ModelPinch: Diverse Task Experts Are Dense, Even in 4 Parameters

Anonymous ACL submission

Abstract

Adapting large language models to downstream tasks typically relies on gradient-based fine-tuning, which requires optimizer state, learning-rate schedules, and substantial compute. Recent gradient-free alternatives skip learning entirely by perturbing all model weights with Gaussian noise and ensembling the top-K, but their per-step cost still scales with model size. We introduce *ModelPinch*, a gradient-free post-training procedure that samples random perturbations within an ultra-low-dimensional subspace defined by a frozen singular value decomposition and a fixed random projection, and aggregates the top-K candidates by majority vote. Across multiple open-weight model families and seven reasoning, coding, language, and chemistry benchmarks, *ModelPinch* matches full-weight gradient-free search on approximately half of all evaluated model-benchmark configurations while operating at a search dimension five to nine orders of magnitude smaller. Controlled ablations identify a single critical component - a low-rank orthogonal projection with appropriate rank and orthogonality - that is robust to the specific basis vectors or singular values used. These results demonstrate that effective post-training adaptation can be achieved through geometric, rather than learned, constraints on the weight perturbation.

1 Introduction

Post-training of large language models has steadily reduced the number of trainable parameters needed to adapt a frozen backbone. LoRA (Hu et al., 2022) introduced low-rank weight updates and brought the count from billions down to tens of millions, and SVD-aware variants tightened the budget further by reusing the pretrained singular structure for placement and rank allocation (Zhang et al., 2023; Meng et al., 2024; Gu et al., 2024; Wang et al., 2025). A more recent line goes one or two orders of magnitude lower again, sharing matrices across layers or compressing each layer's update into a single latent vector (Bałazy et al., 2024; Kopiczko et al., 2024; Li et al., 2024, 2025), with TinyLoRA (Morris et al., 2026) reaching the extreme by training as few as a dozen scalar parameters with reinforcement learning. Throughout, adaptation still requires a gradient signal and an optimiser. A parallel line questions whether a gradient signal is necessary at all. Well-pretrained weights lie in flat regions populated with good solutions (Keskar et al., 2017; Mirzadeh et al., 2021), and ensembling predictions across small Gaussian perturbations of the pretrained weights without any parameter update has been shown to recover meaningful gains (Mehrtash et al., 2020). Neural Thickets (Gan and Isola, 2026) extends this with train-split scoring and top-K selection (RandOpt), finding that at scale the pretrained weights are surrounded by a dense population of task-specific experts, so random perturbation plus majority voting reaches gradient-based post-training, at the cost of perturbing the full weight tensor with per-step noise proportional to model size.[1]

The two threads have not met (Figure 1). Extreme parameter-efficient methods learn a tiny subspace by gradient descent; gradient-free methods sample random perturbations across the full weight tensor. The natural cell of this two-by-two has been left empty. Could random sampling work *inside* the ultra-low-dimensional subspace that TinyLoRA defines, with no learning at all?

We propose *ModelPinch*[2], which samples random Gaussian perturbations of a single small vector inside the SVD-projected subspace introduced by TinyLoRA, applies the induced weight delta in place, and aggregates the top-K survivors by majority vote on a 200-problem training split. There

[1] See Appendix A for details.

[2] https://anonymous.4open.science/r/ModelPinch-F3E0/README.md

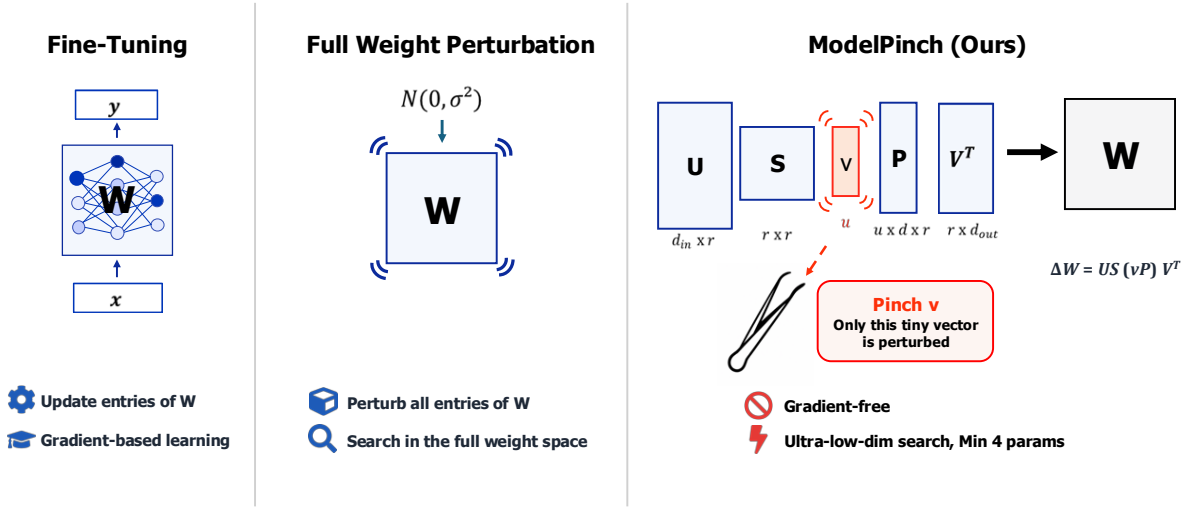

Figure 1: Three paradigms for adapting a pretrained weight matrix W. (a) Gradient-based fine-tuning, which updates the entries of W and requires gradients and an optimiser. (b) Gradient-free full-weight perturbation, which samples Gaussian noise over every entry of W at a per-step cost of $\mathcal{O}(\#\text{params})$. (c) *ModelPinch* (ours) - perturbs only a small vector v (as few as four scalars) inside the frozen TinyLoRA decomposition $\Delta W = US(vP)V^\top$ (with U, S, P, V kept fixed). The search is likewise gradient-free, but its per-step cost is $\mathcal{O}(v_{\text{dim}})$, independent of model size.

is no learning, no optimiser, no gradient computation, and the per-step perturbation cost depends only on the size of the subspace, not on the size of the model.

Contributions

- *ModelPinch* - a gradient-free post-training method that perturbs TinyLoRA's ultra-low-dimensional subspace and aggregates the top-K by majority vote.
- On 49 (model, benchmark) cells across four model families, *ModelPinch* matches Full-Weight RandOpt on half the cells at a search dimension five to nine orders of magnitude smaller.
- Four ablations identify the low-rank orthogonal projection as the only load-bearing ingredient; SVD frame, singular values, and hyperparameters are interchangeable.

2 ModelPinch

ModelPinch occupies the missing cell of a simple 2×2 (Table 1). RandOpt-style methods perturb *all* model weights without gradients (Gan and Isola, 2026); TinyLoRA learns a tiny vector inside an SVD-projected subspace using gradient-based RL (Morris et al., 2026). *ModelPinch* keeps TinyLoRA's choice of *where* to perturb but replaces *how* to search with random sampling and majority-vote ensembling. The resulting method has no learning signal, no optimizer state, and a perturbation cost that is independent of model size.

	Full weights	**TinyLoRA $\mathcal{V}$**
Gradient	SFT / RL	TinyLoRA+GRPO
Random	RandOpt	*ModelPinch* (ours)

Table 1: *ModelPinch* fills the empty cell: random perturbation in the TinyLoRA subspace.

2.1 Preliminaries: TinyLoRA Subspace

Let $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be a frozen weight of one target linear layer (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `down_proj`, `gate_proj`). We compute its truncated SVD once, $W \approx U\,S\,V^\top$ with rank r (we use r=2), and freeze U, S, V. For each target layer m we additionally sample a fixed random tensor $P_m \in \mathbb{R}^{u \times r \times r}$ with a deterministic seed.

We then group layers of the same module type by an integer NTIE. Layers $\{m : \lfloor \text{layer_idx}(m)/\text{NTIE} \rfloor = g\}$ share a single trainable vector $v_g \in \mathbb{R}^u$. The total search dimension v_{dim} is

$$v_{\text{dim}} = u \cdot \sum_{t \in \mathcal{T}} \lceil L_t / \text{NTIE} \rceil, \qquad (1)$$

where $\mathcal{T}$ is the set of module types and L_t the number of layers per type. For

	Full-Weight RandOpt	TinyLoRA+GRPO	*ModelPinch* (ours)
Search dimension	$\sim 10^{8\text{-}10}$	$v_{\text{dim}} \sim 10^{0\text{-}3}$	$v_{\text{dim}} \sim 10^{0\text{-}3}$
Typical σ	$5{\times}10^{-4}$-$5{\times}10^{-3}$	learned	10^{-2}-1
Noise generation cost / step	$\mathcal{O}(\#\text{params})$	-	$\mathcal{O}(v_{\text{dim}})$
Memory beyond base model	one perturbation buffer	optimiser + grads	ΔW_m cache
Gradients required	no	yes	no
Bit-deterministic given seeds	yes	no	yes
Output	ensemble of weights	one trained $v^\star$	ensemble of vs

Table 2: *ModelPinch* keeps RandOpt's gradient-free, ensemble-based search but inherits TinyLoRA's ultra-low-dimensional subspace, allowing several orders of magnitude lower perturbation cost and a much wider usable noise scale.

