Block-Level Weight-Space Structure Persists Under Post-Training: An Empirical Study Across LLM Families

Zhaohui Wang [1]

Abstract

Modern LLMs are deployed as families of post-trained variants (base, instruct, chat, code) derived from a shared set of pre-trained weights. We present an empirical study of how post-training transforms weight-space geometry, covering eight configurations across four architecture families (Qwen2.5, Llama-3.1/3.2, Mistral, Gemma-2). We identify a *granularity gap*: post-training modifies every tensor (zero of 291–339 tensors remain byte-identical, so hash-based deduplication achieves 0% savings), yet preserves block-level structure (mean cosine similarity exceeds 0.99 and relative Frobenius distance stays below 0.13). Post-training therefore acts as a *structured perturbation* that shifts every parameter while leaving block-level geometry intact. The property is not universal: independently trained specializations (e.g., Qwen2.5-Coder) attain cosine similarity $\sim$0.64 with the general base, indicating a disconnected region of weight space. Perturbation magnitude varies systematically with model scale, architecture family, and post-training recipe. As a practical application, we build **LinkerLLM**, a lazy loader that aliases shareable blocks across co-resident variants, achieving 18–48% GPU memory savings and enabling up to five 7B-parameter variants on a single 24 GB consumer GPU. Five of eight configurations retain $\geq$94% of the unshared variant's quality on MMLU, ARC-Challenge, HellaSwag, and WinoGrande; the remaining three (Mistral-7B, Gemma-2-2B, Llama-3.2-1B) have one below-threshold benchmark each (87–91%), which we report transparently rather than gate the block-sharing decision on a single threshold.

[1]Viterbi School of Engineering, University of Southern California, Los Angeles, USA. Correspondence to: Zhaohui Wang <zwang000@usc.edu>.

Workshop on Weight-Space Symmetries, held in conjunction with the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. 2026.

1. Introduction

How do training procedures transform the geometry of neural network weights? Linear mode connectivity (Frankle et al., 2020; Entezari et al., 2022) established that models fine-tuned from the same initialization remain connected by low-loss paths, and Git Re-Basin (Ainsworth et al., 2023) extended this to independently trained models via permutation alignment. We study a specific, practically motivated instance of this question: *what happens to weight-space structure when a pre-trained LLM undergoes post-training?* Post-training (SFT, RLHF, DPO (Grattafiori et al., 2024; Yang et al., 2024)) produces specialized variants (instruct, chat, code) sharing a common origin.

Our study reveals a previously undocumented **granularity gap**: post-training perturbs the weight space at the tensor level (every tensor changes; hash-based deduplication is useless) while preserving structure at the block level (cosine $>$0.99, Frobenius ratio $<$0.16 across all eight tested configurations). The property is not universal: independently trained specializations such as Qwen2.5-Coder-7B fall outside the block-connected neighborhood (cos $\sim$0.64). We further verify that block-level alignment is *not* a consequence of the architecture's permutation symmetry: a function-preserving permutation drops the cosine by three orders of magnitude. As a practical application, we build **LinkerLLM**, a runtime loader that aliases shareable blocks across co-resident variants, achieving 18–48% GPU memory savings on 7B-class models on a 24 GB consumer GPU.

Related work. The mode connectivity literature characterizes basins at the whole-model level (Frankle et al., 2020; Entezari et al., 2022; Ainsworth et al., 2023; Theus et al., 2025); our analysis refines the picture at the *block* level. Task-vector approaches (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024; Wortsman et al., 2022; Sun & Dredze, 2025) operate on weight-space *deltas*; we characterize the *structure* of those deltas. Multi-model serving systems share LoRA adapters (Sheng et al., 2024; Chen et al., 2024; Hu et al., 2022) or compress deltas (Yao et al., 2025; Wang et al., 2026); our empirical finding explains why tensor-hash dedup fails and motivates block-level sharing instead. Concurrent work (Zhong & Raghunathan, 2025) reads structure off fine-tuned weights for monitoring and control; our focus

is structural quantification and memory-efficient deployment.

2. Methodology

For a transformer block i with sub-parameters $\{W_k^{(i)}\}_{k=1}^K$ (attention projections, MLP weights, layernorms), we define two complementary metrics between a base model and a post-trained variant:

$$\mathrm{sim}(i) = \frac{1}{K}\sum_{k=1}^{K} \frac{\langle \mathrm{vec}(W_{k,\mathrm{base}}^{(i)}),\ \mathrm{vec}(W_{k,\mathrm{var}}^{(i)})\rangle}{\|\mathrm{vec}(W_{k,\mathrm{base}}^{(i)})\| \cdot \|\mathrm{vec}(W_{k,\mathrm{var}}^{(i)})\|} \quad (1)$$

$$\epsilon(i) = \frac{1}{K}\sum_{k=1}^{K} \frac{\|W_{k,\mathrm{base}}^{(i)} - W_{k,\mathrm{var}}^{(i)}\|_F}{\|W_{k,\mathrm{base}}^{(i)}\|_F} \quad (2)$$

Cosine measures *directional* preservation; Frobenius ratio measures *magnitude* of perturbation. Both are needed: a block can have high cosine but large Frobenius (e.g., uniform scaling), or vice versa. A block is *shareable* when both pass per-family thresholds:

$$\mathrm{Share}(i) \iff \mathrm{sim}(i) \geq \tau_{\cos} \ \wedge\ \epsilon(i) \leq \tau_{\mathrm{frob}} \quad (3)$$

We study eight base→instruct pairs across four architecture families (Qwen2.5-0.5/3/7B, Llama-3.2-1/3B, Llama-3.1-8B, Mistral-7B-v0.3, Gemma-2-2B). Weights are compared in float32 after conversion from the model's native precision; full configuration details appear in Appendix A.

3. Results

3.1. The Granularity Gap

Tensor level. For Qwen2.5-7B base vs. instruct, only 2 of 339 tensors are byte-identical (both small bias vectors); for Llama-3.1-8B, zero of 291 match. Post-training produces a *dense* perturbation, with no tensor left untouched.

Block level. Despite per-tensor changes, transformer blocks retain high similarity (Table 1, Figure 1): cosine >0.99 across all eight configurations and Frobenius ratio <0.05 for six of eight families. The perturbation is "block-structured": it modifies every tensor within a block by a small, correlated amount that preserves the block's function. The block-level mean is representative: per-sub-parameter cosines on Qwen2.5-7B block 14 range from 0.9997 to 1.0000 with std. 0.009 (Appendix G).

Correlates of perturbation magnitude. The mean Frobenius ratio $\bar{\epsilon}$ varies with (i) model scale (sub-1B Qwen models receive 3–4× larger perturbation than ≥3B), (ii) training methodology (predominantly-SFT Mistral-v0.3 has $\bar{\epsilon}$=0.006 while heavier-RLHF Llama-3.2 reaches 0.110–0.156), and (iii) release vintage (Mistral-v0.1→v0.3 and Gemma-1→Gemma-2 show order-of-magnitude tightening between 2023 and 2024 releases, though Llama-2→Llama-3.1 is essentially flat). These correlations are confounded by simultaneous differences in architecture and data; controlled studies would be needed to disentangle them. Full analysis: Appendix I.

Table 1. Block-level similarity between base and instruct variants. Families ordered by mean Frobenius ratio (ascending = more preserved).

