerns architectural and systems efficiency rather than a new deployment or dataset release. Standard risks associated with language-model training data and downstream model use remain applicable. The authors have reviewed and followed the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The default full-width LRKV parameterization, cache-size formula, and non-reconstructing decode form are specified in section 3, and Appendix F (model shapes, optimizer, schedules, token budgets, and the count-once cache-accounting convention we use throughout). The reduced-width $r{=}8$ points ($d_s{=}16/32/48/96$) are labelled separately, and their exact configurations will accompany the code release. Per-scale hyperparameters are tabulated in Appendix H.

Two points bear directly on whether our numbers can be reproduced. First, the 100B-token matched-cache endpoints and cache-accounting convention are reported in Table 13 and Appendix B. These endpoints are means over three random seeds, and the table reports the observed differences directly. Second, Appendix I.5 specifies the non-materializing decode path, stored cache state, operator sequence, and benchmark protocol. subsection 4.3 reports the numerical check against explicit reconstruction.

We will release training code and configurations, including the non-materializing kernel and the scripts used for the reported analyses.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, 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)*, 2023.

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

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, et al. Smollm2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Srinadh Bhojanapalli, Ayan Chakrabarti, Andreas Veit, Michal Lukasik, Himanshu Jain, Frederick Liu, Yin-Wen Chang, and Sanjiv Kumar. Leveraging redundancy in attention with reuse transformers. *arXiv preprint arXiv:2110.06821*, 2021.

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 *The Thirteenth International Conference on Learning Representations*, 2025.

Vivek Chari, Guanghui Qin, and Benjamin Van Durme. Kv-distill: Nearly lossless learnable context compression for llms. *arXiv preprint arXiv:2503.10337*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Choromanski, Valerii Likhosherstov, et al. Rethinking attention with performers. In *ICLR*, 2021.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Ronglong Fang and Yuesheng Xu. Addressing spectral bias of deep neural networks by multi-grade deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 114122–114146, 2024. doi: 10.52202/079017-3625.

Gemma Team. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Jingcheng Hu, Houyi Li, Yinmin Zhang, Zili Wang, Shuigeng Zhou, Xiangyu Zhang, Heung-Yeung Shum, and Daxin Jiang. Multi-matrix factorization attention. *arXiv preprint arXiv:2412.19255*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. `https://kellerjordan.github.io/posts/muon/`, 2024. Published December 8, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

Malik Khalaf, Yara Shamshoum, Nitzan Hodos, Yuval Sieradzki, and Assaf Schuster. QKV projections require a fraction of their memory. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2506.02939.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

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

Xingtai Lv, Ning Ding, Kaiyan Zhang, Ermo Hua, Ganqu Cui, and Bowen Zhou. Scalable efficient training of large language models with low-dimensional projected attention. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 14588–14599. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.emnlp-main.808/.

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. arXiv:2502.07864.

Paul Michel, Omer Levy, and Graham Neubig. Are Sixteen Heads Really Better than One? In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 14014–14024, 2019.

NVIDIA. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. Technical report, NVIDIA, 2025. URL https://research.nvidia.com/labs/nemotron/files/NVIDIA-Nemotron-3-Nano-Technical-Report.pdf.

NVIDIA. Nemotron 3 Super: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. Technical report, NVIDIA, 2026a. URL https://research.nvidia.com/labs/nemotron/files/NVIDIA-Nemotron-3-Super-Technical-Report.pdf.

NVIDIA. Nemotron 3 Ultra: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. Technical report, NVIDIA, 2026b. URL https://research.nvidia.com/labs/nemotron/files/NVIDIA-Nemotron-3-Ultra-Technical-Report.pdf.

OpenAI. GPT-4. https://openai.com/index/gpt-4/, 2023. Accessed: 2025-10-02.

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. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Yuting Peng, Yang Wang, Zhou Fang, Lipeng Zhu, Yingnan Deng, and Yifei Duan. Revisiting LoRA: A smarter low-rank approach for efficient model adaptation. In *2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA)*, pp. 1248–1252. IEEE, 2025.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 5301–5310. PMLR, 2019.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for kv cache compression. *arXiv preprint arXiv:2408.05646*, 2024.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2407.08608.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5797–5808. Association for Computational Linguistics, 2019. URL https://aclanthology.org/P19-1580/.

Hong Wang and Kelly Wang. Complete characterization of gauge symmetries in transformer architectures. In *NeurIPS 2025 Workshop on Symmetry and Geometry in Neural Representations*, 2025.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

An Yang et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Dingyu Yao, Bowen Shen, Zheng Lin, Wei Liu, Jian Luan, Bin Wang, and Weiping Wang. TailorKV: A hybrid framework for long-context inference via tailored KV cache optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20340–20359. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.1043/.

David Yunis, Kumar Kshitij Patel, Samuel Wheeler, Pedro Savarese, Gal Vardi, Karen Livescu, Michael Maire, and Matthew R Walter. Approaching deep learning through the spectral dynamics of weights. *arXiv preprint arXiv:2408.11804*, 2024.

Ted Zadouri, Hubert Strauss, and Tri Dao. Hardware-efficient attention for fast decoding. *arXiv preprint arXiv:2505.21487*, 2025.

Yifan Zhang, Yifeng Liu, Huizhuo Yuan, Zhen Qin, Yang Yuan, Quanquan Gu, and Andrew C. Yao. Tensor product attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

A Detailed downstream task scores

Table 2 reproduces the per-cell numerical scores summarised visually in Figure 6. Each cell reports accuracy (%) on the corresponding benchmark after supervised midtraining; bold values mark the best method within a (scale, benchmark) cell, and the *Comb.* column is the unweighted average across the five benchmarks. HE = HumanEval.

Table 2: **Downstream task performance after midtraining (128M, 2.5B, and 6.3B scales) – full numerical values.** LRKV achieves the highest combined accuracy across all three scales (18.9%, 37.9%, 40.4%) on five diverse benchmarks. Combined score (Comb.) is the average across all five benchmarks. HE = HumanEval.

	128M						2.5B						6.3B					
Model	**ARC-E**	**ARC-C**	**MMLU**	**GSM8K**	**HE**	**Comb.**	**ARC-E**	**ARC-C**	**MMLU**	**GSM8K**	**HE**	**Comb.**	**ARC-E**	**ARC-C**	**MMLU**	**GSM8K**	**HE**	**Comb.**
Standard MHA	26.4	26.0	27.2	1.0	8.8	17.9	66.6	47.1	39.3	10.2	**13.4**	35.3	72.7	53.6	42.7	10.6	**14.6**	38.8
GQA	**28.0**	25.4	25.7	0.5	4.6	16.8	65.4	49.6	40.5	10.0	**13.4**	35.8	69.0	48.2	40.9	9.9	13.4	36.3
MQA	**28.0**	27.1	27.8	**1.3**	2.4	17.3	65.2	47.3	40.2	10.3	3.7	33.3	67.2	47.0	40.5	8.2	4.3	33.4
MLA	27.3	27.8	26.3	1.1	7.7	18.0	67.5	51.5	41.9	10.8	12.8	36.9	69.4	51.3	40.9	9.5	13.4	36.9
LRKV	26.7	**30.2**	**28.4**	0.3	**9.1**	**18.9**	**70.7**	**53.8**	**42.2**	**11.3**	11.7	**37.9**	**75.0**	**58.0**	**44.5**	**11.5**	12.8	**40.4**

B Matched-Budget Comparison and Serving Capacity

The main comparison in Table 1 places each method at its standard operating point, so the rows differ in memory. This appendix reports the 100B-token matched-cache comparison against MLA and the measured serving-capacity results.

100B-token matched-cache frontier. The expanded 128M sweep includes exact or near-exact LRKV–MLA matches at 8.3%, 25%, and 50% of MHA's KV cache. The 8.3% LRKV point reduces the shared cached width to d_s=16; with H=6, d_h=128, and r=8, its cache ratio is $\frac{d_s+Hr}{Hd_h} = \frac{16+6\cdot 8}{6\cdot 128} = 8.33\%$. The 25% and 50% points retain the default full-width shared branch and match MLA to within 0.3 percentage points.

Table 3: **Exact or near-exact cache matches at 128M and 100B training tokens.** Scalar test endpoints for the configurations in Table 13. Negative Δ favours LRKV.

Budget	**LRKV config**	**LRKV CE / BPB**	**MLA config**	**MLA CE / BPB**	**ΔCE**
8.3%	d_s=16, r=8 (8.3%)	**2.891 / 0.873**	d_c=128 (8.3%)	2.901 / 0.876	**−0.010**
25%	r=11 (25.3%)	**2.889 / 0.871**	d_c=384 (25.0%)	2.894 / 0.875	**−0.005**
50%	r=43 (50.3%)	**2.880 / 0.864**	d_c=768 (50.0%)	2.891 / 0.873	**−0.011**

LRKV has lower observed CE and BPB at all three matched targets. Each scalar endpoint is the mean of three runs with different random seeds. Across the three seeds the 128M LRKV CE ranges and standard deviations are: r=8 range 0.0020, s.d. 0.00113; r=8 with d_s=16 range 0.0015, s.d. 0.00079; r=11 range 0.0042, s.d. 0.0023; r=43 range 0.0068, s.d. 0.00362; r=64 range 0.0040, s.d. 0.00219. Every LRKV–MLA gap in Table 3 exceeds the corresponding three-seed range (12.7, 2.2, and 3.0 LRKV seed standard deviations at 8.3%, 25%, and 50%), so the ordering is not a seed effect.

Cache-matched grid across scales. Figure 9 plots the lower block of Table 1. Panel (a) gives the LRKV minus MLA BPB gap at every scale and budget tier; all sixteen cells are negative. Panel (b) shows final BPB against KV cache at each scale with LRKV solid and MLA dashed. We do not fit a scaling law to this grid: the 128M row is trained for 100B tokens and the others for 50B, the 2.5B and 6.3B rows are single runs, and the 6.3B budgets are rounded to the nearest integer rank (8.6%, 11.7%, 18.0%, and 53.1%), so the grid is a set of paired measurements rather than a fitted surface.

