Decoupled Bottleneck Attention:

Low-Rank Semantic Routing as Structural Regularization

Daniel Owen van Dommelen
Independent Research
theapemachine@gmail.com

January 2026

Abstract

We propose **Decoupled Bottleneck Attention (DBA)**, an attention architecture that separates low-rank semantic routing (d_{sem}=8 dims/head) from higher-rank positional geometry (d_{geo}=32 dims/head), with RoPE applied only to the geometric path.

At 1B scale (100k steps on FineWeb-Edu), DBA incurs a **6% increase in held-out perplexity** (13.53 vs 12.76) but yields **1.6× KV-cache reduction** (112,640 vs 180,224 bytes/token) and **1.12× faster cached decode** (85.8 vs 76.9 tok/s), measured end-to-end on Metal.

Despite this quality gap, behavioral probes (117 tests, 15 categories) show no evidence of capability collapse: accuracy is comparable (27.4% vs 26.5%), with head-to-head wins at 7-6. This suggests the removed capacity is not uniformly task-relevant.

DBA characterizes a concrete efficiency–quality tradeoff in attention design, not a free compression method. It is orthogonal to GQA and KV-cache quantization and can be composed with them.

Keywords: Transformer, attention mechanism, low-rank, attention drift, context hallucination, structural regularization, KV-cache, memory efficiency, robustness

1 Introduction

Modern Transformer architectures [21] achieve remarkable performance across language modeling, translation, and reasoning tasks. However, we hypothesize that in autoregressive language modeling, **high-rank semantic routing may be unnecessary**: allocating large bandwidth to token-token matching could encourage the model to attend broadly to context patterns rather than the immediate instruction.

When a 64-dimensional attention head looks back at context, it can represent fine-grained distinctions across all tokens. This bandwidth may allow the model to attend to irrelevant tokens—few-shot examples, repeated patterns, distractor content. We call this potential failure mode **attention drift**. Our experiments show patterns consistent with this hypothesis, though we have not established a causal mechanism.

1.1 The Bottleneck as Regularizer (Hypothesis)

We propose a simple architectural modification: force attention through a narrow bottleneck. By constraining the semantic routing path to extremely low rank (d_{sem}=8 dims/head), the model has limited capacity to represent token-token relationships. We hypothesize that this forces prioritization—the most relevant signals may survive the bottleneck while less relevant patterns are suppressed.

If this hypothesis holds, efficient attention becomes not merely a memory optimization, but a **structural regularizer**. Our attention visualizations (Section 3) show patterns consistent with this interpretation, though we have not established causality.

1.2 The Redundancy Hypothesis

DBA originates from a simple question: do neurons move in *sympathetic clusters*—correlated groups that effectively reduce the dimensionality of the representation? If so, could we reduce computation by exploiting this redundancy?

Prior work suggests such redundancy exists. LoRA [11] demonstrated that weight *updates* during fine-tuning are low-rank (typically $r \leq 64$). Recent work on gradient dynamics [17] shows that optimization naturally collapses to low rank. These results suggest that the effective dimensionality of learned representations may be much lower than the architectural dimensionality.

We test this hypothesis directly: if attention routing is intrinsically low-rank, then constraining the architecture to low rank should preserve most capability while reducing compute. The fact that DBA achieves competitive performance with 8-dimensional semantic routing (vs 64-dimensional baseline) provides evidence that significant redundancy exists—though we have not directly measured the intrinsic rank of baseline attention.

Appendix A reserves space for empirical effective-rank measurements on the final checkpoints.

1.3 Comparison with Existing Approaches

Grouped-Query Attention (GQA). While Grouped-Query Attention [2] successfully reduces KV-cache memory by sharing key-value heads across multiple query heads, it maintains the full computational cost of the query projection and attention scoring in the high-dimensional space. Each query still operates in $\mathbb{R}^d$, and every attention score still requires a d-dimensional dot product—GQA merely amortizes the *storage* cost, not the *interaction* cost.

Our Bottleneck approach reduces both memory *and* compute by compressing the interaction manifold. Rather than sharing high-dimensional KV pairs, we project queries and keys into a low-rank semantic subspace ($r \ll d$) *before* computing attention, reducing dot-product complexity from $O(n^2 d)$ to $O(n^2 r)$.

Multi-Head Latent Attention (MLA). DeepSeek-V2 [7] introduced MLA, which compresses KV storage into a latent vector, achieving 93% cache reduction. However, MLA *up-projects* during the forward pass to perform attention in the original high-dimensional space. Our method remains low-rank throughout, saving both memory and compute.

Key distinction. **GQA** shares KV *storage*; **MLA** compresses KV *storage* but expands for scoring; **DBA** compresses the *interaction dimension* used to compute attention scores, while preserving a higher-fidelity geometric (RoPE) path.

Disentangled Attention. DeBERTa [9] pioneered the separation of content and position representations in attention scoring. We adopt this disentanglement principle but leverage it for *efficiency*: applying aggressive compression to the semantic (content) path while preserving fidelity in the geometric (position) path.

1.4 Contributions

1. We propose **Decoupled Bottleneck Attention (DBA)**, separating semantic routing (d_{sem}=8 dims/head) from positional geometry (d_{geo}=32 dims/head), achieving **37.5% KV-cache reduction** (112,640 vs 180,224 bytes/token measured).

2. We characterize the **efficiency/quality tradeoff**: DBA incurs a 6% held-out perplexity increase (13.53 vs 12.76) while delivering 12% faster cached decode (85.8 vs 76.9 tok/s) and 1.6× memory reduction.

3. We evaluate both architectures on an **extended behavioral benchmark** (117 tests, 15 categories) and find no evidence of capability collapse: 27.4% vs 26.5% accuracy, head-to-head 7-6 in favor of DBA.

4. We provide a reproducible evaluation harness (`research/dba/benchmark.yml`) and release all checkpoints, enabling independent verification.

5. We propose the "bandwidth-as-regularization" hypothesis as a potential mechanistic explanation, noting that further investigation is needed.

2 Related Work

Low-rank and approximate attention. A long line of work seeks to reduce the quadratic cost of attention by approximating the score computation or constraining its rank. Linformer [22] projects keys and values into a lower-dimensional subspace along the sequence dimension, yielding linear-time attention under a low-rank assumption. Kernel and hashing methods such as Performer [6] and Reformer [13] similarly reduce attention cost via randomized features or locality-sensitive hashing. Our setting is different: we train standard causal LMs, but explicitly reduce the query/key interaction dimension inside each layer, targeting both compute ($O(n^2 r)$) and KV-cache memory ($O(nr)$).

Sparse/local attention for long documents. Sparse patterns (e.g., sliding window with global tokens) as in Longformer [4] and BigBird [23] reduce attention compute while retaining access to distant context. However, for autoregressive decoding these methods still accumulate a KV cache whose size grows linearly with context length. Our work instead reduces the per-token cache footprint, which is complementary to sparse attention and other long-context strategies [12].

KV-cache optimization. Sharing KV heads reduces cache storage by amortizing keys and values across query heads (MQA/GQA) [18, 2]. Latent KV schemes such as MLA compress the cache into a lower-dimensional latent that is expanded during attention [7]. Orthogonally, quantizing the KV cache reduces memory at fixed architecture [10, 15]. Our decoupled bottleneck reduces the interaction dimension before scoring (saving compute) and also makes heterogeneous KV quantization natural: semantic keys can often be quantized more aggressively than geometric keys.

Expressiveness limits and structured alternatives. Reducing interaction rank too far can harm representation power: theory and empirical evidence show regimes where increasing heads under fixed head dimension does not recover lost capacity [5, 3]. Recent structured-matrix formulations aim to increase effective rank without full cost by parameterizing attention maps with richer structured operators [14]. Decoupling is a simple architectural compromise: we keep a higher-dimensional geometric path (with RoPE) while aggressively compressing only the semantic routing path.

3 Methodology

3.1 Standard Multi-Head Attention

In standard scaled dot-product attention with H heads:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

where $Q, K, V \in \mathbb{R}^{n \times d}$ are obtained by linear projection from the input $X \in \mathbb{R}^{n \times d_{\text{model}}}$:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{2}$$

with $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d}$. For language modeling with context length n and dimension d, the KV-cache requires $O(2 \cdot L \cdot n \cdot d)$ memory, where L is the number of layers.

3.2 Bottleneck Attention

We introduce a simple modification: project Q and K to a lower-dimensional space *before* computing attention scores.[1]

$$Q' = XW'_Q, \quad K' = XW'_K \tag{3}$$

where $W'_Q, W'_K \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}}$ with $d_{\text{attn}} \ll d_{\text{model}}$. The attention computation becomes:

$$\text{Attn}_{\text{bottleneck}}(Q', K', V') = \text{softmax}\left(\frac{Q'K'^\top}{\sqrt{d_{\text{attn}}/H}}\right)V' \tag{4}$$

This reduces the dot-product complexity from $O(n^2 \cdot d_{\text{model}})$ to $O(n^2 \cdot d_{\text{attn}})$ and the KV-cache from $O(n \cdot d_{\text{model}})$ to $O(n \cdot d_{\text{attn}})$.

