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 strong performance across language modeling and reasoning tasks, but their inference cost scales poorly with context length. In autoregressive decoding, the key–value (KV) cache grows linearly with sequence length and dominates memory bandwidth, making long-context inference expensive even when compute is abundant.

Most efficiency work targets storage rather than interaction: sharing KV heads (MQA/GQA), compressing cached representations, or reducing precision. These approaches reduce memory footprint, but they preserve the full-dimensional token–token interaction used to compute attention scores. As a result, the interaction bandwidth of attention remains unchanged.

This paper asks a more basic question: how much token–token interaction bandwidth is actually required for useful language model behavior? In particular, do language models need high-rank semantic routing to follow instructions, retrieve targets, and reason over context—or does excess bandwidth primarily improve next-token prediction on noise and distractors?

1.1 Decoupled Bottleneck Attention

We propose Decoupled Bottleneck Attention (DBA), an attention architecture that explicitly constrains the dimensionality of semantic routing while preserving higher-fidelity positional geometry. Queries and keys are decomposed into two additive paths: (i) a semantic path with extremely low rank (8 dimensions per head), responsible for content-based routing, and (ii) a geometric path with higher rank (32 dimensions per head), responsible for positional relationships and rotary position embeddings (RoPE). Crucially, RoPE is applied only to the geometric path, while the semantic path operates on pure content similarity.

This design imposes a narrow bottleneck on token–token interaction without uniformly compressing representations: the value projection remains high-dimensional, and positional structure is preserved. The resulting model reduces attention parameters and KV-cache footprint by 37.5% at 1B scale, yielding a 1.6× reduction in KV-cache memory and a 12% increase in cached decode throughput.

These efficiency gains come at a cost. DBA incurs a 6% increase in held-out perplexity compared to a standard attention baseline. However, across an extended behavioral probe suite spanning 117 tests in 15 categories, we find no evidence of systematic capability collapse: behavioral accuracy remains comparable, with head-to-head wins split 7–6 in favor of DBA. Attention visualizations reveal more concentrated routing under noise, consistent with—but not proving—the hypothesis that constraining semantic interaction bandwidth suppresses attention drift.

We emphasize that DBA is not presented as a free compression method or a drop-in replacement for standard attention in quality-sensitive settings. Rather, it characterizes a concrete efficiency–quality trade-off: a predictable loss in next-token prediction accuracy in exchange for substantial inference-time memory and bandwidth savings. The contribution of this work is empirical and architectural, not mechanistic proof.

1.2 The Redundancy Hypothesis (Supporting Perspective)

DBA is motivated primarily by an architectural question about interaction bandwidth, not by a claim that language representations are globally low-rank. However, prior work suggests that significant redundancy exists in learned representations and optimization dynamics, providing a plausible explanation for why aggressive interaction bottlenecks do not immediately collapse behavior.

Several lines of evidence point to effective dimensionality being substantially lower than architectural dimensionality. LoRA [10] demonstrates that fine-tuning updates concentrate in low-rank subspaces, often with rank ≤ 64. Recent analyses of gradient dynamics [17] show that gradient rank decreases monotonically during training, approaching rank one in late stages of optimization. These findings suggest that, in practice, model updates and routing decisions operate on a constrained manifold.

Importantly, this does not imply that all components of attention can be compressed uniformly. Attention serves multiple roles: routing between tokens, encoding positional relationships, and aggregating values. DBA exploits this asymmetry by constraining only the semantic routing component of attention, while preserving a higher-dimensional geometric path for positional structure. Redundancy in semantic routing makes this constraint tolerable, but it is not assumed to be free.

Under this view, redundancy is not the contribution but the enabling condition: if token–token routing decisions are intrinsically low-entropy, then limiting the dimensionality of semantic interaction should primarily remove capacity devoted to modeling noise and weak correlations. The empirical results in this paper support this interpretation, but do not establish it as a causal mechanism.

1.3 Comparison with Existing Approaches

A large body of work has sought to reduce the memory and compute costs of attention. These approaches differ not only in how much they compress, but in what aspect of attention they reduce: storage, precision, or interaction bandwidth.

Grouped-Query Attention (GQA). Grouped-Query Attention [1] reduces KV-cache memory by sharing key–value pairs across multiple query heads. This amortizes storage cost but preserves the full-dimensional query–key interaction: each attention score is still computed in the original high-dimensional space. As a result, GQA reduces memory footprint but does not reduce the computational cost or representational capacity of token–token interaction.

Latent KV Compression (MLA). Multi-Head Latent Attention (MLA), introduced in DeepSeek-V2 [6], compresses the KV cache into a lower-dimensional latent representation. However, this latent is expanded during the forward pass to compute attention in the original high-dimensional space. MLA therefore reduces storage but preserves the full interaction bandwidth during scoring. In contrast, DBA remains low-rank throughout the attention computation, reducing both KV storage and dot-product complexity.

Low-rank and approximate attention. Prior work on low-rank or approximate attention (e.g., Linformer [22], Performer [5]) reduces attention cost by projecting keys and/or values along the sequence dimension or by approximating the softmax kernel. These methods target asymptotic complexity for long sequences. DBA operates in a different regime: we train standard causal language models but reduce the dimensionality of query–key interaction within each layer, directly targeting the bandwidth of token–token routing and the size of the KV cache.

Disentangled attention. Disentangled attention mechanisms, such as DeBERTa [8], separate content-based and position-based components of attention scoring. DBA adopts this conceptual separation but applies it for a different purpose. Rather than improving expressivity, DBA exploits disentanglement to apply aggressive compression to the semantic routing path while preserving a higher-dimensional geometric path for positional structure. This allows substantial reduction in interaction bandwidth without uniformly degrading positional fidelity.

Key distinction. Existing methods primarily reduce what is *stored* (KV sharing, compression, quantization) or how scores are *approximated*. DBA reduces what is *interacted*: the dimensionality of semantic token–token matching used to compute attention scores. This makes DBA orthogonal to KV-sharing and quantization methods and allows these techniques to be composed to further trade quality for efficiency.

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 [5] and Reformer [12] 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 ($O(n^2r)$) and KV-cache memory ($O(nr)$).

Sparse/local attention for long documents. Sparse patterns (e.g., sliding window with global tokens) as in Longformer [3] 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 [11].

KV-cache optimization. Sharing KV heads reduces cache storage by amortizing keys and values across query heads (MQA/GQA) [18, 1]. Latent KV schemes such as MLA compress the cache into a lower-dimensional latent that is expanded during attention [6]. Orthogonally, quantizing the KV cache reduces memory at fixed architecture [9, 14]. 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 [4, 2]. Recent structured-matrix formulations aim to increase effective rank without full cost by parameterizing attention maps with richer structured operators [13]. Decoupling is a simple architectural compromise: we keep a higher-dimensional geometric path (with RoPE) while aggressively compressing only the semantic routing path.