Peak memory and serving capacity. All decode measurements in this section are on a single NVIDIA B200 (2.5B backbone: 36 layers, 18 heads, d_h=128, bfloat16). Each point is a fresh process: prefill to context−256 tokens, 16 warm-up decode steps, then 256 timed steps between `torch.cuda.synchronize()` calls; tok/s = batch×256 / seconds; we report the median of three runs. Peak memory is `max_memory_allocated` (GiB) including the chunked prefill; KV is the measured size of the cache tensors. Baselines use the fused SDPA path (`enable_gqa`, no K/V expansion); MLA (absorbed) uses batched matmuls over the shared latent; LRKV uses the optimized

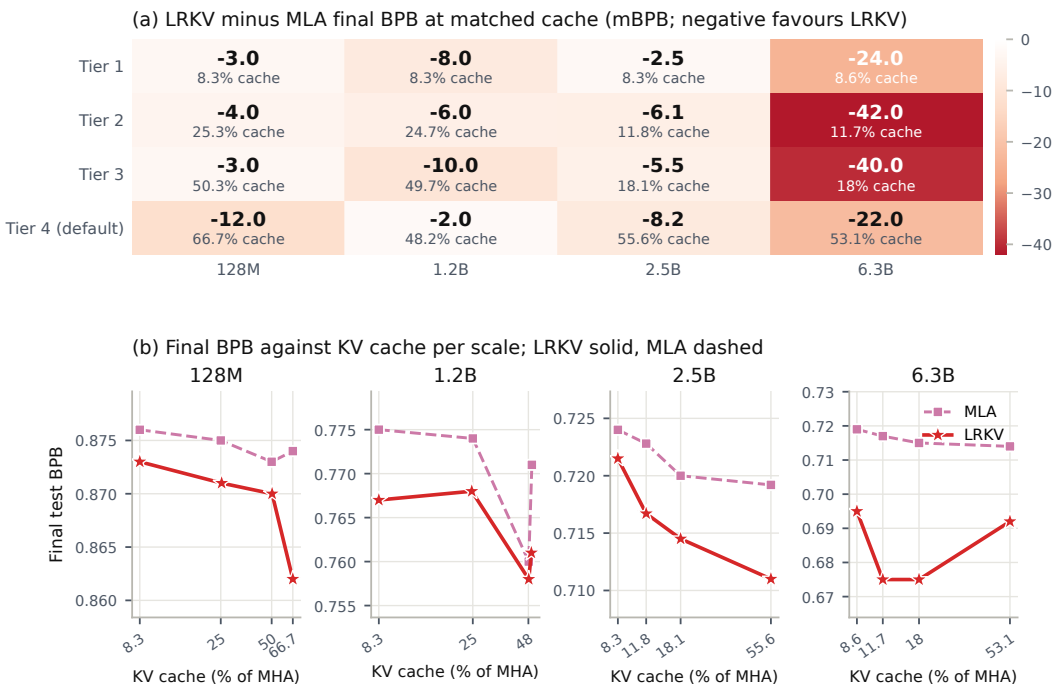


Figure 9: **Cache-matched LRKV vs. MLA across four scales. (a)** LRKV minus MLA final BPB (mBPB) for each scale and budget tier of Table 1; the cache percentage of the LRKV configuration is printed in each cell. **(b)** Final BPB against KV cache at each scale, LRKV solid and MLA dashed, with the MLA latent width d_c set to each LRKV budget. Tiers 1–3 are 8.3%, 25%, and 50% at 128M and 1.2B and 8.3%, 11.8%, and 18.1% at 2.5B and 6.3B; tier 4 is the default LRKV operating point.

kernel of Appendix I.5. Raw logs and the unmodified-kernel numbers are in `benchmarks/b200/` of the code release.

Table 4: **Decode throughput and memory at 2.5B on one B200.** tok/s at fixed batch and context (medians of three); peak and KV at batch 8 / 8K. "OOM" = did not fit; "–" = not run. LRKV r=8 is the full-width configuration (11.8% cache). MLA (absorbed) additionally caches decoupled per-head RoPE keys, so its cache (4.23 GB, 20.8% of MHA) exceeds the latent-only 8.3% counted in Table 1.

Method	8K ctx, tok/s			32K ctx, tok/s		batch 8 / 8K	
	b=8	b=32	b=128	b=8	b=32	Peak (GB)	KV (GB)
Standard MHA	397.0	867.1	OOM	331.1	OOM	27.2	20.29
GQA (6 KV)	392.6	1593.2	2519.1	397.0	1036.7	13.2	6.76
MQA	383.0	1494.4	6098.5	383.2	1537.3	7.4	1.13
MLA (absorbed, d_c=384)	336.0	597.7	726.1	124.1	182.9	16.1	4.23
MLA (latent, d_c=384)	65.0	66.6	84.1	17.0	20.8	10.1	1.69
LRKV (r=8)	422.7	1734.9	3094.6	463.4	871.1	9.2	2.40
LRKV (r=64)	425.8	1770.5	OOM	441.8	OOM	19.4	11.27

At a fixed memory budget LRKV r=8 holds 4.6M resident tokens at every context, 8.6× MHA and 2.8× GQA; MQA holds more but at 5.6% of the cache and with its quality gap (Table 1). With the optimized kernel LRKV's throughput at its own boundary exceeds GQA's at GQA's boundary.

C Related Work (Full Version)

The memory footprint of the key-value (KV) cache has become a central bottleneck in deploying large autoregressive Transformers. While early work improved the *computational* efficiency of attention via sparsity, kernelization, or low-rank approximations of the attention matrix (Child et al., 2019; Beltagy et al., 2020; Wang et al., 2020; Katharopoulos et al., 2020; Choromanski et al., 2021), these

Table 5: **Capacity boundary on one B200: largest batch that completes prefill and 256 decode steps in three fresh runs**, and decode throughput at that boundary for 8K and 128K context. Measured boundaries agree with the analytic cache size (Appendix I.1) to within one sequence for MHA, GQA, MQA and LRKV; the MLA variants are limited by prefill/up-projection transients rather than by their cache.

	max batch					tok/s at boundary	
Method	8K	16K	32K	64K	128K	8K	128K
Standard MHA	66	33	16	8	4	914.3	94.9
GQA (6 KV)	199	100	50	25	12	2729.5	286.5
MQA	1203	600	303	151	75	15357.6	1612.4
MLA (absorbed, d_c=384)	306	144	64	20	2	738.9	15.1
MLA (latent, d_c=384)	203	113	56	28	12	83.9	5.4
LRKV (r=8)	566	281	141	71	35	3698.5	274.9
LRKV (r=64)	118	59	29	14	7	2562.1	160.2

methods do not reduce KV-cache size and therefore provide limited benefit for decoding-heavy LLMs where memory, rather than compute, dominates inference cost.

Sharing K/V Projections. A widely adopted strategy for reducing KV memory is to share key and value projections across attention heads. Multi-Query Attention (MQA) (Shazeer, 2019) uses a single shared set of K/V projections, and Grouped-Query Attention (GQA) (Ainslie et al., 2023) extends this by sharing within small head groups. These approaches achieve substantial memory savings but sacrifice head-level expressivity. Analyses of pretrained models (Clark et al., 2019; Michel et al., 2019) consistently show that attention heads specialize for different syntactic or semantic functions; collapsing their projections can therefore degrade modeling quality. LRKV is motivated by this tension: we preserve the efficiency of shared projections while reintroducing diversity through low-rank head-specific residuals.

Low-Rank Parameterization of Attention Weights. Our formulation is related to low-rank reparameterization techniques such as LoRA (Hu et al., 2022), which add low-rank updates to weight matrices for parameter-efficient fine-tuning. Follow-up work extends this idea to training-time regularization or improved factorization schemes (Peng et al., 2025). However, these methods intervene on the *optimization pathway*, not on the structure of stored activations. LRKV applies the same low-rank principle directly to attention projections, introducing low-rank *representational* deviations that shape the K/V features encoded in the cache.

Recent work has also explored low-rank or factored $Q/K/V$ projections to improve parameter efficiency or computational throughput (Khalaf et al., 2026). The concurrent work of Lv et al. (2024) investigates low-rank parameterizations of attention projections more broadly. These methods typically replace full-rank projections with low-rank ones to reduce computation. LRKV differs conceptually: we preserve a full-rank shared projection and add only low-rank head-specific residuals, enabling memory reduction while maintaining full-rank representational capacity.

Complementary Efficiency Methods. LRKV is orthogonal to other efficiency techniques: sparse/kernelized attention (Child et al., 2019; Beltagy et al., 2020; Wang et al., 2020) reduce computational complexity but not KV cache size; architectural optimizations like FlashAttention (Shah et al., 2024) improve throughput through better memory access patterns; quantization and pruning reduce precision or parameters. For head geometry analysis, we use gauge-invariant bilinear forms with centered Gram matrices (Appendix E).

KV-Cache Compression and Long-Context Modeling. A complementary body of research aims to compress or restructure the KV cache itself. Approaches include clustering or distilling K/V states (Chari et al., 2025), exact hybrid and compressed buffers (Yao et al., 2025), token selection or pooling (Li et al., 2024), and architectural changes that replace attention with recurrent alternatives (Sun et al., 2023). Unlike these activation-level methods, LRKV reduces the amount of information stored for every retained token by changing how each head generates its K/V features.

Comparison to Multi-Latent Attention. Multi-Latent Attention (MLA) (DeepSeek-AI et al., 2024) reduces KV cache memory by compressing token representations into a shared low-dimensional latent space before caching. During attention, interaction with the cache therefore requires projecting queries and/or values through this latent bottleneck. This achieves strong memory compression but

couples all heads through the same latent representation and restricts positional encoding choices (e.g., requiring partial RoPE).

LRKV addresses a different source of redundancy. Rather than compressing each token's hidden features into one joint latent, LRKV preserves full token-level resolution and reduces redundancy *across heads* by factorizing each head's KV projection into a shared full-rank base plus low-rank head-specific residuals. This additive structure retains the original head-feature dimensionality and supports exact full-dimensional RoPE in the integrated kernel. LRKV and MLA therefore address different sources of redundancy: MLA uses a joint hidden-feature bottleneck, whereas LRKV uses structured sharing across heads.

Factorized or Shared KV Mechanisms. Concurrent work explores structured KV-cache reduction via factorized representations (e.g., MFA/MFA-KR; TPA (Hu et al., 2024; Zhang et al., 2025)). LRKV provides a middle ground between fully shared (MQA/GQA) and fully independent projections: the full-rank shared base captures global structure, while low-rank residuals preserve head-level variability.

Positioning against adjacent factorized methods. Low-rank attention is not new, and we do not claim it is. LRKV's specific architectural claim is narrower: an *additive* decomposition of each head's projection into a shared full-rank base and a head-specific low-rank residual, with the rank interpolating continuously between complete sharing and full per-head independence, and with the cached state remaining in the original feature dimensionality so that full-dimensional RoPE applies unchanged. Table 6 separates that claim from the nearest alternatives along the three axes that matter for a cache-reduction method: what is factorized, what is actually cached, and what the parameterization implies for positional encoding.

Table 6: **Positioning of LRKV against adjacent factorized-attention methods.** We report no head-to-head experiments against MFA/MFA-KR, TPA, Palu or TransMLA; only MLA is included as a trained baseline in this paper. This table is a scope statement, not an empirical comparison.