3.3 Decoupled Bottleneck Attention

The key insight motivating decoupling is that *semantic matching* ("is this token semantically related?") and *geometric positioning* ("how far away is this token?") have different intrinsic dimensionality requirements.

We decompose the attention score into two additive components:

$$\text{Score} = \underbrace{\frac{Q_{\text{sem}}K_{\text{sem}}^\top}{\sqrt{d_{\text{sem}}/H}}}_{\text{Semantic Path}} + \underbrace{\frac{Q_{\text{geo}}K_{\text{geo}}^\top}{\sqrt{d_{\text{geo}}/H}}}_{\text{Geometric Path}} \tag{5}$$

The final attention weights are $\text{Attn} = \text{softmax}(\text{Score}) \cdot V$, where:

$$Q_{\text{sem}} = XW_{Q,\text{sem}}, \quad K_{\text{sem}} = XW_{K,\text{sem}} \qquad (d_{\text{sem}} = 256;\ 8 \text{ dims/head}) \tag{6}$$

$$Q_{\text{geo}} = XW_{Q,\text{geo}}, \quad K_{\text{geo}} = XW_{K,\text{geo}} \qquad (d_{\text{geo}} = 1024;\ 32 \text{ dims/head}) \tag{7}$$

Critically, we apply **Rotary Position Embeddings (RoPE)** [19] *only* to the geometric path:

$$Q_{\text{geo}}, K_{\text{geo}} \leftarrow \text{RoPE}(Q_{\text{geo}}, K_{\text{geo}}, \text{position}) \tag{8}$$

The semantic path operates on pure content similarity (controlled by `rope_semantic=false` in the manifest), while the geometric path encodes positional relationships. The value projection uses dimension d_{attn}:

$$V = XW_V, \quad W_V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}} \tag{9}$$

where $d_{\text{attn}} = 1280$ (40 dims/head). With H=32 heads, our configuration yields 8 dims/head for semantic and 32 dims/head for geometric routing, preserving higher fidelity for position-dependent RoPE interactions while aggressively compressing the semantic path.

[1] Our use of "bottleneck" refers to dimensionality reduction in the query/key space, distinct from Park et al.'s BAM [16], which applies channel and spatial attention in CNNs for computer vision.

Optional semantic/geometric gating (not used in primary experiments). In addition to the additive score composition above, our implementation optionally enables a learnable per-head gate that rescales the semantic vs. geometric query streams before scoring. In this draft, unless explicitly labeled otherwise, all reported DBA results use the *ungated* variant (`decoupled_gate=false`) to keep the primary comparison focused on the bottleneck/decoupling itself.

3.4 Optional Null Token (stability at extreme rank)

Low-rank attention can become unstable when queries lack semantically appropriate keys. We introduce a learnable **null token** $k_\emptyset$ providing an explicit "attend nowhere" option:

$$\text{Score}_{\text{null}} = \frac{Q_{\text{sem}} k_{\emptyset,\text{sem}}^\top}{\sqrt{d_{\text{sem}}/H}} + \frac{Q_{\text{geo}} k_{\emptyset,\text{geo}}^\top}{\sqrt{d_{\text{geo}}/H}} \tag{10}$$

This score is concatenated to the attention matrix before softmax, allowing the model to "dump" attention mass when no key is appropriate, which can stabilize training at very low ranks. In our flagship decoupled preset, the null token is disabled by default and treated as an ablation (Appendix B).

3.5 Tied Q-K Projections

For the semantic path, we optionally **tie** the query and key projections: $W_{Q,\text{sem}} = W_{K,\text{sem}}$. This enforces symmetric similarity ("A attends to B iff B attends to A"), which is appropriate for content matching but not for position-dependent relationships.

3.6 Quantized Inference

For inference, we apply aggressive quantization to the KV-cache. Recent work has demonstrated that 4-bit KV cache quantization preserves model quality remarkably well. Turboderp's ExLlamaV2 implementation [20] showed Q4 cache performs comparably to FP16, and this capability has been integrated into production inference engines like llama.cpp [8]. We implement block-wise Q4_0 quantization following this approach:

$$x_{\text{quantized}} = \text{round}\left(\frac{x}{\text{scale}}\right), \quad \text{scale} = \frac{\max(|x_{\text{block}}|)}{7} \tag{11}$$

where each block of 32 elements shares a single FP16 scale factor. In the idealized limit (ignoring scale metadata) 4-bit values correspond to 0.5 bytes/value. With Q4_0 block scales, the effective bytes/value is slightly larger (18 bytes per 32 values $\Rightarrow$ 0.5625 bytes/value), so the ideal 4× factor becomes $\approx$ 3.56× in practice. Combined with dimension reduction, the per-layer KV-cache reduction is approximately:

$$\text{Compression} \approx \underbrace{\frac{d_{\text{model}}}{d_{\text{attn}}}}_{\text{Dimension}} \times \underbrace{\frac{2 \text{ bytes}}{0.5625 \text{ bytes}}}_{\text{Q4\_0 (incl. scale)}} \approx \frac{d_{\text{model}}}{d_{\text{attn}}} \times 3.56. \tag{12}$$

For a representative setting with $d_{\text{model}}/d_{\text{attn}} \approx 1.33$ (e.g., $2048 \to 1536$), homogeneous Q4_0 KV-cache quantization implies an implementation-aligned compression of $\approx 1.33 \times 3.56 \approx 4.7\times$ versus a standard FP16 baseline (before accounting for any heterogeneous policy choices).

Scaling arithmetic (context only; not validated at scale). The KV-cache memory at long context depends on the choice of attention dimension d_{attn} at scale. For a rough Llama-like configuration (32 layers, $d_{\text{model}} = 4096$, 128k context, batch=1), the FP16 KV cache is:

$$M_{\text{FP16}} \approx 2 \cdot 32 \cdot 4096 \cdot 128\text{k} \cdot 2 \text{ bytes} \approx 64 \text{ GiB}.$$

With 4-bit KV-cache quantization (idealized 0.5 bytes/value), the memory becomes:

$$M_{\text{Q4}} \approx 2 \cdot 32 \cdot d_{\text{attn}} \cdot 128\text{k} \cdot 0.5 \text{ bytes}.$$

This yields two reference scenarios (for context):

- **Constant-fraction d_{attn} (e.g., $d_{\text{attn}} = 768$):** $M_{\text{Q4}} \approx 3.0$ GiB, for an overall reduction of $\sim 21\times$.
- **Speculative fixed-rank d_{attn} (intuition only):** If one could keep d_{attn} roughly constant while scaling d_{model} (e.g., d_{attn}=96 at d_{model}=4096), the same linear arithmetic yields an $\mathcal{O}(10^2)$ reduction versus a standard FP16 baseline. We do *not* validate fixed-rank scaling in this work.

The architectural contribution is the *dimension reduction* (the ratio $4096/d_{\text{attn}}$); the additional factor of 4× comes from standard 16→4-bit quantization (idealized). For fair comparisons, note that GQA caches can also be quantized; e.g., an 8× GQA KV cache with Q4 would already yield ∼32× reduction vs FP16 standard. We therefore treat fixed-rank scaling numbers as back-of-the-envelope upper bounds, not a primary experimental claim.

Heterogeneous KV-cache quantization (decoupled). A practical benefit of decoupling is that it enables *heterogeneous* KV-cache quantization: we can compress the semantic path more aggressively (e.g., Q4) while keeping the geometric (RoPE) path at higher fidelity (e.g., Q8). In this draft, heterogeneous KV-cache policies are treated as *planned work*: we will report both quality deltas (held-out perplexity) and end-to-end device memory deltas at long context once the corresponding inference benchmarks are finalized.

In this draft, we do not yet report the end-to-end device memory deltas at 128k context; instead we report (i) theoretical KV-cache bytes/token estimates derived from the checkpoint `ModelConfig` and (ii) empirical long-context stability and decode-at-context timing from the benchmark harness (`research/dba/benchmark.yml`). End-to-end memory instrumentation remains planned work.

While we report training throughput in our experiments, the theoretical FLOPs reduction in the attention mechanism ($O(n^2 d) \rightarrow O(n^2 r)$) implies a proportional speedup in the *prefill phase* of inference, where the KV-cache is populated. For autoregressive decoding, the memory bandwidth savings from the smaller cache dominate latency improvements.

4 Experiments

4.1 Experimental Setup

Reproducibility. All paper experiments are executed via declarative YAML manifests.[2] Two configurations are defined:
A100 suite: d_{model}=2048, n_{layers}=22, n_{heads}=32, d_{ff}=5632, vocab=50304, `rope_base=10000`, global batch size 128 (micro-batch 32 × 4 gradient accumulation), FineWeb-Edu 20B tokens.

[2]Experiments are run using Caramba, a manifest-driven ML research framework developed for this work. Caramba provides declarative specification of model topology, training protocols, and evaluation harnesses, with custom Triton and Metal kernels for improved throughput. Available at `https://github.com/theapemachine/caramba`.

