THE REFERENCE MATTERS: DISPLACEMENT PROBING FOR WEIGHT-SPACE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight-space learning predicts a model's attributes directly from its parameters, and probing-based methods do so efficiently by reading a model's responses to learned probes. Prior work shows that probing succeeds within a Model Tree—models fine-tuned from a shared pretrained origin—but not why. We trace this to the shared origin itself: since a tree's leaves start from the same weights, what distinguishes them is their displacement from that origin. This motivates probing the displacement rather than the raw weight, an intervention we call the **DiSplacement Probe (DSProbe)**. We develop this idea along two axes: extending it to second-order Gram responses, whose displacement we show encodes the learned features induced by fine-tuning; and treating the reference as a design choice, learning a shared component to subtract beyond the pretrained origin. Across four architectures on the Model Jungle benchmark, the resulting **Refined-DSProbe (R-DSProbe)** sets a new state of the art, matches prior accuracy with substantially fewer probes, and supports a single encoder shared across Model Trees, reducing metanetwork parameters by $3.61\times$. [1]

1 INTRODUCTION

The number of publicly available neural networks is growing rapidly, yet most are released with little documentation of what they were trained on. Weight-space learning offers a way to characterize such models directly from their parameters: it trains metanetworks that take another network's weights as input and predict its attributes—such as its training data or generalization behavior—without external metadata (Eilertsen et al., 2020; Unterthiner et al., 2020; Schürholt et al., 2021).

Among these, probing-based methods are the most scalable: rather than processing the full weight tensor, they read a model's responses to a few learned probes, which are compact and permutation-equivariant (Kahana et al., 2025). ProbeX (Horwitz et al., 2025) showed that public models form *Model Trees*—checkpoints fine-tuned from a shared ancestor—within which a single layer suffices to predict attributes, and MVProbe (Heo et al., 2026) enriched this with higher-order, Gram-based views. Both establish *that* these choices work, but not *why*: what structure in weight space makes probing responses capture the model attributes?

We trace this to the shared pretrained origin of a Model Tree. Because its leaves begin fine-tuning from the same origin, that origin cancels when leaves are compared, so what distinguishes them is their displacement from their reference. This motivates probing the *displacement* rather than the raw weight, an intervention we call the *DiSplacement Probe* (DSProbe). We develop it along two axes. First, the same reasoning extends to the second-order Gram, whose change along training encodes learned features (Radhakrishnan et al., 2024; Cha et al., 2026); probing the Gram displacement thus records not merely where a weight has moved but what features that movement has learned. Second, we treat the reference itself as a design choice: rather than fixing it to the pretrained origin, we learn a shared component to subtract, isolating the variation that may be more task-specific; we call this learned-reference extension the *Refined DiSplacement Probe (R-DSProbe)*. All variants keep the architecture and training of prior probing methods unchanged, altering only the probe input.

[1] Code is available at `https://anonymous.4open.science/r/R-DSProbe-148C`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

On the Model Jungle dataset (Horwitz et al., 2025), R-DSProbe sets a new state of the art for source-class membership prediction across all four architectures (ResNet (He et al., 2016), SupViT (Dosovitskiy et al., 2021), MAE (He et al., 2022), and DINO (Caron et al., 2021)).

Our contributions are as follows:

- We identify the shared pretrained origin as an effective reference that makes leaf-specific variation more distinguishable, and introduce DSProbe to probe this displacement rather than the raw weight.
- We extend DSProbe to second-order Gram responses and give a feature-learning account of their gain, showing that their benefit comes from the learned features induced by fine-tuning rather than from expressivity alone.
- We treat the reference as a design axis rather than a fixed origin, learning a shared component to subtract; this *Refined-DSProbe (R-DSProbe)* reaches state-of-the-art accuracy on Model Jungle across all four architectures.

2 Related Work

2.1 Weight-Space Learning

Early works predict properties such as training configuration (Eilertsen et al., 2020) and generalization behavior (Unterthiner et al., 2020) from the raw parameters of individual models. This line of work has been extended to yield general representations for multiple downstream tasks (Herrmann et al., 2024; Schürholt et al., 2021; 2022; 2024).

A key challenge with model weights is their high dimensionality and parameter space symmetry (Zhao et al., 2025). Subsequent works incorporate inductive biases for learning from model weights using graph-based representations (Kofinas et al., 2024; Lim et al., 2024; Zhang et al., 2023) and symmetry-aware metanetworks (Navon et al., 2023; Zhou et al., 2023a;b). While symmetry-aware, these approaches still operate directly on parameter space, which limits their scalability to large models.

To address these challenges, probing-based approaches (Kahana et al., 2025; Herrmann et al., 2024; Horwitz et al., 2025; Heo et al., 2026) have emerged as a dominant paradigm. By characterizing model weights through their responses to learnable probes, these methods naturally avoid weight symmetry, improving scalability to large models. ProbeGen (Kahana et al., 2025) introduces a probe generator to impose structure across learned probes. ProbeX (Horwitz et al., 2025) observes that public models are organized into *Model Trees*, and improves scalability by extending probing to a single hidden layer within a tree. Most recently, MVProbe (Heo et al., 2026) expands this linear-layer probing with transposed and Gram-based views to recover information missed by the linear response.

While both ProbeX and MVProbe have improved the scalability and expressiveness of single-layer probing, a fundamental question remains unresolved. Our work addresses this gap by showing how shared initialization induces task-relevant model variation and how different probing views expose this variation.

2.2 Feature Learning

Weight space reflects training-induced changes in the learned representations. Feature learning studies how training reshapes the representations learned by a network. Training can align learned features with task-relevant directions (Baratin et al., 2021), while fine-tuning can substantially alter pretrained representations (Kumar et al., 2022). Such representational changes have been studied through their geometry (Kornblith et al., 2019; Chou et al., 2025). The Neural Feature Ansatz (Radhakrishnan et al., 2024) relates a layer's weight Gram matrix to the outer-product structure of its feature gradients, with subsequent analyses further characterizing this relationship (Tansley et al., 2026). Cha et al. (2026) show that changes in the weight Gram during training track the evolution of feature covariance.

3 BACKGROUND

3.1 GOAL OF WEIGHT-SPACE LEARNING

For a task τ with dataset $D_\tau = \{(x_i, y_i)\}_{i=1}^{n_\tau}$ and target function f_τ, we train f_θ by minimizing a task-specific loss $\mathcal{L}_\tau$ measuring the mismatch between f_θ and f_τ:

$$\theta^{(\tau)} = \arg\min_\theta \mathcal{L}_\tau(\theta), \qquad \mathcal{L}_\tau(\theta) = \frac{1}{n_\tau} \sum_{i=1}^{n_\tau} \ell_\tau \left(f_\theta(x_i), f_\tau(x_i) \right). \tag{1}$$

Weight-space learning turns this setup around: instead of solving for $\theta^{(\tau)}$, it treats the trained parameters themselves as data. Given a collection of trained models $C = \{(\theta^{(i)}, a^{(i)})\}_{i=1}^{N}$, where $a^{(i)}$ is a model attribute of interest (e.g., a source-class label or a CLIP embedding), the goal is to predict the attribute $a^{(i)}$ directly from the parameters $\theta^{(i)}$.

3.2 SINGLE-LAYER PROBING WITHIN MODEL TREES

Early work characterizes an entire model through its responses $f_\theta(\mathbf{U})$ to a learnable probe matrix $\mathbf{U}$ (Kahana et al., 2025). Recently, ProbeX (Horwitz et al., 2025) extends probing to a single hidden layer. Let $\theta = \{W_l\}_{l=1}^{L}$ denote the layer weight matrices of a model and let $W_l \in \mathbb{R}^{m \times n}$ be the selected layer, where n and m are input and output dimensions. The probing network for W_l is

$$\Phi_l = W_l \mathbf{U} \in \mathbb{R}^{m \times r}, \qquad \mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r] \in \mathbb{R}^{n \times r}, \qquad \hat{a}^{(i)} = g(\Phi_l), \tag{2}$$

where Φ_l denotes the probe response of W_l and g denotes an encoder mapping these responses to the predicted attribute, with probes and encoder trained jointly.

3.3 MULTI-VIEW PROBING OF A SINGLE LAYER

With a fixed probe matrix $\mathbf{U}$ of rank $r < n$, the linear response $W_l\mathbf{U}$ does not uniquely characterize W_l. MVProbe (Heo et al., 2026) targets the limited expressivity of first-order probing by introducing second-order Gram views. The first-order row and column views capture directional responses, while their Gram counterparts capture pairwise relationships, together providing four complementary views:

$$\Phi_{\text{row}} = W_l \mathbf{U}_{\text{row}}, \quad \Phi_{\text{col}} = W_l^\top \mathbf{U}_{\text{col}}, \quad \Phi_{\text{row-ker}} = W_l W_l^\top \mathbf{U}_{\text{row-ker}}, \quad \Phi_{\text{col-ker}} = W_l^\top W_l \mathbf{U}_{\text{col-ker}}.$$

4 MOTIVATION

What makes within-tree variation easier to probe?

ProbeX (Horwitz et al., 2025) observes that public models can be organized into Model Trees. A Model Tree has a shared pretrained checkpoint as its *root* and, as its *leaves*, the models obtained by fine-tuning that root on different source tasks. ProbeX shows that within a tree, a single hidden layer can suffice to predict model attributes. It attributes this improvement to reduced *nuisance variation*, i.e., weight variation unrelated to the target attribute, but does not explain why model attributes are accessible from the remaining variation. We show that this variation arises from task-conditioned displacement from the shared pretrained origin in Section 5.1.

Is expressivity the right objective for probing?