Semantic subspaces in attention. Menary et al. [15] analyze attention through the lens of *semantic subspaces*—independent subspaces of the latent representation that can fully determine attention distributions. They show that Pre-Norm (the standard normalization placement) forces semantic subspaces to be orthogonal, and model "circuit collapse" when this constraint is violated. Our decoupled architecture explicitly separates semantic and geometric subspaces, which may naturally satisfy these orthogonality requirements. This theoretical framing supports our empirical finding that attention can be decomposed into independent routing pathways without capability collapse.

3 Methodology

3.1 Standard Multi-Head Attention

In standard scaled dot-product attention with H heads:

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

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 Decoupled Bottleneck Attention (DBA) is that attention serves multiple distinct roles, which need not share the same dimensionality. In particular, *semantic routing*—deciding which tokens are relevant based on content similarity—and *positional geometry*—encoding relative position and distance—have different intrinsic bandwidth requirements.

DBA decomposes the attention score into two additive components, corresponding to these roles:

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

The semantic path operates on a highly compressed representation, with $Q_{\text{sem}}, K_{\text{sem}} \in \mathbb{R}^{n \times d_{\text{sem}}}$ and $d_{\text{sem}} = 256$ total dimensions (8 per head with $H = 32$). This path is responsible for content-based token–token matching and constitutes the primary interaction bottleneck in DBA.

The geometric path preserves higher dimensionality, with $Q_{\text{geo}}, K_{\text{geo}} \in \mathbb{R}^{n \times d_{\text{geo}}}$ and $d_{\text{geo}} = 1024$ total dimensions (32 per head). Rotary Position Embeddings (RoPE) [19] are applied *only* to this geometric path:

$$Q_{\text{geo}}, K_{\text{geo}} \leftarrow \text{RoPE}(Q_{\text{geo}}, K_{\text{geo}}, \text{position}) \tag{6}$$

This separation ensures that positional structure is preserved at higher fidelity, while semantic routing is forced through a narrow bottleneck.

Crucially, the value projection is not compressed to the semantic dimension. Instead, values are projected to

$$V = XW_V, \quad W_V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}}, \quad d_{\text{attn}} = d_{\text{sem}} + d_{\text{geo}} = 1280 \tag{7}$$

corresponding to 40 dimensions per head. This design preserves representational capacity downstream of routing: DBA constrains *which* tokens interact, not *what* information is aggregated once routing has occurred.

[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.

The final attention output is computed as:

$$\mathrm{Attn} = \mathrm{softmax}(\mathrm{Score}) \cdot V \tag{8}$$

By reducing the dimensionality of semantic query–key interaction while preserving a higher-dimensional geometric path and value representation, DBA directly reduces the bandwidth of token–token interaction. This yields proportional reductions in attention parameters, dot-product complexity, and KV-cache footprint, without uniformly compressing representations or positional structure.

Semantic/geometric gating. Our implementation optionally enables a learnable per-head gate $g_h = \sigma(\gamma_h) \in [0, 1]$ that rescales the semantic and geometric query vectors before score computation:

$$Q'_{\mathrm{sem},h} = 2g_h \cdot Q_{\mathrm{sem},h}, \quad Q'_{\mathrm{geo},h} = 2(1 - g_h) \cdot Q_{\mathrm{geo},h} \tag{9}$$

where γ_h is a learnable per-head logit initialized to zero (so $g_h = 0.5$ initially, preserving the ungated baseline). This allows each head to learn whether to emphasize semantic routing ($g_h \to 1$) or geometric routing ($g_h \to 0$). Ablation experiments (Section 4.6) show that gating provides a substantial improvement (-5.67 PPL), suggesting that heads benefit from specializing along the semantic/geometric axis. The primary 100k-step experiments in this paper use the *ungated* variant (`decoupled_gate=false`) to isolate the contribution of decoupling itself; gated DBA represents a promising direction for future work.

3.4 Optional Null Token (Stability at Extreme Rank)

Aggressively constraining semantic routing can introduce a specific failure mode: when a query lacks any semantically appropriate key, the attention mechanism is forced to distribute probability mass across weak or irrelevant matches. At very low interaction rank, this can lead to unstable or diffuse routing.

To address this, we optionally introduce a learnable **null token** $k_\varnothing$ that provides an explicit "attend nowhere" option. The null token contributes an additional score:

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

This score is concatenated to the attention matrix prior to softmax, allowing the model to allocate probability mass away from all input tokens when no strong semantic match exists.

The null token is intended purely as a stability mechanism for extreme bottlenecks. In the primary DBA configuration evaluated in this paper, the null token is **disabled by default** and all reported results use the ungated, no-null variant unless explicitly stated otherwise. We treat the null token as a design ablation (Appendix B) rather than a core component of DBA.

3.5 Tied Q-K Projections

For the semantic path, we optionally **tie** the query and key projections: $W_{Q,\mathrm{sem}} = W_{K,\mathrm{sem}}$. 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 [7]. 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 $4\times$ factor becomes $\approx 3.56\times$ 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 $4\times$ comes from standard 16→4-bit quantization (idealized). For fair comparisons, note that GQA caches can also be quantized; e.g., an $8\times$ GQA KV cache with Q4 would already yield $\sim$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 ($O(n^2d) \to O(n^2r)$) 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:

Table 1: Experimental configurations (from manifest presets).

Suite	Configuration	d_{sem}	d_{geo}	Per-head	Steps	Status
22L (1B params)	Baseline	—	—	64	100k	Complete
	Decoupled (sem8/geo32/v40)	256	1024	8+32	100k	Complete
12L (550M params)	Baseline (×3)	—	—	64	10k	Complete
	Bottleneck (×3)	—	—	40	10k	Complete
	Decoupled (×3)	256	1024	8+32	10k	Complete
	GQA 32Q/4KV (×3)	—	—	64	10k	Complete

22L suite: d_{model}=2048, n_{layers}=22, n_{heads}=32, d_{ff}=4096, vocab=50304, `rope_base=10000`, global batch size 32 (micro-batch 4 × 8 gradient accumulation), FineWeb-Edu 20B tokens. Run on A100. Manifest: `config/presets/dba_paper_rerun.yml`.
12L suite: Same per-layer architecture but n_{layers}=12 (∼550M params), FineWeb-Edu 1B tokens, 3 seeds (1337, 1338, 1339) for statistical validation. Run on A100 with global batch size 16 (micro-batch 4 × 4 gradient accumulation). Manifest: `config/presets/dba_paper_local.yml`.
The canonical training invocations for the primary comparison are:

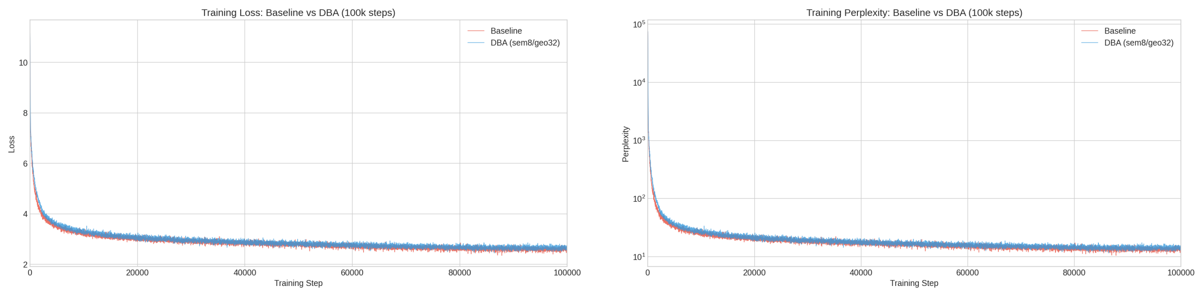
```
python -m caramba.cli run config/presets/dba_paper_rerun.yml --target baseline
python -m caramba.cli run config/presets/dba_paper_rerun.yml --target decoupled
```

Datasets. Training uses tokenized FineWeb-Edu dumps: `fineweb_20b.npy` (∼20B tokens) for the 22L suite, and `fineweb_1b.npy` (∼1B tokens) for the 12L suite. For lightweight post-hoc evaluation we use `fineweb_100m.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).