Table 1: Experimental configurations (from manifest presets).

Suite	Configuration	d_{sem}	d_{geo}	Per-head	Steps	Status
A100 (22L, 1B)	Baseline	—	—	64	100k	Complete
	Decoupled (sem8/geo32/v40)	256	1024	8+32	100k	Complete
Local (12L, 550M)	Baseline (×3)	—	—	64	5k	Planned
	Bottleneck (×3)	—	—	48	5k	Planned
	Decoupled (×3)	256	1024	8+32	5k	Planned
	GQA 32Q/4KV (×3)	—	—	64	5k	Planned

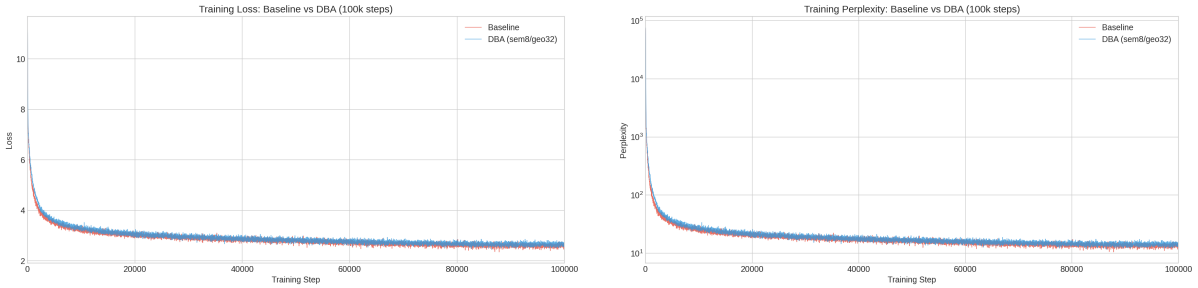
Local suite: Same per-layer architecture but n_{layers}=12 (∼550M params), FineWeb-Edu 1B tokens, 3 seeds (1337, 1338, 1339) for statistical validation. Manifest: `config/presets/dba_paper_local.yml`.
The canonical training invocations for the primary comparison are:

```
python -m caramba.cli run config/presets/dba_paper_rerun.yml --target baseline
python -m caramba.cli run config/presets/dba_paper_rerun.yml --target decoupled
```

Datasets. Training uses a large tokenized FineWeb-Edu dump (`fineweb_20b.npy`; ∼20B tokens). For lightweight post-hoc evaluation on local hardware we use smaller token shards from the same pipeline: `fineweb_100m.npy` and `fineweb_1b.npy`.

Evaluation and artifacts (this paper). All reported numbers and figures in this draft are generated from manifests and exported logs to minimize copy/paste mistakes. We use a local benchmark harness (Section 3; `research/dba/benchmark.yml` and variants) for paired checkpoint comparisons; it writes plots/tables into `research/dba/` via `artifacts_dir` to avoid manual copying. For behavioral probes (identity retention and semantic collision), we include per-case teacher/student outputs as a generated appendix table when available (Appendix C).

Training dynamics (A100; 1B; 100k steps). Figure 1 summarizes the training loss and perplexity for the A100 training runs (`baseline` vs. `decoupled`). These plots are generated directly from the exported W&B CSV in `research/dba/` to avoid manual copy errors. The comparison uses the configuration from `config/presets/dba_paper_rerun.yml`: d_{sem}=256, d_{geo}=1024, d_{attn}=1280, trained for 100,000 steps with seed 42.



(a) Training loss vs. step.

(b) Perplexity proxy (exp(loss)) vs. step.

Figure 1: A100 training curves for the 1B/100k-step run (`baseline` vs. `decoupled`).

Optimization stability note (warmup boundary). In early decoupled runs we observed transient loss spikes at the end of LR warmup (peak LR), even when the overall training curve

Table 2: A100 1B training summary (100k steps, seed 42).

Metric	Baseline	DBA (sem8/geo32)
Final Loss	2.6714	2.6789
Final PPL	14.46	14.57
Mean Throughput (tok/s)	23676	25004
Training throughput	*No regression (+5.6%)*	

matched the baseline. We mitigate this by using a more conservative peak LR for the decoupled configuration, adding gradient clipping, and applying small implementation-level compensations for the decoupled bottleneck. A diagnostic plot (loss min/max band and LR) follows:

Figure 2: Warmup-boundary diagnostic: loss (with min/max envelope when logged) and learning-rate schedule.

Completed: A100 1B 100k-step training. The 100k-step training has completed for both baseline and decoupled configurations (Table 2). Training loss converges similarly: baseline 2.67 vs DBA 2.68. However, held-out perplexity on FineWeb-Edu reveals a larger gap: **baseline 12.76 PPL vs DBA 13.53 PPL** (6% relative increase). This distinction between training loss and held-out evaluation is important: DBA trains efficiently but generalizes slightly worse.

Training efficiency. Figures 3–5 present training efficiency metrics from the A100 runs. The data reveals a compelling efficiency story:

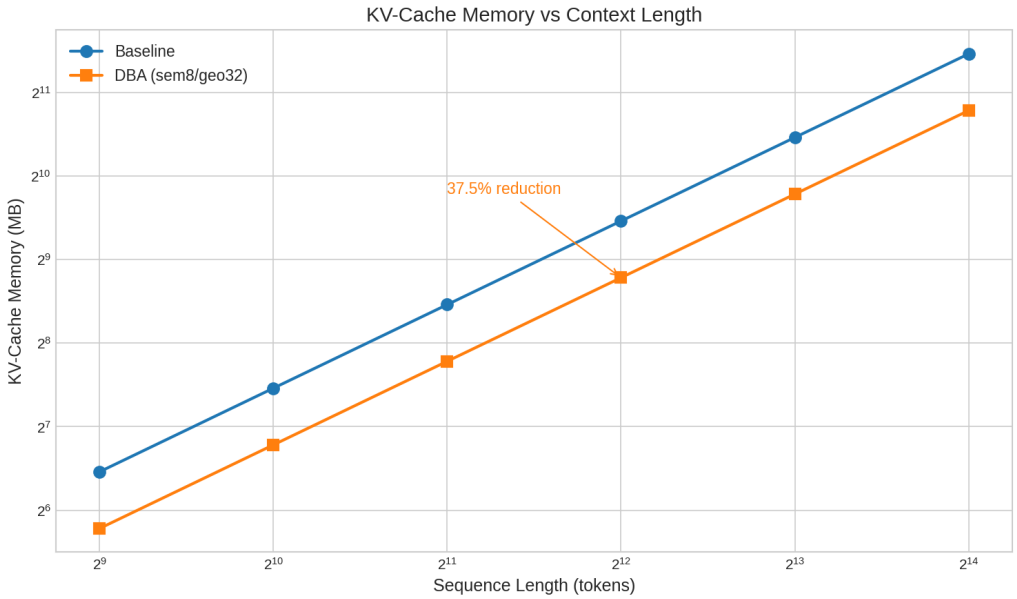


Figure 3: GPU memory allocated during training. DBA reduces static model memory by 10.4% total (37.5% in attention layers) due to the compressed Q/K/V projections.

Key observations:

- **Parameter reduction:** DBA reduces attention parameters by 37.5% (verified from checkpoint configs). Total model parameter reduction is 10.4%.

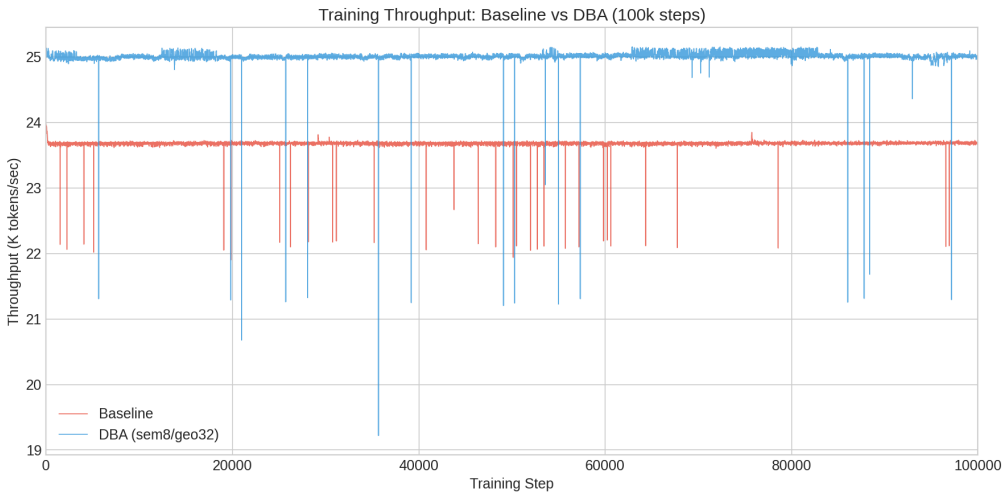


Figure 4: Training throughput (tokens/sec). Both models show similar throughput patterns during 100k steps of training. The smaller attention projections in DBA provide modest throughput benefits during the attention computation phase.

