

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 *semantic routing* (low-rank query–key interaction) from *positional geometry* (higher-rank routing), with RoPE applied only to the geometric path.

At 1B parameters (22 layers, 100k steps on FineWeb-Edu), **gated DBA (sem8)** reduces steady-state VRAM residency by $\sim 23\%$ ($77\% \rightarrow 59\%$ GPU memory allocated) while improving training throughput by $\sim 8\%$ ($23,675 \rightarrow 25,548$ tok/s), yielding a $\sim 1.4\times$ **improvement in tokens/s per % VRAM**. An **ungated DBA** variant isolates the effect of decoupling alone and shows smaller but consistent efficiency gains.

Under the behavioral evaluations used here (ungated DBA), we observe **no evidence of capability collapse**: on 117 probes, accuracy is comparable (26.5% vs 27.4%); on an expanded 490-probe suite, DBA *outperforms* baseline (39.3% vs 32.0%).

Overall, DBA makes *interaction bandwidth* a tunable axis in attention design and characterizes an efficiency–perplexity tradeoff in this setting.

Keywords: Transformer, attention mechanism, low-rank, structural regularization, KV-cache, memory efficiency

1 Introduction

Modern Transformer architectures [?] achieve strong performance but their inference cost scales poorly with context length. The key–value (KV) cache grows linearly with sequence length and dominates memory bandwidth.

Most efficiency work targets storage rather than interaction: sharing KV heads, compressing cached representations, or reducing precision. These approaches preserve the full-dimensional token–token interaction used to compute attention scores.

This paper asks: how much *semantic interaction bandwidth* is actually required for the behaviors we evaluate? We propose Decoupled Bottleneck Attention (DBA), which separates low-rank semantic routing from higher-rank positional geometry. The architecture and results are summarized in Table 1.

Table 1: DBA in context: comparison of attention efficiency methods.

Method	Reduces	Full interaction?	Composable?
GQA [?]	KV storage	Yes	Yes
MLA [?]	KV storage	Yes (expands)	Yes
Linformer [?]	Sequence complexity	Approximated	No
DBA (ours)	Interaction bandwidth	No	Yes

Existing methods reduce what is *stored* or *approximated*. DBA reduces what is *interacted*: the dimensionality of the query–key subspace used for semantic token–token matching. We refer to this as *interaction bandwidth*.

Empirically, DBA produces a notable pattern in this training regime: held-out perplexity worsens, yet behavioral probe accuracy does not collapse and can improve under expanded evaluation. This work does not claim that perplexity is uninformative; rather, it reports an instance where next-token likelihood and the behavioral measures used here are not tightly coupled, and uses DBA as a controlled way to study interaction bandwidth as an architectural constraint.

2 Related Work

Low-rank and approximate attention. Linformer [?] projects along the sequence dimension; Performer [?] and Reformer [?] use kernel approximations or hashing. DBA instead reduces the query/key interaction dimension within each layer, targeting both compute ($O(n^2r)$) and KV-cache ($O(nr)$).

Sparse attention. Longformer [?] and BigBird [?] use sparse patterns to reduce compute. DBA reduces per-token cache footprint and is complementary.

KV-cache optimization. MQA/GQA [?, ?] share KV heads; MLA [?] compresses to a latent expanded at runtime. Both preserve full interaction bandwidth. DBA reduces the interaction dimension itself and enables heterogeneous quantization (e.g., Q4 semantic, Q8 geometric).

Expressiveness limits. Low-rank attention can harm representation power [?, ?]. DBA mitigates this by compressing only the semantic path while preserving geometric fidelity.

Semantic subspaces. Menary et al. [?] show attention operates through independent semantic subspaces. DBA explicitly separates semantic and geometric subspaces, which may satisfy their orthogonality requirements.

3 Methodology

3.1 Standard Multi-Head Attention

In standard scaled dot-product attention with H heads:

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

where $Q, K, V \in \mathbb{R}^{n \times d}$ are obtained by linear projection from input $X \in \mathbb{R}^{n \times d_{\text{model}}}$. The KV-cache requires $O(2 \cdot L \cdot n \cdot d)$ memory.