Method	What is factorized	What is cached	Setting	Positional-encoding implication
LRKV (ours)	Per-head K/V projection = shared full-rank base + additive rank-r residual	Shared K/V features once, plus per-head rank-r latents	Trained from scratch	Exact full-dimensional RoPE; residual rotation is fused and not persisted
MLA (DeepSeek-AI et al., 2024)	Joint low-dimensional K/V compression with absorbed projections	Joint content latent, plus a decoupled RoPE key	Trained from scratch	Decoupled/partial RoPE required for absorption
MFA / MFA-KR (Hu et al., 2024)	Low-rank factorization of the QK bilinear form; KR reuses the key representation for values	Compact shared key/value state	Trained from scratch	Circuit-level factorization; positional treatment differs
TPA (Zhang et al., 2025)	Context-dependent tensor-product factors for Q, K and V	Contextual low-rank factors	Trained from scratch	RoPE integrated into the factors by design
Palu (Chang et al., 2025)	Post-training low-rank decomposition of existing projections	Compressed intermediates, full K/V reconstructed on demand	Retrofit of a trained model	RoPE-aware reconstruction with co-designed kernels
TransMLA (Meng et al., 2025)	Conversion of a GQA-style model into an MLA parameterization	MLA latent cache	Conversion plus fine-tuning	Inherits MLA's absorption constraints

The clearest split in Table 6 is by *setting*. Most of the low-rank KV literature, including Palu, TransMLA, and the post-hoc SVD family, factorizes or converts an already-trained checkpoint. Those methods inherit whatever K/V geometry pretraining happened to produce and then approximate it, which is why they need reconstruction kernels, RoPE-aware corrections, or recovery fine-tuning. LRKV instead makes the shared-plus-residual decomposition the parameterization the model is *trained* in, so the factored form is exact by construction rather than an approximation of something denser, and the rank is a design variable set before training rather than a compression ratio chosen after it. That is the distinction we claim, and it is a distinction of setting and parameterization, not of demonstrated empirical superiority.

MFA/MFA-KR and TPA are also trained from scratch, so the distinction is architectural rather than one of setting. LRKV keeps a full-rank shared base with an additive per-head residual and caches state in the original feature dimensionality, while MFA factorizes the QK circuit and TPA uses context-dependent tensor-product factors. The trained comparisons in this paper focus on MHA, GQA, MQA, and MLA, spanning full, grouped, complete-sharing, and latent-cache baselines.

Analyzing attention head diversity. Prior work has measured head redundancy using attention pattern similarity (Michel et al., 2019; Voita et al., 2019), activation-based metrics such as CKA

(Kornblith et al., 2019), or SVCCA (Raghu et al., 2017). Recent work explicitly characterizes gauge symmetries in attention parameterizations (Wang & Wang, 2025), recognizing that per-head rotations preserve attention function. We analyze the functional operators $\mathbf{W}^Q(\mathbf{W}^K)^\top$ using gauge-invariant bilinear forms and centered Gram matrices. This gives a parameterization-invariant description of operator geometry that complements activation-based metrics.

D Shared Subspaces, Spectral Bias, and Information Structure

Prior empirical analyses of pretrained Transformers have shown that attention heads exhibit substantial redundancy and occupy overlapping subspaces (Michel et al., 2019; Clark et al., 2019; Bhojanapalli et al., 2021; Rahaman et al., 2019). To assess this structure in a gauge-invariant manner, we analyze attention bilinear forms $\mathbf{A}_h = \mathbf{W}_h^Q(\mathbf{W}_h^K)^\top$ and measure head diversity via PCA on their centered Gram matrix (subsection 4.3). LRKV reaches 93.5% effective rank versus 94.0% for standard MHA at 2.5B scale, so its operator-space spread closely tracks MHA despite the shared K/V bases. The result is consistent with the additive parameterization capturing a shared component while retaining head-specific directions; it does not establish that this geometry causes the loss improvement.

Spectral bias in neural networks provides one possible interpretation of shared and residual structure (Rahaman et al., 2019; Fang & Xu, 2024): optimization often fits high-variance modes before finer residual structure. LRKV is compatible with this interpretation because each head lies in an affine family around $\mathbf{W}_{\text{shared}}^{K,V}$, with a compact learned residual subspace. Our measurements do not identify the shared branch with specific spectral modes, so we treat this connection as motivation rather than a fitted decomposition of dominant and lower-variance components.

Residual rank as a control knob for diversity. In LRKV, the residual rank r controls a continuous spectrum between fully shared keys/values and fully independent per-head projections. At a high level, the shared base $\mathbf{W}_{\text{shared}}^{K,V}$ serves as a common coordinate system capturing features that are broadly useful across heads, while the low-rank residuals provide a budgeted mechanism for head specialization. When r is small, heads are strongly coupled through the shared representation; as r grows, the model can allocate additional degrees of freedom to capture head-specific patterns. This view motivates treating r as an architectural hyperparameter with a clear interpretation: it controls cache cost and the capacity of the per-head residual. At the tested 2.5B operating point, moderate rank gives 93.5% PCA-based effective rank versus 94.0% for standard MHA (subsection 4.3) while substantially reducing cache memory.

Decomposition geometry and (non-)orthogonality. The factorization $\mathbf{U}_h\mathbf{B}_h^\top$ is non-identifiable under invertible changes of basis in its rank-r latent dimension. The objective also imposes no orthogonality constraint between the shared and residual terms. We therefore use overlap measures such as cosine similarity or subspace angles only as diagnostics of the learned decomposition, not as uniquely identified components or an optimization guarantee.

Connection to low-rank approximation. LRKV can also be interpreted as learning a structured low-rank approximation of the family of per-head projections. In classical matrix approximation, the optimal rank-r representation is given by truncated SVD with respect to a chosen error metric. LRKV instead learns shared and head-specific residual factors jointly with the network under the task loss. It makes no post-hoc approximation-optimality claim; its factors are selected by end-to-end task optimization.

Scaling, normalization, and stability. Because LRKV is additive, the norms of the shared and residual components can evolve independently during training. Post-projection RMSNorm attenuates sensitivity to absolute scale, but parameter magnitudes alone do not determine attention behavior or model quality. We therefore report them as a decomposition diagnostic alongside operator geometry and held-out loss.

E GAUGE-INVARIANT HEAD GEOMETRY VIA PCA IN BILINEAR FORM SPACE

This section presents a gauge-invariant PCA-based methodology for analyzing attention-head geometry across model scales. It combines bilinear-form comparison, centered Gram-matrix analysis, and a variance-explained interpretation of operator-space diversity.

E.1 METHODOLOGY: PCA IN THE SPACE OF BILINEAR FORMS

Gauge invariance motivation. Comparing attention heads via raw weight matrices $\mathbf{W}_h^K$ or $\mathbf{W}_h^Q$ is not meaningful because attention is invariant to coupled per-head rotations: for any orthogonal $\mathbf{R}_h$, the transformations $\mathbf{W}_h^Q \leftarrow \mathbf{W}_h^Q \mathbf{R}_h$ and $\mathbf{W}_h^K \leftarrow \mathbf{W}_h^K \mathbf{R}_h$ leave attention outputs unchanged. The gauge-invariant object is the bilinear form $\mathbf{A}_h = \mathbf{W}_h^Q (\mathbf{W}_h^K)^\top$, which determines attention logits.

Inner product on bilinear forms. We define similarity between heads i and j using the Frobenius inner product on their bilinear forms:

$$\langle \mathbf{A}_i, \mathbf{A}_j \rangle_F = \text{tr}(\mathbf{A}_i^\top \mathbf{A}_j) = \text{tr}((\mathbf{W}_i^K)^\top \mathbf{W}_j^K (\mathbf{W}_j^Q)^\top \mathbf{W}_i^Q). \tag{5}$$

Normalized by individual norms, this yields a similarity score $s_{ij} \in [-1, 1]$ forming the Gram matrix $\mathbf{G}$.

Centering for PCA. The Gram matrix $\mathbf{G}$ can be centered without materializing the mean bilinear form:

$$\mathbf{G}_{\text{centered}}[i,j] = \mathbf{G}[i,j] - \frac{1}{H}\sum_k \mathbf{G}[i,k] - \frac{1}{H}\sum_k \mathbf{G}[k,j] + \frac{1}{H^2}\sum_{k,\ell} \mathbf{G}[k,\ell]. \tag{6}$$

This is the kernel PCA centering trick (Schölkopf et al., 1998), enabling PCA in the abstract space of bilinear forms using only their inner products.

Interpretation as variance decomposition. The eigenvalues $\{\lambda_i\}$ of $\mathbf{G}_{\text{centered}}$ represent variance explained by each principal component in bilinear form space. We compute:

- **Variance explained:** $v_i = \lambda_i / \sum_j \lambda_j$ (fraction of total variance in i-th PC)
- **Cumulative variance:** $\sum_{j=1}^{k} v_j$ (variance captured by first k PCs)
- **Effective rank (PCA-based):** $\exp(-\sum_i v_i \log v_i)$ via Shannon entropy

High effective rank means that variance is spread across many operator-space directions; low effective rank means that it is concentrated in fewer principal components.

Comparison to uncentered analysis. Prior work typically uses uncentered Gram matrices, whose eigenvalues include both the mean direction and variance around the mean. Centering isolates the latter. MQA's complete KV sharing creates a dominant mean direction, while its higher centered rank is consistent with query-side differentiation around that shared baseline.

E.2 COMPLETE RESULTS ACROSS SCALES

E.3 EFFECTIVE RANK AT 128M SCALE

At 128M scale with 6 attention heads, LRKV ($r = 64$) reaches 83.0% mean centered effective rank against 82.4% for Standard MHA. The $r = 16$ configuration reaches 81.5%, while MQA reaches 78.4% despite complete KV sharing. The MQA centered-versus-uncentered pattern is consistent with query-side differentiation at this scale. Per-layer 128M rank curves omit GQA because the checkpoint set is incomplete for that method.

E.4 PCA EIGENVALUE SPECTRA

The 2.5B eigenvalue spectra provide an alternative view of head diversity, complementing the aggregate effective rank metric. The PCA-based effective rank summarizes the entire spectrum via entropy, while the plot shows the full eigenvalue distribution. Key observations:

Table 7: **PCA-based effective rank comparison.** LRKV's centered operator-space effective rank lies within one percentage point of Standard MHA at both scales. MQA's centered and uncentered values differ substantially.

	128M (6 heads)		2.5B (18 heads)	
Model	**Uncentered**	**PCA-based**	**Uncentered**	**PCA-based**
Standard MHA	95.4%	82.4%	98.9%	**94.0%**
LRKV (r=64)	96.2%	**83.0%**	98.3%	**93.5%**
GQA	–	–	95.4%	91.7%
LRKV (r=16)	88.8%	81.5%	–	–
MQA	72.6%	78.4%	86.2%	91.0%
MLA	83.9%	78.5%	92.7%	91.6%

PCA-based percentages are relative to maximum possible effective rank (number of heads). No completed 128M GQA checkpoint was supplied, so no 128M GQA geometry is reported. MQA's 4.8pp change from uncentered to PCA-based rank at 2.5B is consistent with query-side variation around a shared mean.