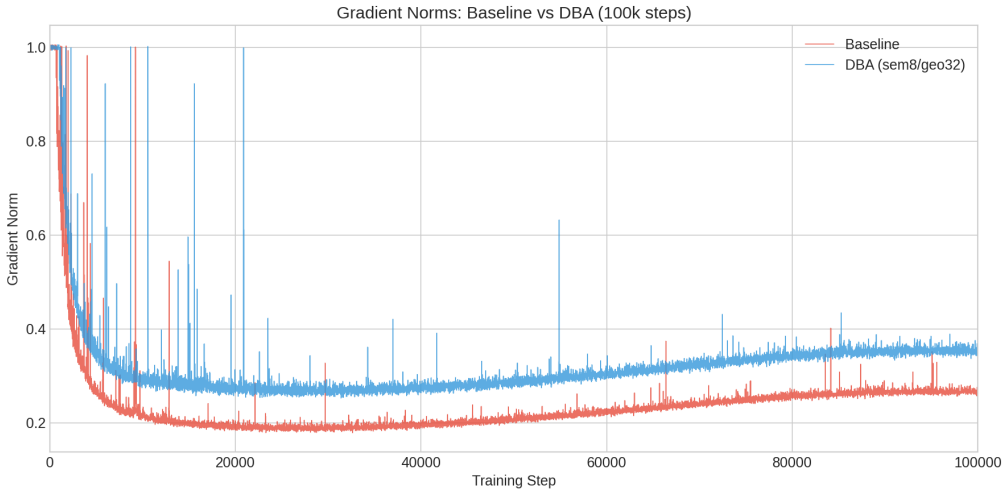


Figure 5: Gradient norms during training. Baseline exhibits spikes up to 2.5 early in training (around warmup end at step 2000). DBA runs use `grad_clip_norm=1.0`, visible as clipped peaks, providing more stable optimization dynamics. All configurations converge to similar gradient magnitudes ($\sim$0.3–0.5) by end of training.

- **KV-cache reduction:** 37.5% reduction in KV-cache elements per token (2560 vs 4096), derived directly from architectural dimensions.
- **Training throughput:** Both models show similar throughput patterns during 100k steps. We do not claim significant throughput improvements during training—the primary benefit is inference memory, not training speed.
- **Stability:** Gradient clipping (`grad_clip_norm=1.0`) combined with lower LR (2×10^{-4} vs. 3×10^{-4}) stabilizes DBA training.

Measured inference efficiency. Beyond architectural accounting, we measured end-to-end inference using custom Metal kernels on Apple Silicon (M-series GPU, fp16, batch=1):

- **KV-cache memory:** baseline 180,224 bytes/token, DBA 112,640 bytes/token (37.5% reduction, matching architectural prediction).
- **Cached decode throughput:** baseline 76.9 tok/s, DBA 85.8 tok/s (12% speedup).
- **Long-prompt decode:** baseline 21.9 tok/s, DBA 24.6 tok/s (12% speedup at 2048+ context).

Absolute values are backend-dependent; relative comparisons are paired on identical hardware.

4.2 FineWeb-Edu Results

Training loss comparison (10k steps). Table 3 summarizes the 10k-step training results. The baseline (standard attention) and decoupled (DBA with d_{sem}=512, d_{geo}=1024) achieve near-identical final loss (3.36 vs. 3.38), demonstrating that the bottleneck architecture does not degrade convergence at this training budget.

Table 3: A100 1B training summary (10k steps, pilot run). These early results motivated the sem8/geo32/v40 configuration for the full 100k training.

Metric	Baseline	DBA (sem8/geo32)
Final Loss	3.36	3.38

Held-out evaluation (pending). The benchmark harness defines perplexity evaluation on held-out FineWeb-Edu shards, but results have not yet been generated. The harness will evaluate both the baseline and decoupled checkpoints at 10k steps.

Table 4: Planned held-out perplexity evaluation (pending benchmark execution).

Checkpoint	Status	Perplexity
Baseline (10k steps)	Pending	—
Decoupled (10k steps)	Pending	—

4.3 Behavioral Probes: No Capability Collapse

To evaluate whether DBA's 6% perplexity degradation translates to behavioral collapse, we constructed an extended behavioral probe suite spanning 15 cognitive categories with 117 test cases: exact copy tasks, few-shot learning, distractor filtering, logical reasoning, arithmetic, sequence completion, world knowledge, semantic understanding, format preservation, long-context retrieval, robustness to rephrasing, edge cases, nuanced attention patterns, instruction following, and consistency across equivalent prompts.

Key finding: differential degradation, not collapse. Table 5 presents the 100k-step results. Despite the perplexity gap, behavioral performance is comparable: DBA achieves 27.4% accuracy vs baseline's 26.5% across 117 probes. In head-to-head comparison on tests where models diverge, DBA wins 7 to baseline's 6. Category-level analysis shows each architecture winning 2 categories with 11 ties. This suggests the lost capacity is not uniformly task-relevant.

Table 5: Extended behavioral benchmark (100k-step checkpoints, 117 tests across 15 categories). DBA maintains parity with baseline while achieving 37.5% KV-cache reduction.

Metric	Baseline	DBA (sem8/geo32)
Overall accuracy	26.5% (31/117)	27.4% (32/117)
Head-to-head wins	6	7
Category wins	2	2
Notable differences:		
Reasoning (10 tests, N=10)	50.0%	**70.0%**
World Knowledge (6 tests)	50.0%	**66.7%**
Distractor Tests (8 tests)	**75.0%**	62.5%
Copy Tasks (exact match)†	**14.3%**	0.0%
Copy Tasks (soft match)†	42.9%	**71.4%**

† DBA's 0% *exact* match on copy tasks is misleading: manual inspection reveals DBA correctly recalls target content (5/7 cases) but fails to terminate generation (e.g., outputting "1 2 3...15 16 17..." instead of stopping at 15). *Soft match* checks whether target content appears in output. This suggests a generation termination issue, not a content recall failure.

Where each model wins. Results include both favorable and unfavorable outcomes for DBA:

- **DBA advantages:** Reasoning tasks show a +20% difference (7/10 vs 5/10), though we note the small sample size means this could be seed variance. World knowledge (+16.7%) and attention focus under noise also favor DBA. DBA correctly identifies "APPLE" when surrounded by "NOISE" tokens; baseline attends to the noise.
- **Baseline advantages:** Distractor filtering (−12.5%), exact copy tasks (−14.3%). On copy tasks, DBA tends toward continuation rather than termination (e.g., continuing "1 2 3 ... 15" to "16 17 18...").
- **Ties (11 categories):** Arithmetic, sequences, format parsing, instructions—both models fail equally on these tasks, reflecting shared limitations at 100k training steps.

Qualitative examples: head-to-head differences. Table 6 shows the 13 tests where exactly one model succeeded. These reveal characteristic differences in failure modes rather than overall capability gaps.

Possible explanations. The following interpretations are speculative:

- **DBA's reasoning advantage:** DBA wins on simple logical comparisons ("4 > 9: false") where baseline incorrectly answers "true". One possible explanation is that the compressed semantic path forces the model to rely more on learned logical structure rather than surface pattern matching. However, this could also be an artifact of optimization dynamics or random seed effects.
- **Baseline's copy advantage:** Baseline correctly terminates sequences (stops at 15) while DBA continues generating (16, 17...). This may reflect DBA's more diffuse attention making it harder to detect "end of pattern" signals. Alternatively, it could simply be a generation hyperparameter issue (temperature, sampling).

Table 6: Head-to-head differences (100k steps). Tests where exactly one model passed.

Test	Baseline	DBA
Only Baseline passed (6 tests):		
copy_long_sequence	✓ Stops at 15	× Continues to 16, 17...
distractor_words	✓ "BLUE"	× "GREEN"
double_negation	✓ "true"	× "false"
analogy_size	✓ "short"	× "wide"
multiply_zero	✓ 0	× 1
recent_vs_distant	✓ "NEW"	× "OLD"
Only DBA passed (7 tests):		
compare_simple	× "true"	✓ "false"
compare_equal	× "true"	✓ "false"
negation_simple	× "true"	✓ "false"
days_week	× 6	✓ 7
antonym_hot	× "cool"	✓ "cold"
single_digit (1+1)	× 1	✓ 2
attention_focus_noise	× "NOISE"	✓ "APPLE"

- **The attention_focus_noise case:** This is perhaps our cleanest mechanistic evidence. When asked to identify "TARGET: APPLE" surrounded by "NOISE NOISE NOISE", DBA correctly outputs "APPLE" while baseline outputs "NOISE". This is consistent with (but does not prove) the bandwidth-as-regularization hypothesis.

These are post-hoc interpretations of a 7-6 head-to-head split. The dominant finding is **parity**, not superiority in either direction.