Qwen2.5-1.5B this gives v_{dim}=4 at the *ultra_tiny* setting (u=1, NTIE=100) and 896 at *large* (u=8, NTIE=1); per-model exact values are in Table 11. The induced weight delta for layer m is

$$\Delta W_m(v) \;=\; U_m\, S_m \left(\textstyle\sum_{i=1}^{u} v_{g(m),i}\, P_{m,i} \right) V_m^\top. \tag{2}$$

We refer to $\mathcal{V} = \{v \in \mathbb{R}^{v_{\text{dim}}}\}$ as the *Pinch subspace*, a low-dimensional affine slice of weight space whose geometry is fully determined by the frozen (U, S, V, P). Unless noted otherwise we use u=1, NTIE=10 (the sensitivity sweep is in Section 4). We use the SVD-derived frame because it is the most natural starting point; Section 7 shows that any orthogonal frame of the same rank works equally well.

2.2 ModelPinch: Random Perturbation in the Pinch Subspace

ModelPinch keeps RandOpt's gradient-free top-K selection but searches only inside an ultra-low-dimensional subspace, replacing gradient-based fitting of v with random Gaussian sampling (Table 2).

Given a base model with the frozen (U, S, V, P) components installed, a perturbation is fully specified by a pair (seed, σ),

$$v(\text{seed}, \sigma) \;=\; \sigma \cdot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_{v_{\text{dim}}}), \tag{3}$$

where ϵ is generated by a PyTorch RNG seeded with seed. Equation (2) maps v to additive deltas $\{\Delta W_m\}_m$ which are added in-place to the corresponding weights. Restoration subtracts the same ΔW_m, recovering the base weights bit-exactly.

For each model and benchmark we draw N unique seeds and assign σ values round-robin from a finite grid Σ_{TL}. Each perturbed model is scored by its mean reward on a small training split of the target task (we use 200 problems unless noted otherwise); the resulting *empirical fitness* is noisy but cheap, and we never compute gradients of it. Algorithm 1 summarises the procedure.

Algorithm 1 *ModelPinch* search and ensembling

Require: Base model θ_0; frozen subspace components (U, S, V, P); training split $\mathcal{D}_{\text{train}}$; test split $\mathcal{D}_{\text{test}}$; reward R; population N; sigma grid Σ_{TL}; ensemble sizes $\{K_j\}$

Sample N unique seeds $s_1, \ldots, s_N$
for $i = 1, \ldots, N$ **do**
  $\sigma_i \leftarrow \Sigma_{\text{TL}}[\, i \bmod |\Sigma_{\text{TL}}|\,]$
  $v_i \leftarrow \sigma_i \cdot \epsilon(s_i)$ ▷ $\epsilon \sim \mathcal{N}(0, I)$
  $\theta_i \leftarrow \theta_0 + \sum_m \Delta W_m(v_i)$
  $f_i \leftarrow \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{x \in \mathcal{D}_{\text{train}}} R(\theta_i; x)$
  Discard θ_i
end for
$\mathcal{S}^\star \leftarrow$ top-K_{max} of $\{(s_i, \sigma_i, f_i)\}$ by f_i
for each $K \in \{K_j\}$ and each $x \in \mathcal{D}_{\text{test}}$ **do**
  Generate answer $a_{i,x}$ under each of the top K models
  $\hat{a}_x \leftarrow \arg\max_a |\{i \le K : a_{i,x} = a\}|$
end for
return Reward of $\hat{a}$ on $\mathcal{D}_{\text{test}}$ for each K

2.3 Selection and Majority-Vote Ensembling

We follow RandOpt's selection and ensembling (Gan and Isola, 2026) - top-K by training fitness for $K \in \{1, 10, 40, 50, 100\}$, then majority-vote on the task-specific extracted answer (numeric for math, formula for Countdown, class for ROCStories, with analogous extractors for MBPP code and USPTO-50k); ties are broken by first occurrence. Voting is at the answer level so different reasoning paths can still agree, and both search and ensembling are bit-deterministic given $(\text{model}, r, u, \text{NTIE}, \text{seed}, \Sigma_{\text{TL}})$.

Benchmark	Train	Test	Max tok.
GSM8K	200	1,319	1024
MATH-500	200	300	2048
OlympiadBench	200	474	2048
Countdown	200	300	1024
MBPP	200	764	2048
ROCStories	200	19,633	64
USPTO-50K	200	4,902	64

Table 3: Benchmark splits used in this work. All evaluations use the instruct chat template and greedy decoding.

3 Experiments

3.1 Experiment Setting

We evaluate *ModelPinch* on seven public benchmarks (Cobbe et al., 2021; Hendrycks et al., 2021; He et al., 2024; Gan and Isola, 2026; Austin et al., 2021; Mostafazadeh et al., 2016; Schwaller et al., 2019) spanning math reasoning, arithmetic puzzles, code generation, commonsense, and chemical reaction prediction (Table 3); each uses a fixed 200-problem training split for *ModelPinch* fitness and the standard test split for evaluation, with task-specific rewards (binary for math/code, partial credit for Countdown, classification accuracy for ROCStories and USPTO-50K).

We use Qwen2.5 (Qwen Team, 2025) at four sizes (0.5B, 1.5B, 3B, 7B) and three cross-family checks - LLaMA-3.2-3B-Instruct, LLaMA-3.1-8B-Instruct, and Mistral-7B-v0.1. Models are loaded in bf16 and served via vLLM (Kwon et al., 2023) with up to four engines per GPU using fractional memory allocation.

Each (model, benchmark) cell runs *ModelPinch* with population $N{=}1000$ for math benchmarks and $N{=}5000$ for ROCStories and USPTO-50K, over the five TinyLoRA configurations (*ultra_tiny* through *large*, parameters in Table 5 caption). Sigma is $\{0.01, 0.05, 0.1, 0.5, 1.0\}$ for *ModelPinch* and $\{5{\times}10^{-4}, 10^{-3}, 2{\times}10^{-3}\}$ for the Full-Weight RandOpt baseline; the reported best ensemble is the maximum over $K \in \{1, 10, 40, 50, 100\}$.

Base is the unmodified instruct model (greedy); Full-Weight RandOpt shares the population, top-K selection, and majority vote of *ModelPinch* but replaces the perturbation step of Algorithm 1 with a Gaussian on every model parameter. A TinyLoRA+GRPO (Morris et al., 2026; Shao et al., 2024) gradient-based reference is reported in Section 3.3.

3.2 Main Results

ModelPinch wins or ties on 25 of 49 cells, Full-Weight RandOpt on 25, with a median absolute gap below 2 accuracy points (Table 5). This is achieved at v_{dim} between 4 and 1152 against $\sim 10^{8\text{-}10}$ parameters for Full-Weight RandOpt - five to nine orders of magnitude smaller search dimension and per-step noise cost. The wider usable sigma range (Table 2) compensates for the smaller dimension.

Gains scale with model size on reasoning-heavy benchmarks - Countdown moves from $.044 \rightarrow .060$ (0.5B) to $.084 \rightarrow .467$ (3B) to $.292 \rightarrow .583$ (7B), and LLaMA-3.1-8B shows the same pattern with GSM8K $.444 \rightarrow .834$. This is consistent with Neural Thickets (Gan and Isola, 2026) - better-pretrained weights are surrounded by a denser cloud of task-specialised solutions that carries through to the Pinch subspace. MBPP gains are modest because most failures are syntactic, and ROCStories sits near base because cloze accuracy is dominated by next-sentence selection. Figure 2 (*left*) gives the per-task bar chart; Figure 5 and Figure 7 (appendix) visualise the weight-space landscape and ensemble accuracy distribution.