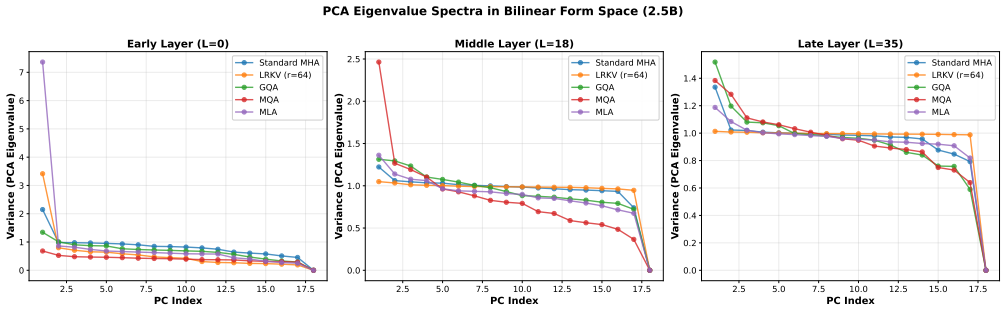


Figure 10: **PCA eigenvalue spectra in bilinear form space (2.5B).** Eigenvalues of centered Gram matrices at early, middle, and late layers show how variance is distributed across principal components. LRKV's spectra closely track Standard MHA, with similar leading eigenvalues and tail decay. MQA shows slightly elevated later eigenvalues, consistent with query-side variation after the shared mean is removed. Early layers have somewhat larger leading eigenvalues across methods, while deeper layers distribute variance more uniformly across modes.

- **LRKV tracks Standard MHA spectra:** At 2.5B scale (Figure 10), LRKV's eigenvalue curves closely track Standard MHA across early, middle, and late layers, extending the aggregate effective-rank comparison to the full spectrum.
- **MQA is sensitive to centering:** Compared with its uncentered analysis, MQA retains substantial variance in later centered components. This is consistent with query-side variation around the shared KV mean, but does not establish a behavioral compensation mechanism.
- **Depth-dependent patterns:** Early layers show slightly higher leading eigenvalues across all methods. Middle and late layers have more uniform spectra, indicating less concentration in the leading operator-space directions without assigning a behavioral interpretation.

E.5 Cumulative Variance Explained

The cumulative variance plot answers the question: "How many principal components capture most of the measured operator variation?" This complements the entropy-based effective rank.

Key findings:

- **LRKV retains broad operator-space variation:** At 2.5B, LRKV requires nearly all available PCs to capture 90% variance, closely tracking Standard MHA in this metric.
- **No dominant principal component:** The relatively linear cumulative variance curves indicate that no single PC captures disproportionate variance. Even the first PC accounts for only 15–20% of total centered variance.

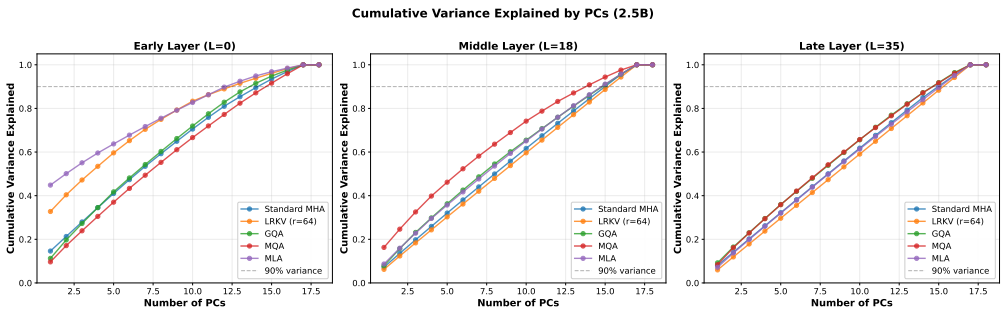


Figure 11: **Cumulative variance explained in bilinear form space (2.5B).** Plots show how many principal components are needed to capture X% of total variance at early, middle, and late layers. Standard MHA and LRKV require ∼16–17 PCs for 90% variance out of 18, so their measured operator variation spans most available directions. GQA requires ∼15–16 PCs, while MQA and MLA require ∼14–15 PCs. The 90% threshold (dashed line) provides a common descriptive dimensionality measure. All methods have relatively linear curves, with no single centered component dominating.

- **MQA's centered variance remains distributed:** Despite complete KV sharing, MQA's cumulative variance curve spreads across several PCs rather than concentrating in the first few. This is consistent with query-side variation but does not identify its behavioral effect.

E.6 Comparison: Uncentered vs PCA-based Effective Rank

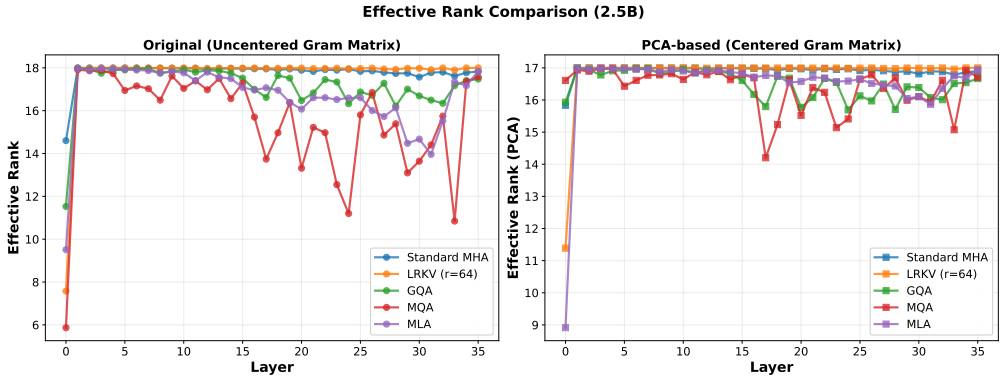


Figure 12: **Uncentered versus PCA-based effective rank at 2.5B.** The comparison is consistent across all 36 layers. Standard MHA and LRKV have high uncentered effective rank (17–18), while MQA is lower (15–16). After centering, the methods occupy a narrower range (16–17). MQA's increase is consistent with a strong shared mean plus query-side variation around that mean. LRKV closely tracks Standard MHA in both views; GQA and MLA are intermediate.

The side-by-side 2.5B comparison of uncentered versus PCA-based effective rank shows why centering changes the diagnostic:

MQA's centering sensitivity. At 2.5B scale, MQA rises from 86.2% uncentered to 91.0% centered, a 4.8 percentage point change. This quantifies how much of the uncentered concentration is removed with the shared mean. Relative to Standard MHA, the gap changes from 12.7pp uncentered to 3.0pp centered.

LRKV's consistency across metrics. LRKV's 4.8pp change from uncentered (98.3%) to centered (93.5%) effective rank is similar to Standard MHA's 4.9pp change. In this diagnostic, LRKV therefore tracks MHA both before and after mean removal.

Scale-dependent centering effects. The gaps between uncentered and PCA-based metrics are larger at 128M scale (LRKV: 13.2pp gap) than at 2.5B (4.8pp gap). Thus the diagnostic is more

sensitive to mean removal at the smaller scale. These two checkpoint sets do not identify the mechanism behind that difference.

E.7 Eigenvalue Spectra of Uncentered Similarity Matrices

For completeness, we also show the eigenvalue spectra of uncentered head similarity matrices, which have been used in prior work for head diversity analysis.

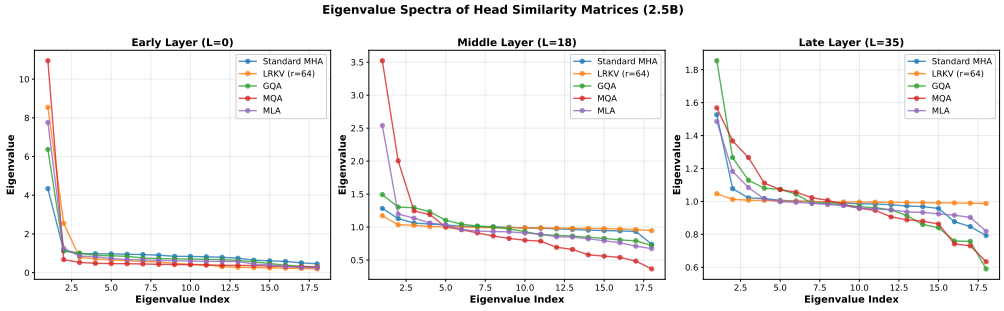


Figure 13: **Uncentered eigenvalue spectra at early, middle, and late layers (2.5B scale).** The eigenvalue distribution of uncentered head similarity matrices $\mathbf{S}$ shows broader separation between methods than the centered analysis. LRKV closely tracks Standard MHA across depth. MQA has elevated leading eigenvalues and faster tail decay, indicating stronger correlation in the uncentered view; much of the separation narrows after the shared mean direction is removed. Early layers have slightly higher leading eigenvalues across methods, while deeper layers distribute variance more uniformly across modes.

E.8 Head Similarity Heatmaps

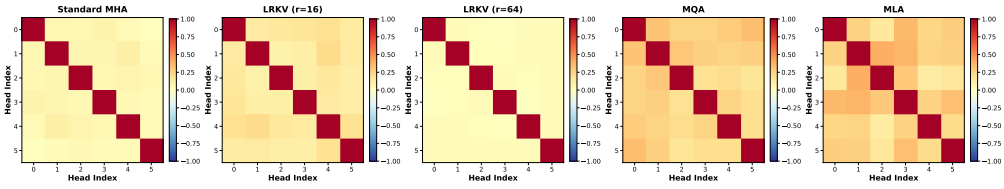


Figure 14: **Head similarity matrices at 128M scale.** With fewer heads (6 versus 18), individual head relationships are more visible. LRKV (r=64) closely tracks Standard MHA, with moderate positive correlations but no dominant clusters. LRKV (r=16) has slightly stronger correlations, consistent with its reduced effective rank. The mean off-diagonal similarity for LRKV (r=64) is within 0.02–0.04 of Standard MHA across layers. These uncentered heatmaps show the raw pairwise similarities before mean removal.

The heatmaps provide direct visualization of head relationships in the uncentered similarity space, complementing the PCA-based aggregate metrics. Key observations:

- **LRKV tracks MHA's correlation structure:** At both scales, LRKV's similarity matrices closely match Standard MHA's patterns. Off-diagonal entries have comparable magnitudes and spatial distributions in the analyzed checkpoints.
- **No strong block structure:** Standard MHA and LRKV both show diffuse off-diagonal correlations rather than clear clusters before centering.
- **GQA shows group structure.** At 2.5B scale with 6 KV groups of 3 heads each, GQA's heatmap has visible within-group structure and stronger similarity among heads that share KV projections.
- **Quantitative confirmation:** The mean off-diagonal similarity for LRKV (r=64) is within 0.02–0.04 of Standard MHA across layers, confirming the visual similarity is not merely qualitative. This tight correspondence holds in both uncentered and PCA-based analysis.

E.9 Key Findings from PCA Analysis

1. LRKV tracks MHA's centered effective rank. At 2.5B scale, LRKV reaches 93.5% PCA-based effective rank versus 94.0% for Standard MHA, a 0.5pp gap, while using 55.6% of MHA's KV cache. The pattern is consistent across the 36 analyzed layers.

