Bottleneck Attention: Hard-Wiring Low-Dimensional Routing in GPT-Style Transformers

Danny (The Ape Machine) | December 2025

Abstract

Adaptive Low-Rank Training (ALRT) suggests that Transformer attention contains extreme redundancy: in prior experiments, query/key (Q/K) projections compress to ranks approximately 9-14 with mean approximately 11.2 out of 512, implying that attention pattern computation may live in a small comparison subspace. Motivated by this signal and a first-principles 'question everything' approach to architecture design, we test a minimal hypothesis: decouple residual stream width d_model from attention subspace dimension d_attn, using Q,K,V: R^{d_model}->R^{d_attn} with d_attn << d_model, and map attention output back via R^{d_attn}->R^{d_model}. On a controlled WikiText-2 word-level setup (6-layer GPT-style LM; d_model=512, 8 heads, context 256), reducing d_attn from 512 to 128 (4x) and to 32 (16x, with a learned null key/value 'attend nowhere' option) yields only modest degradation in best validation loss (+2.1% and +4.4% relative), while attention compute and KV-cache memory scale linearly with d_attn. These results support the interpretation that, in this regime, attention primarily acts as a low-dimensional routing primitive distinct from high-rank feature processing in MLP blocks.

1. Introduction

Transformers remain architecturally conservative despite dramatic changes in scale, hardware, and use cases since their introduction. Many design choices persist because they work, not because they are known to be optimal.

ALRT was motivated by a simple suspicion: much of Transformer parameterization is redundant, and this redundancy is not uniform across components. Empirically, ALRT finds strong projection-type asymmetry: attention Q/K projections compress far more aggressively than attention output and MLP projections, suggesting routing may be intrinsically low-dimensional while writing back is not.

This report tests the most literal architectural consequence: if attention pattern computation truly lives in a small subspace, we can hard-wire that by decoupling attention's internal dimension from the residual stream width. Unlike dynamic rank-resizing, this is a static architectural change trained from scratch.

2. Background: ALRT and the Low-Rank Attention Signal

ALRT parameterizes linear maps as low-rank factors W = UV and adapts rank during training using stable rank estimation, r_stable(W) = ||W||_F^2 / sigma_max(W)^2.

Across GPT-style language modeling experiments, ALRT reports that query/key projections are the most compressible, with mean ranks around 11.2 out of 512, while attention output and MLP projections retain higher effective ranks. A natural architectural interpretation is that attention's comparison space is low-dimensional, even if the residual stream is not.

3. Method: Bottleneck Attention

We replace full-width Q/K/V with projections into a smaller attention subspace: W_Q, W_K, W_V in R^{d_model x d_attn}, and project the attention output back via W_O in R^{d_attn x d_model}. Attention score and apply operations scale in d_attn, while the residual stream stays at d_model.

Optional 'attend nowhere' (null attention): softmax forces each query to allocate probability mass to some key. We append a learned per-head null key/value (extra column) that is always visible, allowing the model to place mass on a no-op sink when no context token is relevant.

Optional learned per-head temperature: when d_attn is small, dot products can become noisy and softmax can saturate. A learned logit scale per head (temperature) improves stability in tiny head dimensions.

4. Experimental Setup

Dataset: WikiText-2 using a minimal word-level whitespace tokenizer with line boundaries and for OOV. The single wiki.train.tokens file is split 90/10 into train/validation within the script.

Model: decoder-only GPT-style Transformer with pre-norm blocks; 6 layers; d_model=512; 8 heads; d_ff=2048; context length 256; dropout 0.1; weight tying between input embedding and LM head.

Optimization: AdamW with lr=3e-4, weight decay=0.01, betas=(0.9,0.95), batch size 32, 200 steps/epoch, 30 epochs (6000 steps), gradient clipping 1.0. Evaluation every 200 steps (50 eval iters). Hardware: Apple MPS backend.

5. Results

Table 1 summarizes best validation metrics across three configurations. Reducing d_attn yields a smooth degradation curve rather than a catastrophic cliff.

Model	d_attn	Params (M)	Best val loss	Best val ppl
Baseline	512	36.06	5.3688	214.61
Bottleneck	128	31.34	5.4800	239.85
Extreme+Null	32	30.16	5.6070	272.33

Table 1: WikiText-2 (word-level) results. Best validation metrics over training.

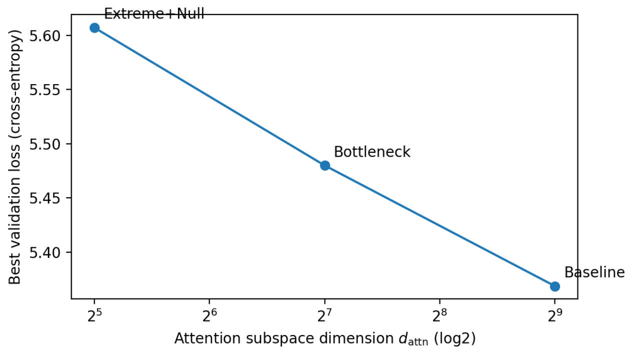

Figure 1: Best validation loss vs attention subspace dimension d_attn (log2 x-axis).