[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`.

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`).

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%)*	

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 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:
Key observations:

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

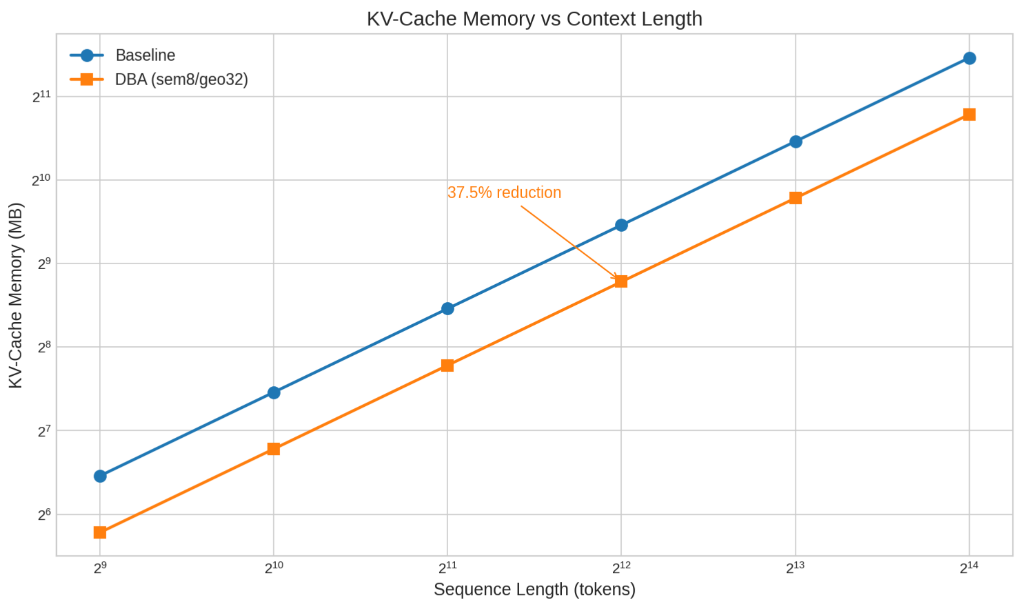


Figure 3: GPU memory allocated during training. DBA reduces static model memory by 10.4% total (37.5% in attention layers) due to the compressed Q/K/V projections.

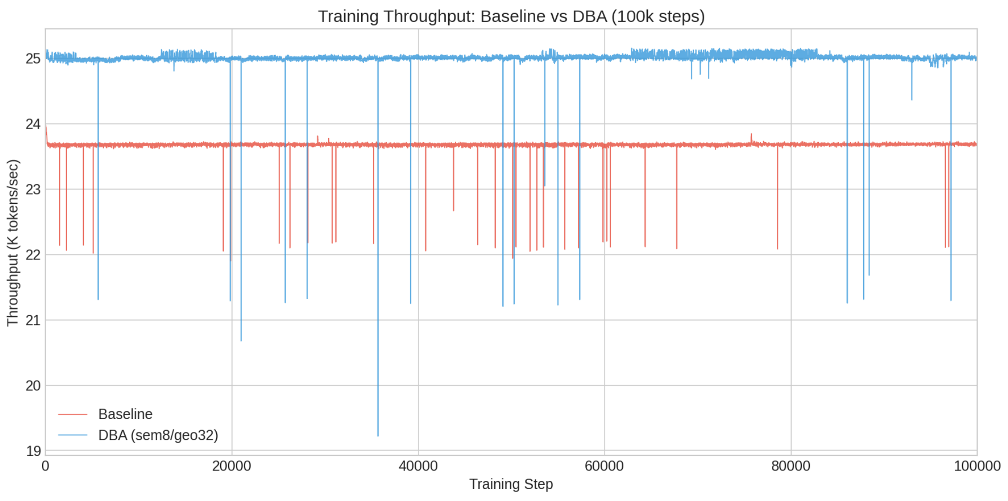


Figure 4: Training throughput (tokens/sec). Both models show similar throughput patterns during 100k steps of training. The smaller attention projections in DBA provide modest throughput benefits during the attention computation phase.

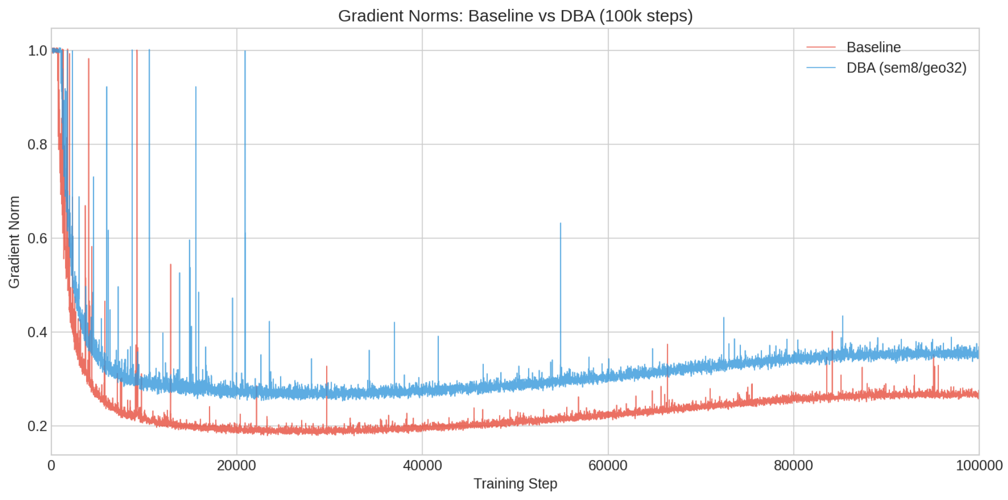


Figure 5: Gradient norms during training. Baseline exhibits spikes up to 2.5 early in training (around warmup end at step 2000). DBA runs use `grad_clip_norm=1.0`, visible as clipped peaks, providing more stable optimization dynamics. All configurations converge to similar gradient magnitudes (∼0.3–0.5) 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}=256, d_{geo}=1024; i.e., 8+32 dims/head) achieve near-identical final loss (3.36 vs. 3.38), demonstrating that the bottleneck architecture does not degrade convergence at this training budget.