3.2 Bottleneck Attention

We project Q and K to a lower-dimensional space *before* computing attention scores:¹

$$Q' = XW'_Q, \quad K' = XW'_K \quad (2)$$

where $W'_Q, W'_K \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}}$ with $d_{\text{attn}} \ll d_{\text{model}}$. This reduces dot-product complexity from $O(n^2 \cdot d_{\text{model}})$ to $O(n^2 \cdot d_{\text{attn}})$ and KV-cache proportionally.

¹Our use of “bottleneck” refers to dimensionality reduction in the query/key space, distinct from Park et al.’s BAM [?].

3.3 Decoupled Bottleneck Attention

The key insight is that *semantic routing* and *positional geometry* have different bandwidth requirements. DBA decomposes the attention score into two additive components:

$$\text{Score} = \underbrace{\frac{Q_{\text{sem}} K_{\text{sem}}^\top}{\sqrt{d_{\text{sem}}/H}}}_{\text{Semantic routing}} + \underbrace{\frac{Q_{\text{geo}} K_{\text{geo}}^\top}{\sqrt{d_{\text{geo}}/H}}}_{\text{Geometric routing}} \quad (3)$$

The semantic path uses $d_{\text{sem}} = 256$ total dimensions (8 per head with $H = 32$). The geometric path uses $d_{\text{geo}} = 1024$ (32 per head). RoPE [?] is applied *only* to the geometric path:

$$Q_{\text{geo}}, K_{\text{geo}} \leftarrow \text{RoPE}(Q_{\text{geo}}, K_{\text{geo}}, \text{position}) \quad (4)$$

The value projection is not compressed:

$$V = XW_V, \quad W_V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}}, \quad d_{\text{attn}} = d_{\text{sem}} + d_{\text{geo}} = 1280 \quad (5)$$

This preserves representational capacity downstream of routing: DBA constrains *which* tokens interact, not *what* information is aggregated.

Semantic/geometric gating. Our implementation optionally enables a learnable per-head gate $g_h = \sigma(\gamma_h) \in [0, 1]$:

$$Q'_{\text{sem},h} = 2g_h \cdot Q_{\text{sem},h}, \quad Q'_{\text{geo},h} = 2(1 - g_h) \cdot Q_{\text{geo},h} \quad (6)$$

Ablations show gating provides substantial improvement (-5.67 PPL at 10k steps). We validate this at scale with a separate 100k-step gated run (Section 4.5). In this paper, we report **gated DBA as the recommended configuration** and include the ungated variant as an ablation to isolate the decoupling contribution.

4 Experiments

We present experiments in chronological order: (1) architecture selection at 22 layers/10k steps, (2) main results at 22 layers/100k steps, (3) ablations at 12 layers/10k steps, and (4) gated DBA results.

4.1 Experimental Setup

All models use $d_{\text{model}}=2048$, 32 heads, and are trained on FineWeb-Edu with AdamW.² Table 2 summarizes configurations.

4.2 Architecture Selection (22L 1B 10k)

We selected **sem8** for 100k-step runs based on 10k-step comparisons: best efficiency score ($1.51\times$) with 37.5% KV-cache reduction and competitive downstream accuracy (1.2% gap). Subsequent 10k ablations identified **semantic/geometric gating** as the strongest design improvement; we therefore ran an additional 100k-step gated sem8 model to compare directly at the primary scale.

²Code and checkpoints: <https://github.com/theapemachine/caramba>.

Table 2: Experimental configurations. All use vocab = 50304, batch 32, $d_{\text{model}}=2048$.