Family	Blocks	cos range	ϵ range	ϵ mean
Mistral-7B-v0.3	32	0.999+	0.004–0.01	0.006
Qwen2.5-3B	36	0.999+	0.01–0.02	0.011
Qwen2.5-7B	28	0.999+	0.01–0.02	0.015
Llama-3.1-8B	32	0.999+	0.04–0.05	0.045
Qwen2.5-0.5B	24	0.998	0.04–0.05	0.045
Gemma-2-2B	26	0.998–0.999	0.04–0.08	0.049
Llama-3.2-3B	28	0.991–0.995	0.10–0.12	0.110
Llama-3.2-1B	16	0.990–0.995	0.12–0.17	0.156

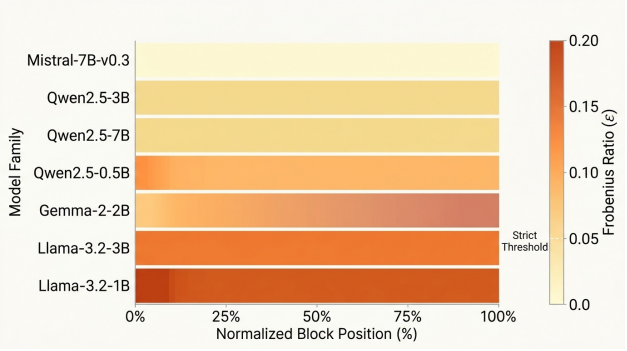

Figure 1. Per-block Frobenius ratio $\epsilon(i)$ across model families (base→instruct). Mistral and Qwen families have uniformly low perturbation (all blocks near-frozen). The Llama-3.2 series shows 10–20× larger per-block perturbation than Mistral, consistent with more aggressive RLHF; the Llama-3.1-8B variant lies between these regimes ($\bar{\epsilon} = 0.045$), suggesting that perturbation magnitude is not a fixed property of the Llama family but varies with the specific post-training recipe used per release.

Perturbation magnitude vs. downstream capability gain. Figure 2 pairs the per-family $\bar{\epsilon}$ with the MMLU gain $\Delta\mathrm{MMLU} = \mathrm{MMLU}_{\mathrm{instruct}} - \mathrm{MMLU}_{\mathrm{base}}$ measured under matched lm-eval-harness settings (5-shot, n=30 per subject, single seed). Pearson $r = 0.819$ on $n = 8$ families: families with the smallest $\bar{\epsilon}$ (Mistral, Qwen-3B, Qwen-7B) show essentially no MMLU change or a small (<2 pt) decrease under instruction-tuning, while families with larger $\bar{\epsilon}$ (Gemma-2-2B, Llama-3.2-1B, Llama-3.2-3B) accrue 5–9 pt of MMLU gain. We caution against over-interpreting this on $n = 8$: SFT/RLHF objectives do not explicitly optimize MMLU, the $n = 30$ subset has ±1.2 pt noise, and capability gain on instruction-following benchmarks (IFEval, AlpacaEval) is the more direct training target. The positive correlation does, however, support the qualitative reading that larger weight-space perturbation accompanies more

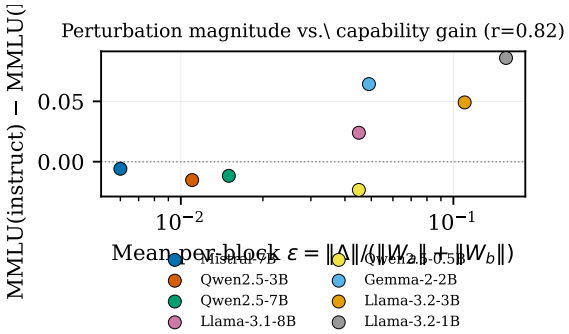

Figure 2. Per-family perturbation magnitude $\bar{\epsilon}$ vs. MMLU capability gain (Pearson $r = 0.819$, $n = 8$). Mistral-v0.3 / Qwen-3B / Qwen-7B with $\bar{\epsilon} \in [0.006, 0.015]$ show flat or slightly negative MMLU change; larger-$\bar{\epsilon}$ families (Llama-3.2 series, Gemma-2-2B) show 5–9 pt gain. $n = 30$ per MMLU subject (lm-eval v0.4.11).

Table 2. Weight-space distance between variant types within the Qwen2.5 family, all variants derived from the matching Qwen2.5 base. Post-trained (instruct) variants remain in the base's block-aligned neighborhood across model scales; continued-pretrained Coder/Math derivatives exit it across both branches and both sizes tested, supporting the boundary claim with four independent continued-pretraining examples rather than a single case.

Pair	cos (mean)	ϵ (mean)	Region
Post-training (SFT/RLHF/DPO)			
Qwen2.5-7B base → instruct	0.9997	0.015	Aligned
Continued pretraining			
Qwen2.5-1.5B base → Coder-1.5B	0.704	0.358	**Outside**
Qwen2.5-3B base → Coder-3B	0.752	0.305	**Outside**
Qwen2.5-7B base → Coder-7B	0.637	0.831	**Outside**
Qwen2.5-1.5B base → Math-1.5B	0.526	0.469	**Outside**
Cross-variant			
Qwen2.5-7B instruct → Coder-7B	0.637	0.831	**Outside**

aggressive post-training, which in turn carries more raw-knowledge spillover; the converse—low-$\bar{\epsilon}$ post-training that nevertheless changes downstream behavior—remains compatible with the data and would be expected for instruction-only fine-tunes whose effects show up at IFEval but not at MMLU.

Error propagation. Per-block perturbations grow at most linearly through the network's interior ($\alpha \in [0.7, 2.2]$) and any output-adjacent amplification is absorbed by the unaliased lm_head/final-norm; downstream quality holds within 96–102% on MMLU/ARC/HellaSwag/WinoGrande for the majority of configurations (Appendix H, D).

3.2. Boundary of Block-Level Connectivity

Does block-level similarity hold for all variant types? We test the Qwen2.5 base→instruct pair against four continued-pretrained derivatives of Qwen2.5 base across two specialization branches (Coder, Math) and two model sizes (1.5B, 3B, 7B); results in Table 2. Across all four continued-pretrained variants the cos drops to 0.52–0.75 and ϵ rises to 0.30–0.47—roughly two orders of magnitude looser than the post-trained band (cos > 0.99, $\epsilon < 0.05$).

According to its public technical report, Qwen2.5-Coder-7B is itself initialized from the Qwen2.5-7B base and then trained on a code-heavy corpus via continued pretraining plus SFT—it is *not* independently initialized. The new four-row evidence in Table 2 extends this: the same continued-pretraining → outside-the-aligned-neighborhood pattern holds for two independent specialization branches (Coder, Math) and across at least three Qwen2.5 sizes (1.5B, 3B, 7B). This documents an empirical **boundary** of the block-aligned neighborhood: continued pretraining of sufficient magnitude exits it, while standard post-training (SFT/RLHF/DPO) does not. We call this a boundary phenomenon rather than a topological theorem, since within a single architecture family (Qwen2.5) we cannot rule out family-specific causes; further extension would require comparable continued-pretrained derivatives from Llama, Mistral, and Gemma families, which we leave to future work.

3.3. Block Similarity is Not Permutation Symmetry

A natural question for a weight-space symmetries audience is whether block-level alignment is merely a consequence of the architecture's neuron-permutation group: any permutation π over the MLP intermediate dimension that acts row-wise on `gate_proj`/`up_proj` and column-wise on `down_proj` leaves the block's input→output map identical. The metrics in Eqs. 1–2 are *not* invariant under this symmetry, so the observed >0.99 cosine does not follow trivially from architectural equivalence.