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 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

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 Systematic Capability Collapse

To assess whether DBA's perplexity degradation translates into degraded model behavior, we evaluated both architectures on an extended behavioral probe suite spanning 117 tests across 15 cognitive categories. These probes target diverse capabilities, including exact copy tasks, few-shot learning, distractor filtering, logical reasoning, arithmetic, sequence completion, semantic understanding, instruction following, and long-context retrieval.

Headline result. Despite a 6% increase in held-out perplexity, DBA exhibits no systematic capability collapse. Overall behavioral accuracy is comparable to the baseline (27.4% vs. 26.5%), with head-to-head comparisons split 7–6 in favor of DBA. At the category level, each architecture wins two categories, with eleven ties.

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.

Differential degradation rather than uniform decline. Where differences do occur, they are uneven across task types. DBA outperforms the baseline on several reasoning-oriented probes and world-knowledge questions, while the baseline performs better on distractor filtering and exact copy tasks. This pattern suggests that the reduced semantic interaction bandwidth does not uniformly impair behavior, but alters failure modes.

A notable failure mode for DBA appears in exact copy tasks requiring precise termination. DBA frequently recalls the target content correctly but continues generating beyond the expected endpoint (e.g., continuing "1 2 3 . . . 15" to "16 17 18 . . ."). Manual inspection indicates this reflects a termination or stopping issue rather than a failure to recall content. When evaluated under a soft-match criterion (checking for correct recall regardless of termination), DBA succeeds in 5 of 7 cases.

Qualitative differences and attention focus. Table 6 shows the 13 tests where exactly one model succeeded. These cases illustrate characteristic differences rather than consistent superiority. In particular, DBA correctly identifies salient targets under heavy distractor noise (e.g., outputting "APPLE" when surrounded by repeated "NOISE" tokens), while the baseline occasionally attends to the distractor region. This behavior is consistent with DBA's more constrained semantic routing.

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"

Interpretation. Taken together, these results suggest that DBA's perplexity degradation reflects reduced capacity to model weak correlations and contextual noise, rather than a broad loss of functional capability. The behavioral probes indicate that constraining semantic interaction bandwidth changes *how* the model fails, not *whether* it functions. 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 visualization. Figures 7–9 visualize attention patterns for the `attention_focus_noise` probe. Baseline attention heads exhibit diffuse spread across distractor tokens, whereas DBA heads concentrate attention near the target anchor. These visualizations align with the observed behavioral outcome but do not establish a causal mechanism.

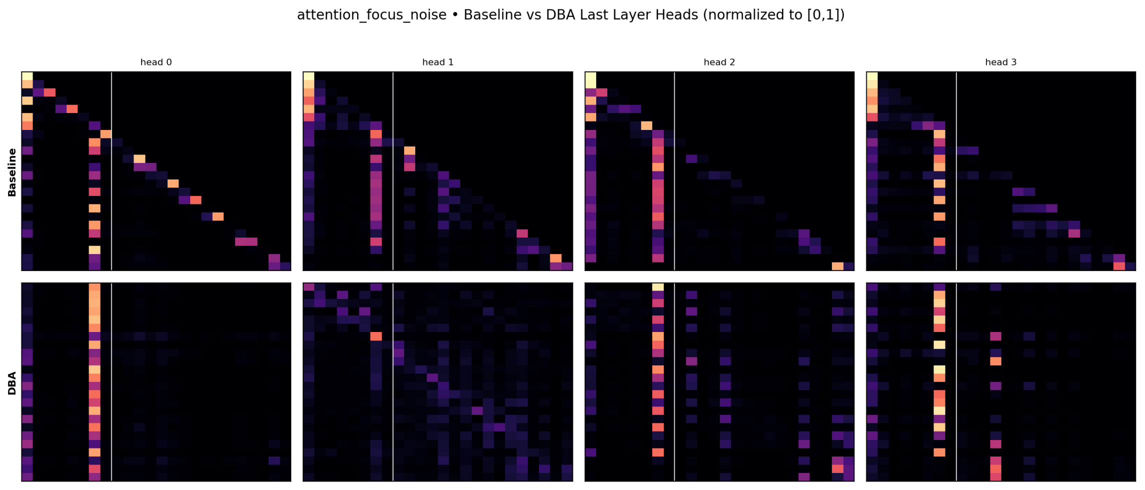


Figure 7: Last-layer attention heads for `attention_focus_noise` (100k checkpoints). Top row: baseline. Bottom row: DBA. The vertical white line marks the target anchor. **Key observation:** Baseline heads (top) show diffuse attention spreading into NOISE tokens (left of anchor), while DBA heads (bottom)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

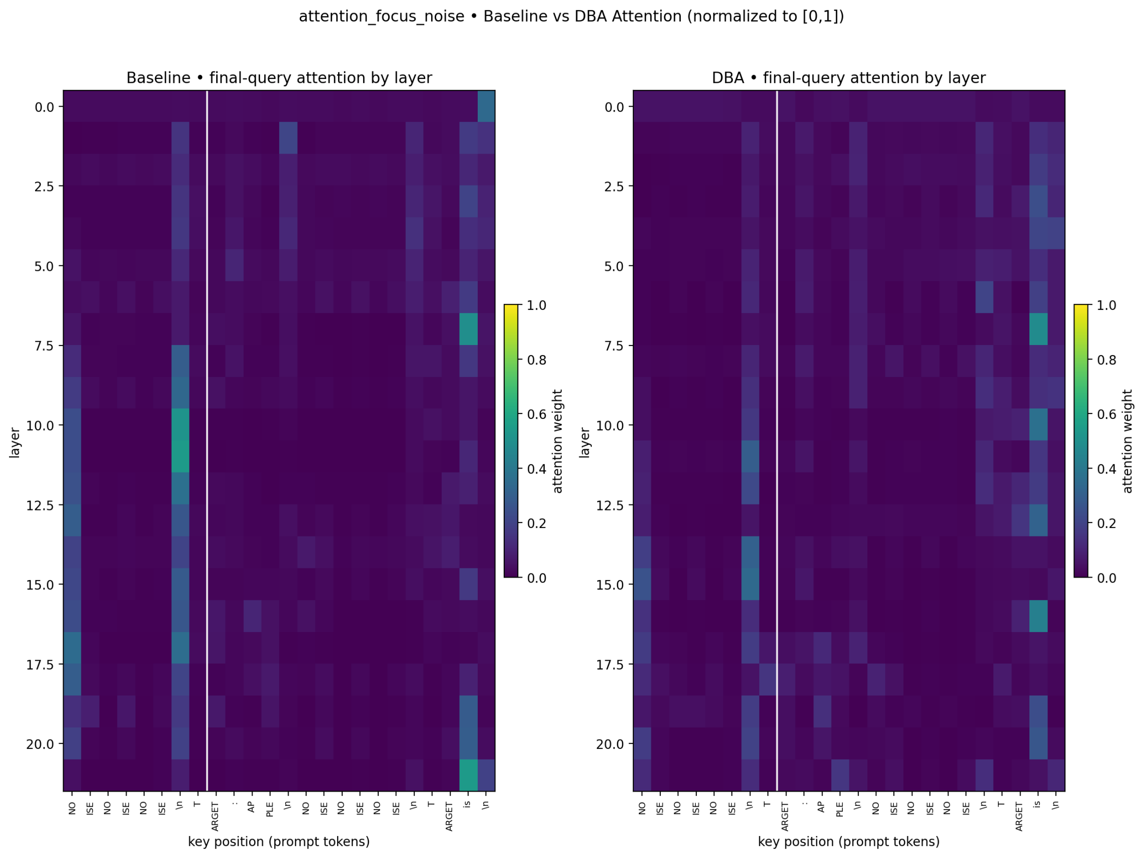


Figure 8: Attention heatmaps across all layers for `attention_focus_noise`. Left: baseline. Right: DBA. Rows are layers (depth); columns are prompt tokens. The vertical white line marks the anchor separating distractor region from target.