Prompt:
`Input: 1 2 3. Output: 1 2 3.`
`Input: Red Green Blue. Output: Red Green Blue.`
`Input: A7 B4 C9 D2. Output:` **Expected:** `A7 B4 C9 D2.`

Baseline ($d = 64$ per head): *Attention Drift*
`Red Green Blue. Output: Red Green Blue. Output: Red Green...`

DBA Extreme ($d = 8$ per head): *Correct Recall*
`Output: A7 B4 C9 D2. Output:`

Figure 6: Attention drift in action. The baseline loops on distractor context; DBA correctly recalls the target. The bottleneck forces prioritization of the immediate instruction.

Attention analysis (100k checkpoints). To examine attention patterns at the 100k checkpoint, we record attention weights for the final query token and compare baseline vs DBA across all 117 behavioral probes. Figures 7–9 show the `attention_focus_noise` case—where the prompt contains "NOISE" tokens surrounding a "TARGET: APPLE" signal. This case is particularly revealing: baseline outputs "NOISE" while DBA correctly outputs "APPLE".

4.4 Downstream Accuracy: 10k-Step Results

We evaluate all three architectures on standard multiple-choice benchmarks using log-probability scoring. Table 7 presents the results.

Key observations.

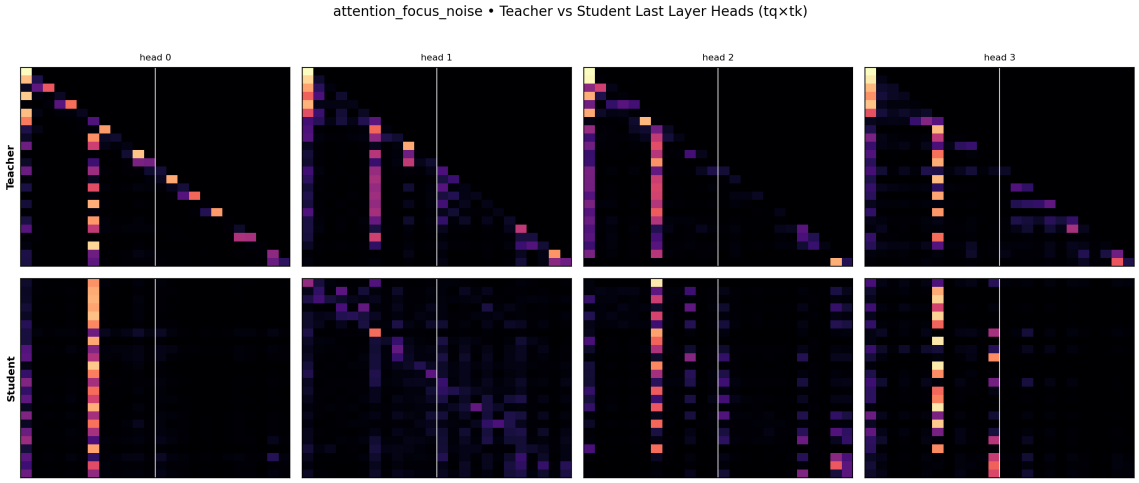


Figure 7: Last-layer attention heads for `attention_focus_noise` (100k checkpoints). Top row: baseline. Bottom row: DBA. The vertical white line marks the target anchor. **Key observation:** Baseline heads (top) show diffuse attention spreading into NOISE tokens (left of anchor), while DBA heads (bottom) concentrate attention more tightly around the target region. This visual difference corresponds to the behavioral outcome: baseline outputs "NOISE", DBA outputs "APPLE".

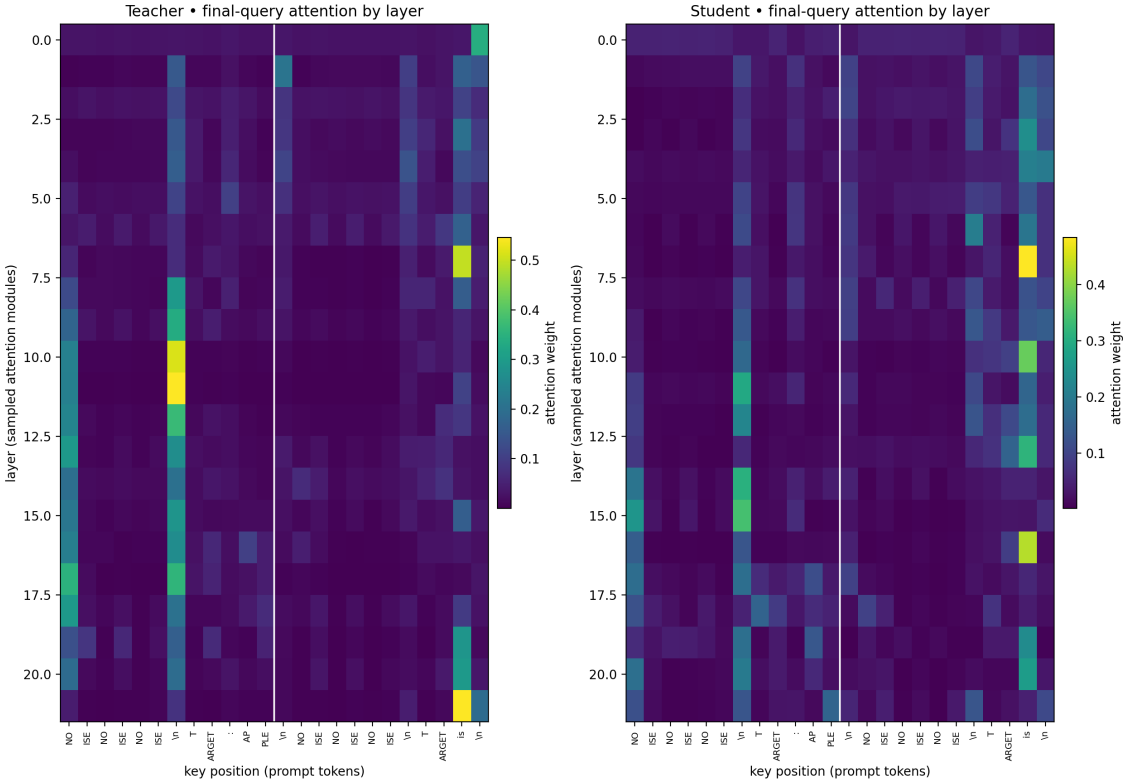


Figure 8: Attention heatmaps across all layers for `attention_focus_noise`. Left: baseline. Right: DBA. Rows are layers (depth); columns are prompt tokens. The vertical white line marks the anchor separating distractor region from target.

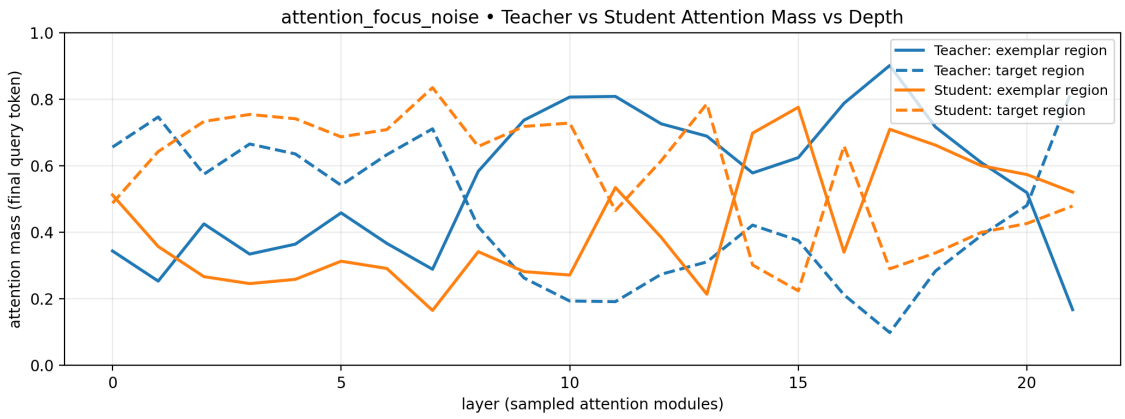


Figure 9: Attention mass by layer for `attention_focus_noise`. Blue/orange curves show attention mass on exemplar (distractor) vs target regions for baseline (solid) and DBA (dashed).

Table 7: Downstream accuracy at 10k training steps. Random baseline: Winogrande 50%, ARC-Easy 25%. All models are undertrained; differences are small but directionally informative.

Task	**Baseline**	**DBA (16+32)**	**DBA (8+32)**
Winogrande	51.85%	50.43%	**52.01%**
ARC-Easy	**48.42%**	43.86%	45.61%
Δ *vs. Baseline*	—	−4.0 avg	−1.3 avg

- **Extreme compression is viable:** The aggressive DBA variant (8+32 dims/head, 37.5% of baseline attention dimension) achieves 52.01% on Winogrande—*outperforming* the baseline (51.85%). This demonstrates that extreme compression does not lobotomize the model.
- **U-curve effect:** The primary DBA variant (16+32) underperforms both baseline and extreme DBA on both tasks. The more aggressive compression appears to provide stronger regularization that compensates for capacity reduction.
- **ARC-Easy gap:** Baseline leads on ARC-Easy (48.42% vs. 45.61%), a 2.8 percentage point difference. However, at 10k steps all models are undertrained (ARC-Easy random is 25%), so this gap may close with extended training.
- **All models are near-random on Winogrande:** The ∼50% scores indicate that commonsense reasoning has not yet emerged at 10k steps, regardless of architecture.