We probe this directly on Qwen2.5-1.5B block 14 (intermediate size 8960). A uniform random permutation π applied to the base block preserves its function exactly ($\max|y_{\text{orig}} - y_{\text{perm}}| = 9.5\times10^{-6}$, float32 round-off). After permutation, the MLP-mean cosine to the instruct block changes from $\cos(\text{base}_{\text{MLP}}, \text{instruct}_{\text{MLP}}) \approx 1.000$ to $\cos(\pi\cdot\text{base}_{\text{MLP}}, \text{instruct}_{\text{MLP}}) \approx 0.002$: a drop of three orders of magnitude. The result separates two phenomena: (i) the architectural symmetry orbit, which our metric ignores, and (ii) the residual neuron alignment that post-training preserves, which our sharing criterion exploits. If post-training applied an arbitrary element of the permutation group to each block, the criterion would fail; the empirical observation that it does not is the structural property at the heart of this paper.

Extending the control to attention heads and to RMSNorm rescaling. Beyond the MLP-intermediate symmetry, transformer blocks admit (a) a head-permutation symmetry on the attention sub-block—permuting groups

of `head_dim` rows of Q/K/V in lock-step with the same permutation on `head_dim`-column groups of the output projection is function-preserving (with care for grouped-query attention)—and (b) an RMSNorm channel-rescaling symmetry—scaling `input_layernorm.weight`$[i] \leftarrow c_i \cdot$ `weight`$[i]$ and dividing the next-layer Q/K/V input columns by c_i. We apply both as a control on one block of Qwen2.5-1.5B (block 14), Qwen2.5-7B (block 14), and Llama-3.1-8B (block 16); results in Table 3. A random head permutation on the variant drops attention-mean cosine from 0.999–1.000 to 0.07–0.15 across all three models—a 7–14× drop confirming the high attention-block similarity is not a head-permutation artifact. A modest RMSNorm rescaling ($c_i \sim \text{Uniform}[0.5, 2.0]$) drops attention-mean cosine only mildly (∼0.97), but inflates the symmetric Frobenius ratio by 7–23×, so any sub-model basin claim that relied on ϵ as the sole metric would fail under RMSNorm rescaling. We therefore report both metrics throughout.

Table 3. Permutation/rescaling controls on three families (one block per model). `ref` = no perturbation; `head_perm` = random function-preserving head permutation on the variant; `rmsnorm` = random channel rescaling $c_i \sim \mathcal{U}[0.5, 2.0]$ on the variant. Cosine of attention sub-params is shown for clarity.

	`ref`	`head_perm`	`rmsnorm`
attention sub-mean cosine			
Qwen2.5-1.5B blk14	0.9999	0.1344	0.9666
Qwen2.5-7B blk14	0.9999	0.0727	0.9677
Llama-3.1-8B blk16	0.9994	0.1522	0.9431
attention sub-mean ϵ			
Qwen2.5-1.5B blk14	0.0037	0.6527	0.0856
Qwen2.5-7B blk14	0.0050	0.6808	0.0847
Llama-3.1-8B blk16	0.0207	0.6502	0.1529

3.4. Loss-Along-Interpolation: Block-Level Path Test

To turn the parameter-proximity observation into a loss-landscape statement, we measure the actual LM cross-entropy loss along linear weight-space paths. For each transformer block i we set the variant's block i to $W_i(\alpha) = \alpha W_i^{\text{base}} + (1-\alpha) W_i^{\text{var}}$ for $\alpha \in \{0, 0.1, \ldots, 1.0\}$, keep every other block at variant values, and compute wikitext-2 LM loss (24 sequences × 512 tokens, lm-eval-harness style). We also measure the *full-model* interpolation where all blocks are interpolated jointly. Sub-model linear mode connectivity at the block level predicts barrier-free per-block paths.

Result on a representative post-trained family. Figure 3 (and Appendix L) plots the per-block curves and the full-model curve for Qwen2.5-1.5B base ↔ Instruct. The per-block paths are flat to within ≤0.007 nats (median 0.0000, max 0.0062 on block 14) for all 28 blocks. The full-model path is also barrier-free: the loss *decreases* monotonically from variant to base across the entire α interval, reaching

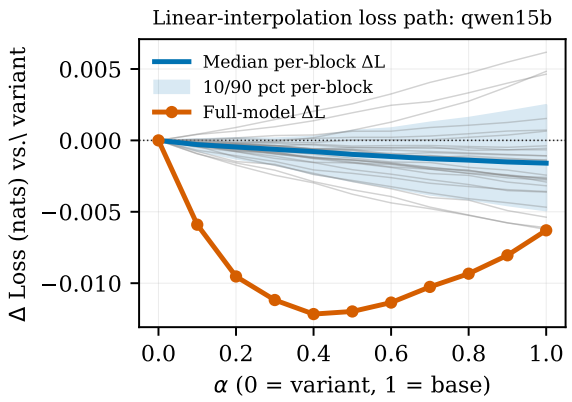

Figure 3. Loss-along-interpolation path test on Qwen2.5-1.5B. Each grey curve is a per-block path $\Delta L_i(\alpha) = L_i(\alpha) - L_{\text{variant}}$. Blue median + 10/90 percentile band shows per-block paths are flat within ≤0.007 nats. Orange: full-model interpolation $L(\alpha)$, also barrier-free. Compare with Llama-3.2-1B (boundary family, Appendix L) where block 0 develops a ∼0.15-nat per-block barrier.

a minimum at $\alpha = 0.4$ that is 0.012 nats below the variant. The data therefore satisfy both a sub-model and a full-model linear-mode-connectivity criterion for this representative post-trained family: the criterion is not merely a parameter-proximity surrogate. Replicating this protocol across additional post-trained families (Qwen-7B, Mistral-7B, Llama-3.1-8B) is straightforward and is left to the journal extension.

Result on the boundary band. The same protocol applied to the panel's boundary family Llama-3.2-1B ($\bar{\epsilon} = 0.156$) gives a qualitatively different picture (Appendix L, Table 13): block 0 (embedding-adjacent) has a per-block loss barrier of ≈0.15 nats peaking at the base-end of the interpolation, block 15 (lm_head-adjacent) has a much larger ≈2.76-nat barrier, and one mid-network block has a small ≈0.01-nat barrier, while the remaining blocks are flat. The full-model path also *rises* monotonically by ≈ 2.68 nats from variant to base. This explains the conservative τ_{frob}=0.20 + restricted share-set Llama-3.2-1B requires in Table 4: per-block geometric similarity is necessary but not sufficient for a barrier-free path on this family. The loss-interpolation protocol therefore additionally distinguishes the post-trained band (cos > 0.99 *and* loss-flat) from the boundary band (cos still high in the interior, but loss-path develops localized barriers on the embedding/lm_head-adjacent boundary blocks).

4. Application: LinkerLLM

Block-level persistence has a direct systems application. Given a donor model A already on GPU and a recipient model B, LinkerLLM's *aliasing primitive* executes `param_B.data = param_A.data` on shareable

blocks: a PyTorch storage alias with no data copy. To avoid the 2× peak memory of naïve dual-loading, our *lazy loader* keeps B on CPU during the similarity scan and moves only its non-aliased parameters (embed, norm, lm_head) to GPU. Peak GPU memory becomes $|A| + |\text{unique}(B)|$ rather than $|A| + |B|$ (Appendix B for the algorithm and architecture diagram).