Configuration	d_{sem}	d_{geo}	Head	d_{ff}	Steps	Seeds
22L Baseline	—	—	64	4096	10k	1337
22L sem16	512	1024	16+32	4096	10k	1337
22L sem8	256	1024	8+32	4096	10k	1337
22L Baseline	—	—	64	5632	100k	42
22L Decoupled	256	1024	8+32	5632	100k	42
12L Baseline	—	—	64	5632	10k	1337-39
12L Bottleneck	—	—	40	5632	10k	1337-39
12L Decoupled	256	1024	8+32	5632	10k	1337-39
12L GQA 32Q/4KV	—	—	64	5632	10k	1337-39
22L Gated	256	1024	8+32	5632	100k	42

4.3 Main Results (22 Layers, 100k Steps)

Training dynamics. Training loss converges smoothly across runs, with no divergence or late-stage regressions. For held-out perplexity, the ungated decoupled variant differs: **baseline 12.76 PPL vs DBA 13.53 PPL** (6.04% relative increase). For **gated DBA**, the held-out perplexity increase relative to baseline is **3.43%**.

Training efficiency. DBA reduces attention parameters by 37.5% (10.4% total). KV-cache elements drop by 37.5% (2560 vs 4096 per token). In steady-state W&B measurements (Table 3), **ungated DBA** improves throughput by **+5.6%** ($23,675 \rightarrow 25,009$ tok/s) while reducing GPU memory allocated from **77% to 68%**. **Gated DBA** improves throughput by **+7.9%** ($23,675 \rightarrow 25,548$ tok/s) while reducing GPU memory allocated to **59%**, yielding the best combined efficiency: **1.41× tokens/s per % VRAM** vs baseline.

Table 3: Performance, memory usage, and efficiency comparison (steady-state averages).

Metric	Baseline	Decoupled (sem8)	Gated sem8
Tokens / second (tok/s)	23,675	25,009	25,548
Speedup vs. baseline	1.00×	1.056×	1.079×
Step time (ms)	2,768	2,621	2,565
Step time reduction	–	–5.3%	–7.3%
GPU memory allocated (%)	77%	68%	59%
GPU memory reduction	–	–11.7%	–23.4%
Params + optimizer memory (GB)	17.43	15.62	15.62
Memory reduction (params+opt)	–	–10.36%	–10.36%
Tokens/s per GB (params+opt)	1,359	1,601	1,636
Efficiency vs. baseline	1.00×	1.18×	1.20×
Tokens/s per % VRAM	307	368	433
Efficiency vs. baseline	1.00×	1.20×	1.41×
Gradient norm (mean)	~1.0	~1.0	~1.0
Training stability	Stable	Stable	Stable

Table 4: Training step time breakdown (steady-state averages).

Metric	Baseline	Decoupled (sem8)	Gated sem8
Total step time (ms)	2,768	2,621	2,565
Step time vs. baseline	1.00×	0.947×	0.927×
Forward + backward (ms)	2,578	2,525	2,440
Fwd/bwd change vs. baseline	–	–2.1%	–5.4%
Optimizer step (ms)	189	95	125
Optimizer change vs. baseline	–	–50.0%	–33.9%
Optimizer share of step (%)	6.8%	3.6%	4.9%

4.3.1 Behavioral Probes: No Evidence of Collapse Under Probes

We evaluated 117 behavioral probes across 15 categories targeting exact copy, few-shot learning, distractor filtering, reasoning, arithmetic, and long-context retrieval.

Despite 6% higher perplexity, we do not observe a broad degradation across the probe suite; differences are task-specific. DBA outperforms on reasoning and world-knowledge; baseline performs better on distractor filtering and exact copy.

To stress-test this finding, we evaluated an expanded benchmark suite (490 probes, 18 categories) with additional adversarial, binding, and multi-hop tests. Under this suite, DBA scores 39.3% vs 32.0% soft accuracy, head-to-head 7–3.

Attention visualization. Baseline heads show diffuse attention spreading into distractor tokens; DBA heads concentrate attention near salient targets (as seen in our attention visualizations).