MVProbe (Heo et al., 2026) frames probe expressivity as a central motivation for introducing second-order views. It shows that there exist distinct weight matrices W_l and W_l' whose first-order probe responses are identical, $W_l\mathbf{U} = W_l'\mathbf{U}$ while remaining distinguishable under second-order views $W_l^\top W_l \mathbf{U} \neq W_l'^\top W_l' \mathbf{U}$ (Heo et al., 2026, Thm. 4.1). However, expressivity alone does not fully explain why the second-order responses are particularly effective, as the same argument extends to higher order:

Theorem 1 (Expressiveness of Higher-Order Probes)**.** Let $\mathbf{U} \in \mathbb{R}^{n\times r}$ have full column rank, where $1 \le r < n$. For each integer $q \ge 0$, define the alternating higher-order features:

$$\Phi_{2q}(\mathbf{X}) := (\mathbf{X}^\top \mathbf{X})^q \mathbf{U} \in \mathbb{R}^{n\times r}, \qquad \Phi_{2q+1}(\mathbf{X}) := \mathbf{X}(\mathbf{X}^\top \mathbf{X})^q \mathbf{U} \in \mathbb{R}^{m\times r}. \tag{3}$$

Then, for any integer $k \ge 2$ satisfying

$$n - r \ge \left\lfloor \frac{k}{2} \right\rfloor, \qquad m \ge \left\lceil \frac{k}{2} \right\rceil,$$

there exist distinct matrices $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^{m\times n}$ such that

$$\Phi_j(\mathbf{X}_1) = \Phi_j(\mathbf{X}_2), \qquad 1 \le j < k, \qquad \text{but} \qquad \Phi_k(\mathbf{X}_1) \ne \Phi_k(\mathbf{X}_2). \tag{4}$$

Consequently, adding k-th-order branches can separate weight matrices that are indistinguishable by all lower-order responses. However, this representational advantage does not by itself guarantee better attribute prediction. Indeed, ablations in MVProbe (Heo et al., 2026, Table 9) and our higher-order ablation in Table 1 show that third- and fourth-order responses can underperform the second-order Gram despite comparable or greater probe capacity. Its effectiveness must therefore be explained beyond expressivity alone. Instead, its effectiveness can be understood through the feature-learning signal encoded by Gram-changing updates.

Table 1: Model Jungle accuracy (%) when augmenting MVProbe with higher-order branches. Capacity controls match the number of branches without introducing higher-order responses.

Method	ResNet	SupViT	MAE	DINO
MVProbe	91.09	92.41	81.24	77.88
Degree-3	**93.66**	92.78	81.12	77.18
Capacity control	92.36	92.75	81.49	78.63
Degree-3+4	93.46	92.83	81.36	77.04
Capacity control	92.22	**93.24**	**82.02**	**78.72**

For a layer $z = Wh$, Cha et al. (2026) define the virtual feature update $h^+ = h - \eta \nabla_h \mathcal{L}$ and the resulting *virtual covariance shift* (VCS): $h^+(h^+)^\top - hh^\top$. They show that removing the component of the parameter gradient that leaves the weight Gram unchanged largely preserves, and in some settings improves, training performance, indicating that Gram-changing updates capture a substantial part of the feature-learning signal. In Section 5.2, we connect this result to task-conditioned finetuning and show how reference subtraction exposes the corresponding Gram displacement. Details of Theorem 1 and Table 1 are in Appendix C

5 METHOD

5.1 DISPLACEMENT PROBE: SEEING THE TREE BY STANDING AT ITS ROOT

Task-conditioned displacement accumulates in the trained weight.

Let $\theta_t = \{W_{l,t}\}_{l=1}^L$ denote the model parameters at step t, with $W_{l,t} \in \mathbb{R}^{m\times n}$ the selected layer, and let $\mathcal{L}_\tau$ denote the task loss (Section 3.1). Under gradient descent, the selected layer is updated by

$$W_{l,t+1} = W_{l,t} - \eta \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t). \tag{5}$$

The per-step displacement $-\eta \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)$ is evaluated at the current model state θ_t, measuring its mismatch from the target function f_τ. Each update thus encodes in the selected layer a task-conditioned displacement that locally steers f_{θ_t} toward f_τ. Summing equation 5 over the full trajectory gives

$$W_l = W_{l,0} + \Delta_W, \qquad \Delta_W = -\eta \sum_{t=0}^{T-1} \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t), \tag{6}$$

where $W_l := W_{l,T}$. Thus, Δ_W is the accumulated task-conditioned displacement, formed by gradients evaluated along the full-model trajectory $\{\theta_t\}_{t=0}^{T-1}$.

Models within a tree share a common pretrained origin.

Within a Model Tree, the shared pretrained root W_{pre} replaces the initial weight $W_{l,0}$,

$$\theta_0^{(\tau)} = \theta_{\text{pre}}, \qquad W_{l,0}^{(\tau)} = W_{\text{pre}}, \qquad \forall \tau. \tag{7}$$

In what follows, we index each leaf $W_{l,T}^{(\tau)}$ by its source task τ. Comparing leaves fine-tuned on tasks τ and τ', the root cancels already in their first-step difference, leaving only the difference of task-specific loss gradients, both evaluated at the same root state $\theta_0 = \theta_{\text{pre}}$:

$$W_{l,1}^{(\tau)} - W_{l,1}^{(\tau')} = -\eta \left[\nabla_{W_{l,0}} \mathcal{L}_\tau(\theta_0) - \nabla_{W_{l,0}} \mathcal{L}_{\tau'}(\theta_0)\right] = -\eta \, \nabla_{W_{l,0}} \left(\mathcal{L}_\tau - \mathcal{L}_{\tau'}\right)(\theta_0). \tag{8}$$

Indeed, at the first step, all leaves share the same parameter state, so each task induces a distinct gradient direction in the shared parameter space. After fine-tuning, every leaf can be written as $W_{l,T}^{(\tau)} = W_{\text{pre}} + \Delta_W^{(\tau)}$, so the shared pretrained component cancels when comparing leaves. Thus, within a tree, inter-leaf differences are entirely determined by the displacements $\Delta_W^{(\tau)}$ accumulated through fine-tuning on their respective source tasks. This directly motivates probing $\Delta_W = W_l - W_{\text{pre}}$ in place of the trained weight W_l itself: When W_{pre} is the fine-tuning initialization, Δ_W is exactly the accumulated task-conditioned displacement in equation 6. We call this intervention the *DiSplacement Probe* (DSProbe); it retains the architecture of single-layer probing methods, while changing the input to the *displacement* from its raw counterpart.

5.2 DiSplacement Probe for Weight Gram

Each Gram update is a task-conditioned feature displacement.

To extend the displacement view from weights to their Gram responses, we examine how the Gram matrices evolve during task-conditioned fine-tuning, following the Gram-update calculation of Cha et al. (2026). For the selected layer at step t, let H_t denote the input features induced by task τ, and let $Z_t = W_{l,t} H_t$ denote the corresponding output features. Using the chain rule and a first-order expansion of the gradient update in equation 5, the two Gram orientations evolve as

$$\begin{aligned} W_{l,t+1}^\top W_{l,t+1} - W_{l,t}^\top W_{l,t} &= -2\eta \, \mathrm{Sym}\big(\nabla_{H_t} \mathcal{L}_\tau(\theta_t) \, H_t^\top\big) + \mathcal{O}(\eta^2), \\ W_{l,t+1} W_{l,t+1}^\top - W_{l,t} W_{l,t}^\top &= -2\eta \, \mathrm{Sym}\big(\nabla_{Z_t} \mathcal{L}_\tau(\theta_t) \, Z_t^\top\big) + \mathcal{O}(\eta^2), \end{aligned} \tag{9}$$

where $\mathrm{Sym}(A) := (A + A^\top)/2$. Following the VCS interpretation of Cha et al. (2026), the first relation describes the first-order virtual covariance shift of the layer input H_t, while the second gives the analogous shift of its output Z_t. Thus, a task-conditioned update of the layer Gram admits a direct feature-space interpretation. Moreover, by the shared-origin argument of Section 5.1, all task leaves at the first step start from the same root parameters. Their Gram shifts are therefore expressed in the same layer-wise coordinate system, differing only through the task-conditioned update. The derivation of equation 9 is in Appendix D.

Task-conditioned Gram changes accumulate in the trained model.

The single-step Gram shifts accumulate along the fine-tuning trajectory. Summing the first-order updates in equation 9 gives

$$(W_l^{(\tau)})^\top W_l^{(\tau)} = W_{l,0}^\top W_{l,0} + \Delta_H^{(\tau)}, \qquad W_l^{(\tau)} (W_l^{(\tau)})^\top = W_{l,0} W_{l,0}^\top + \Delta_Z^{(\tau)}, \tag{10}$$

where $\Delta_H^{(\tau)}$ and $\Delta_Z^{(\tau)}$ denote the accumulated task-conditioned feature-space shifts along the optimization trajectory. As with the first-order weight in Section 5.1, we remove the shared pretrained component and probe only the displacement, retaining its probing architecture and training procedure while replacing the raw Gram inputs with

$$\Delta_H = W_l^\top W_l - W_{\text{pre}}^\top W_{\text{pre}}, \qquad \Delta_Z = W_l W_l^\top - W_{\text{pre}} W_{\text{pre}}^\top, \tag{11}$$

Here, Δ_H and Δ_Z record what features that movement has learned. Together with the first-order displacements Δ_W and $\Delta_W^\top$, which record where the weight has moved, they yield the following displacement-based response formulations for DSProbe:

$$\Phi_W = \Delta_W \mathbf{U}_W, \qquad \Phi_{W^\top} = \Delta_W^\top \mathbf{U}_{W^\top}, \qquad \Phi_H = \Delta_H \mathbf{U}_H, \qquad \Phi_Z = \Delta_Z \mathbf{U}_Z. \tag{12}$$