3.3 Comparison with GRPO

On Qwen2.5-0.5B / GSM8K we compare *ModelPinch* and TinyLoRA+GRPO (Morris et al., 2026; Shao et al., 2024) inside the same $v_{\text{dim}}{=}12$ subspace, with Full-Weight RandOpt as a full-space reference (Table 4). *ModelPinch* reaches .491, GRPO only .433, and Full-Weight RandOpt .525. The gradient-free search is more effective than gradient-based GRPO training in this ultra-low-dimensional subspace (Shao et al., 2024; Mania et al., 2018).[3]

Method	Acc.
GRPO	.433
ModelPinch tiny ($K{=}100$)	<u>.491</u>
Full-Weight RandOpt ($K{=}50$)	**.525**

Table 4: *ModelPinch* outperforms GRPO training inside the same $v_{\text{dim}}{=}12$ subspace (Qwen2.5-0.5B / GSM8K).

[3] GRPO hyperparameters are listed in Appendix K.

Model	Benchmark	Base Full-Weight	*ModelPinch* (ours) utiny (v_{dim}=4)	tiny (12-16)	small (40-64)	medium (192-288)	large (768-1152)	RandOpt Full-Weight
Qwen2.5-0.5B	GSM8K	.434	.475	.491	.486	.483	.504	**.525**
	MATH-500	.313	.367	.337	.353	.340	.390	**.407**
	OlympiadBench	.108	.139	.120	.118	.118	.118	**.148**
	Countdown	.044	.007	.010	**.060**	.050	.050	.020
	MBPP	.270	.270	.272	**.277**	.268	.264	.251
	ROCStories	.319	.318	.318	.318	.320	**.321**	.318
	USPTO-50K	.315	.304	.292	.315	.316	**.322**	.315
Qwen2.5-1.5B	GSM8K	.549	.605	.631	.643	.638	.632	**.682**
	MATH-500	.453	.553	.577	**.580**	.527	.553	.567
	OlympiadBench	.215	.302	.308	.289	.298	.306	**.319**
	Countdown	.063	.113	.133	.103	.077	.100	**.153**
	MBPP	.500	.507	.503	**.531**	.530	.517	.521
	ROCStories	.466	.459	.453	.465	.477	.402	**.499**
	USPTO-50K	.073	**.315**	**.315**	**.315**	.313	**.315**	**.315**
Qwen2.5-3B	GSM8K	.790	.846	**.848**	.847	.837	.789	.842
	MATH-500	.603	**.740**	.723	.717	.707	.690	.707
	OlympiadBench	.344	.428	.420	.428	**.430**	.416	.428
	Countdown	.084	.460	.423	**.467**	.437	.407	**.467**
	MBPP	.605	.619	.615	.623	.636	.623	**.645**
	ROCStories	.538	.540	.547	.538	.497	.386	**.557**
	USPTO-50K	.063	.315	**.316**	.315	.315	.315	.239
LLaMA-3.2-3B	GSM8K	.635	.721	.742	**.755**	.750	.738	.732
	MATH-500	.310	.157	.227	**.330**	.207	.243	.257
	OlympiadBench	.125	.169	.192	.181	.186	.188	**.198**
	Countdown	.044	.387	.477	.500	.483	.467	**.523**
	MBPP	.429	.470	.404	.462	.384	**.495**	.471
	ROCStories	.414	.418	.425	.442	**.447**	.435	.424
	USPTO-50K	.178	.335	.316	.325	.255	.327	**.352**
Qwen2.5-7B	GSM8K	.811	.839	.869	.889	.857	.861	**.898**
	MATH-500	.703	.750	.763	**.777**	.757	.753	.757
	OlympiadBench	.397	.487	.498	.500	.496	.508	**.519**
	Countdown	.292	.563	.583	.577	.583	.567	**.597**
	MBPP	.687	.673	**.704**	.691	.685	.692	.696
	ROCStories	.684	.685	.697	.691	.688	.681	**.698**
	USPTO-50K	.266	.326	.318	.296	**.367**	.320	.334
Mistral-7B-v0.1	GSM8K	.354	.424	.422	.447	.412	.422	**.472**
	MATH-500	.050	.077	.080	.067	.067	.067	**.093**
	OlympiadBench	.013	.015	.015	**.021**	**.021**	.019	**.021**
	Countdown	.096	.010	.047	.059	.066	.046	**.125**
	MBPP	.296	.319	.304	.305	.312	.310	**.343**
	ROCStories	.343	.416	.397	.398	.394	**.449**	.416
	USPTO-50K	.271	.323	**.331**	.315	.323	.315	.316
LLaMA-3.1-8B	GSM8K	.444	.691	.756	.821	.834	.803	**.847**
	MATH-500	.330	**.587**	.557	.530	.533	.520	.560
	OlympiadBench	.162	.253	.264	.253	.279	**.295**	.291
	Countdown	.514	.541	.538	.520	.534	.530	**.557**
	MBPP	.564	.592	.613	.615	.617	.630	**.632**
	ROCStories	.052	.260	.527	.430	.520	.563	**.590**
	USPTO-50K	.013	.317	**.378**	.291	.315	.315	.328

Table 5: Per-cell best ensemble accuracy across the 7 × 7 model-benchmark grid for *Base* (unperturbed), *RandOpt* (full-weight perturbation), and *ModelPinch* (ours, ultra-low-dimensional perturbation). Despite searching five to nine orders of magnitude fewer parameters, *ModelPinch* wins or ties on 25 of 49 cells, matching Full-Weight RandOpt's 25. Per-row **best** and second-best are marked; per-model exact v_{dim} is in Table 11.

4 Are the Hyperparameters at Their Sweet Spots?

The main experiments fix a single configuration of the five *ModelPinch* hyperparameters (SVD truncation rank r, fitness-evaluation set size n_{fit}, population N, noise scale grid σ, and ensemble-size grid K), chosen to match TinyLoRA's defaults. Whether these settings sit at saturation or simply happen to work has not been quantified.

We sweep each axis independently around the baseline (r=2, n_{fit}=200, N=1000, $\sigma \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$, $K \in \{1, 10, 40, 50, 100\}$) on Qwen2.5-0.5B / GSM8K with three random seeds (42, 1, 2); low σ shifts the grid one decade down to $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ and high σ one decade up to $\{0.1, 0.5, 1.0, 5.0, 10.0\}$. Twelve variant configurations replace one axis at a time (Table 6); all other settings match the baseline.

The twelve variant configurations land in a .444 to .492 band, all within roughly 5 absolute points of *Base* (.491) and several below it. No single axis sweep dominates - the highest cell (O2, N=100: .492) is statistically indistinguishable from *Base*,

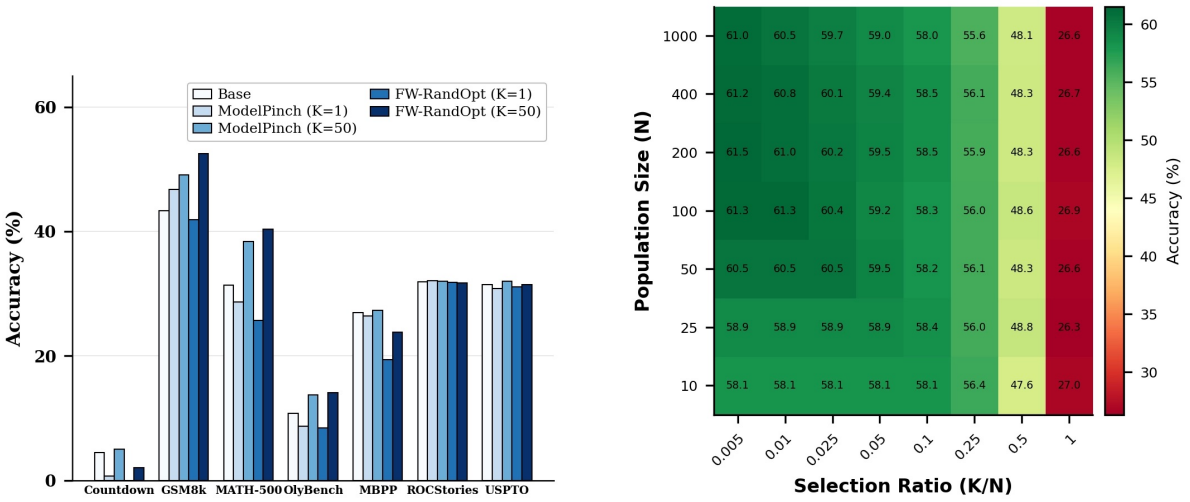

Figure 2: *Left* is the ensemble effect on Qwen2.5-0.5B across seven benchmarks - base vs. *ModelPinch* and Full-Weight RandOpt, each at $K{=}1$ (single perturbation) and $K{=}50$ (top-50 majority vote); the $K{=}50$ ensemble consistently matches or exceeds Full-Weight RandOpt at the same K. *Right* is a joint sweep over population size N and selection ratio K/N on Qwen2.5-0.5B / GSM8K (Section 4), averaged across all five TinyLoRA capacities; accuracy is essentially flat across N above $N \approx 100$ and degrades only at the largest selection ratios.