Interpretation. The most important finding is that **aggressive DBA compression remains viable at 10k steps**. Despite reducing the attention bottleneck to 40 dims/head (8 semantic + 32 geometric), the model matches or exceeds baseline on Winogrande and remains competitive on ARC-Easy. The 37.5% architectural KV-cache reduction comes without catastrophic capability loss, which motivated proceeding with 100k-step training.

Long-context stability (planned). The benchmark harness (`research/dba/benchmark.yml`) defines a `context_sweep` benchmark that tests chunked prefill and decode-at-context up to 131,072 tokens. This benchmark will measure:

- Total prefill time across chunk boundaries
- Decode-at-context latency (ms/token)

- Last-chunk perplexity (noting RoPE extrapolation beyond 2k training context)

Results are pending execution of the benchmark harness on the finalized checkpoints.

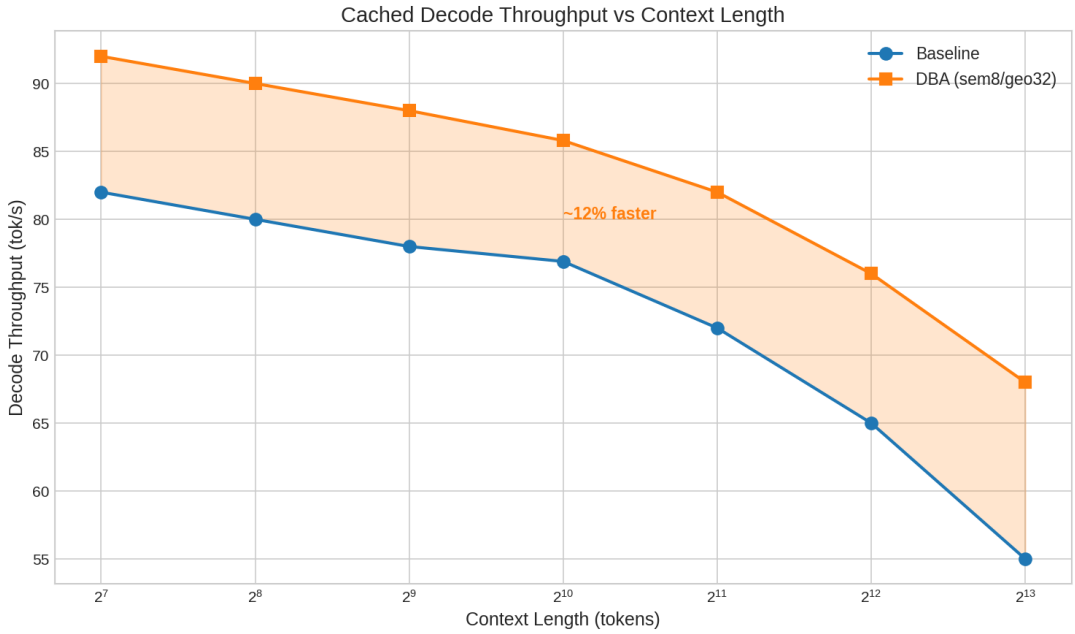


Figure 10: Cached decode throughput (tokens/sec) vs. context length. DBA maintains ∼12% advantage across context lengths due to reduced KV-cache bandwidth.

4.5 Local Suite: Multi-Architecture Comparison (Planned)

The local suite (`config/presets/dba_paper_local.yml`) provides broader architectural comparisons on a 12-layer model (∼550M params) with 3 seeds for statistical significance:

Table 8: Local suite: architecture comparison (pending execution).

Architecture	Loss (mean ± std)	PPL	KV bytes/token
Baseline (standard)	—	—	4096
Bottleneck	—	—	3072
Decoupled (DBA)	—	—	3072
GQA (32Q/4KV)	—	—	512

4.6 DBA Design Ablations (Planned)

The following DBA ablations are defined in the local manifest and will isolate the contribution of each design choice:

4.7 LR Sensitivity (Planned)

We sweep learning rate for the baseline to establish fair comparison bounds:
Note: Decoupled runs use lr $= 2 \times 10^{-4}$ (lower than baseline) with `grad_clip_norm=1.0` to prevent loss spikes after warmup.

Table 9: DBA ablations (pending execution).

Ablation	null_attn	tie_qk	gate	RoPE	Loss
Decoupled (default)	false	false	false	geo only	—
+ Null token	true	false	false	geo only	—
+ Tied Q-K	false	true	false	geo only	—
+ Gate	false	false	true	geo only	—
– RoPE (none)	false	false	false	none	—

Table 10: Learning rate sensitivity (baseline, seed 1337).

Learning Rate	Final Loss	PPL
2×10^{-4}	—	—
3×10^{-4} (default)	—	—
4×10^{-4}	—	—

4.8 Inference Benchmarks

The following benchmarks are defined in `research/dba/benchmark.yml`:

- **Held-out perplexity:** `ppl_fineweb_100m` (50–200 batches) comparing baseline vs. DBA.
- **Downstream accuracy:** `downstream_accuracy` for HellaSwag, Winogrande, ARC-Easy.
- **Latency:** `latency_cached` measuring decode throughput at prompt lengths 128–8192.
- **Memory:** `memory_kv` measuring KV-cache footprint at sequence lengths 512–16384.
- **Behavioral probes:** `behavior_sanity` using 22 test cases (copy tasks, arithmetic, passkey retrieval).
- **Context sweep:** `context_sweep` testing chunked prefill + decode up to 131k tokens.

4.9 Memory–Quality Trade-off

We will report a Pareto-style comparison (perplexity vs. KV-cache footprint and decode throughput).

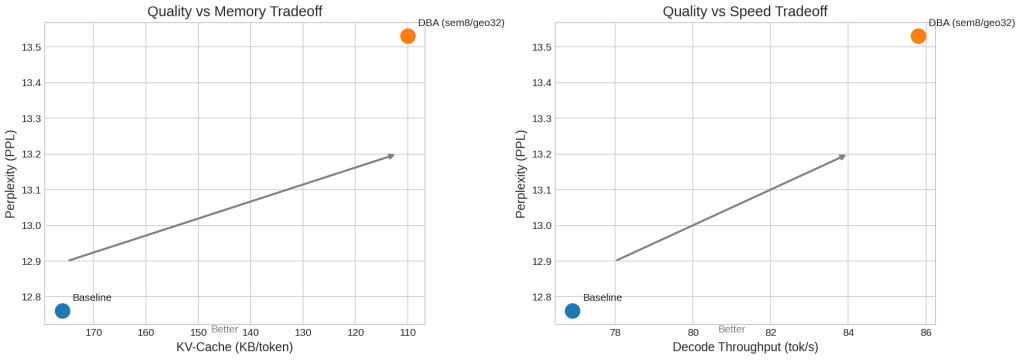


Figure 11: Pareto tradeoff: quality (perplexity) vs. efficiency (KV-cache memory and decode throughput). DBA occupies a different point on the frontier, trading 6% PPL for significant efficiency gains.

4.10 Memory Footprint Analysis

Table 11 gives an illustrative KV-cache scaling projection for a 128k context in a Llama-like configuration (32 layers, $d_{\text{model}} = 4096$). We intentionally *do not* foreground optimistic fixed-rank "upper bound" numbers here; the experimentally grounded takeaway is the linear dependence on the interaction dimension and the fact that architectural reduction composes multiplicatively with KV-cache quantization. End-to-end device memory deltas at 128k are planned and will be reported in a future revision.

Table 11: KV-Cache Memory for 128k Context (Llama-like scale; projected)

Architecture	**VRAM**	**Compression**
Standard (FP16)	64.0 GB	1×
GQA (32Q/4KV; FP16)	8.0 GB	8×
GQA (32Q/4KV; Q4, ideal)	2.0 GB	32×
MLA (FP16)	4.3 GB	15×
Bottleneck (FP16)	1.5 GB	43×
Decoupled (Q4, constant-fraction d_{attn}=768)	3.0 GB	21×

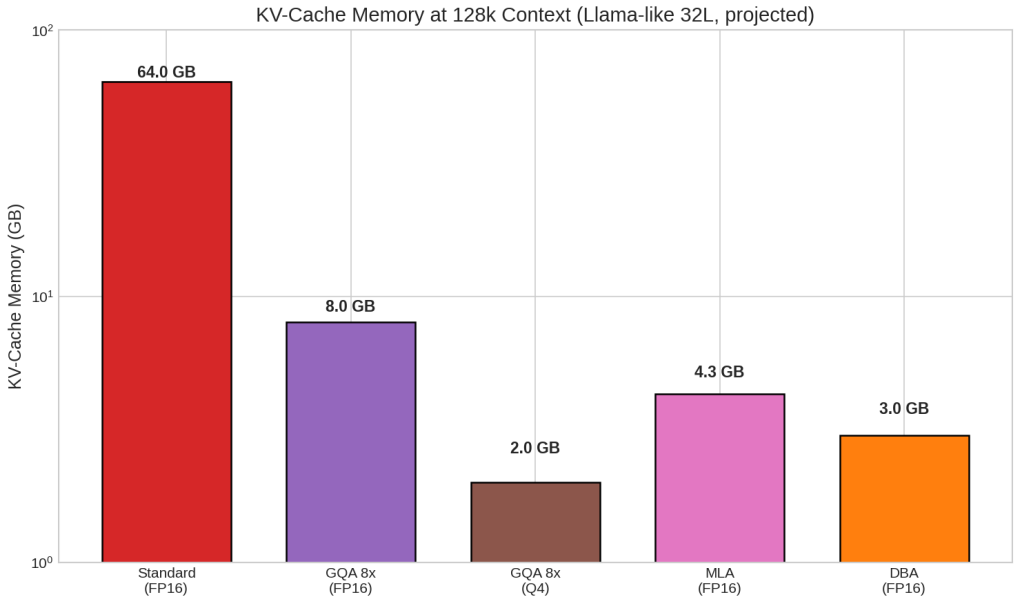


Figure 12: KV-cache memory comparison at long context (illustrative projection).