2. MQA is sensitive to centering. MQA's 4.8pp increase from uncentered (86.2%) to centered (91.0%) effective rank at 2.5B indicates that the shared mean accounts for much of its uncentered concentration. The analogous change at 128M is 5.8pp. These patterns are consistent with query-side variation but do not by themselves establish a causal compensation mechanism.

3. MLA sits between GQA and MQA in diversity space. MLA reaches 91.6% PCA-based effective rank at 2.5B, between GQA (91.7%) and MQA (91.0%). At 128M, MLA (78.5%) and MQA (78.4%) are effectively tied in this metric.

4. Scale-dependent behavior. The gap between uncentered and PCA-based metrics shrinks from 128M to 2.5B (e.g., LRKV: 13.2pp at 128M versus 4.8pp at 2.5B). This shows a scale-dependent difference in the effect of centering across the analyzed checkpoints, without establishing a developmental mechanism.

5. Rank tracks geometry and quality. At 128M, r=16 (12.5% of head dimension) achieves 81.5% PCA-based effective rank, while r=64 (50%) achieves 83.0%. This 1.5pp difference accompanies a performance difference of 0.870 versus 0.862 BPB. The result is consistent with residual capacity contributing to quality, although the diversity checkpoints and endpoint sweep do not establish causality.

E.10 Comparison to Prior Diversity Metrics

Our PCA-based approach differs from prior work in several key respects:

CKA/SVCCA (Kornblith et al., 2019; Raghu et al., 2017). These methods compare activation similarities, not functional operators. While CKA uses centered Gram matrices, it analyzes activation space rather than weight-space bilinear forms. We instead apply centering to attention operators to separate their mean direction from residual operator-space variation.

Attention pattern similarity (Michel et al., 2019; Voita et al., 2019). These behavioral metrics measure which tokens heads attend to on specific inputs and depend on the input distribution. Our approach instead provides a data-independent description of parameter-space operator geometry; it is not a substitute for behavioral analysis.

Raw weight comparison. Directly comparing $\mathbf{W}_h^K$ or $\mathbf{W}_h^Q$ matrices is not gauge-invariant: arbitrary per-head rotations that preserve attention function can make identical heads appear different or vice versa. The bilinear form $\mathbf{W}^Q(\mathbf{W}^K)^\top$ is the minimal gauge-invariant object for comparison.

Uncentered Gram matrices. Several prior analyses (Clark et al., 2019; Bhojanapalli et al., 2021) form head similarity matrices (often using uncentered correlations or kernel similarities) and interpret their spectra directly. We apply explicit kernel centering in the space of gauge-invariant attention operators, separating shared mean structure from variance around the mean. Without centering, mean structure and spread around the mean contribute to the same spectrum. The centered analysis reports the latter separately and does not by itself identify behavioral specialization.

Together, gauge-invariant bilinear forms, centered Gram matrices, and a variance-explained interpretation provide an operator-space view of head geometry that is complementary to activation-based analyses.

F EXPERIMENTAL SETUP (FULL DETAILS)

F.1 CACHE ACCOUNTING AND MATCHED-BUDGET COMPARISON.

Because MLA stores a single joint K/V latent, we count its cache once rather than separately for keys and values. All cache percentages in this paper use the same denominator, the standard MHA cache $2LHd_h$. Under this convention, LRKV has cache ratio $\frac{M_{\text{LRKV}}}{M_{\text{MHA}}} = \frac{d_h+Hr}{Hd_h} = \frac{1}{H} + \frac{r}{d_h}$, where the first term is the shared full-rank K/V cache and the second is the per-head low-rank latent cache. Thus low-rank LRKV has a nonzero shared-cache floor. At 2.5B scale (H=18, d_h=128) each LRKV budget in Table 1 is matched by an MLA with d_c set to the same ratio: LRKV reaches 0.7215 versus MLA's 0.7240 BPB at 8.3% (d_s=48, r=8), 0.7167 versus 0.7228 at 11.8% (r=8), 0.7145 versus 0.7200 at 18.1% (r=16), and 0.7110 versus 0.7192 at 55.6% (r=64). LRKV therefore traces a favorable memory–quality frontier: aggressive ranks reach MLA-like cache budgets with lower loss, while moderate ranks exceed MHA quality with substantial KV-cache savings.

Table 8: **Mathematical comparison of attention mechanisms.** We compare how different mechanisms parameterize key/value projections and where compression is applied. Here $X \in \mathbb{R}^{T\times d}$ is the token sequence, $d_h = d/H$ is the head dimension, H is the number of heads, and $r, d_c \ll d_h$.

Method	KV projection form	Rank constraint	Compression axis
Standard MHA	$W_h^{K,V} \in \mathbb{R}^{d\times d_h}$ (independent per head)	None (full rank)	None
MQA	$W_h^{K,V} = W_{\text{shared}}^{K,V}\ \forall h$	$\text{rank}(W_h^{K,V}) = \text{rank}(W_{\text{shared}})$	Across heads (complete sharing)
GQA	$W_h^{K,V} = W_{g(h)}^{K,V}$ for group $g(h)$	Full rank within group	Across heads (group-wise sharing)
MLA	$W_h^{K,V} = W_{\text{down}} W_{\text{up},h}^{K,V}$	$\text{rank}(W_h^{K,V}) \le d_c$	Within each token's hidden features (latent bottleneck)
LRKV (ours)	$W_h^{K,V} = W_{\text{shared}}^{K,V} + U_h^{K,V}(B_h^{K,V})^\top$	$\text{rank}(W_h^{K,V} - W_{\text{shared}}^{K,V}) \le r$	Across heads (additive low-rank deviations)

F.2 PRETRAINING CONFIGURATION

We pretrain from scratch a family of decoder-only Transformer language models with parameter counts ranging from **128M to 6.3B** on the **FineWeb-Edu** dataset (Penedo et al., 2024). Unless otherwise stated, model scale (e.g., 128M, 2.5B, 6.3B) refers to the parameter count of the *base architecture with standard multi-head attention (MHA)*. Alternative attention mechanisms replace only the attention module while keeping all other architectural components fixed. Their attention-specific parameter counts follow their respective formulas and can therefore differ from the MHA backbone count.

MLA baseline implementation. Our MLA baseline follows the DeepSeek-style latent design (DeepSeek-AI et al., 2024). Each layer projects the input to a joint K/V latent $\mathbf{Z} = \mathbf{X}\mathbf{W}_{\text{down}} \in \mathbb{R}^{L\times d_c}$ with a single bias-free linear map and caches only $\mathbf{Z}$; per-head keys and values are recovered by separate up-projections $\mathbf{K}_h = \mathbf{Z}\mathbf{W}_{\text{up},h}^K$ and $\mathbf{V}_h = \mathbf{Z}\mathbf{W}_{\text{up},h}^V$ to the full head width d_h=128, and queries use a full-rank projection. RoPE is partial: the first 32 of the 128 dimensions of every query and key are rotated and the remaining 96 carry no positional encoding, so the rotated slice does not block absorbing the key up-projection into the query at decode time. Queries and keys are RMS-normalized before the dot product, as in every other variant. The latent width d_c is the only architectural degree of freedom and is set to the cache budget of each tier in Table 1; the MLP, embeddings, optimizer, schedule, and data are shared with the other variants, and the learning rate is grid-searched per method with the same 5B-token protocol (see Optimization Details below). At 128M the latent-width ablation improves monotonically with d_c (2.901 to 2.891 CE from 8.3% to 50% cache, Table 13). The decode benchmarks evaluate this parameterization through two paths: a latent form that up-projects the cached latent at every step, and an absorbed form with decoupled per-head RoPE keys (Appendix I.5).

Model Architecture Summary. Table 9 provides complete architectural specifications for all model scales evaluated in this work.

Training Configuration Summary. Table 10 provides complete training configuration details across all model scales.

Table 9: **Complete model architectural specifications.** All models use $d_{\text{head}} = 128$ and FFN expansion ratio of 4 (i.e., $d_{\text{FFN}} = 4 \times d_{\text{model}}$).

Scale	Layers	Heads	d_{model}	d_{FFN}	Vocab Size
128M	12	6	768	3072	50304
512M	16	12	1536	6144	50304
1.2B	24	12	1536	6144	50304
2.5B	36	18	2304	9216	50304
6.3B	32	32	4096	16384	50304

Table 10: **Training configuration across model scales.** Batch size refers to number of sequences; total tokens per batch = batch size $\times$ context length. All models trained on FineWeb-Edu with Muon+AdamW optimization.

Scale	Context	Batch Size	Tokens/Batch	Total Tokens	Compute (EF)	Purpose
128M	2048	16	32,768	100B	31.0	Ablations
512M	8192	4	32,768	50B	153.6	Long context
1.2B	2048	8 (GA=2)	32,768	50B	304.7	Scaling study
2.5B	2048	8 (GA=2)	32,768	50B	635.2	Main experiments
6.3B	2048	8 (GA=2)	32,768	50B	1603.3	Scaling study

GA=gradient accumulation steps. EF=ExaFLOPs. The 128M sweep uses a common 100B-token budget for late-training ablations; other runs use a fixed 50B-token budget.

Optimization Details. For each model scale and attention mechanism, we select the learning-rate setting with a 5B-token grid search before the full pretraining run. All non-learning-rate optimizer choices and the search horizon are shared across methods. We use a hybrid optimizer strategy: the **Muon optimizer** (Jordan et al., 2024) for Transformer weight matrices and **AdamW** for embeddings, with component-specific reference learning rates (matrix: 0.02, embedding: 0.2, unembedding: 0.004). All models use cosine learning rate decay with linear warmup (2000 steps), weight decay 0.1, and gradient clipping at 1.0.

Hardware and Precision. All experiments are run on a single **8$\times$H200 GPU** node in mixed-precision (bfloat16) mode. Batching and optimizer hyperparameters are provided in Appendix H.

Evaluation Metrics. Training progress is monitored using the **cross-entropy loss** and **bits-per-byte (BPB)** on held-out validation split of FineWeb-Edu. These metrics allow consistent comparison across model scales and serve as the primary indicators of data efficiency and convergence quality.

F.3 Mid-training Configuration

After pretraining, we perform **supervised fine-tuning** (mid-training) on a curated mixture of instructional data to improve downstream task performance. The mid-training dataset consists of three components:

1. **SmolTalk** (Ben Allal et al., 2025) with 460K conversational examples for general instruction-following
2. **MMLU auxiliary train** (Hendrycks et al., 2021) with 100K multiple-choice questions spanning diverse academic subjects
3. **GSM8K** (Cobbe et al., 2021) with 8K grade-school math problems including calculator tool use

This yields a total training mixture of approximately **568K examples**. A held-out validation mixture, disjoint from the midtraining examples, uses proportional sampling (24K SmolTalk, 5.2K MMLU, and 420 GSM8K examples).