ID	Axis	Values	Acc.
Base	-	-	*.491*
O1	K	+200	.476 ± .024
O2	N	100	.492 ± .015
O3	N	500	.471 ± .031
O4	N	2000	.464 ± .054
O5	n_{fit}	50	.461 ± .052
O6	n_{fit}	100	.455 ± .056
O7	n_{fit}	500	.458 ± .013
O8	r	1	.461 ± .023
O9	r	4	.461 ± .022
O10	r	8	.452 ± .049
O11	σ	low	.444 ± .048
O12	σ	high	.471 ± .071

Table 6: One-axis-at-a-time sweep on Qwen2.5-0.5B / GSM8K (three seeds); twelve variants land in a narrow .444-.492 band around *Base* (.491).

and the lowest (O11, σ low: .444) sits 4.7 points below. The widest dispersions appear at high σ (O12: ±.071), large rank (O10: ±.049), and small n_{fit} (O5, O6: ±.052, .056), all consistent with regimes that increase per-perturbation variance. The rank insensitivity matches the low intrinsic rank of adapter tuning, where even rank one is competitive (Hu et al., 2022); since our r is the perturbation-basis SVD truncation rank, this is a consistency, not an equivalence. Figure 2 (*right*) confirms the same plateau as a joint $N \times K/N$ sweep on GSM8K averaged across all five TinyLoRA capacities, mirroring the $N \times K$ saturation surface reported for RandOpt (Gan and Isola, 2026). No axis carries the configuration substantially above or below *Base*, so *ModelPinch*'s effectiveness does not hinge on any single tuned knob.

5 Where in the Model Should We Pinch?

The main experiments perturb every target module of Qwen2.5-0.5B uniformly. Whether this coverage is necessary or whether a layer-group (early/mid/late) and module-type (attention/MLP) restriction suffices at lower v_{dim} is open. We compare twelve combinations: three layer groups (early covering layers 0-7, mid 8-15, late 16-23 of the 24-layer model) crossed with three module-type subsets (attention-only, MLP-only, both), plus three all-layer variants (*Base* at $v_{\text{dim}}{=}12$, attention-only and MLP-only at reduced v_{dim}). v_{dim} ranges from 2 to 12 across the cells. Three seeds (42, 1, 2).

Table 7 shows that all twelve cells land in a narrow 0.423-0.452 band (range 2.9 accuracy points), and even the smallest $v_{\text{dim}}{=}2$ restriction stays within 6.8 points of *Base* (.491). Restricting to a single layer group costs at most 6.8 accuracy points (O7 mid/attn: .423), and the late-layer "both" cell (O9: .452, $v_{\text{dim}}{=}4$) tops the restricted-cell band at one third of *Base*'s dimension. In early layers attention-only and MLP-only are tied, but in mid and late groups MLP-only exceeds attention-only by 1.3 and 1.7 points respectively, with the late-MLP cell (O11: .447, $v_{\text{dim}}{=}2$) the strongest

ID	Layers	Modules	v_{dim}	Acc.
Base	all	both	12	*.491*
O1	all	attn-only	6	.444 ± .009
O2	all	MLP-only	6	.448 ± .018
O3	early	both	4	.445 ± .020
O4	early	attn-only	2	.431 ± .005
O5	early	MLP-only	2	.431 ± .014
O6	mid	both	4	.444 ± .012
O7	mid	attn-only	2	.423 ± .003
O8	mid	MLP-only	2	.436 ± .006
O9	late	both	4	.452 ± .002
O10	late	attn-only	2	.430 ± .017
O11	late	MLP-only	2	.447 ± .008

Table 7: Layer group × module type ablation on Qwen2.5-0.5B / GSM8K (three seeds); all twelve cells stay in a narrow 0.423-0.452 band. *Base* is the all-layer / all-module default (v_{dim}=12).

sub-baseline configuration. Figure 6 (appendix) visualises the same cells against the base model.

The orthogonal-projection signal identified in Section 6 is therefore distributed across the entire model, and the best subset (O9 late + both, v_{dim}=4) comes within 3.9 points of *Base* at one third of the dimension. Late layers and MLP modules contribute marginally more at the same v_{dim}, consistent with prior characterizations of where adaptation-relevant signal sits in pre-trained transformers (Geva et al., 2021; Zhang et al., 2023; Song et al., 2026; Lee et al., 2023). The dominant lever is the orthogonal-projection structure itself rather than which part of the model carries it.

6 Is the TinyLoRA Subspace Itself Necessary?

We isolate the two structural components of $W' = W + USvPV^\top$ - the SVD top-r prior U, S, V and the vP mixing - and remove each independently (Table 8). R1 drops vP mixing ($W' = W + USvV^\top$); R2 replaces the SVD prior with a same-shape random matrix ($W' = W + vMP$); R3 drops both ($W' = W + vM$).

Variant	SVD	vP	Acc.
Base	✓	✓	*.491*
R1	✓	-	.489 ± .010
R2	-	✓	.004 ± .001
R3	-	-	.005 ± .001

Table 8: R2 and R3 (no SVD prior) collapse to within noise of zero; R1 (no vP) is indistinguishable from *Base* (Qwen2.5-0.5B / GSM8K, three seeds).

The SVD top-r prior is therefore essential to *ModelPinch*, whereas the vP mixing is not. This mirrors, in a gradient-free regime, the finding of PiSSA (Meng et al., 2024) that the principal singular subspace of W is the high-leverage direction for adaptation. However, R2 removes the orthogonal-projection structure and the values inherited from W's SVD at the same time, so these results alone cannot tell which of the two is load-bearing the orthogonal-projection structure itself, or the specific values it carries. We dissect this distinction in Section 7.

7 What Frozen Components Make Pinch Work Best?

What structure the projection must have (low-rank and orthogonal) was settled in Section 6; *what content* inside that structure carries the signal is open. LoRA-XS (Bałazy et al., 2024) and TinyLoRA (Morris et al., 2026) both adopt SVD-derived U, V frames as inductive priors, but neither has tested whether those specific values are necessary or whether any structurally faithful frame suffices, especially under gradient-free search.

Holding v_{dim}=12 and the rest of the baseline fixed, we replace each frozen (U, S, V, P) component with a structurally faithful but information-free random alternative. U, V are replaced with random orthonormal frames (*orth-rand*); S with a $\mathcal{N}(0, 1)$ diagonal (*diag-rand*) or a diagonal at the top-r singular-value mean (*diag-uniform-mean*); and P with *orth-rand* or *diag-rand*. Eight configurations result (Table 9), on Qwen2.5-0.5B / GSM8K with three seeds.

ID	Replaced	Replacement	Acc.
Base	-	-	*.491*
O1	S	diag-rand	.488 ± .002
O2	S	diag-uniform-mean	.500 ± .007
O3	U, V	orth-rand	.497 ± .007
O4	U, V, S	orth-rand + diag-rand	.473 ± .004
O5	U, V, S	orth-rand + diag-uniform-mean	.496 ± .001
P1	P	orth-rand	.491 ± .002
P2	P	diag-rand	.494 ± .005

Table 9: Replacing each frozen TinyLoRA component with a structure-preserving random alternative keeps seven of eight cells within ∼1 point of *Base*; only O4 drops to .473 (Qwen2.5-0.5B / GSM8K, three seeds).

Seven of the eight configurations fall in a .488 to .500 band. Only O4 (.473), which replaces U, V and S jointly, falls outside it, 1.7 points below *Base*; every single-component substitution is benign. The solution density $\delta(\sigma{=}0)$, the fraction of

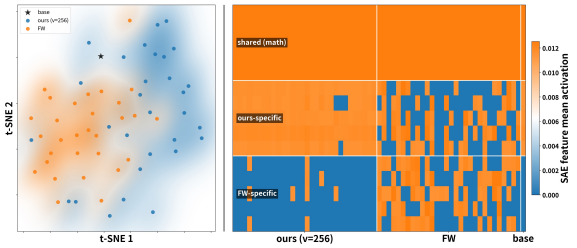

Figure 3: Representation analysis on MATH-500 using LLaMA-3.1-8B. *Left*: t-SNE of representation embeddings for the base model, Full-Weight RandOpt (FW), and *ModelPinch* (v_{dim}=256). *Right*: sparse autoencoder (SAE) feature activation heatmap across the compared methods.

perturbations that beat the base model, rises with the injected randomness, from .216 at *Base* to .518 at O4, yet ensemble accuracy stays flat or declines; a more random prior produces more individually-adequate perturbations, but their answers diverge enough that majority voting does not turn the surplus into a gain.