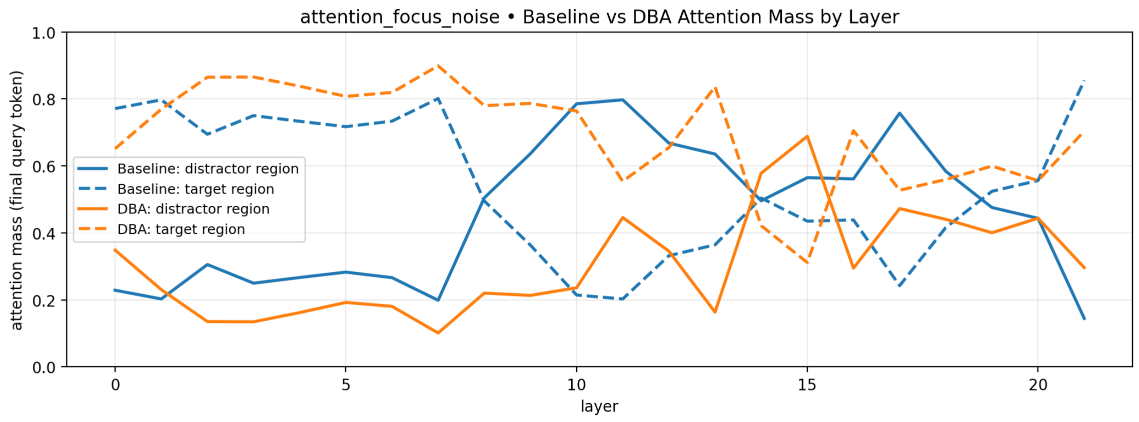


Figure 9: Attention mass by layer for `attention_focus_noise`. Blue/orange curves show attention mass on exemplar (distractor) vs target regions for baseline (solid) and DBA (dashed).

4.4 Architecture Selection: 10k-Step Downstream Evaluation

Prior to the 100k-step training runs, we conducted exploratory 10k-step runs to select the DBA configuration for scale-up. Three architectures were compared: baseline, DBA (16+32 dims/head), and the more aggressive DBA (8+32 dims/head). Table 7 presents downstream accuracy results that informed this selection.

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

Key observations.

- **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.

Architecture selection. The key finding is the **U-curve effect**: the more aggressive DBA (8+32) outperformed both baseline and the intermediate DBA (16+32) on Winogrande, while remaining competitive on ARC-Easy. This counterintuitive result—that more aggressive compression improved downstream performance—motivated selecting the sem8/geo32/v40 configuration for the 100k-step scale-up. The behavioral probe results in Section 4.3 confirm this selection: the aggressive bottleneck acts as effective regularization rather than a capacity constraint.

100k-step downstream validation. Table 8 presents downstream accuracy after extended training. The ARC-Easy gap observed at 10k steps persists but does not widen, while DBA's Winogrande advantage strengthens (+2.5 pp vs baseline). Micro-averaged across both tasks (weighted by test set size: 1267 Winogrande, 570 ARC-Easy), DBA leads 56.0% vs 54.5%.

Table 8: Downstream accuracy at 100k training steps. Random baseline: Winogrande 50%, ARC-Easy 25%.

Task	n	**Baseline**	**DBA (8+32)**
Winogrande	1267	50.2%	**52.7%**
ARC-Easy	570	**64.0%**	63.3%
Micro avg	1837	54.5%	**56.0%**

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.

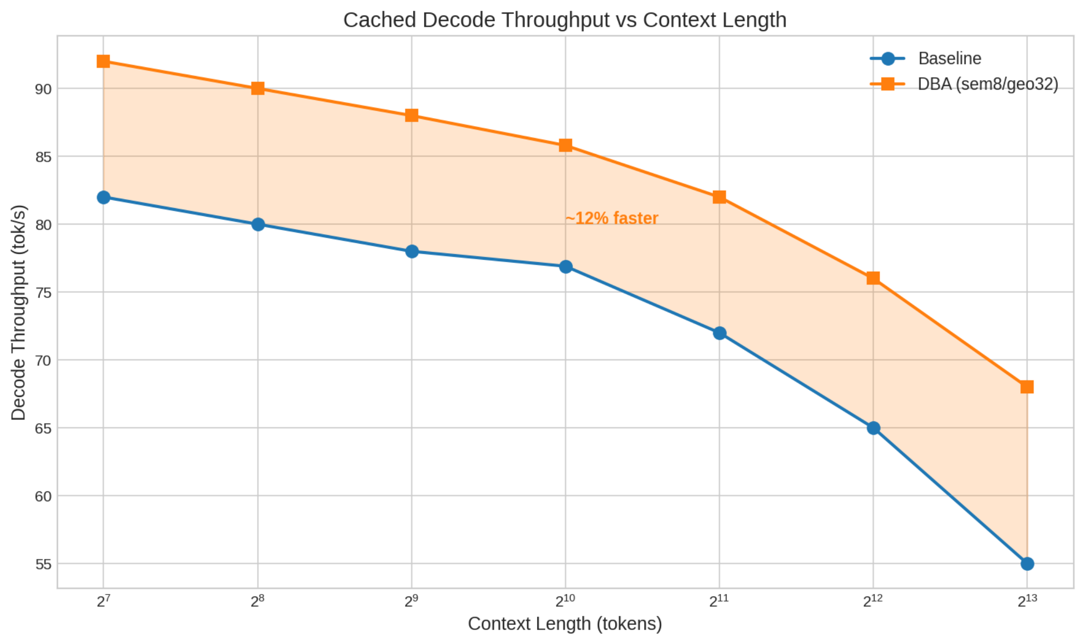


Figure 10: Cached decode throughput (tokens/sec) vs. context length. DBA maintains ∼12% advantage across context lengths due to reduced KV-cache bandwidth.

4.5 Local Suite: Multi-Architecture Comparison

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 9: Local suite: architecture comparison (12L, 550M params, 10k steps, 3 seeds).