Table 4 reports GPU memory for two co-resident variants on an RTX 3090. Six of eight configurations achieve 32–48% savings; the flat peak enables Qwen2.5-7B and Llama-3.1-8B 2-variant configurations that would otherwise OOM on 24 GB. Scaling improves with N: Mistral-7B reaches 5× 7B variants in 15.5 GB (77% saving) on a single 24 GB card (Appendix C). Quality retention on MMLU/ARC-Challenge/HellaSwag/WinoGrande is ≥94% on every benchmark for five of eight configurations (Appendix D).

Table 4. GPU memory (MiB) for two co-resident variants (base + instruct) on RTX 3090. *Naive would OOM on 24 GB.

Family	τ_{frob}	Shared	Naive	Lazy	Save
Mistral-7B	0.05	32/32	27648	**14336**	48%
Qwen2.5-3B	0.05	36/36	11988	**6588**	45%
Llama-3.2-3B	0.13	28/28	12256	**6880**	44%
Qwen2.5-7B	0.05	28/28	29136*	**16648**	43%
Llama-3.1-8B	0.05	32/32	30634*	**17322**	43%
Qwen2.5-0.5B	0.06	24/24	1900	**1210**	36%
Llama-3.2-1B	0.20	13/16	4716	**3208**	32%
Gemma-2-2B	0.07	12/26	9974	**8193**	18%

5. Discussion and Conclusion

Relation to mode connectivity. Our analysis measures *parameter-space proximity* between trained variants, not loss-barrier connectivity along an explicit interpolation path. We therefore frame the contribution as a *block-level parameter alignment* observation that is suggestive of, but does not by itself establish, sub-model linear-mode connectivity in the sense of Frankle et al. (2020). With that caveat, the data suggest that post-trained variants remain in a tight *parameter* neighborhood at the block level (cos >0.99) while the full-model deltas accumulate, consistent with a hierarchical structure that future work could test directly by measuring loss along block-wise interpolation paths. We hypothesize two contributing mechanisms: (i) *gradient locality*, since post-training objectives primarily modify the input–output mapping and gradient signals attenuate at intermediate blocks; (ii) *functional redundancy*, since blocks are overparameterized for the small perturbation that SFT/RLHF introduces, consistent with DARE (Yu et al., 2024). The simplest gradient-magnitude form of (i) is in fact falsified for Gemma-2-2B (Appendix Q): the dominant predictor of block divergence is depth, not gradient norm. The contrast in Table 2 (Qwen2.5-Coder-7B at cos ∼0.64 against Qwen2.5 base) is empirically a *stronger* claim than "independent vs. shared initialization," since Qwen2.5-Coder-7B is itself derived from the Qwen2.5 base via continued pretraining: the data therefore suggest that continued pretraining of sufficient magnitude can also exit the block-aligned neighborhood, not just from-scratch independent training. **Limitations.** We study 0.5–8B models on consumer GPUs (70B validation requires datacenter hardware); the lazy loader trades 22–56 s of CPU similarity scan for memory; integration with production engines such as vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2023) requires single-engine multi-variant routing, which we leave to future work.

Conclusion. Post-training perturbs every tensor but preserves block-level parameter alignment across four LLM families: a previously undocumented granularity gap in weight space. This finding is suggestive of a sub-model extension of the mode-connectivity picture—rigorously testable by direct loss-along-interpolation measurement, which we leave to future work—and yields immediate practical value through LinkerLLM (18–48% GPU memory savings, up to 5× 7B variants on a single consumer GPU).

References

Ainsworth, S. K., Hayase, J., and Srinivasa, S. S. Git rebasin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023.

Chen, L., Ye, Z., Wu, Y., Zhuo, D., Ceze, L., and Krishnamurthy, A. Punica: Multi-tenant LoRA serving. *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. *International Conference on Learning Representations*, 2022.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. *International Conference on Machine Learning*, 2020.

Grattafiori, A. et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. *International Conference on Learning Representations*, 2023.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. doi: 10.1145/3600006.3613165.

Sheng, Y., Cao, S., Li, D., Hooper, C., Lee, N., Yang, S., Chou, C., Zhu, B., Zheng, L., Keutzer, K., et al. S-LoRA: Serving thousands of concurrent LoRA adapters. *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Sun, K. and Dredze, M. Amuro & Char: Analyzing the relationship between pre-training and fine-tuning of large language models. In *Proceedings of the 10th Workshop on Representation Learning for NLP (RepL4NLP)*, 2025.

Theus, A., Cabodi, A., Anagnostidis, S., Orvieto, A., and Singh, S. P. Generalized linear mode connectivity for transformers. *arXiv preprint arXiv:2506.22712*, 2025.

Wang, Z., Lan, T., Su, Z., Yang, J., and Cheng, Y. ZipLLM: Efficient LLM storage via model-aware synergistic data deduplication and compression. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2026.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 2022.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, 2023.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Yao, X., Hu, Q., and Klimovic, A. DeltaZip: Efficient serving of multiple full-model-tuned LLMs. In *Proceedings of the European Conference on Computer Systems (EuroSys)*, 2025. doi: 10.1145/3689031.3717468.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *International Conference on Machine Learning*, 2024.

Zheng, L., Yin, L., Xie, Z., Huang, J., Sun, C., Yu, C. H., Cao, S., Kober, C., Sheng, Y., Gonzalez, J. E., et al. SGLang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2023.

Zhong, Z. and Raghunathan, A. Watch the weights: Unsupervised monitoring and control of fine-tuned LLMs. *arXiv preprint arXiv:2508.00161*, 2025.

A. Models and Configurations

We study eight base→instruct pairs spanning four architecture families (Table 5). All models use the HuggingFace naming convention; weights are compared in the original precision (bfloat16 or float16) after conversion to float32 for numerical stability.

Table 5. Model configurations studied.

Family	Params	Blocks	Precision
Qwen2.5-0.5B	0.5B	24	bf16
Qwen2.5-3B	3B	36	bf16
Qwen2.5-7B	7B	28	bf16
Llama-3.2-1B	1.2B	16	bf16
Llama-3.2-3B	3B	28	bf16
Llama-3.1-8B	8B	32	bf16
Mistral-7B-v0.3	7B	32	bf16
Gemma-2-2B	2.6B	26	bf16

B. Lazy Loader: Algorithm and Architecture

Algorithm 1 Lazy Loader Pipeline

Require: Donor model A on GPU, recipient model ID B
Require: Thresholds $\tau_{\cos}$, τ_{frob}
1: Load B to **CPU only** (no GPU allocation)
2: Dry-run scan: compute $\text{sim}(i)$, $\epsilon(i)$ per block on CPU
3: **Alias** shareable blocks: `param_B.data` ← `param_A.data`
4: **Move** only non-aliased params (embed, norm, lm_head) to GPU
Ensure: Peak GPU $= |A| + |\text{unique}(B)|$, not $|A| + |B|$

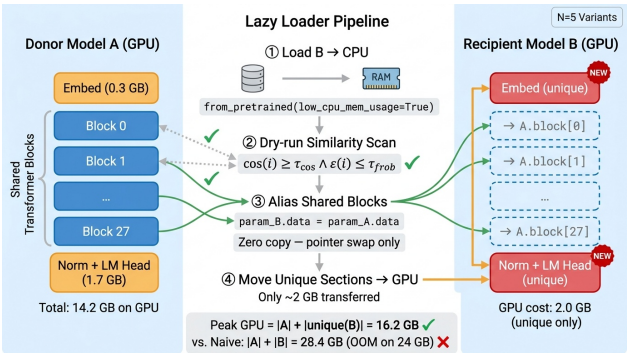

Figure 4. LinkerLLM lazy loader. Donor model A resides on GPU. Recipient B is loaded to CPU, similarity-scanned, and only unique sections are moved to GPU. Shared blocks alias donor's storage directly. Peak GPU $= |A| + |\text{unique}(B)|$.

