Adaptive Low-Rank Training via Spectral Rank Estimation

Dynamic Parameter Efficiency in Transformer Language Models

Technical Report v2 — December 2025

Abstract

We present Adaptive Low-Rank Training (ALRT), a method for training neural networks with dynamically adjusted low-rank weight factorization. Our approach uses stable rank estimation via power iteration to determine the intrinsic dimensionality of learned representations, enabling automatic discovery of layer-specific compression ratios. This report documents the evolution of ALRT through multiple iterations (v10-v18), including exploration of alternative hypotheses such as pressure-based compression and scheduled rank decay. Our latest implementation (v18) demonstrates stable compression from rank 64 to average rank 42 on a 6-layer GPT-style transformer while maintaining strong language modeling performance, with validation perplexity improving from 791 to 4.1 over training. We also present a first-principles interrogation framework for systematic analysis of transformer architectural efficiency, identifying promising directions for future research.

1. Introduction

Modern neural networks are typically overparameterized, with weight matrices containing substantial redundancy. Our previous work demonstrated that transformer attention projections can be compressed from rank 512 to rank 11 (98% reduction) while maintaining model quality. This finding motivates a broader investigation into adaptive rank methods that can automatically discover layer-specific compression ratios during training.

This report documents the evolution of our Adaptive Low-Rank Training (ALRT) methodology through versions 10-18, including several experimental branches that explored alternative compression hypotheses. We present both successful developments and instructive failures, providing a complete picture of the research trajectory.

2. Core ALRT Method

2.1 Low-Rank Parameterization

We parameterize each linear transformation W as the product of two factors: W = UV, where U has shape (out_features, rank) and V has shape (rank, in_features). This reduces parameters from $m \times n$ to $r(m+n)$ when $r \ll \min(m,n)$, and critically reduces gradient computation complexity in the backward pass.

2.2 Stable Rank Estimation

The stable rank provides a continuous, robust measure of effective dimensionality:

$$r\_stable(W) = ||W||^2\_F / \sigma^2\_max(W)$$

We compute this efficiently from the low-rank factors without materializing W:

- **Frobenius norm:** $||W||^2\_F = tr((U^TU)(VV^T))$ — computed in $O(r^2(m+n))$ instead of $O(mn)$
- **Spectral norm:** σ_max estimated via 5 iterations of power method using matvecs through U and V

2.3 Rank Controller

The rank controller maps noisy stable rank estimates to smooth integer rank trajectories:

- **Safety factor (α = 1.5):** Target = α × r_stable, providing headroom for expansion
- **EMA smoothing (β = 0.8):** Prevents oscillation from noisy estimates
- **Change threshold (Δ = 2.0):** Rank only changes when smoothed target differs by ≥Δ
- **Bounded step (s = 4):** Maximum ±4 rank change per update for stability
- **Bidirectional:** Rank can increase if network requires more capacity

2.4 Rank Resizing via Truncated SVD

When rank changes, we preserve learned representations via truncated SVD with "sqrt split":

- Reconstruct W = UV on CPU for numerical stability
- Compute SVD: W = U_svd Σ V_svd
- Truncate to r_new components
- Split singular values: U_new = U_svd √Σ, V_new = √Σ V_svd

The sqrt split distributes energy equally between factors, maintaining balanced gradient flow. After any resize, the optimizer is reinitialized to clear stale momentum terms.

3. Implementation Evolution

3.1 Version History Overview

Version	Approach	Key Features	Status
v10	Spectral energy-based rank	WikiText-2, 6-layer GPT, 37.79M params	✓ Baseline established
v15	Adaptive gradient-based	Per-layer spectral rank, tail energy	✓ Strong differentiation
v16	Pressure cooker	Global pressure, head/block gates	⚠ Over-compressed (rank 4)
v17	Scheduled + regularizer	Fixed schedule, L2 pressure penalty	⚠ Branch (different hypothesis)
v18	**ALRT (stable rank)**	**Power iteration, SVD resize, KV cache**	**✓ Current mainline**

Table 1: Version history showing evolution of ALRT methodology.

3.2 The v15-v16 "Pressure Cooker" Experiments

v16 introduced an aggressive "pressure controller" that attempted to drive compression through multiple mechanisms simultaneously: spectral energy targets, learned head gates, learned block gates, and a global pressure coefficient. While conceptually interesting, this approach collapsed all ranks to the minimum (4) while maintaining high validation loss (~6.3), indicating the model was over-compressed beyond its representational capacity.

Key lesson: Multiple interacting compression mechanisms can destabilize training and lead to catastrophic under-fitting. The ALRT approach of per-layer stable rank estimation with conservative controllers proved more reliable.

3.3 The v17 Branch Divergence

v17 diverged from the ALRT mainline by replacing adaptive rank inference with a predetermined schedule (rank decays linearly from init to final over warmup steps). It also replaced stable rank estimation with an L2 regularization "pressure" term that penalizes later singular components more heavily.

While v17 is not "wrong," it represents a different research hypothesis: scheduled compression with soft regularization versus adaptive compression via spectral analysis. We maintain v17 as a separate branch for future investigation but returned to stable-rank-based ALRT for v18.