5.3 REFINED DISPLACEMENT PROBE

The gains from probing displacements $\Delta_W, \Delta_H, \Delta_Z$ suggest that for weight-space prediction, removing the shared components across the leaves makes task-relevant variation more accessible. We extend this principle to the displacement itself, adaptively removing a component that may remain shared across leaf displacements. Although Δ removes the shared root, the displacements may still admit a task-common component. We introduce a learnable matrix $C \in \mathbb{R}^{m \times n}$, with the same shape as W_l, to capture such a component. Let $\widetilde{\Delta}^{(\tau)}$ denote the remaining task-dependent variation, so that

$$\Delta^{(\tau)} = \widetilde{\Delta}^{(\tau)} + C, \tag{13}$$

and we probe the refined displacement $\widetilde{\Delta}$. We optimize C jointly with the probing objective, allowing the removed component to adapt to the model attribute of interest. Substituting $\Delta_W = W_l - W_{\mathrm{pre}}$ gives $\widetilde{\Delta}_W = W_l - (W_{\mathrm{pre}} + C)$. We denote the resulting shared component by

$$W_* = W_{\mathrm{pre}} + C, \tag{14}$$

so that all displacement representations are refined as

$$\widetilde{\Delta}_W = W_l - W_*, \qquad \widetilde{\Delta}_H = W_l^\top W_l - W_*^\top W_*, \qquad \widetilde{\Delta}_Z = W_l W_l^\top - W_* W_*^\top. \tag{15}$$

The corresponding refined probe responses are

$$\widetilde{\Phi}_W = \widetilde{\Delta}_W U_W, \qquad \widetilde{\Phi}_{W^\top} = \widetilde{\Delta}_W^\top U_{W^\top}, \qquad \widetilde{\Phi}_H = \widetilde{\Delta}_H U_H, \qquad \widetilde{\Phi}_Z = \widetilde{\Delta}_Z U_Z. \tag{16}$$

We refer to this refined-reference extension of DSProbe as ***R-DSProbe*** (Refined DiSplacement Probe). For implementation, we directly optimize the refined reference W_* rather than parameterizing the offset C. Since $W_l^{(\tau)} = W_{\mathrm{pre}} + C + \widetilde{\Delta}_W^{(\tau)}$, we initialize W_* with the mean of the leaf weights within each tree, excluding the held-out models used for evaluation, and then optimize it jointly with the probing objective. This construction extends naturally when the exact W_{pre} is unavailable. Since models can be clustered according to their tree structure directly from weight-space distances (Horwitz et al., 2025), the mean of each cluster provides a natural initialization for W_*, which can then be refined without explicit access to W_{pre}.

6 EXPERIMENTS AND RESULTS

We follow the Model Jungle task settings of MVProbe (Heo et al., 2026) and evaluate two settings: *discriminative*, comprising source-class membership prediction for ResNet (He et al., 2016), SupViT (Dosovitskiy et al., 2021), MAE (He et al., 2022), and DINO (Caron et al., 2021); and *generative*, comprising training-class prediction for Stable Diffusion (Rombach et al., 2022) LoRAs (Hu et al., 2022). We compare DSProbe and R-DSProbe with StatNN (Unterthiner et al., 2020), ProbeX (Horwitz et al., 2025), and MVProbe, matching probe counts, per-branch encoder dimensions, and encoder output dimensions across probing methods. Each method's best-performing layer is selected based on validation accuracy (Figure 1). For SD-LoRA, the probed matrix BA is itself the displacement from the pretrained weight, so DSProbe coincides with MVProbe and we report only R-DSProbe. Dataset specifications, baseline descriptions, and implementation details are provided in Appendix B.

6.1 MAIN RESULTS

Table 2 summarizes the performance of displacement-based probing on Model Jungle. In the discriminative setting, DSProbe already substantially improves over probing the original weights. This supports our central premise that removing the shared pretrained origin exposes variation that is more discriminative of the source task.

Refining the reference further strengthens this effect. By learning W_* and removing task-common variation that remains after pretrained subtraction, R-DSProbe yields the best performance across all four architectures. This holds at the level of individual responses as well: evaluating each response in isolation with a single-branch probe (Appendix G), the pretrained displacement improves 10 of the 12 model–response combinations and the refined displacement improves all 12. The two exceptions are the second-order responses of DINO, which pretrained subtraction alone degrades

Table 2: Main results on Model Jungle. Left: source-class membership accuracy (%) on discriminative models. Right: classification accuracy (%) on Stable Diffusion LoRA models. ProbeX ($\times 4$) matches the parameter budget of the four-branch methods. Mean over five seeds; standard deviations are reported in Tables 6 and 7.

Method	ResNet	SupViT	MAE	DINO
StatNN	65.83	52.43	50.54	50.53
ProbeX ($\times 4$)	89.89	89.17	77.02	69.87
MVProbe	91.09	92.41	81.24	77.88
DSProbe (ours)	96.82	95.93	83.84	77.89
R-DSProbe (ours)	**97.33**	**96.51**	**85.24**	**79.11**

	SD-200		SD-1k	
Method	**ID**	**ZS**	**ID**	**ZS**
ProbeX ($\times 4$)	97.92	93.25	32.22	52.77
MVProbe	99.80	95.25	98.00	97.09
R-DSProbe (ours)	**99.84**	**95.53**	**99.04**	**98.48**

but refinement recovers, explaining the marginal gain of DSProbe over MVProbe on DINO. The consistent improvement from the fixed pretrained reference to the learned reference suggests that the choice of reference itself can be optimized for weight-space prediction.

The same displacement principle remains effective in the generative setting. Here, in-distribution (ID) evaluates test models from classes represented during training, whereas zero-shot evaluates test models from held-out classes not seen during fitting. R-DSProbe outperforms the baselines on both SD-200 and SD-1k, and its strong zero-shot performance in particular indicates that the resulting representation preserves source-dependent structure beyond the classes observed when fitting the probe. Since the model architecture and probing capacity are held fixed, the performance gains reflect the quality of the input representation.

6.2 Layer-wise Comparison

In Figure 1, we compare the layer-wise performance of R-DSProbe against ProbeX and MVProbe. R-DSProbe is not confined to the fixed layer used in the main experiments. Across all four architectures, R-DSProbe generally tracks the same layer-wise performance profile as ProbeX and MVProbe, but shifts the accuracy curve upward over the most informative layers. The improvement is particularly pronounced for ResNet, where R-DSProbe maintains a substantial margin throughout the deeper layers, and remains visible for SupViT, MAE, and DINO despite their markedly different layer-wise profiles.

Importantly, the layer-wise performance is broadly aligned across methods. Layers that are informative for ProbeX or MVProbe also tend to be informative for R-DSProbe, while displacement probing extracts consistently stronger predictive signals from them. This suggests that R-DSProbe improves the accessibility of task-relevant variation within a layer. Prior work shows that ViTs maintain more uniform representations across depth than ResNets (Raghu et al., 2021), whereas in ResNets the target becomes increasingly linearly predictable in later layers (Cha et al., 2026). The resemblance of our layer-wise patterns to these established trends suggests that single-layer probing may offer a useful lens for identifying where task-relevant features occur across depth.

6.3 The Reference Matters: Comparison with Centering

A central choice in DSProbe is to probe the displacement from a reference rather than the raw weight, which raises a natural question: does the gain merely reflect centering the input, or does *what* is subtracted matter? We compare five references under the same four-branch architecture (Table 3): the raw weight (W), a random matrix (W_{rand}) rescaled to the Frobenius norm of W_{pre}, the pretrained origin (W_{pre}), the mean of the leaf weights (W_{mean}, i.e., centering), and the learned reference (W_*). What is subtracted is decisive. Subtracting a norm-matched W_{rand} changes accuracy only marginally and in either direction across architectures, whereas referencing the pretrained origin W_{pre} improves it substantially. This also explains why centering helps: since each leaf is $W_l = W_{\mathrm{pre}} + \Delta_W$, subtracting W_{mean} removes the pretrained component together with the mean displacement, making centering itself a displacement-based representation rather than an alternative to it. The learned reference W_* improves further and attains the highest accuracy on every architecture.

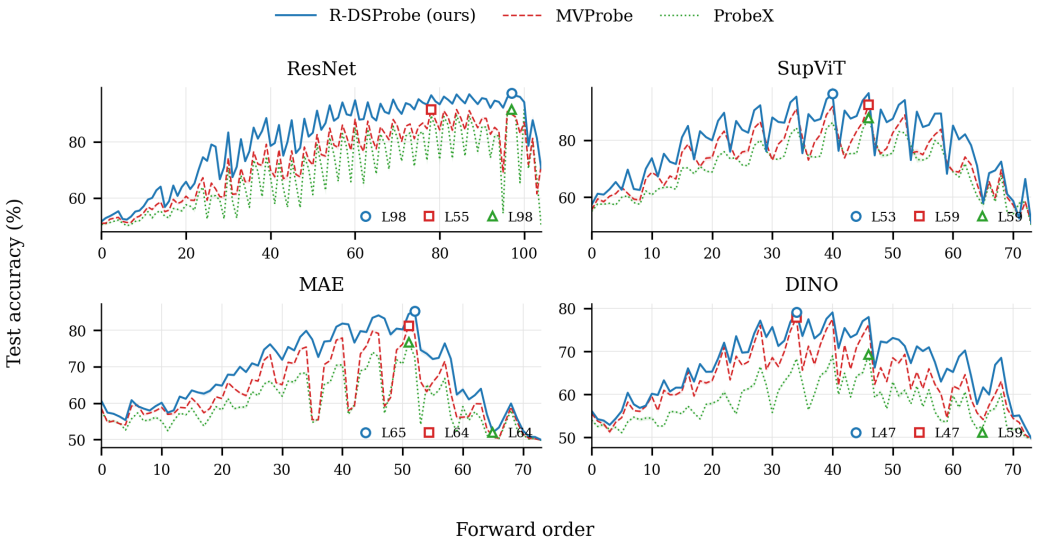


Figure 1: Full-layer test accuracy (%) in forward order for R-DSProbe (blue), MVProbe (red), and ProbeX (green), averaged over five seeds; open markers denote each method's validation-selected layer (indices below each panel).

Table 3: Effect of the reference subtracted before probing, under the four-branch architecture. W_{rand} is rescaled to the Frobenius norm of W_{pre}. Accuracy (%), mean $\pm$ population standard deviation over five seeds.