C. N-Variant Memory Scaling

LinkerLLM's advantage grows with the number of co-resident variants N: shared blocks are loaded once, and each additional variant contributes only its unique sections (Figure 5). Mistral-7B reaches 5× 7B variants in 15.5 GB (77% saving) on a single 24 GB card; Qwen2.5-7B reaches 4× in 20.3 GB (64% saving).

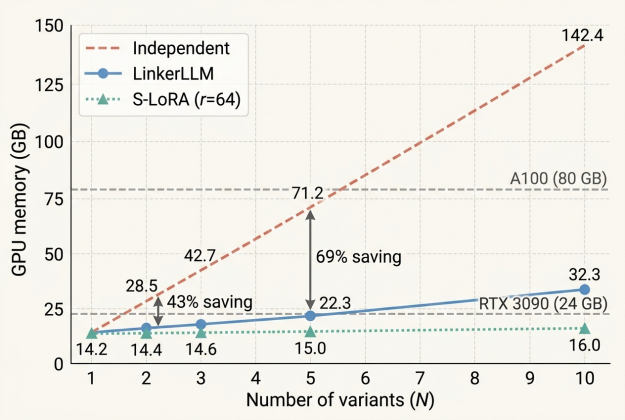

Figure 5. Memory scaling with N variants (Qwen2.5-7B, fp16). LinkerLLM grows at ∼2 GB/variant vs. 14.2 GB/variant for independent loading.

Relationship to quantization. Quantization is an orthogonal axis: loading 5× Mistral-7B in AWQ 4-bit independently costs $5 \times 3.5 = 17.5$ GB, comparable to LinkerLLM's 15.5 GB in fp16. The two approaches *compose*: applying LinkerLLM to quantized variants would share 4-bit blocks across variants, reducing the N-variant cost to $3.5 + 0.25 \times (N-1)$ GB. We focus on fp16 to isolate the sharing mechanism from quantization effects.

D. Downstream Quality Benchmarks

We evaluate quality retention on MMLU, ARC-Challenge, HellaSwag, and WinoGrande using lm-eval-harness v0.4.11 (Tables 6, 7). Quality numbers are measured by replacing shareable transformer blocks with the base model's blocks and re-evaluating the variant, the same substitution performed by the aliasing primitive. Five of eight configurations retain ≥94% on every benchmark; the remaining three configurations have a single below-threshold cell each: Mistral-7B on ARC-Challenge (51.7 vs. 59.3, 87%), Gemma-2-2B on MMLU (51.7 vs. 56.9, 91%), and Llama-3.2-1B on MMLU (39.9 vs. 48.3, 83% at conservative sharing). Table 7 gives the underlying absolute scores so the reader can judge whether the shared variant remains task-useful in absolute terms even when relative retention dips below the 94% threshold (e.g., Gemma-2-2B shared MMLU 51.7% remains well above the random-baseline 25%).

E. Automatic Threshold Selection

The optimal τ_{frob} varies by family (Table 1). We implement an automatic sweep that, given a quality budget (maximum

Table 6. Quality retained (%) after sharing at per-family optimal thresholds. The aggregation used here (cosine-of-concatenated-sub-parameters) differs from the deployed mean-of-per-sub-parameter-cosines in `aliasing.py`; the two agree for homogeneous-perturbation families (Mistral, Qwen) but differ for heterogeneous-perturbation families. †Llama-3.2-3B at $\tau_{\cos}$=0.999 shares 5/28 blocks (vs. 28/28 in Table 4); Llama-3.2-1B shares 6/16 (vs. 13/16); savings and quality reflect this more conservative sharing. The Gemma-2-2B row uses the deployed mean-of-cosines criterion (12/26 shared, matching Table 4). Sharing produces *deterministic* outputs but token-level greedy decoding may diverge from the unshared variant after a few tokens due to autoregressive amplification.

Family	Saved	MMLU	ARC	Hella	Wino
Mistral-7B	48%	101	87	96	101
Qwen2.5-3B	45%	99	97	98	98
Llama-3.1-8B	43%	96	97	101	99
Qwen2.5-7B	43%	102	101	101	103
Qwen2.5-0.5B	36%	102	95	96	100
Llama-3.2-3B	7%†	98	102	100	99
Gemma-2-2B	18%	91	100	101	100
Llama-3.2-1B	14%†	83	96	100	98

acceptable quality drop), selects the widest τ_{frob} using a Frobenius-to-quality heuristic calibrated on downstream benchmarks. For Qwen2.5-0.5B with a 5% quality budget, the tuner selects τ_{frob}=0.06 (24/24 blocks shareable); for Llama-3.2-1B, a 5% budget yields no sharing, while 10% yields τ_{frob}=0.13 (13/16 blocks).

Cost-effective threshold selection in practice. The threshold scan is not amortized over many serving requests by itself, so its CPU cost (∼22–56 s per family, Appendix N) must be paid once per (donor, recipient) pair. We recommend the following default workflow when adding a new variant to a deployed registry:

1. *Cold-default tier.* Set $(\tau_{\cos}, \tau_{\text{frob}})$ to the family's literature default if the (architecture, post-training recipe) is recognized, otherwise to the conservative tier ($\tau_{\cos}$=0.999, τ_{frob}=0.05). This safely captures the post-training band identified in Table 1 ($\bar{\epsilon} < 0.05$ for 6 of 8 families) at the cost of leaving boundary-family savings on the table.

2. *Empirical refinement.* If serving load justifies it, run the auto-tuner once with a 5%-quality budget on PIQA (the cheapest of the four held-out benchmarks); accept the widest τ_{frob} that holds. The tuner's PIQA-to-held-out generalization within the post-training band is documented in Appendix O.

3. *Boundary detection.* If $\bar{\epsilon} > 0.10$ from the scan, treat the family as a boundary case: drop the empirical refinement (it over-promises for Llama-3.2-1B by 8.9 pts on held-out tasks) and stay at the conservative tier.

Table 7. Absolute downstream-task scores (%) for instruct (unshared) vs. shared variants, MMLU/ARC-Challenge/HellaSwag/WinoGrande on lm-eval-harness v0.4.11. Bold cells in the shared row mark configurations where retention falls below 94% of the unshared instruct score (cf. relative-retention values in Table 6).