LoRA-XS (Bałazy et al., 2024) is the only prior work to ablate these same frozen components, and with a learnable middle matrix in the gradient regime it reaches the opposite conclusion, that the SVD prior helps. The difference is learnability. LoRA-XS trains the middle map, so the SVD prior acts as an exploitable initialisation, whereas *ModelPinch* freezes the middle map entirely and searches only a 12-dimensional vector without gradients. Under this constraint neither the SVD-derived directions (O3 vs. *Base*) nor the singular-value content (O2 vs. *Base*) carries the relevant information. This is consistent with LoRA-XS and isolates the mechanism behind their result, namely that the SVD prior is exploitable only by a learnable middle map. Combined with Section 6, *ModelPinch* needs a structured low-rank projection of the correct shape, but absent a learnable middle map the prior's content is dispensable - under random search any structurally faithful frame suffices.

8 Discussion

Our main empirical claim is that random search in a low-rank orthogonal subspace of the pretrained weights matches random search in the full weight space, and the construction of that subspace contributes nothing beyond its right shape. The four ablations decompose this. The subspace must be a proper orthogonal projection (Section 6); but any orthogonal projection works, including Haar-random frames and uniform-mean diagonals (Section 7); where it sits only weakly matters (Section 5); and the search itself is at saturation across two-decade hyperparameter variations (Section 4).

An orthogonal projection preserves row and column norms of ΔW, so adding $USvPV^\top$ is a controlled perturbation in the input/output directions the pretrained model already uses. A non-orthogonal random matrix of the same shape distorts these norms unpredictably, consistent with the .005 collapse we observe. Our finding is that this geometric stability alone carries the signal, without any gradient update.

The same geometric stability shows up at the representation level (Figure 3). In the t-SNE space, samples from *ModelPinch* (v_{dim}=256) remain closer to the base model distribution, while Full-Weight RandOpt shows a noticeably larger shift; the SAE activation heatmap shows a similar trend, with both methods retaining shared mathematical features but Full-Weight RandOpt additionally activating a broader set of optimisation-specific sparse features.

Quantitatively, Full-Weight RandOpt activates 13× more SAE features off-base than *ModelPinch* (4.61% vs. 0.35% of 32,768 features with $|\Delta a| > 1\sigma$; Appendix J). *ModelPinch* preserves the pretrained representation structure while reaching performance comparable to full-weight perturbation.

The base-to-ensemble gap grows monotonically with model size on reasoning-heavy benchmarks in Table 5, so the dense neighbourhood of task-specific experts reported by Gan and Isola (2026) survives an aggressive subspace restriction.

9 Conclusion

We introduced *ModelPinch*, a gradient-free post-training method that perturbs only a low-rank orthogonal subspace of the pretrained weights and aggregates the top-K survivors by majority vote. Across 49 (model, benchmark) cells from four model families, it matches Full-Weight RandOpt at a search dimension five to nine orders of magnitude smaller, with four ablations isolating the orthogonal-projection structure of the subspace as the single load-bearing ingredient. Effective post-training adaptation can thus rest on geometric, not learned, constraints on the weight perturbation.

10 Limitations

Our empirical study has four main limitations. *Model family.* 28 of the 49 cells in Table 5 come from the Qwen2.5 series; cross-family coverage now adds LLaMA-3.2-3B, LLaMA-3.1-8B, and Mistral-7B-v0.1 (21 cells), with OLMo-2-7B still pending. The LLaMA-3.1-8B/ROCStories (base .052), LLaMA-3.1-8B/USPTO-50K (base .013), and 1.5B/USPTO-50K (base .073) cells share an unusually-low-base pattern, and their large absolute gains should be read with a reward-formatting caveat. *Training-fitness noise.* Per-perturbation fitness is the mean reward on a 200-problem training split; the resulting noise is non-trivial and we have not yet quantified how the population variance of the top-K selection translates into ensemble-accuracy variance. *Hyperparameter coverage.* We sweep five TinyLoRA configurations, five sigma values, and four top-K ratios for the main table, but with a single global seed; replicate runs are needed for confidence intervals on Table 5 (the four ablation sections do report mean $\pm$ std over three seeds). *Gradient comparison.* *ModelPinch* versus TinyLoRA trained with GRPO on the same architecture is reported on Qwen2.5-0.5B / GSM8K (Section 3.3); broader head-to-head coverage across the seven models and seven benchmarks of Table 5 is still pending.

Ethical considerations. *ModelPinch* is a gradient-free post-training method that requires only inference-time access to a model's weights. The same property that makes it lightweight also lowers the technical barrier to producing post-trained variants of any open-weight model on commodity hardware, which is in tension with safety alignment that relies on the difficulty of further training. We restrict our experiments to publicly released benchmarks and instruct models, do not produce or distribute alignment-relaxed checkpoints, and recommend that future work on subspace-restricted gradient-free methods report safety probe results alongside capability metrics.

Model and dataset licenses. All models and benchmarks used in this paper are publicly released under permissive or research-use licenses, and our use is consistent with the terms of each. Models - Qwen2.5 (0.5B, 1.5B, 3B, 7B) under Apache 2.0; LLaMA-3.2-3B-Instruct under the LLaMA 3.2 Community License; LLaMA-3.1-8B-Instruct under the LLaMA 3.1 Community License; Mistral-7B-v0.1 under Apache 2.0. Benchmarks - GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021) under MIT; OlympiadBench (He et al., 2024) under Apache 2.0; Countdown released with Neural Thickets (Gan and Isola, 2026); MBPP (Austin et al., 2021) under CC-BY-4.0; ROCStories (Mostafazadeh et al., 2016) released for non-commercial research use; USPTO-50K (Schwaller et al., 2019) derived from US Patent Office data in the public domain. We use each benchmark's standard test split for evaluation only, without redistribution.

11 Use of Generative AI

We used Claude Opus (versions 4.5, 4.6, and 4.7) to assist with prose polishing and code drafting. All original research ideas, experimental design choices, and final decisions were made by the human authors; AI assistance was limited to writing refinement and implementation support.

References

Jacob Austin, Augustus Odena, Maxwell Nye, and 1 others. 2021. Program synthesis with large language models. arXiv:2108.07732.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2024. LoRA-XS: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, and 1 others. 2021. Training verifiers to solve math word problems. arXiv:2110.14168.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Andrew Gan and Phillip Isola. 2026. Neural thickets: Diverse task experts are dense around pretrained weights. ArXiv:2603.12228.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *EMNLP*.

Jihao Gu, Shuai Chen, Zelin Wang, Yibo Zhang, and Ping Gong. 2024. SARA: Singular-value based adaptive low-rank adaption. *arXiv preprint arXiv:2408.03290*.

Chaoqun He and 1 others. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *ACL*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, and 1 others. 2021. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *ICLR*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. VeRA: Vector-based random matrix adaptation. In *ICLR*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, and 1 others. 2023. Efficient memory management for large language model serving with PagedAttention. In *SOSP*.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. 2023. Surgical fine-tuning improves adaptation to distribution shifts. In *ICLR*.

Kaiyang Li, Shaobo Han, Qing Su, Wei Li, Zhipeng Cai, and Shihao Ji. 2025. Uni-LoRA: One vector is all you need. In *NeurIPS*.

Yang Li, Shaobo Han, and Shihao Ji. 2024. VB-LoRA: Extreme parameter efficient fine-tuning with vector banks. In *NeurIPS*.

Qiyao Liang, Jinyeop Song, Yizhou Liu, Jeff Gore, Ila Fiete, Risto Miikkulainen, and Xin Qiu. 2026. The blessing of dimensionality in LLM fine-tuning: A variance-curvature perspective. *arXiv preprint arXiv:2602.00170*.

Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search of static linear policies is competitive for reinforcement learning. In *NeurIPS*.

Alireza Mehrtash, Purang Abolmaesumi, Polina Golland, Tina Kapur, Demian Wassermann, and William M. Wells. 2020. PEP: Parameter ensembling by perturbation. arXiv:2010.12721.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *NeurIPS*.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. 2021. Linear mode connectivity in multitask and continual learning. In *ICLR*.

Tom Morris and 1 others. 2026. Learning to reason in 13 parameters. ArXiv:2602.04118.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, and 1 others. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL*.

Qwen Team. 2025. Qwen2.5 technical report.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. 2020. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *ICLR*.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. ArXiv:1703.03864.

Philippe Schwaller, Teodoro Laino, Théophile Gaudin, and 1 others. 2019. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS Central Science*.

Zhihong Shao and 1 others. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. ArXiv:2402.03300.