Model	**MVProbe** (W)	**Random** ($W - W_{\text{rand}}$)	**DSProbe** ($W - W_{\text{pre}}$)	**Centering** ($W - W_{\text{mean}}$)	**R-DSProbe** ($W - W_*$)
ResNet	91.09 ± 0.33	91.19 ± 0.65	96.82 ± 0.14	95.36 ± 0.12	$\mathbf{97.33 \pm 0.05}$
SupViT	92.41 ± 0.40	92.28 ± 0.12	95.93 ± 0.15	95.47 ± 0.22	$\mathbf{96.51 \pm 0.19}$
MAE	81.24 ± 0.20	82.46 ± 0.19	83.84 ± 0.15	84.80 ± 0.21	$\mathbf{85.24 \pm 0.27}$
DINO	77.88 ± 0.14	76.82 ± 0.56	77.89 ± 0.25	77.84 ± 0.30	$\mathbf{79.11 \pm 0.13}$

6.4 Efficiency of R-DSProbe

So far, our experiments fixed the number of probes at $r{=}128$ to isolate the effect of the input representation. We now turn to efficiency. R-DSProbe adds a single shared reference W_* per tree, increasing the number of trained parameters, but this overhead is only about 1% of the metanetwork and barely increases training compute (Appendix E). The more important axis for efficiency is the number of probes r. Probing is, by design, an approach that replaces a dense metanetwork, characterizing a layer from the responses to a few probes and thereby reducing computation (Horwitz et al., 2025). Reaching the same accuracy with fewer probes therefore means that larger weight matrices can be handled more efficiently. From this perspective, we vary the number of probes r and measure how many probes R-DSProbe needs to match MVProbe's accuracy: R-DSProbe reproduces the accuracy that MVProbe attains at its full budget with far fewer probes, using 1.3–$21.1\times$ fewer FLOPs depending on the architecture (Appendix F).

6.5 Cross-Tree Probing with a Shared Encoder

Models in practice span multiple Model Trees. ProbeX (Horwitz et al., 2025) handles this with a mixture-of-tree-experts, training a separate metanetwork per tree, so the parameter count grows linearly with the number of trees. Most of this cost is avoidable: the encoder $g(\Phi)$ holds 95.9% of the parameters (Appendix E), and since the probe and projection map every tree to a common d_p-dimensional representation, a single encoder and head can be shared across trees, routing only the lightweight per-tree components (probe $\mathbf{U}$ and projection V). Applied to raw-weight probing, however, this sharing costs 2.83 points of mean accuracy (Table 4, $85.78 \rightarrow 82.94$): a raw weight

Table 4: Cross-tree probing with a shared encoder, crossing the probe input with the encoder. Every model is trained uncapped. Values are mean ± population standard deviation over five seeds.

Method	ResNet(L55)	SupViT(L59)	MAE(L64)	DINO(L47)	Mean	Params
Per-tree experts (raw)	91.73 ± 0.38	92.53 ± 0.30	81.26 ± 0.27	77.59 ± 0.17	85.78	137.44M
Per-tree experts (R-DSProbe)	$\mathbf{96.51} \pm 0.13$	$\mathbf{96.20} \pm 0.14$	$\mathbf{84.74} \pm 0.12$	$\mathbf{78.76} \pm 0.16$	**89.05**	139.47M
Shared encoder (raw)	89.55 ± 0.11	91.78 ± 0.42	74.41 ± 1.35	76.03 ± 0.29	82.94	36.62M
Shared encoder (R-DSProbe)	$\mathbf{96.14} \pm 0.20$	$\mathbf{94.32} \pm 0.22$	$\mathbf{83.47} \pm 0.37$	$\mathbf{76.52} \pm 0.27$	**87.61**	38.65M

carries each tree's pretrained origin, so inputs from different trees are not aligned to a common frame and the shared encoder suffers negative transfer. R-DSProbe removes this origin at its source by subtracting a per-tree reference W_*, aligning displacements from different trees into a common frame that one shared encoder can process. The same move then costs less than half as much, 1.44 points ($89.05 \rightarrow 87.61$), at $3.61\times$ fewer parameters ($139.47 \rightarrow 38.65$M); per tree, each expert shrinks from 34.87M to 1.26M ($28\times$), and the saving grows with the number of trees.

7 Discussion

Our results point to a simple principle: in weight-space probing, *what you subtract matters as much as what you probe*. Subtracting the shared pretrained origin already turns raw weights into a more readable signal, and learning the reference beyond it further isolates task-specific variation—prior methods read the raw weight, which is just the displacement from a zero reference. The same lens explains second-order views: higher orders separate ever more matrices yet do not improve over the Gram, so expressivity cannot be the reason. What distinguishes the Gram is what training writes into it—to first order, its change along fine-tuning records the features that fine-tuning induced, not a more expressive encoding of the same weight. Probing views should thus be chosen for what training writes into them, and references for what they remove.

Several controls confirm that the reference is what drives the gain. Evaluated in isolation with a single-branch model, the displacement improves nearly every response, so the effect lies in the input representation, not a particular combination of branches (Appendix G). Nor does it come from extra capacity: a free, more expressive response offset fails to recover it (Appendix H). And it is stable across the probed layer (Appendix I), given a reference initialized near the shared weight structure with capped updates (Appendix J).

A one-line change of input thus becomes a different kind of probing method. R-DSProbe leaves the probing architecture untouched, yet sets a new state of the art on every architecture we study while using a fraction of the probes and a single encoder where prior work forces one expert per tree. The choice of reference, not added capacity, is what drives these gains—the strongest evidence for the view this paper puts forward: the signal is not in the weights but in how they have moved.

8 Conclusion

We introduced the *DiSplacement Probe (DSProbe)*, which reads a layer's displacement from a reference rather than the layer itself, and traced the success of within-tree probing to the shared pretrained origin that this displacement removes. Extended to second-order Gram responses, the displacement encodes the learned features induced by fine-tuning, explaining their value where expressivity alone cannot; treating the reference as a design choice, the *Refined-DSProbe* learns the shared component to subtract and achieves state-of-the-art accuracy on all Model Jungle architectures; it does so with fewer probes, robustness to layer selection, and a single encoder shared across trees.

Several limitations remain. Our analysis is first-order in the learning rate and assumes fine-tuning from the shared root; its behavior under long schedules, adaptive optimizers, or multi-stage training is open, and richer references than a single shared matrix are a natural extension. Our evaluation also covers vision backbones of moderate size, whereas displacement probing should matter most at larger scale: language models and their fine-tuned descendants form vast Model Trees with far larger weight matrices, where analyzing each model with less capacity becomes increasingly valuable. We consider this the most promising direction, and hope that treating the reference as a first-class design axis proves useful for weight-space learning more broadly.

AI USE STATEMENT

Large language models (LLMs) were used in the preparation of this paper as follows. *Writing:* LLMs assisted in drafting and polishing sections of the main text and appendix, and in LaTeX formatting; all drafts were revised and verified by the authors. *Retrieval:* LLMs were used to help locate related work, which the authors read and verified independently. *Proofs:* the theorems and their proofs were formulated by the authors; LLMs were used to check the derivations and to typeset them, and the authors verified every step. *Code:* LLMs assisted in cleaning and documenting the experimental code prepared for public release; the code used to produce the reported results was written and run by the authors.

ETHICS STATEMENT

This work studies predicting model attributes from the weights of publicly released checkpoints, using the public Model Jungle benchmark; it involves no human subjects or private data. Because the method infers properties of a model's training data, it could in principle reveal information a model owner did not document; we view this primarily as a tool for improving the transparency of undocumented public models, and note that it recovers only coarse attributes (e.g., class membership) within a known Model Tree.

REPRODUCIBILITY STATEMENT

All experiments use the public Model Jungle benchmark (Horwitz et al., 2025). Code to reproduce our results, including training scripts and configurations for all methods, is available in the anonymous repository linked on the first page. Method details are given in Section 5, the experimental protocol in Section 6, and the full architecture and hyperparameters in Appendix A. Training details and ablations for the learned reference W_* are in Appendix J, the layer enumeration used in all layer-wise analyses in Appendix K, and complete proofs of the theoretical results in Appendix L. All results are reported as mean $\pm$ standard deviation over five seeds.

REFERENCES

Aristide Baratin, Thomas George, César Laurent, R. Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 2269–2277. PMLR, 2021.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9650–9660, 2021.

Taehun Cha, Daniel Beaglehole, Adityanarayanan Radhakrishnan, and Donghun Lee. The weight gram matrix captures sequential feature linearization in deep networks. *arXiv preprint arXiv:2605.06258*, 2026.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016.

Chi-Ning Chou, Hang Le, Yichen Wang, and SueYeon Chung. Feature learning beyond the lazy-rich dichotomy: Insights from representational geometry. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10700–10740. PMLR, 2025.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: Dissecting the weight space of neural networks. In *European Conference on Artificial Intelligence (ECAI)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 1119–1126. IOS Press, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, 2022.

Eunwoo Heo, Kyeongkook Seo, and Jaejun Yoo. What linear probes miss: Multi-view probing for weight-space learning. In *International Conference on Machine Learning (ICML)*, 2026.

Vincent Herrmann, Francesco Faccio, and Jürgen Schmidhuber. Learning useful representations of recurrent neural network weight matrices. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18205–18227. PMLR, 2024.

Eliahu Horwitz, Bar Cavia, Jonathan Kahana, and Yedid Hoshen. Learning on model weights using tree experts. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 20468–20478, 2025.

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456. PMLR, 2015.

Jonathan Kahana, Eliahu Horwitz, Imri Shuval, and Yedid Hoshen. Deep linear probe generators for weight space learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. Graph neural networks for learning equivariant representations of neural networks. In *International Conference on Learning Representations (ICLR)*, 2024.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations (ICLR)*, 2022.

Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. In *International Conference on Learning Representations (ICLR)*, 2024.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25790–25816. PMLR, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.

Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *Advances in Neural Information Processing Systems*, volume 34, pp. 12116–12128, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. In *Advances in Neural Information Processing Systems*, volume 34, pp. 16481–16493, 2021.

Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27906–27920, 2022.

Konstantin Schürholt, Michael W. Mahoney, and Damian Borth. Towards scalable and versatile weight space learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43947–43966. PMLR, 2024.