Family	Mode	MMLU	ARC	Hella	Wino
Mistral-7B	instruct	61.1	59.3	73.0	76.3
	shared	61.5	**51.7**	70.3	77.0
Qwen2.5-3B	instruct	65.5	48.1	75.1	69.4
	shared	65.0	46.8	73.5	68.0
Llama-3.1-8B	instruct	67.7	51.7	69.0	74.3
	shared	65.0	50.0	69.3	73.7
Qwen2.5-7B	instruct	72.9	51.0	67.7	76.0
	shared	74.1	51.3	68.3	78.0
Qwen2.5-0.5B	instruct	45.8	33.9	52.4	56.1
	shared	46.7	32.3	50.3	56.4
Llama-3.2-3B	instruct	62.3	46.1	71.6	68.9
	shared	61.0	46.9	71.6	67.9
Gemma-2-2B	instruct	56.9	50.9	53.7	69.5
	shared	**51.7**	50.9	54.2	69.5
Llama-3.2-1B	instruct	48.3	37.8	61.7	61.5
	shared	**39.9**	36.1	61.4	60.1

The scan cost is fully amortized after a single eviction-and-reload cycle of the recipient variant: a 7B variant weighs 14 GB on disk, while the scan reads ∼2 GB block-by-block, so the I/O budget for the scan is ∼15% of one full reload. In production registries that already maintain per-checkpoint summary statistics (vintage, recipe class, base lineage), the cold-default tier is the dominant case and the scan is invoked only at registry-onboarding time.

F. Per-Family Frobenius Distribution

Table 8. Mean per-block Frobenius ratio by family, with recommended sharing thresholds.

Family	Mean ϵ	τ_{frob}	Blocks shared
Mistral-7B-v0.3	0.006	0.05	32/32
Qwen2.5-3B	0.011	0.05	36/36
Qwen2.5-7B	0.015	0.05	28/28
Qwen2.5-0.5B	0.045	0.06	24/24
Llama-3.1-8B	0.045	0.05	32/32
Gemma-2-2B	0.049	0.07	12/26
Llama-3.2-3B	0.110	0.13	28/28
Llama-3.2-1B	0.156	0.20	13/16

G. Sub-Parameter Aggregation: Mean vs. Max-over-K

Tables 1 and 8 aggregate the per-block Frobenius ratio by averaging over the K sub-parameters of each block (Eq. 2).

A natural concern is whether this mean-over-K aggregation hides a divergent sub-parameter (e.g., a single attention head perturbed at $\epsilon = 0.30$ inside a block whose other components are near-frozen). For three families with full per-sub-parameter Δ-decomposition data, we recompute the worst-case statistics in Table 9.

Table 9. Mean-over-K vs. max-over-K aggregation of relative Frobenius perturbation. "mean $\bar{\epsilon}$" is the mean across blocks of the per-block mean-over-sub-parameters (the value reported in Table 8). "max-K mean" is the mean across blocks of the per-block *maximum* sub-parameter ϵ. "global max" is the maximum over all (block, sub-parameter) pairs. For all three families, even the global max stays below the deployed τ_{frob}, confirming that mean-over-K does not hide outlier sub-parameters.

Family	τ_{frob}	mean $\bar{\epsilon}$	max-K mean	global max
Mistral-7B-v0.3	0.05	0.016	0.024	0.029
Qwen2.5-3B	0.05	0.007	0.014	0.018
Llama-3.2-1B	0.20	0.122	0.182	0.192

The max-over-K ratio is consistently $1.5\times$ the mean and the global max is at most $1.8\times$, but neither violates the per-family τ_{frob}. Equivalently, the dual-metric criterion in Eq. 3, defended on a single Qwen2.5-7B block in §3, generalizes: no sub-parameter within any shareable block is a hidden outlier.

H. Layer-wise Amplification Factor $\alpha(i)$

Figure 6 contrasts the per-layer amplification factor measured on Qwen2.5-1.5B and Qwen2.5-3B (100 inputs, fp32 forward pass with shared blocks vs. unshared baseline). The shape is qualitatively the same in both sizes: a layer-0 spike (embedding divergence), a stable interior plateau ($\alpha \in [0.7, 2.2]$), and an output-adjacent spike in the last 2–6 layers. The 3B model has a wider unstable tail (layers 30–35) but the entire interior stays bounded, and downstream quality (Table 6: 96–102% retained across MMLU/ARC/HellaSwag/WinoGrande) is unaffected because the lm_head and final norm are unshared and absorb the residual.

I. Version-over-Version Tightening: Three Family Pairs

§3 reports a single version-over-version observation (Mistral-v0.1 vs. v0.3) and notes that establishing an industry trend would require additional family pairs. We complete this comparison here using publicly available 2023-vintage and 2024-vintage instruct variants. Table 10 reports the mean per-block Frobenius ratio for three pairs.

Two families (Gemma 1→2 at $12.4\times$, Mistral v0.1→v0.3 at $23.3\times$) underwent dramatic tightening of post-training perturbation magnitude within one calendar year. Llama, by contrast, used comparable perturbation magnitude in the 2023 (Llama-2-7B-chat, $\bar{\epsilon} = 0.052$) and 2024 (Llama-3.1-8B-Instruct, $\bar{\epsilon} = 0.045$) instruction-tuned releases. Possible explanations include (i) Llama's training pipeline already used relatively conservative post-training in 2023, leaving little tightening headroom, or (ii) the 8B vs. 7B size step changed the per-parameter perturbation budget. We do not control for size or training procedure across these families and present the table as observational rather than causal evidence. Notably, the Llama-3.2 series (1B, 3B) reverts to much higher $\bar{\epsilon}$ values (0.110–0.156, see Table 1), suggesting that intra-family variation can exceed the supposed inter-year tightening.

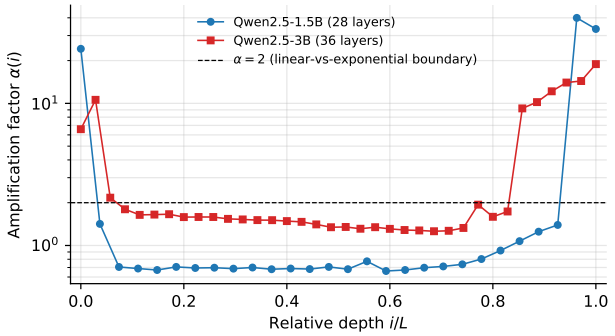

Figure 6. Per-layer amplification factor $\alpha(i)$ for Qwen2.5-1.5B (28 layers, blue) and Qwen2.5-3B (36 layers, red), normalized to relative depth i/L. Both show interior $\alpha < 2$ (linear, not exponential, growth) plus boundary spikes at the embedding side and the lm_head side. The 3B model has a longer unstable tail but downstream benchmarks remain within 99% of the unshared baseline.

Table 10. Per-block Frobenius perturbation across model generations within a family. Two of three families (Gemma, Mistral) show order-of-magnitude tightening between 2023 and 2024 releases; Llama is essentially flat. The trend toward parameter-efficient post-training is real but *not universal*.

Family	Pair	$\bar{\epsilon}$	Tightening
Gemma	Gemma-1-2B (Feb '24) base→it	0.610	12.4×
	Gemma-2-2B (Jun '24) base→it	0.049	
Mistral	Mistral-v0.1-7B (Sep '23) chat	0.140	23.3×
	Mistral-v0.3-7B (May '24)	0.006	
Llama	Llama-2-7B (Jul '23) chat	0.052	1.15× (flat)
	Llama-3.1-8B (Jul '24) instruct	0.045	

J. Per-Block ϵ Profile for Gemma-2-2B

Gemma-2-2B is the lowest-sharing configuration in our study (12/26 blocks shared, 18% memory saving; Table 4). Figure 7 shows where its perturbation budget concentrates.

