Curiouser and Curiouser

Apparent Contradictions in Neural Network Observables

Weyl AI Research

January 2026

Abstract

We hypothesize that constraints which reduce wrong moves matter more than constraints which reduce total moves. This reframes quantization, low-rank adaptation, and sparsity as hallway selection problems rather than capacity-accuracy tradeoffs. A coarse lattice that blocks bad directions may outperform a fine lattice that permits them. We formalize "good constraints" via the signal-to-branching ratio p_t, gradient capture ratio ψ_t, and boundary-crossing rates β_t, β_t^{eff}. Practitioner rule: choose the lowest rank and coarsest precision that keep ψ_t high, $\beta_t, \beta_t^{\text{eff}} > 0$, and $p_t > \frac{1}{2}$. We show this framework resolves apparent contradictions in the literature—why LoRA matches full fine-tuning, why lottery tickets exist, why pruning helps, why control vectors work despite the abundance of equivalent solutions.

1 The Standard View

The conventional understanding of neural network constraints:

Fewer parameters ⇒ less capacity ⇒ worse performance.

This predicts monotonic degradation:

$$\begin{aligned} \text{fp32} &> \text{fp16} > \text{fp8} > \text{fp4} \\ \text{full-rank} &> \text{low-rank} \\ \text{dense} &> \text{sparse} \end{aligned}$$

Each constraint removes representational capacity, so performance should degrade monotonically with constraint strength.

2 The Anomalies

Yet empirically, the hierarchy breaks:

- **LoRA matches full fine-tuning** with proper hyperparameters [1]. The "capacity gap" was configuration, not fundamental.
- **Lottery tickets match dense performance** [2]. Sparse subnetworks, found by pruning, match the original dense network.
- **Int8 training sometimes outperforms fp16** [3]. Lower precision, better results.

- **Pruning improves generalization** [4]. Removing weights helps.
- **L1 regularization works**. Zero is special on the lattice.
- **Stochastic rounding helps** [5]. Noise that explores neighboring cells.

These are not isolated flukes. We observe them repeatedly, in different contexts, with different architectures. All contradict continuous $\mathbb{R}^n$ theory where more parameters = more expressiveness = better.

3 The Apparent Contradictions

The literature contains results that seem mutually incompatible:

"Abundance" Results	**"Structure" Results**
Lottery tickets: 90% of weights prunable [2]	Control vectors: specific directions steer behavior [6]
Git Re-Basin: all solutions equivalent mod permutation [7]	Interpretability: specific neurons encode features [8]
LoRA: rank-16 is enough [9]	Steering: fine-grained behavioral control works [10]
Double descent: huge manifold of minima [11]	Transfer learning: representations reliably transfer [12]
Adam finds solutions easily	Linear probes work [13]

The puzzle: If there's an *abundance* of equivalent solutions discoverable by AdamW, why would specific directions mean anything? Each random seed should produce different internal structure.

And yet control vectors work. Steering vectors transfer. Linear probes find consistent features.

The resolution: The **loss** has gauge symmetry—a huge manifold of equivalent minima [14, 15]. But the **optimizer** breaks the gauge consistently [16]. The broken directions become the control vector directions.

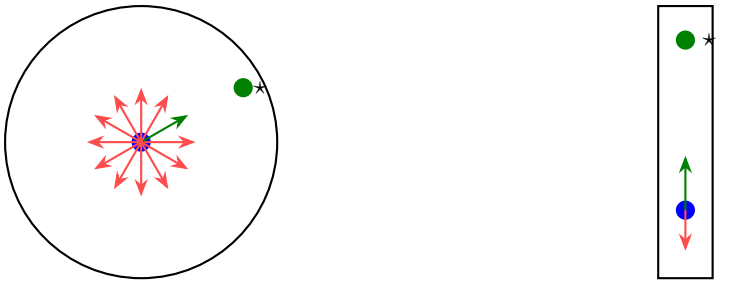
4 The Hypothesis

Hypothesis 1 (Hallway)**.** *Constraints that reduce the number of wrong moves matter more than constraints that reduce the number of total moves. Training succeeds when the signal-to-branching ratio—the fraction of admissible moves that help—is high.*

Consider two training regimes:

1. **Open field:** Many possible moves. Some help, most hurt.
2. **Narrow hallway:** Few possible moves. Most help.

The hallway is better—not despite having fewer moves, but *because* of it.



Open field
$p_t \approx 1/12$

Hallway
$p_t \approx 1/2$

Figure 1: Open field $p_t \approx 1/12$ vs hallway $p_t \approx 1/2$. Higher signal-to-branching ratio accelerates progress by eliminating wrong moves. The hallway has fewer total moves but a higher fraction that help.