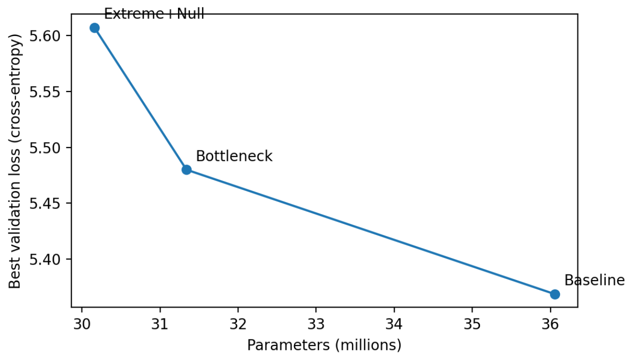

Figure 2: Pareto view: parameters (millions) vs best validation loss.

6. Discussion

Attention as low-dimensional routing: The viability of d_attn=32 suggests that attention can route effectively using a small comparison/mixing subspace, consistent with ALRT's observation that Q/K matrices collapse to tiny effective rank. A useful mental model is 'wide residual stream + narrow router.'

Why degradation is smooth: If high-dimensional geometry were essential for core competence, we would expect collapse once d_attn falls below a critical threshold. Instead, loss increases gradually, suggesting many attention patterns in this regime are representable with relatively few degrees of freedom.

Compute allocation shifts: Bottlenecking attention sharply reduces attention's share of compute, pushing the bottleneck to the FFN and LM head in this setting. KV-cache memory savings, however, are immediate and scale linearly with d_attn, important for long-context inference.

7. Related Work

Low-rank adaptation methods (e.g., LoRA) show that many learned transformations can be modified via low-dimensional updates, implying low intrinsic dimensionality in practice.

Efficient-attention methods address the quadratic cost of attention via sparsity or kernel/low-rank approximations (e.g., Reformer, Linformer, Performer). In contrast, this work tests whether standard attention is over-provisioned in its internal dimensionality even at modest context lengths.

Designs that reduce KV memory (e.g., multi-query or grouped-query attention) are complementary: bottleneck attention reduces both compute and KV dimensionality by shrinking the attention subspace itself.

8. Limitations and Future Work

Context scaling: these experiments use context length 256; required d_attn may grow with longer context.

Positional encoding: learned absolute positions were used; interactions with RoPE/ALiBi remain to be tested.

Variance: results are from single runs; multi-seed sweeps are needed for confidence intervals.

Minimal next experiments: (1) isolate null-slot impact at d_attn=32 with/without null; (2) sweep d_attn in {16,24,32,48,64,96,128}; (3) repeat at longer contexts; (4) log mean attention mass assigned to the null slot by layer/head.

9. Conclusion

ALRT indicates that attention pattern computation is intrinsically low-rank, with Q/K compressing to mean ranks around 11.2/512.

Bottleneck Attention operationalizes this as a static architectural change: compute attention in a reduced subspace d_attn << d_model while preserving a wide residual stream. On WikiText-2 (word-level), d_attn=128 and even d_attn=32 (with a null slot) remain viable with modest loss penalties, implying that the standard convention d_attn=d_model is substantial structural bloat in this regime.

References

A. Vaswani et al. Attention Is All You Need. NeurIPS, 2017.

E. J. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685, 2021.

N. Kitaev, L. Kaiser, A. Levskaya. Reformer: The Efficient Transformer. ICLR, 2020.

S. Wang et al. Linformer: Self-Attention with Linear Complexity. arXiv:2006.04768, 2020.

K. Choromanski et al. Rethinking Attention with Performers. ICLR, 2021.

Adaptive Low-Rank Training via Spectral Rank Estimation: Dynamic Parameter Efficiency in Transformer Language Models (Technical Report), Dec 2025.

Adaptive Low-Rank Training - Technical Report v2, Dec 2025.

Appendix A: Reproducibility Commands

```
Baseline:
python3.12 v19_transformer_attn_bottleneck.py \
  --data ./wiki.train.tokens --out-dir runs/v19_baseline --attn-dim 512

Bottleneck:
python3.12 v19_transformer_attn_bottleneck.py \
  --data ./wiki.train.tokens --out-dir runs/v19_attn128 --attn-dim 128

Extreme + null:
python3.12 v19_transformer_attn_bottleneck.py \
  --data ./wiki.train.tokens --out-dir runs/v19_attn32_null --attn-dim 32 --null-attn
```