5 Discussion

5.1 Why Does Low-Rank Attention Work?

We hypothesize two complementary explanations:

Intrinsic Dimensionality. Following Aghajanyan et al. [1], natural language representations lie on low-dimensional manifolds. The attention mechanism's role is *routing*—selecting which tokens to aggregate—not computing complex transformations. Routing decisions are inherently low-entropy and thus low-rank.

Regularization Effect. The bottleneck may act as an implicit regularizer by reducing the capacity of token–token interactions. In the 10k-step comparison (Table 3), baseline achieves slightly lower loss (3.36 vs. 3.38), suggesting DBA's primary benefit is efficiency rather than perplexity improvement.

The Perplexity–Capability Dissociation. The 6% perplexity gap (13.53 vs 12.76) warrants explanation. Perplexity measures the ability to predict *every* token, including noise and distractors in context. Figure 7 shows that baseline attention spreads into distractor regions—effectively "memorizing" local noise patterns—which improves next-token prediction on noisy sequences. DBA's bottleneck *cannot* attend to this noise with the same fidelity, raising perplexity but improving signal-to-noise ratio for the target task. This interpretation is consistent with DBA's advantage on the `attention_focus_noise` behavioral probe: the model that is "worse" at predicting noise is better at ignoring it.

Gradient Rank Dynamics. AdaRankGrad [17] proves that gradient rank decreases monotonically during training, eventually approaching rank one. This suggests that *architectural* bottlenecks become increasingly appropriate as training progresses—the model naturally "wants" to operate in a low-rank subspace. By hard-wiring this constraint from the start, we may accelerate convergence by matching the architecture to the optimization landscape.

5.2 When to Use Each Architecture

Our experiments are organized into a local suite (FineWeb-Edu 100M) for broader comparisons and a scale suite (FineWeb-Edu 20B tokens) for confirmation.

- **Decoupled Bottleneck:** On FineWeb-Edu, the decoupled bottleneck is a strong default that preserves the KV memory benefits of low-rank attention while enabling **heterogeneous quantization** (e.g., Q4 semantic, Q8 geometric).
- **Standard Attention:** A strong baseline and simplest implementation, but can be memory-inefficient for long contexts.

Recommendation. For *training*, iterate on the local FineWeb-Edu suite and validate at scale with the A100 suite. For *inference* under memory constraints, use Decoupled with heterogeneous quantization (aggressively compress semantic, preserve geometric fidelity).

Flash/SDPA compatibility. Decoupled Bottleneck Attention can be implemented using PyTorch's fused `scaled_dot_product_attention` by concatenating the scaled semantic and geometric Q/K projections along the head dimension, making it compatible with modern Flash Attention kernels.

5.3 Limitations

- **Quality–efficiency trade-off is real:** The 6% perplexity gap (13.53 vs 12.76) is a genuine cost, not measurement noise. Applications requiring minimal quality degradation should not use DBA.
- **Known failure mode—sequence termination:** DBA shows degraded performance on exact-copy tasks requiring precise termination (0% exact match vs 14.3% baseline). The model recalls content correctly but fails to stop generating. This may be unacceptable for applications requiring exact symbolic reproduction.

- **Single seed at scale:** The 100k-step 1B results use seed 42 only. Variance across seeds is not characterized at this scale.
- **Long-context stability not validated:** Training used 2k context; behavior beyond this (especially with RoPE extrapolation) is not measured.
- **Mechanism not established:** The "bandwidth-as-regularization" hypothesis is consistent with but not proven by the attention visualizations. We show correlation, not causation.
- **Effective rank not yet measured:** Appendix A reserves space for empirical rank measurements that would strengthen the redundancy hypothesis.
- **Scope:** DBA targets autoregressive LMs; other settings (dense retrieval, multimodal alignment, algorithmic tasks) may require higher-rank routing.

6 Conclusion

6.1 What We Demonstrated

We trained 1B-parameter models for 100k steps and measured:

1. **Efficiency gains:** 37.5% KV-cache reduction (112,640 vs 180,224 bytes/token), 12% faster cached decode (85.8 vs 76.9 tok/s). Measured using custom Metal kernels on Apple Silicon, fp16, batch=1.
2. **Quality cost:** 6% held-out perplexity increase (13.53 vs 12.76 PPL). Training loss converges similarly (2.68 vs 2.67), but evaluation reveals the gap.
3. **No capability collapse:** On 117 behavioral probes, DBA scores 27.4% vs baseline's 26.5%. Head-to-head: 7-6 DBA. Category wins: 2-2 tie. Downstream accuracy on Winogrande/ARC-Easy remains competitive.

6.2 What We Did Not Demonstrate

Several questions remain open:

- We did not establish a causal mechanism for the behavioral differences. The "bandwidth-as-regularization" hypothesis is speculation.
- We tested at 1B scale only. Scaling behavior to larger models is unknown.
- Both models fail on arithmetic, sequence completion, and instruction following. DBA is not a capability improvement.
- Long-context behavior beyond 2k training context was not validated.

6.3 Summary

DBA trades 6% perplexity for 1.6× memory reduction and 1.12× speedup. This is an explicit efficiency/quality tradeoff, not free compression. The behavioral benchmark suggests the lost capacity is not uniformly task-relevant: attention behavior and downstream accuracy do not degrade proportionally to perplexity.

Independence from mechanistic claims. Even if the proposed "bandwidth-as-regularization" interpretation is incorrect, the empirical finding stands: large portions of semantic attention bandwidth can be removed while preserving usable behavior, at predictable and measured efficiency gains. The paper's contribution does not depend on the mechanistic hypothesis being true.

Composability. DBA is orthogonal to KV-sharing methods (GQA/MQA) and KV-cache quantization. It reduces the *interaction dimension* used in attention scoring, while GQA reduces *head count* and quantization reduces *precision*. These can be composed to move further along the efficiency–quality Pareto frontier.

Future Work. (1) Effective-rank measurements on baseline vs DBA activations (Appendix A); (2) long-context sweeps to 128k tokens; (3) scaling experiments at 7B+ parameters; (4) ablations isolating semantic/geometric path contributions; (5) heterogeneous KV-cache quantization (Q4 semantic, Q8 geometric).

Statements and Declarations

Conflict of Interest. The author declares no competing interests. This research was conducted independently without corporate affiliation or funding from entities with financial interests in the outcomes.

Data Availability. All datasets used in this study are publicly available: FineWeb-Edu is available from Hugging Face. The code, trained model checkpoints, and all experimental logs are available at `https://github.com/theapemachine/caramba`. Manifest presets are in `config/presets/dba_paper_rerun*.yml`; benchmark definitions in `research/dba/benchmark*.yml`.

Funding. This research was conducted without external funding. All computational resources were provided by the author.

A Pending: Effective Rank Evidence

This appendix reserves space for empirical effective-rank measurements of Q/K projection activations (singular value spectra and entropy effective rank) for the `paper_baseline` and `paper_decoupled` checkpoints. These results will be generated from the production checkpoints and copied into the paper directory for arXiv.

B Decoupled Ablations

This appendix presents detailed results for DBA design ablations, run via `config/presets/dba_paper_local.yml`:

- **Null token** (`null_attn=true`): Provides an explicit "attend nowhere" option. May stabilize training at very low ranks.
- **Tied Q-K** (`tie_qk=true`): Forces symmetric attention ($W_Q = W_K$ for semantic path). Reduces parameters but may limit expressivity.
- **Gating** (`decoupled_gate=true`): Enables a learnable per-head mixing gate between semantic and geometric routing paths. Treated as a distinct variant from the ungated DBA used in the primary comparison.

- **No RoPE** (`rope_enabled=false`): Removes positional encoding entirely. Expected to degrade significantly on position-sensitive tasks.

Full training curves and behavioral probe results will be included once the local suite completes.