5 Formalization

Let S be the update subspace (e.g., LoRA's low-rank span) and $R > 0$ a step budget. Let Q be the quantizer induced by the precision stack $\mathcal{P}$.

Realized descent. Let $\mathcal{L}^\sharp_\mathcal{P}$ denote the realized loss under precision stack $\mathcal{P}$ (see Lattice Hypothesis). We call u a **realized descent move** if $\mathcal{L}^\sharp_\mathcal{P}(\theta_t + u) < \mathcal{L}^\sharp_\mathcal{P}(\theta_t)$. When we use $\nabla\mathcal{L}$ below, it is a surrogate (STE or omitted-cast gradients); all measurements use finite-difference tests on $\mathcal{L}^\sharp_\mathcal{P}$.

Locally finite. All stratification and flip-count claims are locally finite: on any compact $K \subset \mathbb{R}^n$, only finitely many strata and thresholds intersect K.

Definition 1 (Admissible Set)**.** *The* ***admissible set*** *at step t is:*

$$\mathcal{A}_t = \{u \in S : \|u\| \le R,\ u = Q(u)\}$$

sampled with μ equal to the optimizer's post-quantization proposal distribution. Set R to the optimizer's median update norm; ensure $\mathcal{A}_t \neq \varnothing$.

Definition 2 (Descent Cone)**.** *The* ***descent cone*** *is:*

$$\mathcal{C}_t = \{u \in \mathcal{A}_t : \nabla\mathcal{L}(\theta_t)^\top u < 0\}$$

Moves with negative directional derivative (surrogate). A move is "wrong" if $\nabla\mathcal{L}^\top u \ge 0$.

Definition 3 (Signal-to-Branching Ratio)**.** *With base measure μ (the optimizer's proposal distribution on $\mathcal{A}_t$):*

$$p_t = \frac{\mu(\mathcal{C}_t)}{\mu(\mathcal{A}_t)} = \Pr_{u\sim\mu}[\nabla\mathcal{L}^\top u < 0]$$

The fraction of admissible moves that are descent directions.

Definition 4 (Gradient Capture Ratio)**.**

$$\psi_t = \frac{\|P_S \nabla\mathcal{L}\|}{\|\nabla\mathcal{L}\|}$$

where P_S is projection onto the update subspace S.

Why ψ_t and p_t matter: For an L-smooth loss and update $u \in S$:

$$\mathcal{L}(\theta_t + u) \leq \mathcal{L}(\theta_t) + \nabla\mathcal{L}^\top u + \frac{L}{2}\|u\|^2$$

Under optimizer distribution π on $\mathcal{A}_t$:

$$\mathbb{E}_\pi[\Delta\mathcal{L}] \lesssim -\|\nabla\mathcal{L}\| \cdot \mathbb{E}_\pi[\|u\| \cos\angle(\nabla\mathcal{L}, u)] + \frac{L}{2}\mathbb{E}_\pi[\|u\|^2]$$

Higher ψ_t and p_t mean more negative cosines and faster progress. **Sufficiency:** If $p_t > \frac{1}{2}$ and R is below the curvature-controlled threshold, the expected step is a descent step. This is the sign-alignment condition from signSGD adapted to constrained optimization.

Definition 5 (Boundary-Crossing Rates)**.** ***Weight rate:***

$$\beta_t = \frac{|\{i : |\Delta\theta_i| > \frac{1}{2}\mathrm{ULP}_i\}|}{|\theta|}$$

Effective rate *(includes activations):*

$$\beta_t^{\text{eff}} = \frac{1}{M}\sum_{m=1}^{M} \mathbf{1}\{\text{cell\_id}_m(\theta_{t+1}) \neq \text{cell\_id}_m(\theta_t)\}$$

where cell_id *is the rounding-decision vector for M probed tensors.*

Both must be reported. *High β_t with low β_t^{eff} indicates boundary crossings in parameters that do not change realized decisions—no real progress.*

Progress requires:

- $\beta_t, \beta_t^{\text{eff}} > 0$: Updates must actually cross cell boundaries on the precision lattice
- High ψ_t: The gradient must live in your update subspace
- $p_t > \frac{1}{2}$: More than half of accessible moves should help

The practitioner's rule: Pick the lowest rank and coarsest precision that keep ψ_t high, $\beta_t, \beta_t^{\text{eff}} > 0$, and $p_t > \frac{1}{2}$.

Generalization check: Compute p_t^{val} and ψ_t^{val} on a held-out batch to verify that a hallway improves validation descent, not only training descent.