Edward Tansley, Estelle Massart, and Coralia Cartis. On the neural feature ansatz for deep neural networks. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 5077–5085. PMLR, 2026.

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

David W. Zhang, Miltiadis Kofinas, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, and Cees G. M. Snoek. Neural networks are graphs! Graph neural networks for equivariant processing of neural networks. In *Second Annual Topology, Algebra, and Geometry in Machine Learning Workshop at ICML*, 2023.

Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *Transactions on Machine Learning Research*, 2025.

Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. In *Advances in Neural Information Processing Systems*, volume 36, pp. 24966–24992, 2023a.

Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Neural functional transformers. In *Advances in Neural Information Processing Systems*, volume 36, pp. 77485–77502, 2023b.

A Implementation Details

A.1 Environment

All experiments use four NVIDIA RTX 6000 Ada GPUs with a batch size of 128. Independent model–layer–seed runs are distributed across GPUs. We evaluate all experiments over five random seeds and report the mean performance.

A.2 Architecture Details

To ensure a controlled comparison of the input representations, we use the same probe network architecture as MVProbe (Heo et al., 2026). Let $W_l \in \mathbb{R}^{m_l \times n_l}$ denote a selected layer, with n_l and m_l its input and output dimensions.

The probe network consists of four parallel branches $\{\Phi_i\}_{i=1}^4$, each operating on a different response derived from W_l. Each branch contains its own set of probe vectors $\mathbf{U}_i$ and an encoder that maps the corresponding response to a fixed-dimensional representation. The four branch representations are then concatenated and passed through a shared encoder and prediction head.

Branch configuration Each branch has its own learnable probe matrix and a bias-free projection layer. Each branch response is first standardized per sample, $\tilde{\Phi}_i = \frac{\Phi_i - \mu(\Phi_i)}{\sigma(\Phi_i)+\epsilon}$, where μ and σ are taken over all entries of the response. The projection then maps each of the r probe responses to a d_p-dimensional vector, followed by a ReLU, and the r projected responses are flattened into an rd_p-dimensional branch representation. The branches below are written for the raw weight W_l, as in MVProbe. DSProbe and R-DSProbe keep every shape and component unchanged and only replace each branch input by its displacement counterpart, Δ for DSProbe and $\widetilde{\Delta}$ for R-DSProbe: $W_l \to \Delta_W$ for Φ_1, Φ_2, $W_l^\top W_l \to \Delta_H$ for Φ_3, and $W_l W_l^\top \to \Delta_Z$ for Φ_4. Note that the second-order inputs are differences of Grams, not the Gram of the first-order displacement.

- **Branch** Φ_1
  - Input: $W_l \in \mathbb{R}^{m_l \times n_l}$
  - Probe: $\mathbf{U}_1 \in \mathbb{R}^{n_l \times r}$
  - Response: $W_l \mathbf{U}_1 \in \mathbb{R}^{m_l \times r}$
  - Projection and Flattening: $\mathbb{R}^{m_l \times r} \to \mathbb{R}^{rd_p}$
- **Branch** Φ_2
  - Input: $W_l^\top \in \mathbb{R}^{n_l \times m_l}$
  - Probe: $\mathbf{U}_2 \in \mathbb{R}^{m_l \times r}$
  - Response: $W_l^\top \mathbf{U}_2 \in \mathbb{R}^{n_l \times r}$
  - Projection and Flattening: $\mathbb{R}^{n_l \times r} \to \mathbb{R}^{rd_p}$
- **Branch** Φ_3
  - Input: $W_l^\top W_l \in \mathbb{R}^{n_l \times n_l}$
  - Probe: $\mathbf{U}_3 \in \mathbb{R}^{n_l \times r}$
  - Response: $W_l^\top W_l \mathbf{U}_3 \in \mathbb{R}^{n_l \times r}$
  - Projection and Flattening: $\mathbb{R}^{n_l \times r} \to \mathbb{R}^{rd_p}$
- **Branch** Φ_4
  - Input: $W_l W_l^\top \in \mathbb{R}^{m_l \times m_l}$
  - Probe: $\mathbf{U}_4 \in \mathbb{R}^{m_l \times r}$
  - Response: $W_l W_l^\top \mathbf{U}_4 \in \mathbb{R}^{m_l \times r}$
  - Projection and Flattening: $\mathbb{R}^{m_l \times r} \to \mathbb{R}^{rd_p}$

Linear Classifier The four branch representations are concatenated into a $4rd_p$-dimensional feature vector and mapped to a 512-dimensional representation by a shared linear encoder. A linear classification head then maps this representation to class logits.

A.3 Hyperparameters

The hyperparameters used in our experiments are in Table 5. We use the ProbeX (Horwitz et al., 2025) architecture with $d = 128$ to evaluate individual displacement responses. For the generative Model Jungle setting, we map frozen CLIP text embeddings to the same 512-dimensional space as the probe representation through a learnable linear projection. Cosine similarities between the probe representation and the projected class embeddings serve as classification logits.

Table 5: Hyperparameters used in our experiments.

DSProbe / MVProbe		ProbeX / ProbeX($\times 4$)		W_* Optimization	
Hyperparameter	**Value**	**Hyperparameter**	**Value**	**Hyperparameter**	**Value**
Epochs	500	Epochs	200 / 500	Learning rate	3×10^{-4}
Learning rate	3×10^{-4}	Learning rate	2×10^{-3} / 3×10^{-4}	Weight decay	10^{-5}
Weight decay	10^{-5}	Weight decay	0 / 10^{-5}	Optimizer	Adam
Batch size	128	Batch size	128	Update cap	10^{-3}
Probes per branch (r)	128	Probes per branch (r)	128 / 512		
Projection (d) / representation	$128/512$	Projection (d) / representation	$128/512$		
Branches	4	Branches	1		
Response standardization	True	Response standardization	None		
Optimizer	Adam	Optimizer	Adam		
Loss	BCE	Loss	BCE		

B Experiment Details

B.1 Datasets

Model Jungle (Horwitz et al., 2025) comprises collections of randomly fine-tuned models derived from common foundation models, emulating the Model Tree setting. We consider two evaluation settings:

- **Discriminative.** Each model is fine-tuned on a random subset of 50 CIFAR-100 (Krizhevsky, 2009) classes, and the task is to predict source-class membership. We evaluate four architectures: ResNet-101 (He et al., 2016), SupViT (Dosovitskiy et al., 2021), MAE (He et al., 2022), DINO (Caron et al., 2021).
- **Generative.** Each Stable Diffusion (Rombach et al., 2022) LoRA (Hu et al., 2022) model is fine-tuned on images sampled from a single ImageNet-1k (Deng et al., 2009) class, using base checkpoints from SD-v1.2 through SD-v1.5. We reconstruct the LoRA update as $W_l = BA$ and predict the training class by mapping the model weights to the corresponding CLIP (Radford et al., 2021) text embedding. Because BA is itself the displacement from the pretrained base weight, the pretrained reference is zero in these coordinates and DSProbe coincides with MVProbe; since only BA is probed, the differing base checkpoints also do not enter the input. This setting therefore evaluates only the refinement stage, with W_* initialized at the mean of the training LoRA updates. We split the classes into 170 seen and 30 unseen classes for SD-200, and 850 seen and 150 unseen classes for SD-1k, evaluating ID and zero-shot accuracy within their respective candidate sets.

B.2 Baselines

We compare DSProbe with three representative weight-space learning methods. Our StatNN baseline adapts the summary-statistic representation of Unterthiner et al. (2020) to source-class membership prediction. For each parameter tensor, we compute seven statistics: the mean, population standard deviation, and the 0th, 25th, 50th, 75th, and 100th percentiles. These statistics are concatenated in parameter-name order across all named parameter tensors of the full model, including weights and biases but excluding buffers such as BatchNorm (Ioffe & Szegedy, 2015) running statistics. The resulting features are used to train an XGBoost (Chen & Guestrin, 2016) classifier with 100 binary membership outputs. ProbeX probes individual layer weight matrices directly. MVProbe (Heo et al., 2026) augments single-layer probing with transposed and Gram-based second-order responses.

We exclude graph-based and other equivariant weight-space methods (Zhang et al., 2023; Navon et al., 2023; Kofinas et al., 2024; Lim et al., 2024; Kahana et al., 2025) from the main experiments because they are computationally prohibitive for the large weight matrices considered here. Prior work also reports divergence for such methods in the single-layer probing setting.

B.3 Implementation Details

To minimize architectural confounding, we match the branch capacity: every probing method uses 512 probe vectors in total (ProbeX places all 512 on its single response, the four-branch methods use $r = 128$ per response), a bias-free per-probe projection to $d_p = 128$ features, and a single linear

Table 6: Full version of the discriminative results in Table 2. Source-class membership accuracy (%) on Model Jungle discriminative models. ProbeX ($\times 4$) matches the parameter budget of the four-branch methods. Values are mean $\pm$ population standard deviation over five seeds. Best per column in bold.

Method	ResNet	SupViT	MAE	DINO
StatNN	65.834 ± 0.316	52.430 ± 0.444	50.543 ± 0.210	50.530 ± 0.246
ProbeX ($\times 4$)	89.891 ± 0.343	89.167 ± 0.417	77.023 ± 0.340	69.873 ± 0.094
MVProbe	91.089 ± 0.329	92.413 ± 0.398	81.238 ± 0.195	77.879 ± 0.144
DSProbe (ours)	96.822 ± 0.143	95.930 ± 0.147	83.838 ± 0.154	77.887 ± 0.248
R-DSProbe (ours)	$\mathbf{97.330 \pm 0.051}$	$\mathbf{96.511 \pm 0.187}$	$\mathbf{85.243 \pm 0.265}$	$\mathbf{79.113 \pm 0.132}$

Table 7: Full version of the generative results in Table 2. Classification accuracy (%) on Stable Diffusion LoRA models. ProbeX ($\times 4$) matches the parameter budget of the four-branch methods. Values are mean $\pm$ population standard deviation over five seeds. Best per column in bold.