Xinyuan Song, Keyu Wang, PengXiang Li, Lu Yin, and Shiwei Liu. 2026. Demystifying the roles of LLM layers in retrieval, knowledge, and reasoning. In *ICASSP*.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. 2025. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In *NAACL*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*.

A Extended Related Work and Method Definitions

This appendix gathers formal definitions of the parameter-efficient and gradient-free post-training methods cited in Section 1, alongside their parameter counts and effective search dimensions. Let $W \in \mathbb{R}^{m\times n}$ denote a frozen pretrained weight, and let $W = U_{\text{full}}\Sigma_{\text{full}}V_{\text{full}}^{\top}$ denote its full SVD with truncated top-r factors $U \in \mathbb{R}^{m\times r}$, $\Sigma = \text{diag}(s_1, \ldots, s_r)$, and $V \in \mathbb{R}^{n\times r}$.

A.1 Low-Rank Adapters with a Trainable Inner Matrix

LoRA (Hu et al., 2022). The weight update factorises as $\Delta W = BA$, with trainable $A \in \mathbb{R}^{r\times n}$, $B \in \mathbb{R}^{m\times r}$, and rank $r \ll \min(m, n)$. Trainable parameters per module are $r(m + n)$.

AdaLoRA (Zhang et al., 2023). Allocates rank adaptively across modules by parameterising $\Delta W = P\Lambda Q$ with diagonal Λ and orthogonality-regularised P, Q; important singular values are kept and the rest pruned during training. Effective parameters per module shrink over training.

SARA (Gu et al., 2024). Selects per-layer rank by inspecting the truncated SVD of W, then trains $\Delta W = BA$ at the chosen ranks.

PiSSA (Meng et al., 2024). Initialises LoRA with the principal singular components of W, so $B_0 = U\sqrt{\Sigma}$ and $A_0 = \sqrt{\Sigma}\,V^{\top}$; both are trainable thereafter, and the residual $W - B_0A_0$ is frozen. Parameter count matches LoRA at rank r.

MiLoRA (Wang et al., 2025). Reverses PiSSA's choice and initialises trainable factors from the *minor* singular components, freezing the principal subspace. Parameter count matches LoRA.

Curvature analysis (Liang et al., 2026). Supports the SVD-aware family more broadly by showing that fine-tuning curvature concentrates along a small set of task-relevant directions, consistent with low-rank parameterisations.

A.2 Ultra-Low-Dimensional Adapters with Frozen Random Projections

LoRA-XS (Bałazy et al., 2024). Freezes LoRA's B and A at SVD-derived orthogonal frames U, V and inserts a trainable $r \times r$ middle matrix R, giving $\Delta W = URV^{\top}$ with r^2 trainable scalars per module.

VeRA (Kopiczko et al., 2024). Shares a single pair of frozen random A, B across all layers and trains only two per-layer scaling vectors. Trainable parameters per module are $m + r$.

V-BLoRA (Li et al., 2024). Maintains a global vector bank and indexes module updates by sparse mixtures over it, with trainable mixture coefficients.

UniLoRA (Li et al., 2025). Compresses each layer's update into a single latent vector mapped through a fixed projection to the full ΔW shape.

TinyLoRA (Morris et al., 2026). Compresses the per-module update to $\Delta W = US(vP)V^{\top}$, where U, S, V come from the top-r SVD of W and $P \in \mathbb{R}^{u\times r\times r}$ is a fixed random projection sampled once per layer with a deterministic seed. A single shared trainable vector $v \in \mathbb{R}^u$ is tied across NTIE consecutive layers; total trainable parameters reduce to $v_{\text{dim}} = u \cdot \sum_t \lceil L_t/\text{NTIE}\rceil$ (as few as twelve scalars for a 1.5B model). TinyLoRA fits v by gradient-based RL (GRPO). *ModelPinch* reuses this exact subspace but replaces gradient fitting with random Gaussian search over v, described in Section 2.

A.3 Gradient-Free Post-Training of Pretrained Weights

MAML and ANIL (Finn et al., 2017; Raghu et al., 2020). The fast-adaptation literature attributes much of the gain from meta-learning and pretraining to feature reuse in the initialisation rather than rapid representation learning, providing the theoretical foundation for the well-pretrained-weights argument used by both RandOpt and *ModelPinch*.

Evolution strategies and random search (Salimans et al., 2017; Mania et al., 2018). Evolution Strategies (ES, 2017) estimates a gradient via Gaussian perturbations and updates a maintained mean parameter; Augmented Random Search (ARS, 2018) shows that random search matches ES and policy-gradient RL in low-dimensional control. Both keep state across iterations, unlike *ModelPinch* and RandOpt, which retain only the top-K survivors and discard intermediate state.

Perturbed-ensembles / PEP (Mehrtash et al., 2020). Draws N i.i.d. samples $\theta_i = \theta_0 + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$, evaluates each on a held-in set, and averages predictions; no parameter update.

Neural Thickets / RandOpt (Gan and Isola, 2026). Extends PEP with train-split fitness scoring and top-K selection. Given a 200-problem training split with reward R, it samples N full-weight perturbations, computes $R(\theta_i)$, and ensembles the top-K by majority vote at inference. Per-step noise generation is $\mathcal{O}(\#\text{params})$ and the effective search dimension equals the model parameter count, $\sim 10^{8\text{-}10}$ for the models in our study.

A.4 Parameter and Search-Dimension Comparison

Table 10 summarises trainable parameter counts and effective search dimensions across the methods above for a representative 1.5B-parameter transformer with $r = 2$, $u = 1$, $\text{NTIE} = 10$, $v_{\text{dim}} = 12$, and 96 target modules.

Method	Grad.	Per-module count	Order
Full fine-tuning	yes	mn	$\sim 10^9$
LoRA / PiSSA / MiLoRA	yes	$r(m+n)$	$\sim 10^7$
AdaLoRA	yes	$\leq r(m+n)$	$\sim 10^7$
LoRA-XS	yes	r^2	$\sim 10^3$
VeRA	yes	$m+r$	$\sim 10^5$
UniLoRA / V-BLoRA	yes	1 vec / layer	$\sim 10^3$
TinyLoRA + GRPO	yes	v_{dim} tied	$\sim 10^1$-10^3
RandOpt	no	mn	$\sim 10^9$
ModelPinch (ours)	no	v_{dim} tied	$\sim 10^1$-10^3

Table 10: Trainable parameter or effective search dimension per target module for the methods cited in Section 1 ($W \in \mathbb{R}^{m\times n}$, rank r; *Order* is the model-wide order of magnitude for a 1.5B transformer). *ModelPinch* occupies the gradient-free, ultra-low-dimensional cell that prior work had left empty.

ModelPinch occupies the gradient-free cell at TinyLoRA's search dimension. The other low-dimensional methods (LoRA-XS, VeRA, UniLoRA, V-BLoRA, TinyLoRA) all keep a gradient backend; the gradient-free methods (PEP, RandOpt) all perturb the full weight tensor. The intersection has not been explored before.

B Per-Model ModelPinch Search Dimensions

Table 11 reports the exact search dimension v_{dim} for each (model, capacity) cell underlying Table 5. Values differ across models because $v_{\text{dim}} = u \cdot \sum_t \lceil L_t/\text{NTIE} \rceil$ depends on the per-module-type layer count of each architecture.

Model	utiny	tiny	small	medium	large
Qwen2.5-0.5B	4	12	40	192	768
Qwen2.5-1.5B	4	12	48	224	896
Qwen2.5-3B	4	16	64	288	1152
LLaMA-3.2-3B	4	12	48	224	896
Qwen2.5-7B	4	12	48	224	896
Mistral-7B-v0.1	4	16	56	256	1024
LLaMA-3.1-8B	4	16	56	256	1024

Table 11: Exact v_{dim} (number of trainable scalars per model) for each *ModelPinch* capacity configuration. Values differ across models because $v_{\text{dim}} = u \cdot \sum_t \lceil L_t/\text{NTIE} \rceil$ depends on the number of layers per module type. As a fraction of total model parameters, the search dimension ranges from $5\times10^{-8}\,\%$ (utiny, LLaMA-3.1-8B) to $1.54\times10^{-4}\,\%$ (large, Qwen2.5-0.5B); Full-Weight RandOpt operates at $100\,\%$.

The *utiny* cell is constant at v_{dim}=4 across all seven models because at u=1, NTIE=100 each architecture collapses to four module-type groups. The *large* cell scales with layer count - 24 layers (Qwen2.5-0.5B) give 768; 28-32 layers (Qwen2.5-1.5B / LLaMA-3.2-3B / Qwen2.5-7B / Mistral) give 896-1024; LLaMA-3.1-8B's 32 layers reach 1024. Larger models therefore search a marginally larger v_{dim}, but the order of magnitude (10^2-10^3) is the same.