6 Explaining the Literature

6.1 LoRA Without Regret

Schulman et al. [1] found empirically:

1. Apply LoRA to all layers (not just attention)
2. Use 10× higher learning rate
3. LoRA matches full fine-tuning in "low regret regime"

Hallway interpretation:

- **"Apply to all layers"** ⇒ maximizes ψ_t. MLPs have 2–3× more parameters than attention. Attention-only LoRA misses most gradient mass. Low ψ_t = hallway to nowhere.
- **"10× learning rate"** ⇒ keeps β_t nonzero. LoRA has fewer parameters; same LR means smaller updates in weight space. Higher LR raises the probability that quantized updates exceed $\frac{1}{2}$ULP (higher β_t), turning nonzero gradients into actual cell exits.
- **"Low regret regime"** = high ψ_t AND high β_t. Hallway exists and you're walking down it.

Why LoRA sometimes fails:

- Attention-only: ψ_t too low, gradient escapes subspace
- Low LR: β_t too low, stuck in cell despite nonzero gradient
- Rank too low for task: hallway doesn't reach good basin

6.2 rsLoRA

Kalajdzievski [17] showed standard α/r scaling causes "gradient collapse" at high rank. The fix: $1/\sqrt{r}$ scaling.

Hallway interpretation: The original scaling made per-parameter updates too small to cross cell boundaries as rank increased. The $1/\sqrt{r}$ scaling maintains β_t across ranks—boundary-crossing rate management.

6.3 Lottery Tickets

Frankle & Carbin [2] showed sparse subnetworks match dense performance. Ramanujan et al. [18] found untrained pruned networks that match trained dense networks.

Hallway interpretation: Most of fp32 parameter space is void—directions that don't help. Pruning removes the void. The remaining sparse network operates in a subspace where directions matter. Lottery tickets are hallways that were always there; pruning reveals them.

6.4 Stochastic Rounding

Gupta et al. [5] showed stochastic rounding enables 16-bit training where round-to-nearest fails.

Hallway interpretation: Deterministic rounding can trap you in a cell—updates below ULP round to zero, $\beta_t = 0$, training stagnates despite nonzero gradients. Stochastic rounding probabilistically crosses boundaries that deterministic rounding cannot. It maintains $\beta_t > 0$ when the learning rate would otherwise be sub-ULP.

6.5 Control Vectors Work

Turner et al. [6] and Zou et al. [10] showed linear directions in activation space steer model behavior.

Hallway interpretation: The gauge symmetry of the loss (huge manifold of equivalent solutions) gets broken by the optimizer. The broken directions have semantic structure—they're not arbitrary. Control vectors exploit these broken gauge directions. The "abundance" is all the hallways that exist on the gauge orbit. The "structure" is which hallways the optimizer actually takes.

7 Quantization as Hallway Selection

A coarse lattice has fewer representable points. Updates that would move toward a nearby bad minimum may:

1. Round to the current point (no movement—blocked hallway)
2. Round to a better point (skip the bad basin entirely)

The lattice geometry determines which hallways exist. Different lattices have different hallway structures. A "worse" lattice (fewer bits) may have better hallways for a specific task.

This explains why:

- **QAT outperforms PTQ** [19]: Training learns which hallways exist on the target lattice. Post-training quantization inherits hallways from a different lattice.
- **Architecture search is precision-dependent**: Different lattices have different hallway structures. The architecture that wins at fp32 may lose at fp4.
- **Same format helps one model, hurts another**: Family-specific loss landscape topology means family-specific hallway geometry.

8 What Would Falsify It

1. **Monotonic degradation across all architectures and tasks.** If constraints always hurt regardless of structure, the hypothesis is wrong.
2. **No architecture $\times$ format interaction.** If format quality is intrinsic (fp32 always beats fp16 always beats fp8), hallway effects aren't real.
3. **Random constraints helping as much as structured constraints.** If randomly zeroing weights helps as much as structured pruning, the "hallway" framing adds nothing.
4. **ψ_t and β_t uncorrelated with training progress.** If the diagnostics don't predict success, the formalization is wrong.

9 Instrumentation

Key diagnostics:

- **Boundary-crossing rate β_t:** Fraction of weight updates $\geq \frac{1}{2}$ ULP
- **Gradient capture ψ_t:** $\|P_S \nabla \mathcal{L}\| / \|\nabla \mathcal{L}\|$ for subspace S
- **ULP-normalized updates:** Histogram of $|\Delta\theta|/\text{ULP}(\theta)$; progress requires mass above 0.5
- **Per-layer ψ_t:** Which layers have gradient escaping the LoRA subspace?

If $\beta_t \to 0$, training stagnates regardless of gradient magnitude. If ψ_t drops at specific layers, those layers need higher rank or full fine-tuning.

Constraints don't just limit where you can go.
They limit where you can go wrong.