Architecture	**Loss (mean ± std)**	**PPL (mean ± std)**	**KV bytes/token**
Baseline (standard)	3.861 ± 0.028	47.51 ± 1.33	98,304
Bottleneck	3.887 ± 0.035	48.77 ± 1.71	73,728
Decoupled (DBA)	3.854 ± 0.048	47.23 ± 2.27	61,440
GQA (32Q/4KV)	3.897 ± 0.090	49.39 ± 4.34	12,288

Key findings. DBA achieves the **lowest perplexity** (47.23) among all architectures while using 37.5% less KV-cache memory than baseline. This suggests that at the 12L/550M scale, the semantic bottleneck acts as effective regularization rather than a capacity constraint. GQA provides the most aggressive memory reduction (8×) but at the cost of higher variance and worse mean PPL.

4.6 DBA Design Variants

The following DBA variants isolate the contribution of each design choice (seed 1337):

Table 10: DBA design variants (12L, 550M params, 10k steps, seed 1337).

Variant	**null_attn**	**tie_qk**	**gate**	**RoPE**	**Loss**	**PPL (Δ)**
Decoupled (default)	false	false	false	geo only	3.906	49.71 (—)
+ Null token	true	false	false	geo only	3.906	49.71 (+0.00)
+ Tied Q-K	false	true	false	geo only	4.031	56.33 (+6.62)
+ Gate	false	false	true	geo only	3.785	**44.04** (−5.67)
– RoPE (none)	false	false	false	none	4.102	60.43 (+10.72)

Key findings. **Gating is the clear winner** (−5.67 PPL), achieving the lowest perplexity (44.04) of any configuration tested. This substantial improvement suggests that heads benefit from learning their own semantic/geometric balance rather than using fixed equal weighting. Some heads may specialize in content-based routing while others focus on positional relationships. **Tied Q-K hurts** (+6.62 PPL), indicating that asymmetric query-key interactions are important even in the compressed semantic path. **Removing RoPE is catastrophic** (+10.72 PPL), confirming that positional structure in the geometric path is essential—the semantic bottleneck cannot compensate for lost positional information. The null token has no measurable effect at this scale, suggesting the bottleneck is not so extreme as to require an explicit "attend nowhere" option.

Implication for future work. The gating result indicates that DBA+gate may be a stronger configuration than the ungated variant used in our primary 100k-step experiments. A natural follow-up would be to train DBA+gate at scale to quantify whether the 12% relative PPL improvement observed at 10k steps persists.

4.7 LR Sensitivity

We sweep learning rate for the baseline to establish fair comparison bounds:

Table 11: Learning rate sensitivity (baseline, 12L, seed 1337).

Learning Rate	Final Loss	PPL
1×10^{-4} (default)	3.875	48.18
2×10^{-4}	**3.664**	**39.02**
4×10^{-4}	3.816	45.44

Key finding. The default learning rate (1×10^{-4}) is suboptimal for the 12L baseline; 2×10^{-4} achieves 19% lower perplexity (39.02 vs 48.18). This suggests the architecture comparison results may underestimate baseline performance. However, since all architectures use the same default LR, relative comparisons remain valid.

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).

4.10 Memory Footprint Analysis

Table 12 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 12: 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×

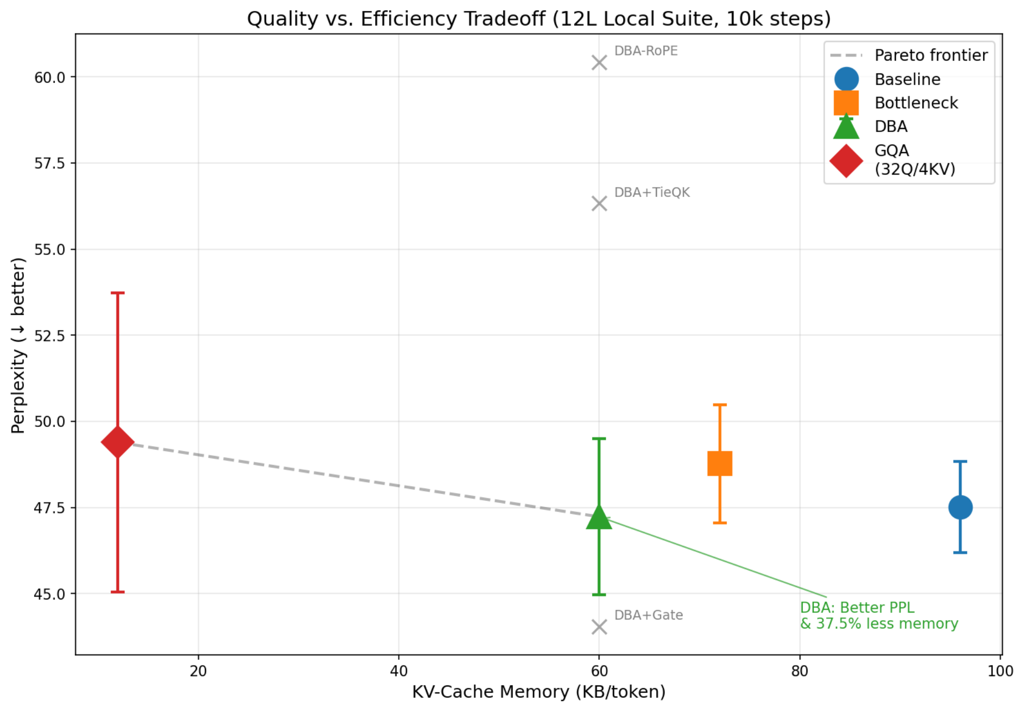


Figure 11: Pareto tradeoff: quality (perplexity) vs. efficiency (KV-cache memory and decode throughput). DBA occupies a different point on the frontier, trading 6% PPL for significant efficiency gains.

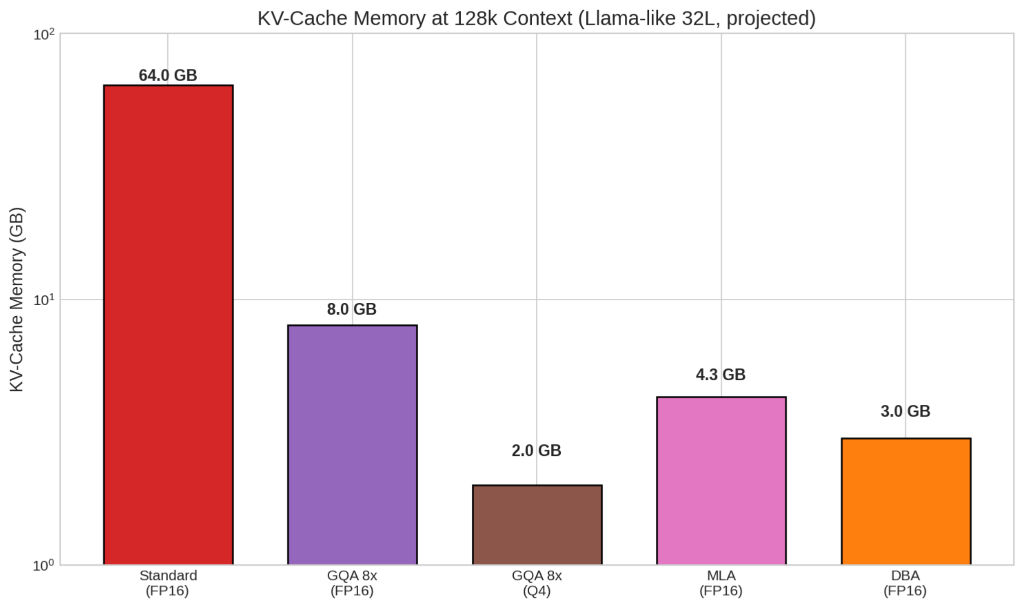


Figure 12: KV-cache memory comparison at long context (illustrative projection).