Table 5: Inference throughput benchmarks (Apple MPS, fp16, KV-cache enabled). Tok/s is computed from total generation time (prefill + decode) as implemented in the latency harness. **latency_cached**: prompt lengths 128–4096, generate 128 tokens, batch 1, fp16 KV-cache. **latency_long_context**: prompt lengths 2048–8192, generate 64 tokens, batch 1, fp16 KV-cache. (This is not the full 131k `context_sweep`.)

Metric	Baseline	Decoupled (sem8)	Gated sem8
Cached generation throughput (tok/s)	74.05	81.95	80.04
Speedup vs. baseline	1.00×	1.107×	1.081×
Long-context cached throughput (tok/s)	24.99	28.00	27.57
Speedup vs. baseline	1.00×	1.120×	1.103×

4.3.2 Inference Latency (Cached and 8k Context)

We report isolated inference-time throughput benchmarks run on Apple MPS³ with KV-cache enabled (fp16). Table 5 summarizes results for `latency_cached` and `latency_long_context` (up to 8k prompt length). A full long-context sweep to 131k tokens (`context_sweep`) is configured but not reported here.

4.4 Design Variant Ablations (12 Layers, 10k Steps)

Among the tested design variants at this scale, **gating produces the largest perplexity improvement** (−5.67 PPL). Tied Q-K degrades perplexity (+6.62 PPL). Removing RoPE degrades perplexity (+10.72 PPL). The null token has no measurable effect at this scale.

4.5 Gated DBA Results (22 Layers, 100k Steps)

The gating ablation showed a 12% relative PPL improvement at 10k steps—the largest effect of any design variant. To validate whether this improvement persists at scale, we trained a gated DBA variant at 22 layers for 100k steps (Table 2, final row). At scale, gating is also the strongest *systems* improvement: it achieves the lowest steady-state VRAM residency (59%) and the highest throughput (25,548 tok/s), and reduces step time primarily via forward/backward savings rather than optimizer time alone (Table 4).

5 Discussion

5.1 Why Does Constrained Semantic Interaction Preserve Behavior?

DBA yields a consistent pattern in this setting: held-out perplexity increases, while the behavioral probe performance we report is preserved and can improve under expanded evaluation. The following non-exclusive interpretations are consistent with this observation: (1) *Perplexity may reward modeling low-value correlations*—high-bandwidth attention may allocate capacity to weak or local correlations that improve next-token likelihood without improving the probe outcomes measured here; (2) *Routing decisions may be intrinsically low-entropy*—if attention primarily performs coarse token selection, high semantic interaction bandwidth may be unnecessary [?, ?]; (3) *Depth-wise deferral*—our attention visualizations show DBA early layers exhibit

³All inference latency results use the PyTorch MPS backend (`device=mps`, `dtype=float16`) with fp16 KV-cache. Hardware/OS: MacBook Pro (M4 Max, 128 GB unified memory; ~96 GB assignable to VRAM), macOS Sequoia 15.6. Absolute tok/s depends on the specific Apple Silicon chip, unified memory configuration, and OS/runtime, but the comparisons in Table 5 are measured on the same host under the same settings.

diffuse attention while later layers concentrate more sharply, which is consistent with constrained bandwidth shifting routing/synthesis toward later depth.

These explanations are hypotheses rather than established mechanisms. In particular, our visualizations are correlational and do not isolate causality. Targeted ablations (e.g., hybrid stacks that switch between full attention and DBA at a chosen depth, or analysis of learned gate values when gating is enabled) would help distinguish among these possibilities.

5.2 Limitations

Perplexity increase. Held-out perplexity increases relative to baseline in this setting: **+6.04%** for ungated DBA and **+3.43%** for gated DBA. Applications that prioritize next-token likelihood may not tolerate this difference.

Termination failures. DBA degrades on exact copy tasks requiring precise stopping. The EOS signal may compete with final-layer semantic aggregation.

Evaluation scope. The 117-probe suite detects capability collapse, not comprehensive task coverage.

Scale and variance. Large-scale results use a single seed. Multi-seed validation exists only at 12L/550M scale.

Long context. Models train on 2k context. Behavior under RoPE extrapolation is not validated.