C Per-Cell Method Comparison Scatter

Figure 4 expands Table 5 into a per-cell scatter against parity. Points above the dashed line favor *ModelPinch* and points below favor Full-Weight RandOpt; the spread shows that the wins-or-ties roughly split evenly across model scales and benchmark families.

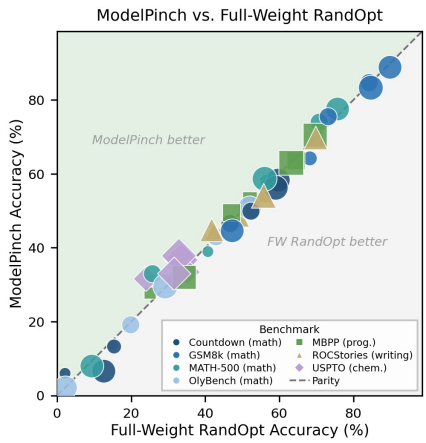

Figure 4: Per-cell scatter of *ModelPinch* (y) vs. Full-Weight RandOpt (x); marker size encodes model scale (0.5B-8B), color benchmark, shape task family. Points above the dashed parity line favor *ModelPinch*.

No single benchmark or model scale dominates either side of parity, supporting the 24/24 wins-or-ties split reported in Section 3.2. The points clustered far above parity are mostly small-base cells (e.g., 1.5B/USPTO-50K, LLaMA-3.1-8B/USPTO-50K) - the same low-base anomalies flagged in the main caption.

D Accuracy Landscape Around Base Weights

Figure 5 visualises the 2D-projected accuracy landscape around each Qwen2.5 base model under 200 random full-weight perturbations. Larger models display a denser red region near base (small perturbations match or improve performance), consistent with the Neural Thickets density argument that underlies the *Larger models gain more* observation in Section 3.2.

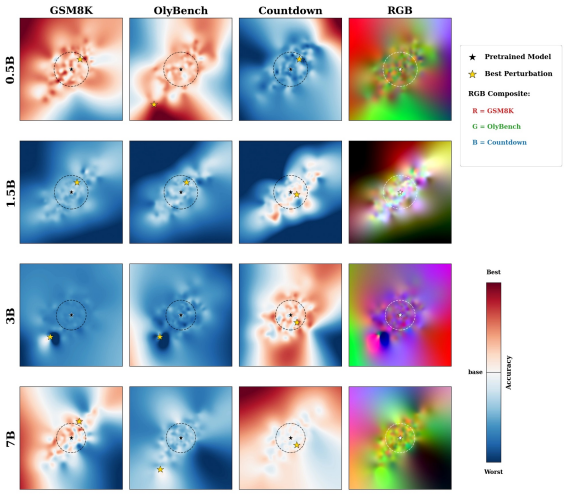

Figure 5: Accuracy landscapes in weight space for Qwen2.5 models (0.5B-7B) under 200 random weight perturbations projected to 2D. Color shows relative accuracy change $(\text{acc} - \text{base})/\text{base} \times 100$ (blue, worse; white, equal; red, better); the dashed circle marks the mean perturbation distance and stars mark the best perturbations. The last column maps GSM8K, OlympiadBench, and Countdown accuracies to R, G, B channels.

The denser red zone at larger scale also explains why *ModelPinch*'s subspace restriction does not collapse the gain - the task-favourable region survives projection onto any low-rank orthogonal slice (Section 7). The RGB panel further shows that the best perturbations for different benchmarks rarely coincide, so single-perturbation cells are noisy and top-K ensembling is doing the work.

E Per-K Breakdown of the Hyperparameter Sweep

Table 12 reports per-K ensemble accuracy for the thirteen sweep configurations of Section 4.

ID	K=1	K=10	K=40	K=50	K=100	K=200
Base	.467	.478	.489	.485	.491	-
O1	.417 ± .037	.460 ± .033	.474 ± .024	.474 ± .023	.472 ± .021	.474
O2	.418 ± .038	.459 ± .025	.479 ± .016	.489 ± .014	.492 ± .015	-
O3	.415 ± .039	.457 ± .039	.462 ± .038	.466 ± .038	.468 ± .027	-
O4	.410 ± .073	.456 ± .050	.462 ± .052	.461 ± .056	.460 ± .060	-
O5	.409 ± .065	.453 ± .051	.453 ± .054	.458 ± .054	.461 ± .051	-
O6	.400 ± .049	.445 ± .054	.450 ± .053	.454 ± .056	.451 ± .052	-
O7	.410 ± .054	.452 ± .015	.456 ± .013	.455 ± .011	.448 ± .023	-
O8	.384 ± .061	.441 ± .041	.458 ± .025	.459 ± .022	.452 ± .028	-
O9	.396 ± .063	.447 ± .029	.461 ± .022	.456 ± .025	.453 ± .032	-
O10	.380 ± .060	.440 ± .052	.451 ± .048	.450 ± .051	.452 ± .049	-
O11	.392 ± .062	.438 ± .042	.437 ± .058	.435 ± .056	.434 ± .051	-
O12	.335 ± .078	.432 ± .076	.467 ± .068	.471 ± .072	.469 ± .069	-

Table 12: Per-K ensemble accuracy on GSM8K, mean ± std across three seeds (Qwen2.5-0.5B). *Base* is a single seed run. O1 is the only configuration extended to K=200, reaching .474 (essentially flat against K=100: .472 ± .021).

The K plateau visible in the table justifies the $K \in \{1, 10, 40, 50, 100\}$ grid used in the main results - no additional accuracy is gained by extending K past ≈ 50, so the reported best-K cells in Table 5 are not artefacts of an under-sized ensemble.

F Analysis by Layer and Module

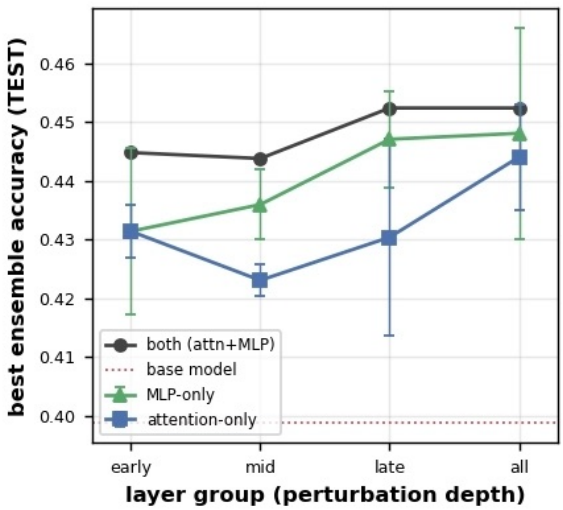

Figure 6: Selective perturbation on Qwen2.5-0.5B (GSM8K) - best-ensemble test accuracy by layer group and module type. Markers are single-seed results (n=1); the dotted line is the base model.

All comparisons fix v_{dim} to isolate location from parameter count. At matched v_{dim}, the late layer

group outperforms the other two in two of three module budgets (attn-only at v_{dim}=2 ties early); among both-module configurations at v_{dim}=4, late reaches .452 against .445 (early) and .444 (mid), and perturbing late layers with both module types comes within four points of the all-layer *Base* (.491) at one third of its dimensions. The depth effect is not a smooth gradient; mid matches or trails early in two of three module conditions, and the signal concentrates in the late layers for both-module and MLP-only cells. This is consistent with AdaLoRA (Zhang et al., 2023) routing budget to top layers, Surgical Fine-Tuning (Lee et al., 2023) handling output-level shifts via the last layer, and Song et al. (2026) placing reasoning in deep layers; Geva et al. (2021) further show that upper feed-forward layers encode semantic rather than surface patterns.

Within every layer group, MLP-only and attention-only configurations share the same v_{dim}. The two are tied in early layers, but MLP-only exceeds attention-only by 1.3 and 1.7 points in mid and late groups (Figure 6). This is consistent with the feed-forward sublayer acting as a key-value store of task-relevant content (Geva et al., 2021) and with AdaLoRA's preferential allocation to feed-forward modules (Zhang et al., 2023); attention primarily routes information between positions rather than storing it. Gradient-based budgeting, selective fine-tuning, and probing reach the same conclusion through independent means.