	SD-200		SD-1k	
Method	**ID**	**Zero-shot**	**ID**	**Zero-shot**
ProbeX ($\times 4$)	97.916 ± 0.654	93.253 ± 1.569	32.224 ± 3.437	52.774 ± 3.884
MVProbe	99.800 ± 0.127	95.250 ± 1.425	97.996 ± 0.283	97.086 ± 0.430
R-DSProbe (ours)	$\mathbf{99.840 \pm 0.150}$	$\mathbf{95.529 \pm 0.753}$	$\mathbf{99.038 \pm 0.234}$	$\mathbf{98.483 \pm 0.240}$

encoder onto a 512-dimensional representation. Unless stated otherwise, all probing methods are trained for 500 epochs with Adam (learning rate 3×10^{-4}, weight decay 10^{-5}) and batch size 128, as summarized in Table 5. StatNN instead uses XGBoost 3.2.0 with CPU histogram-based training and a binary logistic objective for each membership output. We use at most 500 boosting rounds, a learning rate of 0.05, maximum tree depth of 3, a subsampling ratio of 0.9, full feature sampling, and L1 and L2 regularization coefficients of 0 and 1, respectively. Early stopping uses validation binary log loss with a patience of 25 rounds. The selected boosting iteration is then evaluated on the test set. Accuracy is the percentage of correct binary membership predictions across all test models and 100 classes, with probabilities of at least 0.5 classified as positive. Results are averaged over five seeds.

For each probing method and model family we select the probing layer by validation accuracy from the layer-wise sweep of Figure 1, in which the ProbeX baseline follows its original single-branch protocol (200 epochs, 2×10^{-3}, no weight decay); the parameter-matched ProbeX ($\times 4$) reported in Table 2 is evaluated at the layers this sweep selects. The selected layers are 98, 59, 64, and 59 for ProbeX, 55, 59, 64, and 47 for MVProbe, and 98, 53, 65, and 47 for DSProbe and R-DSProbe, for ResNet, SupViT, MAE, and DINO respectively. The ablations that change only the reference (Tables 3, 13, 15 and 17) inherit the DSProbe layers, while the raw-weight row of each is the MVProbe result at MVProbe's own layers, so that no method is evaluated at a layer selected for a different one.

Three comparisons instead hold every condition at one layer set, because the quantity being compared demands it. The branch-composition ablations (Tables 8 and 12) vary the architecture rather than the reference and use 55, 59, 64, and 47 throughout; the cross-tree experiment (Table 4) uses the same set so that per-tree and shared models are counted at identical layers. The probe-count comparison (Figure 2) requires both methods to probe matrices of the same shape; this already holds for SupViT, MAE, and DINO, and for ResNet both methods are evaluated at layer 55, as listed in Table 11.

For SD-LoRA we probe L46 and train with cross-entropy over the candidate classes of each split (170 in-distribution and 30 zero-shot for SD-200, 850 and 150 for SD-1k) rather than the full label set. Class representations are frozen CLIP ViT-L/14 text embeddings of the prompt *a photo of a* $\{$*class*$\}$, mapped into the 512-dimensional representation space by a learned projection; cosine similarities serve as logits. In-distribution and zero-shot checkpoints are selected independently using their corresponding validation candidate sets. The full architecture and the procedure for training W_* are described in Appendix A and Appendix J.

Table 8: Source-class membership prediction accuracy (%) of MVProbe with additional third- and fourth-order branches and capacity-matched controls. All branches are jointly trained with response standardization. Values are mean ± population standard deviation over five seeds. Best per column in bold.

Method	ResNet	SupViT	MAE	DINO
MVProbe	91.089 ± 0.329	92.413 ± 0.398	81.238 ± 0.195	77.879 ± 0.144
Degree-3	$\mathbf{93.657 \pm 0.245}$	92.777 ± 0.240	81.117 ± 0.296	77.183 ± 0.539
Capacity control	92.355 ± 0.558	92.753 ± 0.132	81.488 ± 0.190	78.628 ± 0.205
Degree-3+4	93.457 ± 0.515	92.829 ± 0.165	81.362 ± 0.468	77.043 ± 0.604
Capacity control	92.224 ± 0.410	$\mathbf{93.240 \pm 0.278}$	$\mathbf{82.019 \pm 0.348}$	$\mathbf{78.722 \pm 0.096}$

C Higher-Order Branch Ablation

MVProbe motivates its second-order branches by the additional expressivity.

Theorem 2 (Expressiveness of Second-Order Probes; adapted from MVProbe (Heo et al., 2026, Theorem 4.1))**.** Let $\mathbf{U} \in \mathbb{R}^{n\times r}$ be a probe matrix with $\mathrm{rank}(\mathbf{U}) = r < n$. Define the first- and second-order probe responses as

$$\Phi_1(X;\mathbf{U}) = X\mathbf{U}, \qquad \Phi_2(X;\mathbf{U}) = (X^\top X)\mathbf{U}. \tag{17}$$

Then there exist distinct matrices $X_1, X_2 \in \mathbb{R}^{m\times n}$ such that

$$\Phi_1(X_1;\mathbf{U}) = \Phi_1(X_2;\mathbf{U}), \qquad \text{but} \qquad \Phi_2(X_1;\mathbf{U}) \neq \Phi_2(X_2;\mathbf{U}). \tag{18}$$

Theorem 3 (Expressiveness of Higher-Order Probes)**.** Let $\mathbf{U} \in \mathbb{R}^{n\times r}$ be a probe matrix with $\mathrm{rank}(\mathbf{U}) = r \geq 1$. For each integer $q \geq 0$, define

$$\Phi_{2q}(X;\mathbf{U}) = (X^\top X)^q\mathbf{U}, \qquad \Phi_{2q+1}(X;\mathbf{U}) = X(X^\top X)^q\mathbf{U}. \tag{19}$$

For any integer $k \geq 2$, there exist distinct matrices $X_1, X_2 \in \mathbb{R}^{m\times n}$ such that

$$\Phi_j(X_1;\mathbf{U}) = \Phi_j(X_2;\mathbf{U}), \qquad 1 \leq j < k, \tag{20}$$

but

$$\Phi_k(X_1;\mathbf{U}) \neq \Phi_k(X_2;\mathbf{U}), \tag{21}$$

if and only if

$$n - r \geq \left\lfloor \frac{k}{2} \right\rfloor, \qquad m \geq \left\lceil \frac{k}{2} \right\rceil. \tag{22}$$

Strengthening the sufficient condition in Theorem 1, Theorem 3 gives an exact characterization of when higher-order probe responses can expose weight-space variations that are not observable from lower-order responses alone. We test whether this additional expressivity translates into improved weight-space prediction. We augment MVProbe with third-order responses (*Degree-3*) and with both third- and fourth-order responses (*Degree-3+4*). To separate the effect of higher-order structure from the increased model capacity induced by additional branches, we additionally construct capacity-matched controls with the same number of branches but without introducing higher-order responses.

In Table 8, introducing higher-order branches does not consistently lead to higher weight-space prediction performance. While the Degree-3 variant improves ResNet from 91.089% to 93.657%, the gains are not reproduced across the other model families. On SupViT, MAE, and DINO, the higher-order variants are comparable to or worse than their capacity-matched controls, and adding both third- and fourth-order branches does not provide further improvement. These results suggest that the additional expressivity of higher-order responses does not necessarily translate into more predictive representations for the target model attribute.

D Derivation of the Gram Feature Updates

We derive the two relations in equation 9. Consider the selected layer l at optimization step t,

$$Z_t = W_{l,t} H_t, \tag{23}$$

where H_t and Z_t denote its input and output features, respectively. Since $\mathcal{L}_\tau$ is scalar-valued, the chain rule gives

$$\nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) = \nabla_{Z_t} \mathcal{L}_\tau(\theta_t)\, H_t^\top, \qquad \nabla_{H_t} \mathcal{L}_\tau(\theta_t) = W_{l,t}^\top \nabla_{Z_t} \mathcal{L}_\tau(\theta_t). \tag{24}$$

Consequently,

$$W_{l,t}^\top \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) = \nabla_{H_t} \mathcal{L}_\tau(\theta_t)\, H_t^\top, \tag{25}$$

and, using $Z_t = W_{l,t} H_t$,

$$\nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) W_{l,t}^\top = \nabla_{Z_t} \mathcal{L}_\tau(\theta_t)\, Z_t^\top. \tag{26}$$

We first consider the input-side Gram $W_{l,t}^\top W_{l,t}$. Under the gradient update

$$W_{l,t+1} = W_{l,t} - \eta\, \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t),$$

its one-step change is

$$\begin{aligned} W_{l,t+1}^\top W_{l,t+1} &= \big(W_{l,t} - \eta\, \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)\big)^\top \big(W_{l,t} - \eta\, \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)\big) \\ &= W_{l,t}^\top W_{l,t} - \eta \big[W_{l,t}^\top \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) + \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)^\top W_{l,t}\big] \\ &\quad + \eta^2 \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)^\top \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t). \end{aligned} \tag{27}$$

Therefore,

$$\begin{aligned} W_{l,t+1}^\top W_{l,t+1} - W_{l,t}^\top W_{l,t} &= -2\eta\, \mathrm{Sym}\big(W_{l,t}^\top \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)\big) \\ &\quad + \eta^2 \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)^\top \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) \\ &= -2\eta\, \mathrm{Sym}\big(\nabla_{H_t} \mathcal{L}_\tau(\theta_t)\, H_t^\top\big) + \mathcal{O}(\eta^2), \end{aligned} \tag{28}$$

where $\mathrm{Sym}(A) := (A + A^\top)/2$.