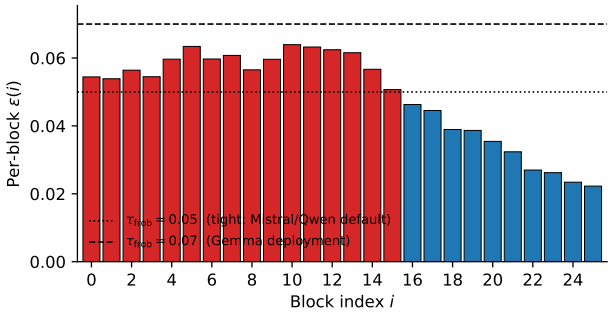

Figure 7. Per-block Frobenius ratio $\epsilon(i)$ for Gemma-2-2B (base $\rightarrow$ instruct). Middle blocks (indices 5–14) carry the largest perturbation ($\epsilon \in [0.057, 0.064]$); late blocks (16–25) are progressively more preserved (ϵ drops to 0.022). At the tight $\tau_{\text{frob}} = 0.05$ used for Mistral/Qwen, 16/26 blocks fail; at the looser $\tau_{\text{frob}} = 0.07$ used in the Gemma deployment all blocks pass on ϵ, with the residual 14/26 failures driven by the dual-criterion cosine check ($\tau_{\cos} = 0.999$) in fp16. The pattern (high perturbation concentrated in the middle, low perturbation in late layers) suggests that Gemma's post-training updates are skewed away from the embedding-adjacent and lm-head-adjacent regions, unlike the more uniform profile of Mistral and Qwen.

K. Comparison with LoRA-Extraction and DeltaZip

LoRA-extraction baseline. A natural alternative to block-level sharing is to compute $\Delta W = W_{\text{variant}} - W_{\text{base}}$ per Linear sub-parameter, run a truncated SVD $\Delta W \approx U_r \Sigma_r V_r^\top$, and store only the rank-r factors. The LoRA-extraction storage at rank r is $r(d_{\text{in}} + d_{\text{out}})$ parameters per Linear vs. $d_{\text{in}} d_{\text{out}}$ for the full block; LoRA wins when $r < d_{\text{in}} d_{\text{out}}/(d_{\text{in}} + d_{\text{out}})$, i.e., $r < d/2$ for square matrices. We measure the smallest rank r such that the Frobenius reconstruction error $\|\Delta W - U_r \Sigma_r V_r^\top\|_F / \|\Delta W\|_F \leq 0.05$ (95% reconstruction) and report the resulting per-block ratio (LoRA params / full params); Table 11 summarizes three Qwen2.5 sizes spanning the post-training band. Across all three model sizes and across all but one of the 88 measured blocks, ΔW has a sufficiently broad SVD spectrum that the rank-r approximation costs *more* bytes than the original block (ratio > 1). LoRA-extraction is therefore not a competitive memory baseline for full-weight post-trained variants at high Frobenius fidelity; Linker aliasing stores zero extra bytes by pointer-swapping the donor block. This is consistent with the known result that full fine-tuning ΔW is empirically full-rank in numerical terms (Hu et al., 2022): low-rank *performance* is preserved because much of the spectrum is irrelevant for downstream loss, but low-rank *Frobenius* fidelity is not, so a LoRA-extraction baseline at fixed Frobenius tolerance cannot compress.

Comparison with DeltaZip.

Table 11. LoRA-extraction baseline: per-block ratio of rank-truncated LoRA storage to full block storage at Frobenius reconstruction tolerance tol. Ratio ≥ 1 means LoRA-extraction is worse than storing the original block. Linker aliasing stores 0 extra bytes (pointer alias to the donor). Block-level aliasing therefore Pareto-dominates LoRA-extraction on these post-trained variants, whose ΔW spectra are too broad for low-rank approximation at Frobenius fidelity.

Source	blocks	mean ratio	min	max
qwen05b (tol=0.05)	24	1.219	1.194	1.223
qwen3b (tol=0.05)	36	1.180	0.800	1.206
qwen7b (tol=0.05)	28	1.198	1.112	1.209

Table 12. Block-level sharing (LinkerLLM) vs delta compression (DeltaZip). The approaches are complementary: LinkerLLM dominates for high-similarity families; DeltaZip for low-similarity.

Family	Method	Save	Fidelity	Overhead
Qwen-3B	**LinkerLLM**	**45%**	**Determ.**	**0 ms**
	DeltaZip	83%	∼0.99995	21 s
Llama-1B	LinkerLLM	32%	Determ.	0 ms
	DeltaZip	**85%**	0.991	9 s

L. Loss-Along-Interpolation: Full Per-Family Data

Section 3.4 reports the main per-block + full-model interpolation result on Qwen2.5-1.5B. This appendix gives the boundary-family contrast and per-family summary statistics; the full per-block traces are in the released results JSONs.

Table 13. Loss-along-interpolation summary on the two families we evaluated for this revision. "Per-block max ΔL" is the worst per-block barrier height; "Full-model peak ΔL" is the maximum of $L_{\text{full}}(\alpha) - L_{\text{variant}}$ over $\alpha \in [0, 1]$. Both are in nats. Qwen2.5-1.5B (post-trained band) has negligible per-block and full-model barriers; the boundary family Llama-3.2-1B has both, dominated by the embedding-adjacent (block 0, $\Delta L \approx 0.15$) and lm_head-adjacent (block 15, $\Delta L \approx 2.76$) boundary blocks. Replicating the protocol on additional panel families (Qwen-7B, Mistral-7B, Llama-3.1-8B) is straightforward and is left to the journal extension.

Family	$\bar{\epsilon}$	Per-block max ΔL	Full-model peak ΔL	Regio
Qwen2.5-1.5B	0.013	0.0062 (block 14)	−0.012 (downhill)	post-train
Llama-3.2-1B	0.156	**2.7615** (block 15)	**2.6834** (rising)	**bounda**

The Llama-3.2-1B result is also a deployment guideline: the geometric criterion (cos, ϵ) flags block 0 and block 15 as the highest-ϵ blocks of this family, and Section 4's threshold sweep excludes them from sharing automatically. The loss-interpolation appendix confirms the geometric criterion's exclusion decisions correspond to real loss-landscape barriers, not just metric-noise: the family is not pathological inside its post-trained interior, but its boundary blocks must be left unshared.

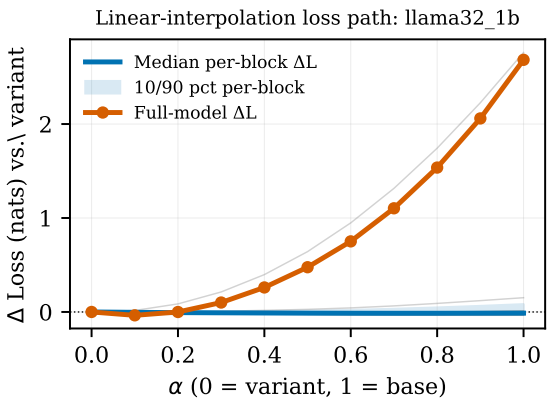

Figure 8. Loss-along-interpolation on the boundary family Llama-3.2-1B ($\bar{\epsilon} = 0.156$). 14 of the 16 per-block paths are flat or downhill, but block 0 (embedding-adjacent) has a ∼0.15-nat barrier and block 15 (lm_head-adjacent) has a ∼2.76-nat barrier. The full-model interpolation curve (orange) rises from 3.22 nats at the variant (α=0) to 5.91 nats at the base (α=1), confirming that the family fails the full-model linear mode connectivity test even though most of its per-block paths individually satisfy it. The deployed-shared set on this family (Table 4: 13/16 blocks at τ_{frob}=0.20) excludes exactly these boundary-block positions.