5 Discussion

5.1 Why Does Constrained Semantic Interaction Preserve Behavior?

DBA demonstrates a consistent pattern across experiments: aggressively constraining semantic interaction bandwidth degrades next-token perplexity while largely preserving behavioral performance. This section discusses several interpretive hypotheses that are consistent with this pattern. These explanations are not mutually exclusive, and we do not establish any of them as causal mechanisms.

Perplexity rewards modeling noise. Next-token perplexity measures the ability to predict every token in context, including weak correlations, distractors, and local noise. Attention mechanisms with high interaction bandwidth can allocate capacity to modeling these patterns, improving perplexity even when such patterns are irrelevant to the task. Constraining semantic routing reduces the model's ability to exploit weak or noisy correlations, which can increase perplexity without impairing task-relevant behavior. The attention visualizations in Section 7 are consistent with this interpretation: baseline attention spreads into distractor regions, while DBA concentrates attention near salient targets.

Routing decisions may be intrinsically low-entropy. Attention can be viewed as a routing mechanism: selecting which tokens to aggregate, rather than performing complex transformations. If routing decisions are intrinsically low-entropy—requiring only coarse distinctions—then large semantic interaction bandwidth may be unnecessary for many tasks. This view aligns with analyses of attention as routing between independent semantic subspaces [15] and helps explain why extreme semantic bottlenecks do not induce immediate behavioral collapse.

Optimization dynamics favor low-rank structure. Independent of architecture, optimization dynamics in large models tend to concentrate updates in low-rank subspaces. LoRA [10] and recent gradient-rank analyses [17] show that effective rank decreases during training. Under this view, architectural bottlenecks may not strongly oppose the optimization trajectory, but instead align with it. This interpretation suggests why DBA trains stably despite reduced interaction capacity, but does not by itself explain the observed perplexity–behavior dissociation.

Regularization through interaction constraint. Finally, constraining semantic interaction bandwidth may act as a form of structural regularization. By limiting the capacity of token–token matching, the model may be forced to prioritize strong, task-relevant signals over diffuse contextual patterns. This interpretation is consistent with DBA's improved performance on probes involving distractor suppression, but we emphasize that this remains a post hoc explanation rather than a demonstrated mechanism.

Summary. DBA's behavior is consistent with the view that high-bandwidth semantic interaction primarily benefits next-token prediction on noisy or weakly relevant context. Constraining this interaction alters failure modes rather than eliminating capability, producing a measurable efficiency–quality trade-off without systematic behavioral collapse.

Depth-wise deferral of semantic routing. Attention visualizations reveal a characteristic pattern in DBA models: early layers exhibit diffuse, low-contrast attention, while later layers concentrate semantic routing more sharply. This contrasts with baseline models, where semantic routing is distributed more evenly across depth. We interpret this as *depth-wise deferral*: under constrained semantic bandwidth, early layers act primarily as feature transformers under residual accumulation, deferring explicit token–token routing decisions to later layers where they are most needed. This pattern helps explain both DBA's robustness (early layers cannot latch onto distractors) and its termination fragility (the EOS signal must compete with final-layer aggregation). Gated DBA partially restores early-layer semantic engagement, which may explain its improved performance.

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

DBA is not a universal improvement over standard attention, and its benefits are bounded by clear trade-offs and unresolved questions.

Quality–efficiency trade-off. The 6% increase in held-out perplexity (13.53 vs. 12.76) observed at 1B scale is a genuine cost, not measurement noise. Applications that require maximal next-token prediction accuracy or precise modeling of weak contextual correlations may not tolerate this degradation. DBA should therefore not be interpreted as a free compression method or a drop-in replacement for standard attention in quality-critical settings.

Failure modes in sequence termination. DBA exhibits degraded performance on exact copy tasks requiring precise termination. In these cases, the model frequently recalls the target content correctly but fails to stop generation at the appropriate boundary. We hypothesize this reflects a fundamental challenge for bottlenecked attention: the end-of-sequence (EOS) signal is a low-redundancy, single-token event that must be detected precisely. Under depth-wise deferral, this signal competes with final-layer semantic aggregation, where the model is simultaneously routing to complete the current prediction. In baseline attention, EOS detection can occur earlier in the network with dedicated capacity; in DBA, the constrained semantic path may lack bandwidth to represent both "continue generating content" and "stop here" simultaneously. This remains a meaningful limitation for tasks requiring exact symbolic reproduction.

Scope of behavioral evaluation. Although the behavioral probe suite spans 117 tests across 15 categories, it is not exhaustive. The probes are designed to detect qualitative capability collapse and shifts in failure modes, not to provide comprehensive task coverage or statistical guarantees. DBA is not presented as a capability improvement, and its behavioral parity with the baseline should be interpreted within this diagnostic scope.

Training budget and variance. At scale (1B parameters, 100k steps), results are reported for a single seed. While qualitative trends are consistent with multi-seed experiments at smaller scale and with earlier training runs, variance at large scale is not fully characterized. Further large-scale seed sweeps would be required to quantify robustness more precisely.

Long-context behavior. Models are trained with a maximum context length of 2k tokens. Behavior at longer contexts, particularly under RoPE extrapolation, has not been validated. Although architectural accounting and short-range benchmarks suggest improved memory efficiency at long context, functional stability beyond the training context remains an open question.

Mechanistic interpretation. The "bandwidth-as-regularization" interpretation and related explanations discussed in Section 5.1 are speculative. While they are consistent with observed attention patterns and behavioral outcomes, this work does not establish causal mechanisms. The paper's primary contribution is empirical and architectural, independent of any particular explanatory framework.

Generality. DBA is evaluated only in the context of autoregressive language modeling. Its applicability to other settings—such as dense retrieval, multimodal architectures, or algorithmic tasks—remains unknown and may require higher interaction bandwidth.

6 Conclusion

This work introduces Decoupled Bottleneck Attention (DBA), an attention architecture that explicitly constrains the bandwidth of semantic token–token interaction while preserving higher-fidelity positional geometry. By decomposing attention into a low-rank semantic routing path and a higher-rank geometric path—with rotary position embeddings applied only to the latter—DBA reduces the dimensionality of attention scoring without uniformly compressing representations.

At 1B scale, DBA yields a 37.5% reduction in KV-cache footprint, a 1.6× memory reduction, and a 12% increase in cached decode throughput, measured end-to-end on identical hardware. These efficiency gains come with a 6% increase in held-out perplexity, reflecting a real and measurable quality cost. However, across an extended behavioral probe suite spanning 117 tests in 15 categories, DBA exhibits no systematic capability collapse: behavioral accuracy remains comparable to a standard attention baseline, with differences manifesting as shifts in failure modes rather than broad degradation.