We train for **one full epoch** over the mid-training mixture with a batch size of **524,288 tokens** and sequence length of 2048. We use a decayed linear schedule for the learning rate, using AdamW

for the embeddings and Muon for the remaining parameters. Models are evaluated every 150 steps using bits-per-byte on the validation set. All mid-training runs use the same hyperparameters across different attention mechanisms to ensure fair comparison.

F.4 Attention Mechanism Configurations

To evaluate the effectiveness of low-rank KV factorization, we compare LRKV against four baseline attention mechanisms across all model scales. Table 11 provides complete configuration details for all methods and scales.

Table 11: **Attention mechanism configurations across model scales.** KV cache percentages are relative to Standard MHA. All models use $d_{\text{head}} = 128$ and 2048-token context (except 512M long-context with 8192 tokens).

	Standard			MQA / GQA			MLA / LRKV		
Scale	**Heads**	**KV Heads**	**Cache**	**MQA KV**	**GQA KV**	**GQA Cache**	**MLA d_c**	**LRKV r**	**LRKV Cache**
128M	6	6	100%	1 (16.7%)	3 (50%)	50%	128 (8.3%)	64	66.7%
512M	12	12	100%	1 (8.3%)	4 (33.3%)	33.3%	256 (8.3%)	51	48.2%
1.2B	12	12	100%	1 (8.3%)	4 (33.3%)	33.3%	256 (8.3%)	51	48.2%
2.5B	18	18	100%	1 (5.6%)	6 (33.3%)	33.3%	384 (8.3%)	64	55.6%
6.3B	32	32	100%	1 (3.1%)	2 (6.3%)	6.3%	704 (8.6%)	64	53.1%
512M-8K[†]	12	12	100%	1 (8.3%)	4 (33.3%)	33.3%	256 (8.3%)	51	48.2%

[†]512M-8K denotes the long-context configuration with 8192-token sequences. All other models use 2048-token context.

KV Cache Calculation. All KV cache percentages are reported relative to Standard MHA's memory usage. For default LRKV, the reported cache size assumes an optimized implementation that caches a width-d_h shared state and per-head low-rank latents $R_h^{K,V}$ rather than the fully reconstructed key-value matrices, following the theoretical analysis in Section 3. The ratio is $\frac{d_h+Hr}{Hd_h} = \frac{1}{H} + \frac{r}{d_h}$. The 8.3% ablation point is a distinct reduced-shared-width configuration with d_s=16 and r=8, whose ratio is $\frac{d_s+Hr}{Hd_h} = \frac{16+6\cdot 8}{6\cdot 128} = 8.33\%$.

These operating points test whether low-rank head-specific residuals can recover quality while retaining a substantially smaller cache than Standard MHA. Equal-cache evidence is reported separately in Table 3.

K/V parameter counts. Table 12 lists the K/V projection parameters per layer implied by each configuration in Table 1.

Table 12: **K/V projection parameters per layer (millions) for the configurations of Table 1.** Computed from the parameterizations: MHA $2Hdd_h$; GQA $2H_{\text{kv}}dd_h$; MQA $2dd_h$; MLA $dd_c + 2d_cHd_h$ (down- and up-projections); LRKV $2dd_s + 2Hr(d+d_h)$. LRKV has fewer K/V parameters than MHA and than its cache-matched MLA counterpart at every scale and tier, so its lower loss is not explained by a larger K/V parameter allocation.

Method / tier	**128M**	**1.2B**	**2.5B**	**6.3B**
Standard MHA	1.18	4.72	10.62	33.55
GQA (Table 1 heads)	0.59	1.57	3.54	2.10
MQA	0.20	0.39	0.59	1.05
MLA, 8.3% tier	0.29	1.18	2.65	8.65
LRKV, 8.3% tier	0.11	0.42	0.92	2.95
MLA, 25% / 11.8% tier	0.88	3.54	3.76	11.80
LRKV, 25% / 11.8% tier	0.31	1.23	1.29	4.02
MLA, 50% / 18.1% tier	1.77	7.08	5.75	18.09
LRKV, 50% / 18.1% tier	0.66	2.51	1.99	6.18
MLA, default tier	2.36	6.82	17.69	53.48
LRKV, default tier	0.88	2.43	6.19	18.35

F.5 RANK ABLATION EXPERIMENTS

To evaluate how residual rank affects LRKV's performance-memory tradeoff, the default full-width 128M sweep changes rank while holding the remaining architecture and training recipe fixed. A separate reduced-shared-width point tests the 8.3% cache regime.

LRKV Rank Ablation. We train 128M parameter models on 100B tokens of FineWeb-Edu with default full-width LRKV ranks $r \in \{8, 11, 16, 32, 43, 64, 128\}$, plus a reduced-shared-width d_s=16, r=8 point at 8.3% cache. All models use 6 heads, $d_{\text{model}} = 768$, and $d_{\text{head}} = 128$. Table 13 presents the complete scalar endpoints supplied for this sweep.

Table 13: **LRKV rank/shared-width ablation and MLA latent-dimension comparison (128M scale, 100B tokens).** All models use 6 heads with $d_h = 128$. Cache percentages are relative to Standard MHA; MLA uses count-once joint-latent accounting.

Configuration	Rank/Latent	KV Cache	CE Loss ↓	BPB ↓	Notes
LRKV Rank Ablation					
LRKV-d_s16-r8	$d_s = 16, r = 8$	8.3%	2.891	0.873	Reduced shared width
LRKV-8	$r = 8$	22.9%	2.891	0.873	Default shared width
LRKV-11	$r = 11$	25.3%	2.889	0.871	25% cache match
LRKV-16	$r = 16$	29.2%	2.886	0.870	–
LRKV-32	$r = 32$	41.7%	2.884	0.868	–
LRKV-43	$r = 43$	50.3%	2.880	0.864	50% cache match
LRKV-64	$r = 64$	66.7%	2.862	0.862	Standard 128M point
LRKV-128	$r = 128$	116.7%	2.859	0.860	>100% cache
MLA Latent Dimension Ablation					
MLA-128	$d_c = 128$	8.3%	2.901	0.876	Minimal memory
MLA-192	$d_c = 192$	12.5%	2.898	0.876	–
MLA-384	$d_c = 384$	25.0%	2.894	0.875	25% cache match
MLA-768	$d_c = 768$	50.0%	2.891	0.873	50% cache match
Baselines					
Standard MHA	6 KV heads	100%	2.903	0.878	Baseline
GQA	3 KV heads	50.0%	2.907	0.880	–
GQA	2 KV heads	33.3%	2.918	0.883	–
MQA	1 KV head	16.7%	2.929	0.886	Complete sharing

Key Findings. The ranks used in the main experiments are fixed, scale-specific configurations chosen to trace the memory–quality frontier; they are not tuned post hoc per row. Performance improves monotonically across the default full-width rank sweep. The r=8 model reaches 2.891 CE / 0.873 BPB and r=64 reaches 2.862 / 0.862 while using 66.7% of MHA's cache. The final r=64 to r=128 CE gap is only 0.003 across three-seed means, and r=128 exceeds MHA's cache, so it is not a preferred compression operating point.

The exact or near-exact LRKV–MLA matches all favour LRKV in the observed endpoints. At 8.3% cache, reduced-shared-width LRKV reaches 2.891/0.873 against MLA-128's 2.901/0.876. At the 25% target, LRKV-r11 (25.3%) reaches 2.889/0.871 against MLA-384's 2.894/0.875. At the 50% target, LRKV-r43 (50.3%) reaches 2.880/0.864 against MLA-768's 2.891/0.873. LRKV also improves on the fixed baselines with no more cache: the 8.3% variant beats MQA at 16.7%, r=43 beats GQA at essentially 50%, and r=64 beats full-cache MHA.

These 100B-token entries are means over three random seeds. Per-configuration seed ranges and standard deviations are reported in Appendix B.

All ablation models are trained for 100B tokens rather than a compute-optimal allocation. This common, long budget tests whether the ordering persists late in training.

F.6 LONG CONTEXT EXPERIMENTS

To evaluate LRKV's effectiveness at extended sequence lengths, we conduct 8192-token context experiments using 512M parameter models trained on 50B tokens (153.6 ExaFLOPs).

Configuration. The 512M models use 16 layers, 12 attention heads, $d_{\text{model}} = 1536$, and $d_{\text{head}} = 128$. We compare LRKV, GQA (4 groups, 33.3% cache), MLA ($d_c = 256$, 8.3% cache), and MQA (8.3% cache) against Standard MHA.

LRKV cache budget. The long-context LRKV model uses r=51. With H=12 and d_h=128, its cache ratio is $\frac{1}{H} + \frac{r}{d_h} = \frac{1}{12} + \frac{51}{128} = 48.2\%$ of MHA's cache. The table below compares each architecture at its tested operating point.

Batching Strategy. To maintain computational throughput while accommodating 4× longer contexts, we adjust batch size to keep total tokens per batch constant. Standard 2048-context runs use batch size 16 (32,768 tokens per batch); 8192-context runs use batch size 4 (32,768 tokens per batch). This ensures comparable gradient noise and optimizer dynamics while scaling to long contexts.

Results. Table 14 presents validation performance at 8192-token context length after 50B tokens (153.6 ExaFLOPs).

Table 14: **Long context performance (512M models, 8K context, 50B tokens).** Relative change is computed against the MHA baseline; negative values denote lower loss.

Method	KV Cache	CE Loss ↓	BPB ↓	vs. MHA
Standard MHA	100%	2.740	0.494	baseline
LRKV	48.2%	**2.665**	**0.481**	**-2.7%**
GQA	33.3%	2.704	0.488	-1.3%
MLA	8.3%	2.714	0.489	-0.9%
MQA	8.3%	2.728	0.492	-0.4%

LRKV achieves the lowest loss while using 48.2% of MHA's cache, outperforming Standard MHA by 2.7%. It also improves on GQA, MLA, and MQA at their tested operating points. The consistent ordering extends LRKV's pretraining advantage from 2K to the 8K training window.

Loss is not retrieval, so we probe retrieval directly. A controlled key/value retrieval probe over the same archived 8K checkpoints tests whether the loss advantage above reflects usable long-context behaviour. The task presents eight candidate records (chance 12.5%), 128 examples per condition and 640 in total, evaluated with the exact factored path.

Table 15: **Controlled 8K key/value retrieval, 512M archived checkpoints.** Eight candidates, chance accuracy 12.5%, and 128 examples per condition. Values are descriptive checkpoint accuracies.

Condition	MHA (100% cache)	LRKV r=51 (48.2%)
Needle at 10% depth	**14.1%**	10.2%
Needle at 50% depth	58.6%	**59.4%**
Needle at 90% depth	**100.0%**	99.2%
Immediate/recent record	85.9%	**97.7%**
Absent-record control	**13.2%**	12.9%

LRKV matches MHA at mid-context depth, with 59.4% against 58.6%, and improves recent-record retrieval from 85.9% to 97.7%. Together with the lower 8K validation loss, these descriptive checkpoint results provide evidence that the 48.2%-cache model retains useful behavior within its trained 8K window.