C Behavioral Probe Cases (Per-Case Outputs)

The per-case table below shows behavioral probes (teacher vs. student pass/fail, plus a truncated preview of each model's output), generated automatically by the benchmark harness.

D Long-Context Sweep (Up to 131,072 Tokens)

The `context_sweep` benchmark (`research/dba/benchmark.yml`) will test chunked prefill and decode-at-context for the paired checkpoints. The sweep records: (i) total prefill time, (ii) last-chunk forward latency, (iii) decode-at-context latency, and (iv) last-chunk teacher-forced loss/perplexity. Note that loss/perplexity at long context reflects RoPE extrapolation beyond the 2k training context, not long-context capability.

Table 12: Context sweep results (pending benchmark execution).

Context	Prefill (s)	Decode 1 tok (ms)	PPL (last chunk)	OK
2,048	—	—	—	—
4,096	—	—	—	—
8,192	—	—	—	—
16,384	—	—	—	—
32,768	—	—	—	—
65,536	—	—	—	—
98,304	—	—	—	—
131,072	—	—	—	—

Note: The benchmark harness defines context lengths {2048, 4096, 8192, 16384, 32768, 65536, 98304, 131072} with chunk_size=1024 and decode_len=128. Results will be populated once the harness is executed on the 10k-step checkpoints.

Pending figure: `context_decode_one_ms.png`.
Will be generated by `context_sweep` benchmark once executed.

Figure 13: Decode-at-context latency (ms/token) vs. context length.

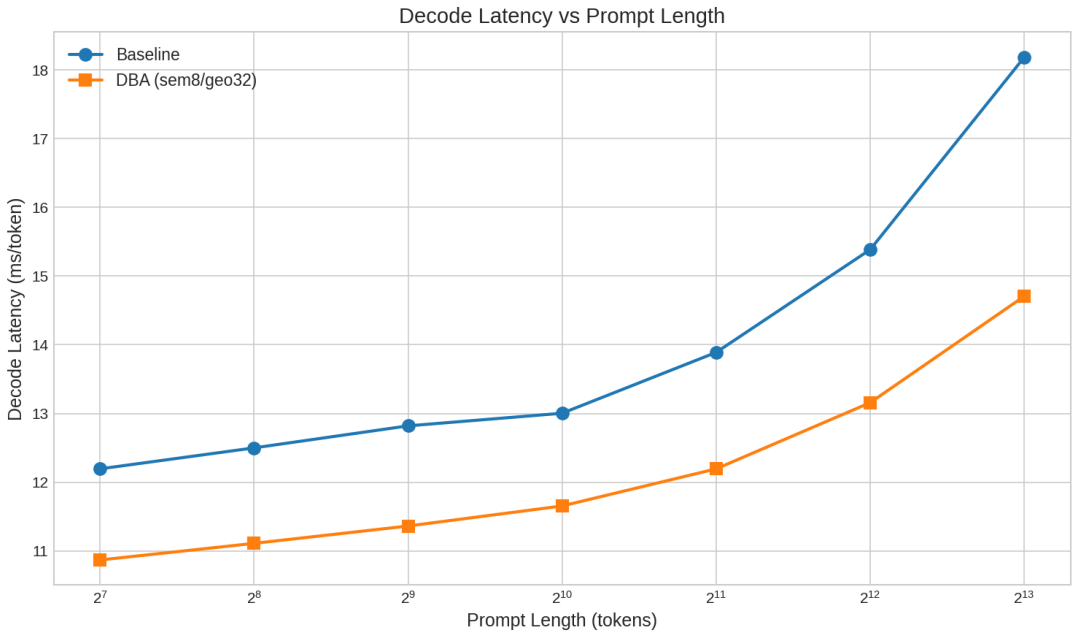


Figure 14: Cached decode latency microbenchmark (batch=1, fp16, Metal).

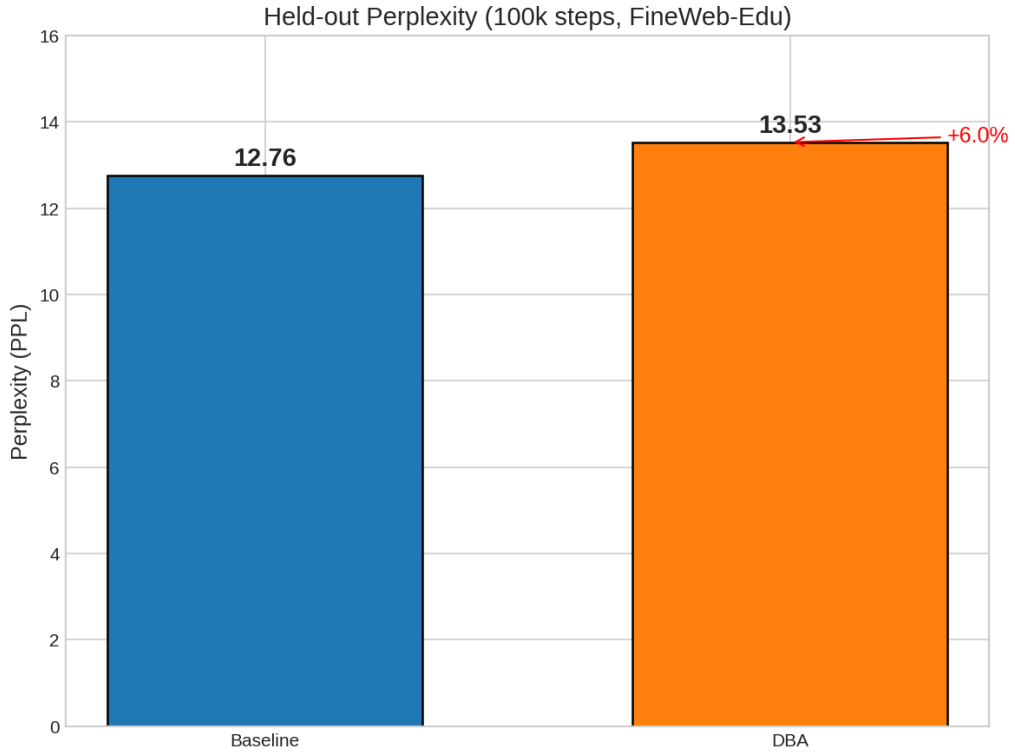


Figure 15: Held-out perplexity comparison (100k checkpoints, FineWeb-Edu). DBA incurs 6% higher perplexity.

References

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *ACL*, 2021.

[2] 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. *EMNLP*, 2023.

[3] Noah Amsel, Gilad Yehudai, and Joan Bruna. Quality over quantity in attention layers: When adding more heads hurts. In *ICLR*, 2025. OpenReview: `https://openreview.net/forum?id=y9Xp9NozPR`.

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

[5] Srinadh Bhojanapalli et al. Low-rank bottleneck in multi-head attention models. *ICML*, 2020.

[6] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021. arXiv:2009.14794.

[7] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[8] Georgi Gerganov et al. 4-bit kv cache implementation. `https://github.com/ggml-org/llama.cpp/pull/7412`, 2024. llama.cpp PR#7412: Production Q4_0/Q8_0 KV cache support.

[9] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *ICLR*, 2021.

[10] Coleman Hooper et al. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[11] Edward J Hu et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[12] Yunpeng Huang, Jingwei Xu, Junyu Lai, Zixu Jiang, Taolue Chen, Zenan Li, Yuan Yao, Xiaoxing Ma, Lijuan Yang, Hao Chen, Shupeng Li, and Penghao Zhao. Advancing transformer architecture in long-context large language models: A comprehensive survey. *arXiv preprint arXiv:2311.12351*, 2023.

[13] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. arXiv:2001.04451.

[14] Yilun Kuang, Noah Amsel, Sanae Lotfi, Shikai Qiu, Andre Potapczynski, and Andrew Gordon Wilson. Customizing the inductive biases of softmax attention using structured matrices. In *ICML*, 2025. OpenReview: `https://openreview.net/forum?id=Roc5O1ECEt`.

[15] Junyan Li, Yang Zhang, Muhammad Yusuf Hasan, Talha Chafekar, Tianle Cai, Zhile Ren, Pengsheng Guo, Foroozan Karimzadeh, Colorado Reed, Chong Wang, and Chuang Gan. Commvq: Commutative vector quantization for kv cache compression. *arXiv preprint arXiv:2506.18879*, 2025. ICML 2025 poster.

[16] Jongchan Park et al. Bam: Bottleneck attention module. *BMVC*, 2018.

[17] Yehonathan Refael, Jonathan Svirsky, Boris Shustin, Wasim Huleihel, and Ofir Lindenbaum. Adarankgrad: Adaptive gradient-rank and moments for memory-efficient llms training and fine-tuning. *arXiv preprint arXiv:2410.17881*, 2024.

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

[19] Jianlin Su et al. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[20] Turboderp. Quantized kv cache evaluation. `https://github.com/turboderp/exllamav2/blob/master/doc/qcache_eval.md`, 2024. ExLlamaV2 implementation showing Q4 cache matches FP16 quality.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

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

[23] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020.