G Parameterisation Variants for the Subspace Necessity Ablation

G.1 Formal Definitions of the Four Variants

Let $W \in \mathbb{R}^{m\times n}$ be a frozen pretrained weight, let $W = U\Sigma V^\top$ with $U \in \mathbb{R}^{m\times r}$, $\Sigma = \text{diag}(s_1, \ldots, s_r)$, $V \in \mathbb{R}^{n\times r}$ denote its top-r singular value decomposition, and let $v \in \mathbb{R}^{v_{\text{dim}}}$ be the perturbed scalar group shared across n_{tie} modules. Throughout we use r=2, u=1, n_{tie}=10, yielding v_{dim}=12 groups across 96 target modules. All four variants match the same scalar count v_{dim}=12 so that random search operates over an identical search space; only the lift from v to ΔW differs (Table 13).

Variant	ΔW	SVD	vP	Rank / Transform
Base	$US(vP)V^\top$	✓	✓	r=2, top-r SVD
R1	$v \cdot USV^\top$	✓	-	r=2, top-r SVD
R2	$v \cdot MP$	-	✓	full-n, raw $\mathcal{N}(0,1)$
R3	$v \cdot M$	-	-	full-n, raw $\mathcal{N}(0,1)$

Table 13: Definitions of the four variants. The rows share an identical scalar count v_{dim}=12 and differ only in how v is lifted to the per-module update ΔW. $M \in \mathbb{R}^{m\times n}$ and $P \in \mathbb{R}^{n\times n}$ for the full-n rows.

Note on R2's rank. P has size $n \times n$, not $r \times r$. The r-rank structure inherited from the SVD prior is part of the SVD prior, so the ablation must not reintroduce it through P. At initialisation we precompute MP and store it per module as a single $m \times n$ matrix; M and P are then freed. Runtime perturbation reduces to $v_{\text{scalar}} \cdot MP$. A rank-matched random baseline would not test whether the SVD direction itself carries the contribution as stringently.

G.2 Per-K Breakdown

Table 14 breaks the four variants down by ensemble size K.

Variant	K=1	K=10	K=40	K=50	K=100
Base	.467	.478	.489	.485	.491
R1	.452 ± .006	.485 ± .014	.479 ± .006	.480 ± .004	.476 ± .012
R2	.004 ± .001	.001 ± .001	.000 ± .000	.000 ± .000	.000 ± .000
R3	.005 ± .001	.003 ± .001	.000 ± .000	.000 ± .000	.000 ± .000

Table 14: Per-K ensemble accuracy on GSM8K (Qwen2.5-0.5B). *Base* is a single seed-42 run; the remaining rows are mean ± std across three seeds.

The R2 / R3 collapse is uniform across the entire K grid - no value of K recovers the SVD-prior-less variants to anything above noise, ruling out an ensemble-size confound.

G.3 σ-Sweep at the Sampling Phase

Table 15 reports the train-accuracy spread across σ for the four variants during the sampling phase.

Variant	σ=0.01	σ=0.05	σ=0.1	σ=0.5	σ=1.0
Base	.588 ± .018	.555 ± .033	.492 ± .074	.018 ± .058	.001 ± .002
R1	.582 ± .018	.561 ± .034	.506 ± .065	.031 ± .075	.001 ± .003
R2	.002 ± .003	.001 ± .003	.001 ± .003	.002 ± .003	.001 ± .003
R3	.004 ± .005	.001 ± .003	.001 ± .003	.001 ± .003	.001 ± .002

Table 15: Mean train accuracy at each σ during the RandOpt sampling phase (Qwen2.5-0.5B / GSM8K, N=1000). The *Base* row is a single seed-42 run; the remaining rows are means across three seeds (mean ± within-run std).

The σ-sweep confirms the per-K finding from another angle - R2 / R3 stay at noise across two decades of σ, so the collapse is not an artefact of an over-aggressive perturbation scale. *Base* and R1 track each other tightly up to σ=0.1 before both degrade, the expected onset of the perturbation regime breaking.

H Parameterisation Invariance under Gradient Training

The Section 7 ablation studies what carries the signal under *gradient-free* search inside a frozen TinyLoRA-shaped subspace. LoRA-XS (Bałazy et al., 2024) runs the analogous component substitution in the *gradient* regime, where the middle map of the TinyLoRA decomposition is learnable. Replacing the SVD-derived U, V with random orthogonal frames there *degrades* accuracy, the opposite of what we observe under random search. The two findings are reconcilable. With a learnable middle map the SVD prior acts as an exploitable initialisation that gradient descent can exploit to find a good $v^\star$. With the middle map frozen, the prior collapses to its structural shape: a low-rank orthogonal projection of the right rank. As shown in *ModelPinch*'s O3 / O5 cells (Table 9), *any* structurally faithful frame supplies that shape, so the SVD-derived directions no longer carry the contribution. This isolates the LoRA-XS finding to the gradient regime, and predicts that gradient-free methods inside any fixed orthogonal frame of the right rank should reach *ModelPinch*-level accuracy.

I ModelPinch Ensemble Test-Set Distributions

Figure 7 shows the test-set accuracy distribution of *ModelPinch* ensembles relative to each model's base on GSM8K and Countdown. The shaded region marks ensembles at or above base; most cells sit comfortably inside it, confirming that the ensemble-level gain in Table 5 is not driven by a few lucky tails.

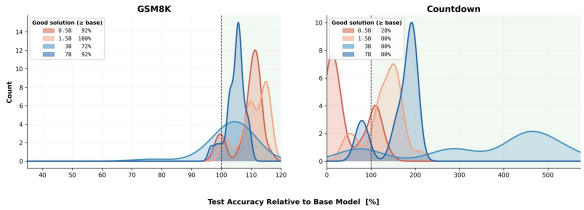

Figure 7: Test-set performance distributions of *ModelPinch* ensembles relative to each model's base accuracy on GSM8K (left) and Countdown (right). The shaded region marks ensembles matching or exceeding base performance. Note - the x-axis shows relative performance; longer tails in smaller models do not equate to higher absolute accuracy.

Smaller models show longer relative tails, but absolute accuracy is dominated by base; the ensemble-level monotonicity in Section 3.2 comes from the at-or-above-base mass rather than from extreme tail values. Countdown's bi-modal distribution at 3B/7B reflects the formula-language nature of the task - single perturbations either find or miss the correct expression, with little in between.

J Representation Analysis Metrics

This appendix reports the quantitative metrics underlying Figure 3. We compare *ModelPinch* (v_{dim}=256) and Full-Weight RandOpt on LLaMA-3.1-8B / MATH-500. Behavioral metrics (LEFT panel) are computed on 100 test problems; representation-level metrics (RIGHT panel) use a 32,768-dimensional SAE trained on the base model's hidden states.

Metric	*ModelPinch*	FW	Ratio
Answer change ratio (Hamming)	15.5%	19.7%	1.27×
Accuracy gap from base	−0.5 pp	−1.77 pp	3.5×
SAE features off-base, $\lvert\Delta a\rvert>1\sigma$	0.35%	4.61%	13.2×
SAE features off-base, $\lvert\Delta a\rvert>2\sigma$	0.11%	2.11%	18.8×
SAE cosine distance to base	0.0028	0.0173	6.2×

Table 16: Behavioral and representation-level deviation from base for *ModelPinch* (v_{dim}=256) and Full-Weight RandOpt on LLaMA-3.1-8B / MATH-500 (100 problems for the behavioral rows, 32,768-feature SAE for the representation rows; base accuracy 25.0%). FW shifts further from base on every metric, with the largest gap at the SAE feature-activation level (13-19× more off-base features).

The behavioral and representation-level ratios disagree by an order of magnitude (1.3-3.5× for behavior vs. 6-19× for features), which is the central message - both methods reach *similar* accuracy on MATH-500, but Full-Weight RandOpt

does so by reshaping a much broader set of internal features. The geometric stability claim of Section 8 thus has a feature-level corroboration distinct from the t-SNE visualisation.

K Pinch vs. Gradient TinyLoRA

The GRPO (Shao et al., 2024) run reported in Section 3.3 trains the same TinyLoRA scalar vector v (v_{dim}=12) that *ModelPinch* searches gradient-free. We use a HuggingFace fp32 gradient pipeline with the hyperparameters in Table 17. The reference policy π_{ref} is the base model (v=0) held frozen throughout training; the KL term against π_{ref} uses the k3 (Schulman) estimator. All other settings follow the GRPO defaults from Shao et al. (2024).

Hyperparameter	**Value**
lr	2×10^{-4}
num_epochs	25
batch_size	64
group_size G	8
clip_eps (PPO clip)	0.2
inner_epochs	1
kl_beta β	0.001
sample_temperature	1.0
sample_top_p	0.95
grad_clip	1.0
optimizer	Adam
seed	42
π_{ref}	base (v=0), frozen
KL estimator	k3 (Schulman)

Table 17: GRPO training hyperparameters used in Section 3.3.