RULER at 32K. We also evaluate the 8K-trained 512M models on RULER at 32K context, four times the training context, without any long-context fine-tuning. LRKV is best overall and in every task category (Figure 4, right, in the main text); MQA, whose cache is smallest, degrades most.

F.7 Downstream Evaluation Tasks

After midtraining, we evaluate all models on five standard benchmarks:

- **ARC-Easy and ARC-Challenge** (Clark et al., 2018): Question answering requiring scientific reasoning
- **MMLU** (Hendrycks et al., 2021): Multi-task language understanding across 57 subjects
- **GSM8K** (Cobbe et al., 2021): Grade-school math word problems
- **HumanEval** (Chen et al., 2021): Python code generation from docstrings

All models use the same evaluation harness and per-task prompting configuration. Exact templates and shot counts will accompany the evaluation code release.

G Pretraining-Downstream Correlation Analysis

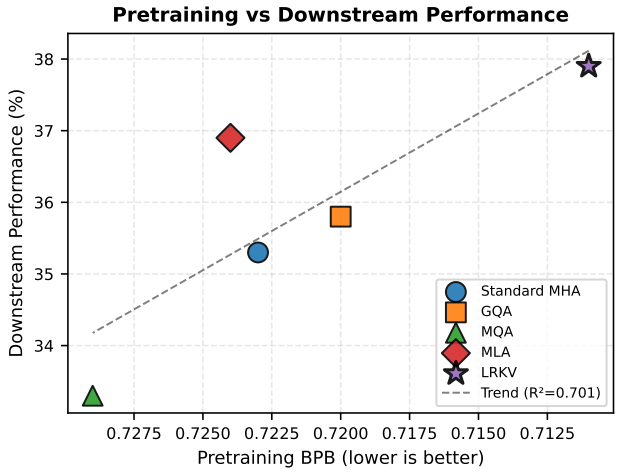


Figure 15: **Descriptive pretraining–downstream association at 2.5B.** Pretraining validation BPB (x-axis, inverted) and downstream combined accuracy for five operating points. The fitted line gives $R^2 = 0.701$ on the Table 1 BPB values (GQA with 6 KV heads, MLA at d_c=384, LRKV r=64). With n=5 and no intervention on BPB, this is a consistency check rather than evidence of causation.

Across the five architectures, lower validation BPB is associated with higher combined benchmark accuracy after the common midtraining recipe (Figure 15). LRKV occupies the endpoint with 0.711 BPB and 37.9% combined accuracy, while Standard MHA reaches 0.723 BPB and 35.3%. The association is descriptive: five architecture-level points cannot identify whether pretraining BPB causes the downstream difference, and the rounded table values give a slightly different fitted R^2 than the underlying full-precision inputs.

H Additional Experimental Details and Computational Analysis

H.1 Batching and Optimization Hyperparameters

For the 128M models, we use a global batch size of **16** sequences (2048 tokens each). For the 1.2B–6.3B models, we use a physical batch size of **8** sequences and **gradient accumulation of 2 steps** to maintain an effective batch size of 16. The separate 512M long-context runs use 4 sequences of 8192 tokens, keeping 32,768 tokens per batch. This setup ensures comparable gradient noise scale and optimizer dynamics across all model sizes. All runs employ weight decay of 0.1, Adam $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping at 1.0.

H.2 Memory-Computation Trade-offs in Attention Mechanisms

Modern attention mechanisms present a fundamental trade-off between cache memory (storage) and per-token computation (latency). We analyze this trade-off across standard attention, query-sharing variants, and compression-based approaches.

Cached-state parameterizations. The compared mechanisms differ in which representation is stored and where projection work occurs:

- **Multi-Latent Attention (MLA)** uses a learned latent bottleneck $Z = XW_{\text{down}}$, caching only Z. During generation, full K/V need not be explicitly materialized; instead, interaction with the cached latent requires per-step projection work, e.g., $qK^\top = (qW_{\text{up}}^K)Z^\top$ and $aV = (aZ)W_{\text{up}}^V$. This introduces additional per-token computation proportional to the latent dimension, and scales linearly with sequence length through dot products with $Z^\top$.
- **LRKV** stores shared full-rank features $(\bar{K}, \bar{V})$ and compact per-head latents (R_h^K, R_h^V). It avoids explicitly materializing per-head K_h, V_h tensors; instead, attention is computed using the associative forms in Equations 3 and 4. Relative to direct full-K/V attention, LRKV adds a low-dimensional residual scan $O(HTr)$ and small projection terms $O(Hrd_h)$, where $r \ll d_h$.

Both MLA and LRKV incur an $O(T)$ interaction with cached states. For LRKV, the additional residual scan is $O(HTr)$ plus T-independent projection work $O(Hrd_h)$. For absorbed MLA, the corresponding scan follows the joint latent and any separate RoPE residual widths.

Quantitative Analysis: Memory vs. Decoding Work. We compare the cache footprint and per-token decoding arithmetic of different attention mechanisms during autoregressive generation. Let T denote the cached sequence length. All attention mechanisms incur an unavoidable sequence-length dependent scan over cached states; the relevant distinction is the dimensionality of the cached states being scanned and whether full per-head K/V tensors must be materialized. Table 16 therefore reports both the cache size stored per token and the dominant T-dependent arithmetic term for one generation step. Constant factors for the query–key and value-aggregation passes are omitted.

Table 16: Memory and per-token decoding characteristics of attention mechanisms. Cache size is measured in floating-point elements per token per layer. The T-dependent work column reports the dominant cached-sequence interaction for one decoding step, omitting constant factors. Projection work denotes additional T-independent per-step projections used by compressed mechanisms.

Method	Cache / Token	T-dependent work / Step	Projection work / Step
Standard Attn.	$2Hd_h$	$O(HTd_h)$	$O(1)$
MQA	$2d_h$	$O(HTd_h)$	$O(1)$
GQA (H_{kv}=G)	$2Gd_h$	$O(HTd_h)$	$O(1)$
MLA	$d_c + d_R$	$O(HT(d_c + d_R))$	$O(Hd_cd_h)$
LRKV	$2(d_h + Hr)$	$O(HT(d_h + r))$	$O(Hrd_h)$

For MLA, d_c denotes the cached latent width and d_R denotes any separately cached RoPE residual. The table assumes optimized MLA-style projection absorption; a naive latent-to-full-K/V expansion over the whole cache would introduce a much larger $O(HTd_cd_h)$ reconstruction cost and is not the intended implementation. For LRKV, the additional work relative to direct full-K/V attention is the low-dimensional residual interaction $O(HTr)$ plus the small projection terms $O(Hrd_h)$; LRKV avoids materializing full per-head K/V tensors.

Key observations.

1. Standard attention, MQA, and GQA directly cache the key/value tensors used by attention. MQA and GQA reduce memory bandwidth by sharing K/V tensors, but each query head still forms its own attention weights, so the dominant arithmetic remains proportional to $O(HTd_h)$.
2. MLA stores a single compressed latent cache rather than separate full per-head K/V tensors. With projection absorption, MLA need not explicitly reconstruct full K/V over the

cached sequence; instead, attention interacts with the latent cache, giving T-dependent work proportional to the cached latent width d_c plus any RoPE residual width d_R.

3. LRKV stores full-rank shared K/V features and low-rank per-head residual latents. During decoding, it computes the exact logits and value outputs using the associative forms in Equations 3 and 4. This adds a low-dimensional residual scan $O(HTr)$ and projection terms $O(Hrd_h)$, rather than reconstructing full per-head K/V tensors.
4. Thus LRKV should not be described as having sequence-length-independent overhead. Its extra residual interaction is T-dependent, but the scanned residual dimension is $r \ll d_h$, so the additional scan is small relative to the standard $O(HTd_h)$ attention interaction while preserving head-specific expressivity.

I KV Cache Accounting

This appendix gives the cache-size accounting used throughout the paper. We report most results as percentages relative to Standard MHA because these ratios are independent of batch size, context length, number of layers, and numerical precision, as long as the same assumptions are used for all methods.

I.1 Notation and Cache Formulas

Let N_ℓ denote the number of Transformer layers, B the batch size, T the cached sequence length, H the number of query heads, H_{kv} the number of KV heads for GQA, d_h the head dimension, r the LRKV residual rank, d_c the MLA compressed latent dimension, and d_R the dimension of any separately cached MLA RoPE residual. Let b_{elem} denote bytes per element; for bfloat16, $b_{\text{elem}} = 2$.

Standard MHA caches both keys and values for every head:

$$M_{\text{MHA}} = 2N_\ell BTHd_h\, b_{\text{elem}}.$$

MQA caches one shared key and one shared value per layer:

$$M_{\text{MQA}} = 2N_\ell BTd_h\, b_{\text{elem}}.$$

GQA caches one key and one value for each KV group/head:

$$M_{\text{GQA}} = 2N_\ell BTH_{\text{kv}}d_h\, b_{\text{elem}}.$$

MLA caches a single joint compressed KV latent, plus any separately cached RoPE residual:

$$M_{\text{MLA}} = N_\ell BT(d_c + d_R)\, b_{\text{elem}}.$$

LRKV caches shared key/value features plus key/value low-rank residual latents:

$$M_{\text{LRKV}} = 2N_\ell BT(d_h + Hr)\, b_{\text{elem}}.$$

Dividing each expression by the Standard MHA cache gives:

$$\frac{M_{\text{MQA}}}{M_{\text{MHA}}} = \frac{1}{H}, \qquad \frac{M_{\text{GQA}}}{M_{\text{MHA}}} = \frac{H_{\text{kv}}}{H},$$

$$\frac{M_{\text{MLA}}}{M_{\text{MHA}}} = \frac{d_c + d_R}{2Hd_h}, \qquad \frac{M_{\text{LRKV}}}{M_{\text{MHA}}} = \frac{d_h + Hr}{Hd_h} = \frac{1}{H} + \frac{r}{d_h}.$$

The factor of 2 in the MHA, MQA, GQA, and LRKV formulas accounts for separately cached key and value states. Optimized MLA stores a single joint latent, so it does not receive this leading key/value factor. In the main tables we set $d_R = 0$ unless otherwise stated; including a separately cached RoPE residual increases the MLA ratio by $d_R/(2Hd_h)$.

The lowest-cache LRKV tier uses a reduced shared cached width d_s ($16/32/48/96$ at 128M/1.2B/2.5B/6.3B) rather than the default $d_s = d_h$. Its ratio is $(d_s + Hr)/(Hd_h)$; with $d_s = 16$, $r = 8$, $H = 6$, and $d_h = 128$, this is 8.33%. This is a distinct configuration from the default full-width shared branch and is labelled explicitly wherever reported.

Table 17: **KV cache and latent cache structure by attention mechanism.** All sizes are per model cache, including all layers.