Generality. DBA is evaluated only for autoregressive language modeling.

6 Conclusion

Decoupled Bottleneck Attention demonstrates that semantic routing and positional geometry can be factored into separate routing paths within attention. In this setting (1B parameters, 22 layers, 100k steps), **gated DBA (sem8)** reduces steady-state VRAM residency from 77% to 59% while improving training throughput from 23,675 to 25,548 tok/s, yielding a **~1.4× improvement in tokens/s per % VRAM**. The ungated variant retains the core decoupling mechanism and shows smaller but consistent gains.

This work establishes *interaction bandwidth* as a tunable axis in attention design, orthogonal to existing techniques like KV sharing and quantization. DBA can be composed with both to further explore the efficiency–perplexity frontier. The behavioral preservation observed here is consistent with the hypothesis that some next-token likelihood gains can come from modeling correlations that are not necessary for the probe outcomes measured in this paper, but this hypothesis is not proven by the current experiments.

Our experiments were conducted at 1B scale with limited training compute. Whether the perplexity gap narrows, holds, or widens at competitive scale remains to be determined. The behavioral outcomes reported here motivate larger-scale investigation and targeted mechanistic ablations.

Statements and Declarations

Conflict of Interest. The author declares no competing interests.

Data Availability. All code, checkpoints, and logs are available at <https://github.com/theapemachine/caramba>.

Funding. This research was conducted without external funding.

A Decoupled Ablations

We evaluated four DBA design variants at 12 layers. The **null token** variant adds a learnable “sink” token but shows no measurable effect at this scale. **Tied Q-K** shares semantic query and key projections, which hurts performance (+6.62 PPL), indicating that asymmetric query–key interactions are important. **Gating** is the best variant (−5.67 PPL): a learnable per-head gate allows each head to balance semantic and geometric contributions. Finally, removing **RoPE** from the geometric path is catastrophic (+10.72 PPL), confirming that positional structure is essential even when decoupled from semantic routing.

B Auto-generated Benchmark Artifacts (100k)

C Benchmark Protocol (Inference-Time)

Inference benchmarks are defined by the manifest `research/dba/benchmark-gated.yml` and executed by the Caramba benchmarking framework (`benchmark/`). In this paper we report `ppl_fineweb`, `latency_cached`, and `latency_long_context`; a full `context_sweep` to 131k tokens is configured in the same manifest but not reported here.

Perplexity (`ppl_fineweb`). Perplexity is computed as $\exp(\bar{\ell})$, where $\bar{\ell}$ is the average token-level cross-entropy over a fixed number of tokens. The benchmark iterates over `fineweb_20b.npy` with `block_size=2048`, `batch_size=1`, `num_batches=200`, computing cross-entropy with `reduction="sum"` and dividing by the total token count. To ensure comparability when model vocabularies are padded (e.g., 50304 vs GPT-2’s 50257), logits are sliced to `valid_vocab_size=50257` before loss computation; the benchmark also fails fast if dataset token IDs exceed `valid_vocab_size`.

Latency (`latency_cached`, `latency_long_context`). Latency uses the KV-cache generation path (`use_cache=true`, `cache_kind=fp16`) with greedy decoding (`temperature=0.0`). For each (prompt length, generation length, batch size) configuration, inputs are deterministic random token IDs sampled uniformly from $[0, \text{valid_vocab_size}]$ with a seed derived from the configuration, and an audit hash of the input IDs is recorded. The benchmark runs `warmup_runs` un-timed iterations. For each of `timed_runs`, it pre-allocates KV caches before timing, then measures prefill time and decode time; reported tok/s is computed from total generation time (prefill + decode) as $\text{gen_len} \cdot \text{batch_size}/t$.

Isolation and reproducibility. For multi-model comparisons, the benchmarking runner isolates the active model on the accelerator by moving non-active models to CPU and clearing backend caches between runs. The benchmark suite emits a machine-readable report (including full config and raw measurement traces) under `research/dba/benchmark100k_gated/`.