These results characterize a concrete efficiency–quality trade-off in attention design. DBA is not a free compression method, nor a drop-in replacement for standard attention in quality-sensitive applications. Instead, it demonstrates that large portions of semantic interaction bandwidth can be removed while preserving functional behavior, at predictable cost and with measurable inference-time benefits.

Importantly, the contribution of this work is architectural and empirical. While several interpretive hypotheses may explain why constrained semantic interaction preserves behavior, no causal mechanism is established, and none is required for the result to hold. DBA's utility does not depend on any particular explanation, but on the observed stability of behavior under aggressive interaction bottlenecks.

More broadly, DBA highlights *interaction bandwidth* as a distinct and tunable axis in attention design, orthogonal to KV sharing, latent compression, and quantization. By making this axis explicit, DBA opens new points on the efficiency–quality Pareto frontier and invites further exploration of structured attention architectures that trade excess interaction capacity for practical inference gains.

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 gate $g_h \in [0, 1]$ that rescales semantic queries by $2g_h$ and geometric queries by $2(1 - g_h)$. This is the **best-performing variant** (-5.67 PPL vs. ungated, -7.3% vs. baseline), suggesting that learned head specialization along the semantic/geometric axis is beneficial. Not used in the primary 100k comparison to isolate the decoupling contribution.
- **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 13: 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.

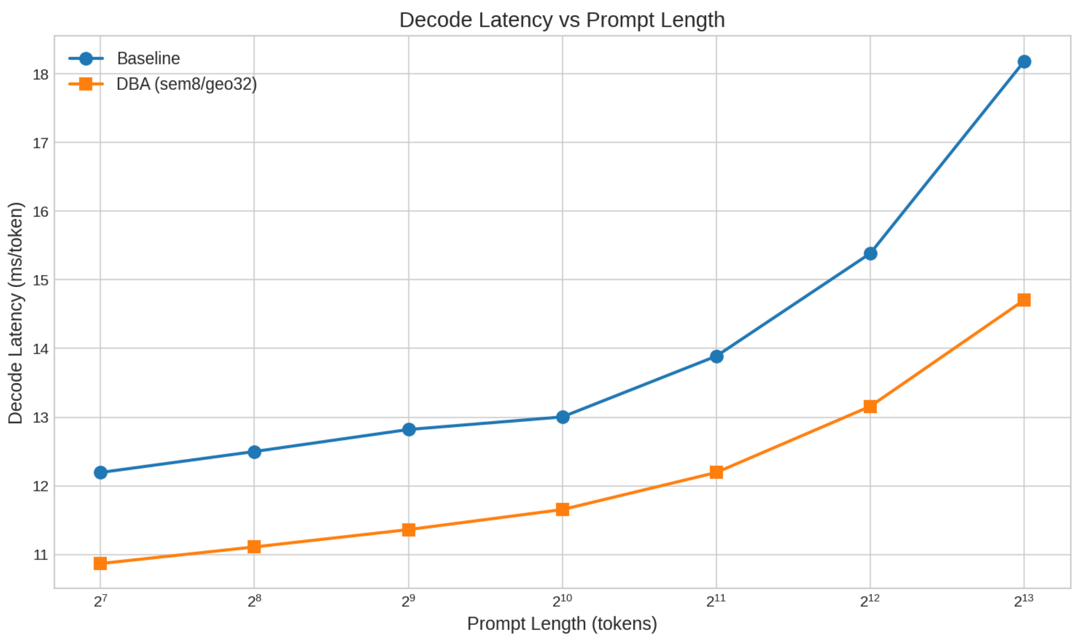


Figure 14: Cached decode latency microbenchmark (batch=1, fp16, Metal).

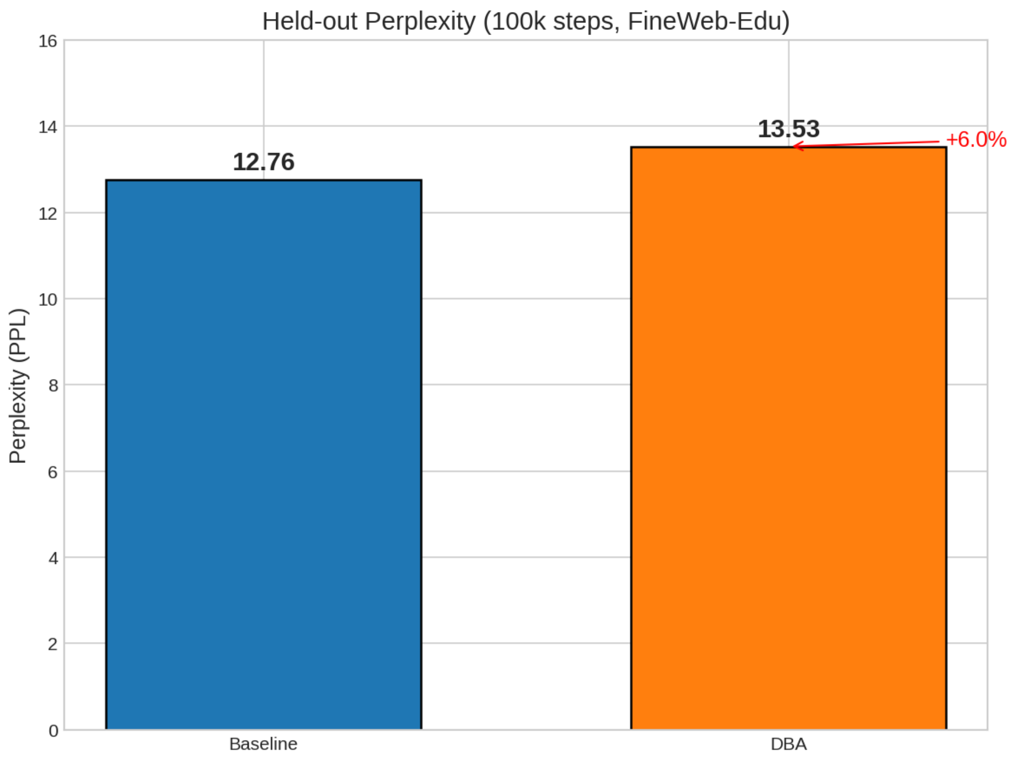


Figure 15: Held-out perplexity comparison (100k checkpoints, FineWeb-Edu). DBA incurs 6% higher perplexity.

References

[1] 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.

[2] 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`.

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

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

[5] 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.

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

[7] 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.

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

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

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

[11] 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.

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

[13] 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`.

[14] 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.

[15] Stephen Menary, Samuel Kaski, and Andre Freitas. Transformer normalisation layers and the independence of semantic subspaces. *arXiv preprint arXiv:2406.17837*, 2024. Shows Pre-Norm requires orthogonal semantic subspaces; analyzes attention as routing between independent subspaces.

[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.