Method	Cached state	Total memory
Standard MHA	K, V each: (N_ℓ, B, T, H, d_h)	$2N_\ell BTHd_h\, b_{\text{elem}}$
MQA	K, V each: $(N_\ell, B, T, 1, d_h)$	$2N_\ell BTd_h\, b_{\text{elem}}$
GQA	K, V each: $(N_\ell, B, T, H_{\text{kv}}, d_h)$	$2N_\ell BTH_{\text{kv}}d_h\, b_{\text{elem}}$
MLA	Latent: (N_ℓ, B, T, d_c) Optional RoPE residual: (N_ℓ, B, T, d_R)	$N_\ell BT(d_c + d_R)\, b_{\text{elem}}$
LRKV	Shared K, V each: (N_ℓ, B, T, d_h) Residual latents R^K, R^V each: (N_ℓ, B, T, H, r)	$2N_\ell BT(d_h + Hr)\, b_{\text{elem}}$

I.2 Cache Structure by Method

Table 17 summarizes the cached states implied by these formulas.

I.3 Scale-Specific Cache Ratios

Table 18 reports the cache ratios for the main configurations used in the paper. These are the ratios of the standard operating points in Table 11; the matched tiers of Table 1 follow from the same formulas. All models use $d_h = 128$.

Table 18: **Cache ratios for the main configurations.** Percentages are relative to Standard MHA. MLA ratios use $d_R = 0$ unless otherwise stated.

Scale	H	MQA	GQA	MLA	LRKV rank	LRKV cache
128M	6	$1/6 = 16.7\%$	$3/6 = 50.0\%$	$128/(2 \cdot 6 \cdot 128) = 8.3\%$	$r = 64$	$1/6 + 64/128 = 66.7\%$
512M	12	$1/12 = 8.3\%$	$4/12 = 33.3\%$	$256/(2 \cdot 12 \cdot 128) = 8.3\%$	$r = 51$	$1/12 + 51/128 = 48.2\%$
1.2B	12	$1/12 = 8.3\%$	$4/12 = 33.3\%$	$256/(2 \cdot 12 \cdot 128) = 8.3\%$	$r = 51$	$1/12 + 51/128 = 48.2\%$
2.5B	18	$1/18 = 5.6\%$	$6/18 = 33.3\%$	$384/(2 \cdot 18 \cdot 128) = 8.3\%$	$r = 64$	$1/18 + 64/128 = 55.6\%$
6.3B	32	$1/32 = 3.1\%$	$2/32 = 6.3\%$	$704/(2 \cdot 32 \cdot 128) = 8.6\%$	$r = 64$	$1/32 + 64/128 = 53.1\%$
512M-8K	12	$1/12 = 8.3\%$	$4/12 = 33.3\%$	$256/(2 \cdot 12 \cdot 128) = 8.3\%$	$r = 51$	$1/12 + 51/128 = 48.2\%$

For the low-rank 2.5B operating points used to study the memory–quality frontier, the same formula gives

$$r = 8,\ d_s = 48: \quad \frac{48 + 18 \cdot 8}{18 \cdot 128} = 8.3\%, \qquad r = 8: \quad \frac{1}{18} + \frac{8}{128} = 11.8\%,$$

$$r = 16: \quad \frac{1}{18} + \frac{16}{128} = 18.1\%,$$

and for the default 2.5B half-cache configuration,

$$r = 64: \quad \frac{1}{18} + \frac{64}{128} = 55.6\%.$$

Thus full-width LRKV has a shared-cache floor of $1/H$; the reduced-shared-width r=8 tier lowers this floor to reach the 8.3% match (8.6% at 6.3B). Every LRKV budget in Table 1 is matched by an MLA with d_c set to the same ratio.

I.4 Absolute Memory Example

The absolute cache memory follows directly from the formulas above. For example, for the 2.5B model with $N_\ell = 36$, $H = 18$, $d_h = 128$, $T = 8192$, $B = 1$, and bfloat16 precision, Standard MHA requires

$$2 \cdot 36 \cdot 8192 \cdot 18 \cdot 128 \cdot 2 \approx 2.72 \times 10^9 \text{ bytes},$$

or approximately 2.72 GB of KV cache per 8K-token sequence. At batch size 4, this becomes approximately 10.9 GB, more than twice the model's bfloat16 parameter memory. We report ratios in the main text to avoid ambiguity between decimal GB and binary GiB conventions.

I.5 END-TO-END DECODE IMPLEMENTATION AND BENCHMARK PROTOCOL

LRKV. The benchmarked end-to-end path stores shared K/V features once and stores the per-head rank-r latents directly. The provided non-materializing kernel consumes those states and evaluates the distributed query-key and value-aggregation forms in Equations 3 and 4. It does not allocate full per-head K_h, V_h tensors. Full-dimensional RoPE is applied through its linear decomposition, with the shared key cached in rotated form and the residual contribution evaluated inside the kernel. The explicit-reconstruction path is retained only as an oracle for the numerical agreement check reported in subsection 4.3.

Optimized LRKV kernel. The kernel used for all 2.5B decode results (full-width shared branch: r=8 at 11.8% cache and r=64 at 55.6%; the d_s=48 variant was not benchmarked) launches one program per (sequence, context chunk, head group). Within a program the shared-branch scores $qK_s^\top$ and outputs PV_s are tensor-core contractions over all heads of the group, and the rank-r residual terms $(qB_h^K)R_h^{K\top}$ and $(PR_h^V)B_h^{V\top}$ are evaluated as block-diagonal tensor-core contractions over the group's Hr residual columns, so every cached byte is read once per program and no per-head K_h, V_h or score matrix is materialized. Per-chunk partial results (running max, sum, value and residual accumulators) are merged by a fused reduce kernel that also applies B^V. The kernel is bit-exact against the module's explicit einsum evaluation on hidden states and cache contents. Its predecessor (a per-head split-T kernel) is 1.5–4.8× slower (2.4–4.8× at batch ≥8) and is kept in the release for reference.

Baselines. MHA, GQA and MQA use PyTorch SDPA with `enable_gqa`, which reads each KV head once (the earlier expand-then-attend path copies grouped K/V to all query heads every step and made GQA and MQA no faster than MHA). MLA (absorbed) attends over the cached latent with batched matmuls and decoupled per-head RoPE keys (d_R=32 per head, so its decode cache is 20.8% of MHA at d_c=384 versus 8.3% for the latent-only form); MLA (latent) caches the latent only and up-projects the whole cache each step. All caches are preallocated at full length.

Hardware and protocol. The 2.5B tables use a single NVIDIA B200 (178 GiB), bfloat16, PyTorch 2.8 / Triton 3.4: one fresh process per point, chunked prefill (32768 tokens per forward, LM head evaluated for the last position), 16 warm-up and 256 timed decode steps, median of three runs. The scale sweep of Figure 5 uses the same code path at batch 1 and 8 with three warm-up and ten timed iterations of 32 tokens, reporting the mean.

Profiling breakdown. Table 19 separates attention-kernel time from the rest of the decode step for the measured points. The systems sweep uses the exact shape grid in Table 20. Its scale names

Table 19: **Where the decode step goes (2.5B, one B200).** Each cell is decode step time / attention-kernel time (ms, one profiled step after warm-up, median over the three runs of that point) / attention share of the step. At batch 1 every method is launch-bound (about 20 ms of kernel launches for 36 layers) and attention is under 15% of the step; from batch 32 upward (batch 128 for GQA) the step becomes attention-bound and the kernel's effective bandwidth decides throughput. "Previous kernel" is the per-head split kernel whose shared K/V re-reads limited it to about 0.3 TB/s; the paper kernel reads each cached byte once. Attention time is taken from a separately profiled step, so shares within profiler overhead of the event-timed step are shown as ≈100%.

	8K context, batch				32K context, batch	
Method	1	8	32	128	8	32
MHA (flash)	20.0 / 1.8 / 9%	20.0 / 4.4 / 22%	36.9 / 29.9 / 81%	OOM	24.2 / 16.1 / 66%	OOM
GQA-6 (flash, fused)	21.4 / 0.9 / 4%	20.2 / 1.6 / 8%	20.0 / 5.8 / 29%	50.9 / 43.7 / 86%	20.1 / 5.7 / 28%	30.6 / 21.7 / 71%
LRKV r=8, previous kernel	27.9 / 2.5 / 9%	28.6 / 10.2 / 36%	41.4 / 36.9 / 89%	140.9 / 143.6 / ≈100%	42.8 / 35.5 / 83%	141.6 / 135.8 / 96%
LRKV r=8, paper kernel	19.2 / 1.3 / 7%	18.9 / 3.2 / 17%	18.4 / 9.5 / 52%	41.5 / 35.7 / 86%	17.2 / 11.1 / 64%	36.7 / 31.2 / 85%
LRKV r=64, paper kernel	18.7 / 1.6 / 8%	18.6 / 4.1 / 22%	18.0 / 11.9 / 66%	OOM	18.0 / 11.3 / 63%	OOM

are rounded labels and should not be read as the trained-backbone definitions in Table 9. The 128M and 2.5B tuples coincide with their training backbones; the remaining rows extend the systems grid independently and are not paired with pretraining-quality values.

Table 20: **Exact model shapes for the decode scale sweep.** All rows use d_h=128. Labels are the rounded names used in Figure 5.

Scale label	Layers	Heads	d_{model}
128M	12	6	768
1.2B	24	16	2048
2.5B	36	18	2304
4.5B	44	22	2816
6.3B	48	24	3072
8.6B	56	28	3584
9.7B	48	32	4096
13B	51	36	4608
20B	64	40	5120
28B	60	48	6144
30B	66	48	6144

J Scope and Limitations

Scope of the decode measurements. The decode results are measured on a single B200. They characterize cache size and operator cost and are not generation-quality measurements; every reported number derives from a logged run in the code release, with the protocol of subsection I.5.

Baseline implementations. All baselines are the implementations supplied with this work under one controlled protocol: SDPA flash attention for MHA, GQA and MQA, and the latent (up-project, then SDPA) and absorbed (batched-matmul) MLA forms. Vendor-specific MLA kernels would move the MLA throughput points but not its cache size, which is fixed by the parameterization.

Kernel regime. The LRKV kernel is bandwidth-bound for r=64; for r=8 it reaches about 1 TB/s effective, so it is above fused GQA at 8K context for every batch and at 32K for batch $\leq$8, but below it at 32K and batch $\geq$16 even though its cache is smaller. This is a kernel-engineering gap rather than a property of the parameterization, and closing it is the first item of future work.

Statistical scope. The 128M and 1.2B quality results are three-seed means with the dispersion reported in Appendix B; the 2.5B and 6.3B results are single runs; the LRKV-versus-MLA ordering at every tier and the ordering of the LRKV default against the baselines match the multi-seed scales. RULER is evaluated at one scale (512M) and one extrapolation length (32K).

Comparisons not run. We compare against MLA, GQA, MQA and MHA trained under the same recipe; MFA/MFA-KR, TPA, Palu and TransMLA are positioned in Table 6 but not trained head-to-head, as they target different settings (contextual factorization, post-training conversion).