3.4 v18: Return to ALRT Principles

v18 returns to the core ALRT methodology with several improvements:

- **Efficient stable rank:** Computed from factors without materializing W
- **Per-layer controllers:** Each of 36 low-rank layers has independent rank trajectory
- **KV-cached inference:** Generation mode with proper caching for efficiency
- **Comprehensive logging:** JSONL file logging with per-layer rank change events
- **Checkpoint support:** Save/resume with complete controller state

4. v18 Experimental Results

4.1 Training Configuration

v18 was trained on a character-level language modeling task with the following configuration:

Parameter	Value
Model dimension (d_model)	512
Number of layers	6
Attention heads	8
Context length	256 tokens
Vocabulary size	284 (character-level)
Low-rank layers	36 (Q/K/V/O + FFN up/down per block)
Initial rank	64
Rank update frequency	Every 1184 steps (~1 epoch)
Hardware	Apple M4 Ultra (MPS backend)

Table 2: v18 training configuration.

4.2 Training Progression

Table 3 shows the training progression over 7600+ steps (~6.4 epochs):

Step	Train Loss	Val Loss	Val PPL	R_min	R_avg	R_max
1	6.67	—	791	64	64.0	64
500	2.38	2.37	10.74	64	64.0	64
1184*	2.11	—	~8.2	60	60.4	64
2368*	1.84	1.75	5.74	56	56.8	64
4736*	1.54	1.50	4.50	48	49.3	61
7600	**1.40**	**1.41**	**4.10**	**40**	**42.4**	**53**

*Table 3: v18 training progression. Steps marked with * indicate rank update events.*

4.3 Key Observations

1. **Stable compression:** Rank decreased smoothly from 64 to average 42.4 (34% reduction) over 6 rank updates, with no training instability.
2. **Layer differentiation emerging:** R_min dropped to 40 while R_max stayed at 53, indicating per-layer adaptation is occurring. Full differentiation (Q/K ~11, O ~23 as seen in v10) requires longer training.
3. **Throughput maintained:** ~88-90k tokens/second throughout training, with only brief dips during rank update SVD computations (~1s per update).
4. **Validation tracks training:** Val loss closely follows train loss (1.41 vs 1.40), indicating no overfitting despite compression.

5. Comparison: v15 Layer Differentiation

The v15 experiment ran for 30 epochs and achieved strong layer differentiation. Final per-layer ranks at step 8000:

Layer Type	Min Rank	Max Rank	Pattern
Q projections	16	26	Low (compressible)
K projections	12	21	Lowest (most compressible)
V projections	16	63	Variable
FFN up (ff1)	32	58	Medium-high
FFN down (ff2)	50	56	High (resists compression)

Table 4: v15 layer-specific rank ranges after 30 epochs, showing clear differentiation by projection type.

The v15 results confirm our hypothesis that different projection types have fundamentally different intrinsic dimensionalities. Q/K projections (which compute attention patterns) compress most aggressively, while FFN down-projections (which map from hidden space back to model dimension) resist compression most strongly.

6. Future Directions: First-Principles Architecture Interrogation

Our ALRT work has revealed that transformer weight matrices contain ~98% redundancy in some projections. This raises a fundamental question: what other architectural inefficiencies are hiding in plain sight?

We have developed a systematic "First-Principles Interrogation" framework for analyzing transformer architecture efficiency. Key questions include:

Attention Mechanism

- If Q/K compress to rank 11 from 512, what does this imply about what attention is actually learning?
- Could Q and K be the same projection? What would we lose?
- Is softmax (forcing attention to be a probability distribution) the right inductive bias?
- Should a token be able to "attend nowhere"—to say "this position has no relevant context"?

Feed-Forward Networks

- Why the 4× expansion factor? Is this ratio still optimal on modern hardware?
- Our data shows FFN up-projections compress more than down-projections. Why the asymmetry?
- If FFN acts as key-value memory (as some research suggests), should it be structured as such explicitly?

Computation Allocation

- Every token gets the same FLOPs regardless of difficulty. Is this efficient?
- How much variation in difficulty exists between tokens? 10×? 100×?
- What signal indicates "this token needs more compute"?

7. Conclusion

This report documents the evolution of Adaptive Low-Rank Training through versions 10-18, including experimental branches (v16 pressure cooker, v17 scheduled compression) that explored alternative hypotheses. Key findings:

1. **Stable rank estimation works:** The power-iteration-based stable rank provides reliable guidance for adaptive compression.
2. **Layer differentiation is real:** Different projection types have fundamentally different intrinsic dimensionalities (Q/K: 12-26, FFN-down: 50-56).
3. **Conservative controllers are essential:** The v16 pressure cooker over-compressed catastrophically; bounded-step controllers prevent this.
4. **v18 provides a solid foundation:** With efficient stable rank computation, per-layer controllers, SVD-based resizing, and proper logging, v18 is ready for larger-scale experiments.

Next steps include running v18 to convergence to observe full layer differentiation, comparing parameter savings at inference time, and exploring the first-principles interrogation questions to identify further architectural inefficiencies.

Appendix A: v18 Usage

Training

```
python3 v18_transformer_lowrank_alrt.py \
    --mode train --data data.txt --out-dir runs/v18
```

Resume Training

```
python3 v18_transformer_lowrank_alrt.py \
    --mode train --data data.txt --out-dir runs/v18 \
    --resume runs/v18/last.pt
```

Generation

```
python3 v18_transformer_lowrank_alrt.py \
    --mode generate --ckpt runs/v18/best.pt \
    --prompt "Once upon a time" --max-new-tokens 400
```

Key Controller Parameters

- **--safety-factor:** Multiplier on stable rank (default: 1.5)
- **--ema-decay:** EMA smoothing factor (default: 0.8)
- **--change-threshold:** Minimum delta to trigger change (default: 2.0)
- **--max-step:** Maximum rank change per update (default: 4)
- **--resize-method:** "svd" (best fidelity) or "truncate" (faster)