10 Connection to Classical Theory

The hallway framework connects to established analyses:

- **Projected/proximal gradient**: Constraints improve descent by removing harmful directions; our p_t quantifies the improvement ratio.
- **SignSGD and gradient compression**: Progress under sign-alignment conditions; $p_t > 1/2$ is the analogous sufficient condition here.
- **Random subspace methods**: Progress with partial gradients when ψ_t is high enough.
- **Error-feedback QSGD**: Achieves $\beta_t > 0$ via residual accumulation, escaping cells that would trap round-to-nearest.

A Estimation Code

```
@torch.no_grad()
def estimate_pt_psit_realized(model, loss_fn, batch,
                              subspace_proj, quantize_step,
                              apply_step, K=64, R=1.0):
    """
    Estimate p_t and psi_t using realized loss (finite difference).

    Args:
        subspace_proj: g -> P_S(g) (project to LoRA subspace)
        quantize_step: u -> Q(u) (apply precision stack)
        apply_step: context manager that temporarily applies u_q
    Returns:
        p_hat: fraction of moves with realized descent
        psi_t: gradient capture ratio (surrogate)
    """
    # Baseline realized loss
    loss_base = loss_fn(model, batch).item()

    # Surrogate gradient for psi_t
    params = [p for p in model.parameters() if p.requires_grad]
    loss = loss_fn(model, batch)
    grads = torch.autograd.grad(loss, params)
    g = torch.cat([gi.reshape(-1) for gi in grads])

    # Gradient capture ratio
    gS = subspace_proj(g)
    psi_t = (gS.norm() / (g.norm() + 1e-12)).item()

    # Sample admissible moves, test REALIZED descent
    p_down = 0
    for _ in range(K):
        z = torch.randn_like(gS)
```

```
        u_dir = gS + 0.1 * z
        u_dir = u_dir / (u_dir.norm() + 1e-12)
        u = R * u_dir
        u_q = quantize_step(u)  # Cast to target lattice

        # Finite-difference test on realized loss
        with apply_step(model, u_q):
            loss_new = loss_fn(model, batch).item()
        if loss_new < loss_base:  # Realized descent
            p_down += 1

    p_hat = p_down / K
    return p_hat, psi_t
```

Generalization variant: Compute p_t^{val} and ψ_t^{val} on a held-out batch to check that a hallway improves validation descent, not only training descent.

Sensitivity: Report p_t under both optimizer proposal and uniform-on-ball to assess measure dependence.

References

[1] Schulman, J., et al. (2025). LoRA without regret. *Thinking Machines Lab*.

[2] Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*.

[3] Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). GPT3.int8(): 8-bit matrix multiplication for transformers at scale. *NeurIPS*.

[4] Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *NeurIPS*.

[5] Gupta, S., Agrawal, A., Gopalakrishnan, K., & Narayanan, P. (2015). Deep learning with limited numerical precision. *ICML*.

[6] Turner, A., Thiergart, L., Udell, D., Leech, G., Mini, U., & MacDiarmid, M. (2023). Activation addition: Steering language models without optimization. *arXiv:2308.10248*.

[7] Ainsworth, S., Hayase, J., & Srinivasa, S. (2023). Git Re-Basin: Merging models modulo permutation symmetries. *ICLR*.

[8] Elhage, N., et al. (2022). Toy models of superposition. *Anthropic*.

[9] Hu, E. J., et al. (2022). LoRA: Low-rank adaptation of large language models. *ICLR*.

[10] Zou, A., et al. (2023). Representation engineering: A top-down approach to AI transparency. *arXiv:2310.01405*.

[11] Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine learning practice and the bias-variance trade-off. *PNAS*.

[12] Neyshabur, B., Sedghi, H., & Zhang, C. (2020). What is being transferred in transfer learning? *NeurIPS*.

[13] Alain, G., & Bengio, Y. (2016). Understanding intermediate layers using linear classifier probes. *arXiv:1610.01644*.

[14] Brea, J., Simsek, B., Ged, F., & Gerstner, W. (2019). Weight-space symmetry in deep networks gives rise to permutation saddles. *arXiv:1907.02911*.

[15] Simsek, B., et al. (2021). Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. *ICML*.

[16] Xie, S., & Li, Z. (2024). Implicit bias of AdamW: ℓ_∞ norm constrained optimization. *ICML*.

[17] Kalajdzievski, D. (2023). A rank stabilization scaling factor for fine-tuning with LoRA. *arXiv:2312.03732*.

[18] Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., & Rastegari, M. (2020). What's hidden in a randomly weighted neural network? *CVPR*.

[19] Jacob, B., et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CVPR*.

[20] Biderman, D., et al. (2024). LoRA learns less and forgets less. *arXiv:2405.09673*.

[21] Fujii, K., et al. (2024). Balancing speed and stability: The trade-offs of FP8 vs. BF16 training in LLMs. *arXiv:2411.08719*.