The output-side Gram $W_{l,t} W_{l,t}^\top$ follows analogously:

$$\begin{aligned} W_{l,t+1} W_{l,t+1}^\top &= \big(W_{l,t} - \eta\, \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)\big) \big(W_{l,t} - \eta\, \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)\big)^\top \\ &= W_{l,t} W_{l,t}^\top - \eta \big[W_{l,t} \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)^\top + \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) W_{l,t}^\top\big] \\ &\quad + \eta^2 \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)^\top. \end{aligned} \tag{29}$$

Hence,

$$\begin{aligned} W_{l,t+1} W_{l,t+1}^\top - W_{l,t} W_{l,t}^\top &= -2\eta\, \mathrm{Sym}\big(\nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) W_{l,t}^\top\big) \\ &\quad + \eta^2 \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t) \nabla_{W_{l,t}} \mathcal{L}_\tau(\theta_t)^\top \\ &= -2\eta\, \mathrm{Sym}\big(\nabla_{Z_t} \mathcal{L}_\tau(\theta_t)\, Z_t^\top\big) + \mathcal{O}(\eta^2). \end{aligned} \tag{30}$$

Thus, to first order in the learning rate, changes in the two weight-Gram orientations correspond to covariance-like shifts in the layer input and output feature spaces, respectively. In particular, the first relation is expressed through the task-conditioned input-feature gradient $\nabla_{H_t} \mathcal{L}_\tau$, whereas the second is expressed through the corresponding output-feature gradient $\nabla_{Z_t} \mathcal{L}_\tau$. This yields the two relations stated in equation 9.

E Overhead of the Learned Reference W_*

As introduced in Section 5.3, R-DSProbe learns a shared reference W_*: one $m \times n$ matrix per tree, adding exactly mn parameters. The reference responses are computed once per batch and shared across its examples. Their evaluation adds $12mnr$ forward matrix-multiplication FLOPs per batch,

Table 9: Parameter and training-compute overhead of the learned reference W_* in R-DSProbe. W_* adds one $m \times n$ matrix per tree; percentages are relative to the full metanetwork (r=128). Training FLOPs are estimated as three times the forward pass (one forward, two backward).

Model	Metanetwork	$+W_*$	Total (incr.)	Train FLOPs (incr.)
SupViT / MAE / DINO	34.39M	+0.59M	34.98M (+1.71%)	3222 → 3243M (+0.66%)
ResNet	34.26M	+0.26M	34.52M (+0.77%)	1661 → 1671M (+0.57%)

Table 10: Metanetwork parameter breakdown (SupViT, 768×768, r=128). The encoder scales linearly with the probe count r and dominates the parameter count, whereas the reference W_* is an r-independent constant.

Component	Shape	Params	Share
Probes $\mathbf{U}$ (4 branches)	$4 \times (768, 128)$	0.39M	1.1%
Projections (4)	$4 \times (128, 768)$	0.39M	1.1%
Encoder	$(512, 65536)$	33.55M	95.9%
Classifier head	$(100, 512)$	0.05M	0.1%
Reference W_*	$(768, 768)$	0.59M	1.7%

or $12mnr/B$ per example for batch size B. Thus, the reference parameter count is independent of r, while its computational cost scales linearly with r and is amortized across the batch.

Table 9 accounts for the parameters and training compute that W_* adds. We report FLOPs as twice the number of multiply–accumulates; the values Heo et al. (2026) report as FLOPs are multiply–accumulate counts, so our numbers are twice theirs under the same computation. Relative to the full metanetwork—34.4M parameters for the ViT-based trees and 34.3M for ResNet—W_* adds only 0.59M and 0.26M, about 1% overall (0.77–1.71%), and increases the training-FLOP estimate, computed as three times the forward-pass FLOPs, by less than 0.7% at batch size $B = 128$. The computational overhead is small because the reference responses are shared across the batch.

Table 10 breaks the metanetwork down by component. The encoder dominates, at 95.9% of the parameters, because it scales linearly with the number of probes r ($4 \times d_p \times r \times d_{\text{rep}}$); the probes and projections are a further 2.2%. This distinction becomes relevant under the probe-count sweep of Appendix F: reducing r shrinks the encoder, and hence the whole metanetwork, while W_* stays fixed.

F Probe-Count Efficiency

R-DSProbe reuses the multi-branch probing architecture of MVProbe and changes only the input representation to the displacement. The computational cost of both methods is therefore dominated by the same four probe-response branches, each of which costs $O(mnr)$ (Heo et al., 2026), where r is the number of probes and (m, n) the weight-matrix size. This cost depends only on r and (m, n), not on the identity—depth or type—of the probed layer. For this comparison, we count matrix-multiplication FLOPs for the full probing path—the weight responses, branch projections, encoder, prediction head, and the per-batch reference products. The two methods share every term of this path except the reference products, so at the same r they differ only by the batch-amortized $12mnr/B$ term charged to R-DSProbe (0.6–0.7% at B=128, r=128; Appendix E) whenever the probed layers have the same (m, n), even if the layers themselves differ. We exploit this to construct a sweep that is iso-cost up to this term: each method is evaluated at its own validation-selected layer, and we compare only when the two selected layers share the same size, so that the sweep varies the probing capacity r alone.

Table 11 lists the layers used in the comparison and their weight sizes. For SupViT, MAE, and DINO, the validation-selected layers of the two methods share the same 768×768 size and thus satisfy the iso-cost condition. For ResNet, the two methods' optimal layers differ in size (R-DSProbe: L98, MVProbe: L55), so their FLOPs would not match at a given r; to preserve iso-cost we evaluate both methods at MVProbe's optimal layer (L55, 1024×256), an unfavorable setting for R-DSProbe since it is not evaluated at its own best layer. In every sweep we vary only the number of probes r

Table 11: Layers used in the probe-count efficiency comparison and their weight sizes. For SupViT, MAE, and DINO each method is evaluated at its own validation-selected layer; these layers share the same size (m, n); for ResNet the two optimal layers differ in size, so both methods are evaluated at MVProbe's optimal layer (an unfavorable setting for R-DSProbe).

Model	**MVProbe**	**DSProbe**	**R-DSProbe**	**Size** (m, n)
ResNet	L55	L55	L55	1024×256
SupViT	L59	L53	L53	768×768
MAE	L64	L65	L65	768×768
DINO	L47	L47	L47	768×768

and keep all other training settings identical to the main experiments, computing the metanetwork's FLOPs at each r from the weight size of the probed layer.

The same weight-matrix sizes are used for the parameter and training-compute accounting in Table 9, with $r = 128$. The reported parameter overhead of 0.77–1.71% therefore applies to these fixed-shape configurations. Figure 2 shows the accuracy–r curve for each architecture. R-DSProbe exceeds MVProbe across the entire range of r on all four architectures, and reaches the accuracy that MVProbe attains at its full budget (r=128) with far fewer probes: r=6 (ResNet), 24 (SupViT), 32 (MAE), and 96 (DINO). Because the two methods share the same weight size within each panel, this reduction in probes translates directly into a reduction in compute, shown by the bars beside each panel: $21.13\times$, $5.30\times$, $3.97\times$, and $1.32\times$ fewer FLOPs for ResNet, SupViT, MAE, and DINO, respectively.

Because the encoder scales linearly with r while W_* does not (Appendix E), these probe reductions also shrink the metanetwork itself. Reducing SupViT to r=24 cuts the metanetwork from 34.98M to 7.40M parameters ($4.7\times$), and ResNet to r=6 from 34.52M to 2.23M ($15.5\times$); in each case, the reference contributes a fixed number of parameters independent of r: 0.59M for SupViT and 0.26M for ResNet. The learned reference is therefore a fixed cost that displacement probing more than offsets by reaching the same accuracy at a much smaller probe budget. Although the relative parameter share of W_* grows as r decreases, its parameter count remains unchanged (0.59M / 0.26M). The FLOP reductions in Figure 2 include the per-batch reference computation, and the corresponding parameter overhead is detailed in Table 9. These counts are analytic and require no model bank; the released code reproduces them directly from the layer shapes and probe counts.

G Comparing Displacements and Their Counterparts

To evaluate how informative each response is for weight-space prediction, we evaluate each response independently using the same single-branch architecture. For each response, we compare the raw weight response and its displacements: W_l, Δ_W, $\widetilde{\Delta}_W$ for the first-order response; $W_l W_l^\top$, Δ_Z, $\widetilde{\Delta}_Z$ and $W_l^\top W_l$, Δ_H, $\widetilde{\Delta}_H$ for the second-order responses. All nine variants share the probing capacity and the optimization settings: the probe count, the projection and representation widths, the number of epochs, the optimizer, the learning rate, the weight decay and the batch size. The response-standardization and probe-initialization protocol instead follows the response type, as in the main experiments: the first-order conditions use the original unstandardized ProbeX protocol, while the Gram conditions use response standardization and Xavier probe initialization. The three variants within a response type are therefore identical except for the input representation, which is the comparison this table makes; accuracies are compared within a response type rather than across them.

Table 12 shows that displacement probing generally improves weight-space prediction performance. The pretrained displacement improves 10 of the 12 model–response combinations. Refining the displacement further improves every combination, achieving the best performance in all 12 comparisons.

The benefit of refinement is particularly clear for DINO. While pretrained displacement degrades both second-order responses, from 72.16% to 70.48% for Δ_Z and from 72.23% to 71.77% for Δ_H, the refined responses recover and surpass their raw counterparts, reaching 72.66% and 73.71%, re-

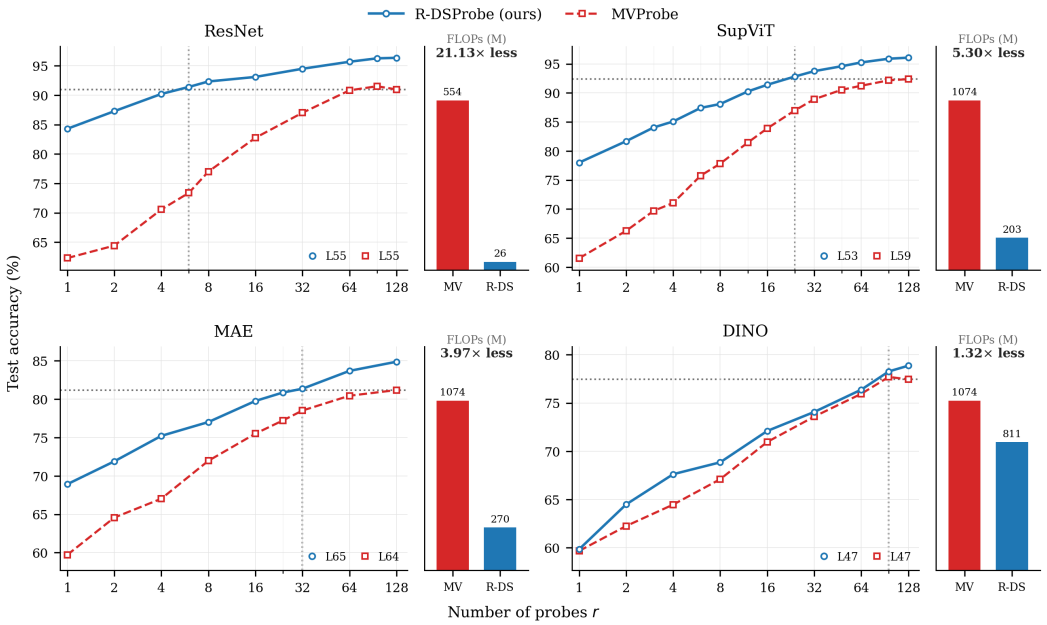


Figure 2: Probe-count efficiency across four architectures. Left: test accuracy (%) vs. number of probes r for R-DSProbe (blue) and MVProbe (red). Right: matrix-multiplication FLOPs of the full probing path, including the per-batch reference products, for MVProbe at r=128 versus R-DSProbe at the smallest r matching its accuracy.