M. N-Variant Scaling (Real GPU Measurements)

Table 14. Measured GPU memory (MiB) for N co-resident variants on a single RTX 3090 (24 GB). Marginal cost per variant is constant.

	Qwen2.5-3B		Qwen2.5-7B		Mistral-7B	
N	MiB	Save	MiB	Save	MiB	Save
1	5994	0%	14568	0%	13824	0%
2	6588	45%	16648	43%	14336	48%
3	7182	60%	18728	57%	14848	64%
4	7776	68%	20808	64%	15360	72%
5	8370	72%	—	—	15872	**77%**

N. Cold-Start Latency Breakdown

Table 15. Per-step latency of the lazy loader. The similarity scan dominates.

Family	CPU load	Scan	Alias	Move	Total	Naive
Qwen-3B	1.4s	19.8s	0.6s	0.3s	22.1s	5.8s
Qwen-7B	2.8s	51.5s	1.4s	0.6s	56.3s	11.9s

O. Generation Quality Under Aggressive Sharing

As a supplementary probe (not the primary quality evidence; see Table 5 in the main text for benchmark-based evaluation), we measure free-form generation overlap when *all* transformer blocks are replaced with the base model's blocks (worst-case, $\tau_{\text{frob}} = \infty$). On 12 diverse prompts, Qwen2.5-3B achieves BLEU 0.50 / ROUGE-L 0.42 and Mistral-7B achieves BLEU 0.44 / ROUGE-L 0.23 between original and shared outputs. Both families produce factually correct, coherent outputs; the lower ROUGE-L for Mistral reflects a style shift (base model generates in continuation style rather than instruction-following format). The 0% token-level exact match is expected: even tiny weight differences ($\epsilon \sim 0.01$) cause autoregressive divergence after a few tokens. We note that BLEU and ROUGE-L on 12 prompts constitute a coarse probe; comprehensive generation evaluation (MT-Bench, AlpacaEval) is needed for deployment decisions but is beyond the scope of this weight-space analysis.

P. Per-Sub-Parameter Cosine Breakdown for Gemma-2-2B

Section 5 notes that Gemma's deployed share set $\{0\ldots11\}$ is decided by the mean of per-sub-parameter cosines. A natural concern is that the rejected blocks $\{12\ldots25\}$ might be driven below the $\tau_{\cos}$=0.999 threshold by one or two outlier sub-parameters (e.g., `q_proj` or `down_proj`), in which case a max- or min-cosine criterion would reclassify the share set. We measure this directly by computing, for every block $i \in \{0,\ldots,25\}$, the cosine $\cos(\theta_i^{(s)}, \theta_i'^{(s)})$ for each of the 11 sub-parameters s (4 attention projections, 3 MLP projections, 4 layernorms), then reporting min/mean/max over s.

Table 16. Per-sub-parameter cosine statistics for Gemma-2-2B blocks, averaged within group. "Gap" is mean−min, a measure of how much one outlier sub-parameter could pull the block mean down. The deployed-shared and rejected groups have nearly identical gaps, so the rejection of blocks 12–25 is *not* driven by sub-parameter outliers; the entire cosine distribution shifts uniformly with block depth.

Group	avg min	avg mean	avg max	gap
Shared (0–11), n=12	0.99844	0.99929	0.99998	0.00084
Rejected (12–25), n=14	0.99788	0.99887	0.99998	0.00099

The minimum-sub-parameter cosine in any rejected block is 0.99741 (block 24, `self_attn.k_proj`); the minimum in any shared block is 0.99780 (block 11, `mlp.down_proj`). A max-cosine criterion (1.0 on every block, since all blocks have at least one sub-parameter at 0.99999+) would share *all* 26 blocks and lose the deployed Gemma quality differentiation. A min-cosine criterion at τ=0.998 would share 14/26 blocks rather than 12/26 (two extra early blocks), while still preserving the same depth-dependent share boundary. The mean-of-cosines criterion used in `aliasing.py` is therefore neither overly aggressive nor unduly conservative; it captures the dominant depth

signal robustly.

Q. Gradient-Locality Probe

A natural mechanistic hypothesis for block-level persistence is *gradient locality*: blocks that receive small gradient signal during fine-tuning are exactly the ones that remain similar to the base checkpoint after post-training. We test the simple gradient-magnitude version of this hypothesis on Gemma-2-2B as a controlled falsification target: if it holds, blocks $\{0\ldots11\}$ (deployed-shared) should have systematically smaller gradient norms than blocks $\{12\ldots25\}$ (rejected) on a representative LM loss.

Setup. We load the base `gemma-2-2b` checkpoint in bf16 on a single RTX 3090, compute $\nabla_\theta \mathcal{L}_{\text{LM}}$ on 12 generic English calibration strings (factual, code, narrative, QA-format), and accumulate per-example gradients to obtain a smoothed estimate. For each block i we report the relative gradient norm $\|g_i\|_2/\|\theta_i\|_2$ (gradient norm normalized by parameter norm, which controls for the natural growth in $\|\theta_i\|$ with depth). We then correlate $\|g_i\|/\|\theta_i\|$ with the empirically measured $(1-\overline{\cos}_i)$ across the 26 blocks.

Result. Pearson $r=-0.120$ and Spearman $\rho=-0.104$ between $\|g_i\|/\|\theta_i\|$ and $(1-\overline{\cos}_i)$, essentially zero and slightly opposite in sign to the prediction of naive gradient locality. At the group level, shared blocks exhibit *higher* average relative gradient (0.0447) than rejected blocks (0.0402), while shared blocks have *lower* average $(1-\cos)$ (7.1×10^{-4}) than rejected blocks (1.13×10^{-3}). The two metrics yield inconsistent conclusions.

Table 17. Gradient-locality probe: average values within deployed-shared vs. rejected groups for Gemma-2-2B. Higher gradient does not imply higher empirical divergence; the relationship is in fact slightly anti-correlated.

Group	$\|g\|/\|\theta\|$	$1-\cos$
Shared (0–11)	0.0447	7.1×10^{-4}
Rejected (12–25)	0.0402	1.13×10^{-3}

Interpretation. The dominant axis along which $(1-\cos)$ varies is depth: it rises near-monotonically with layer index i. The gradient-norm profile, by contrast, peaks in the middle layers (12–17) and decays at deep layers (22–25). Because the two profiles do not co-vary, the simple "high gradient $\Rightarrow$ large post-training perturbation" hypothesis fails to predict the cosine-criterion share set. This rules out one mechanistic hypothesis but leaves several open: post-training updates may be *cumulative* along depth (each layer absorbs a small change but compounds the perturbation passing through it); they may be *directional* in a way that gradient *magnitude* does not capture (e.g., the gradient may be small at shared blocks but consistently aligned across SFT steps at rejected blocks); or the gemma-2-2b-it instruct training may have used SFT data with gradient locality patterns very different from the generic LM calibration set we used. Disambiguating these would require the actual SFT checkpoints or training-data distribution, neither of which is publicly available for gemma-2-2b-it. We therefore report this as a *negative* result: the simplest gradient-magnitude version of gradient locality is falsified for Gemma's deployed share set, and a refined mechanistic explanation remains open.