Table 12: Single-branch source-class membership accuracy (%) of raw, DSProbe, and R-DSProbe responses on Model Jungle discriminative models. Values are mean ± population standard deviation over five seeds. Best per model within each response type in bold.

	First-order			Left Gram			Right Gram		
Model	W	Δ_W	$\widetilde{\Delta}_W$	$WW^\top$	Δ_Z	$\widetilde{\Delta}_Z$	$W^\top W$	Δ_H	$\widetilde{\Delta}_H$
ResNet	89.18 ± 0.30	91.10 ± 0.28	$\mathbf{91.78 \pm 0.28}$	90.69 ± 0.54	93.82 ± 0.24	$\mathbf{93.96 \pm 0.31}$	87.47 ± 0.30	94.45 ± 0.22	$\mathbf{94.85 \pm 0.23}$
SupViT	87.78 ± 0.22	91.11 ± 0.36	$\mathbf{92.04 \pm 0.27}$	89.30 ± 0.31	92.35 ± 0.46	$\mathbf{93.10 \pm 0.19}$	90.70 ± 0.13	93.65 ± 0.24	$\mathbf{94.07 \pm 0.14}$
MAE	76.73 ± 0.10	77.35 ± 0.21	$\mathbf{78.42 \pm 0.29}$	77.84 ± 0.12	78.85 ± 0.26	$\mathbf{80.26 \pm 0.30}$	78.26 ± 0.42	79.28 ± 0.24	$\mathbf{80.38 \pm 0.14}$
DINO	68.29 ± 0.26	69.47 ± 0.55	$\mathbf{71.57 \pm 0.31}$	72.16 ± 0.12	70.48 ± 0.32	$\mathbf{72.66 \pm 0.16}$	72.23 ± 0.18	71.77 ± 0.20	$\mathbf{73.71 \pm 0.21}$

spectively. These results are consistent with the relatively modest gain of DSProbe over MVProbe under the fixed pretrained reference, particularly for DINO (Table 2): the improvement comes primarily from the first-order response, while the second-order displacements provide less gain. Once the reference is refined, both second-order responses also surpass their raw counterparts, yielding more consistent accuracy.

H Is the Reference Just a Learnable Response Offset?

Because the probe is linear, subtracting a reference from the weight is equivalent to adding a constant, model-independent offset to the response. For a fixed probe U and a reference W_{ref} shared by every leaf of a tree,

$$(W_l - W_{\text{ref}})\,U = W_l U + B, \qquad B = -\,W_{\text{ref}}\,U, \tag{31}$$

and likewise $(W_l^\top W_l - W_{\text{ref}}^\top W_{\text{ref}})\,U = W_l^\top W_l\,U + B'$ for the second-order branches. DSProbe is therefore MVProbe with a per-branch additive offset applied to the response before standardization and the projection, and R-DSProbe is the case where that offset is produced by a single learned matrix W_* shared across the four branches. This raises a natural concern: if the reference merely acts as a response offset, its benefit might come from the added capacity of that offset rather than from what is subtracted.

This concern is sharpened by the fact that a free offset is both more expressive and cheaper than ours. A per-branch offset $B \in \mathbb{R}^{m\times r}$ learned independently for each of the four branches strictly

Table 13: A free learnable offset on the response versus the reference it is meant to subsume, under the four-branch architecture. B is a per-branch offset added to the response before standardization and projection; R-DSProbe produces the offset from a single shared W_*. All learned variants are uncapped. Accuracy (%), mean over five seeds; best per column in bold.

Probe input	Method	ResNet	SupViT	MAE	DINO
$W_l U$	MVProbe	91.09	92.41	81.24	77.88
$W_l U + B$ (init 0)	Free offset	91.11	92.59	78.49	77.70
$W_l U + B$ (init $-W_{\text{mean}} U$)	Free offset	96.70	94.92	83.28	78.50
$W_l - W_*$	**R-DSProbe (ours)**	**97.32**	**95.95**	**85.03**	**79.01**

contains our parameterization, in which all four offsets are generated by the single matrix W_*; it is also smaller, at 0.39M parameters against 0.59M for W_* on a 768×768 layer. If such a free offset matched R-DSProbe, the contribution would reduce to choosing a good initialization for a bias term. We therefore measure it directly, replacing W_* with a free per-branch offset B added to the response and trained end to end, under two initializations—zero and the centering point $B = -W_{\text{mean}} U$—and compare against R-DSProbe (Table 13).

Two findings follow. First, a free offset initialized at zero adds 0.39M trainable parameters yet recovers none of the gain, showing no consistent improvement over MVProbe across the four trees: capacity in the response is not what the reference supplies. Second, a free offset initialized at the centering point starts exactly where $W - W_{\text{mean}}$ starts, yet the results show no consistent gains over mean centering alone and on SupViT and MAE drifts below it. With both offsets learned freely under the same initialization, the weight-space reference W_* still outperforms the free response-space offset on every architecture (e.g., MAE 85.03 vs. 83.28, SupViT 95.95 vs. 94.92), isolating the parameterization—not the training constraint—as the operative difference.

Expressiveness is therefore not the operative variable; what matters is where the offset lives. Our reference is parameterized in weight space, so the quantity actually subtracted, $-W_* U$, is recomputed from the current probes at every step and remains the response of a single reference as U is learned—the four branches move together because they are generated by one matrix. A free offset instead lives in response space: it is untied from U, and the two drift apart as training proceeds. The coupling that makes our parameterization *less* expressive is a plausible source of its predictive advantage. We accordingly state the claim of Section 7 as: the gains come from subtracting a reference defined in weight space and shared across branches, not from additional capacity in the probing head.

I Robustness to Layer Selection

Probing-based methods predict from the weights of a single layer, so they must choose which layer to probe. Because a network's representations are heterogeneous across depth, accuracy varies substantially from layer to layer, and the optimal layer differs across architectures and tasks and is not known in advance. Finding it requires training and validating a metanetwork at several layers, and a method that is sensitive to this choice can degrade sharply when the wrong layer is selected. A probing method should therefore be robust to layer selection. Heo et al. (2026) likewise compare across informative intermediate layers on this basis, using the layer-index range $\{20, \ldots, 73\}$. That index, however, follows the repository's lexicographic enumeration rather than the forward pass (Appendix K), so we instead reorder the layers into forward order and define the informative range there. In forward order the accuracy forms a unimodal profile that peaks at the central depth and decreases toward both ends (Figure 1); we therefore compare the two methods over the central 70% of the forward order, excluding the low-information ends (the early embedding and shallow blocks, and the final classifier).

Table 14 compares R-DSProbe and MVProbe over the central 70% of the forward order. R-DSProbe outperforms MVProbe at every layer in this range on ResNet, MAE, and DINO (a 100% win rate), and at 82.7% of layers on SupViT, for a mean win rate of 95.7%. Moreover, both the median and the minimum accuracy over this range are higher than MVProbe's on all four architectures. This means

Table 14: Robustness to layer selection over the central 70% of the forward order (Figure 3). Win rate is the fraction of layers where R-DSProbe exceeds MVProbe; median and min report the typical and worst-case accuracy (%) across those layers.

Model	Win rate (%)	Median (%) R-DSProbe / MVProbe	Min (%) R-DSProbe / MVProbe
ResNet	100.0	87.7 / 77.2	58.9 / 55.7
SupViT	82.7	86.3 / 78.6	67.2 / 64.1
MAE	100.0	73.9 / 65.3	57.4 / 55.4
DINO	100.0	71.2 / 65.9	59.9 / 54.6
Average	95.7	79.8 / 71.7	60.9 / 57.4

that R-DSProbe retains higher performance even when the optimal layer is not selected exactly, indicating robustness to layer selection.

J Effect of Reference Initialization and Training Details

We further examine how the initialization and training configuration of the learned reference W_* affect R-DSProbe. In Table 15, we first compare initialization from the mean training weight W_{mean}, the pretrained weight W_{pre}, a random matrix W_{rand}, and a random matrix with matched norm to W_{pre}. The initialization of the reference has a substantial effect on downstream accuracy. Initializing from either W_{mean} or W_{pre} consistently outperforms random initialization, indicating that a reference close to the shared weight structure provides a substantially better starting point for refinement. In particular, each leaf admits the decomposition $W_l = W_{pre} + C + \widetilde{\Delta}_W$ following equation 13. Taking the expectation over leaves gives

$$\mathbb{E}_l[W_l] = W_{mean} = W_{pre} + C + \mathbb{E}_l[\widetilde{\Delta}W]. \tag{32}$$

Hence, W_{mean} decomposes into the pretrained origin W_{pre}, the additional shared component C, and the mean refined displacement $\mathbb{E}[\widetilde{\Delta}_W]$, making it a natural alternative initialization for W_*. Empirically, W_{mean} initialization outperforms W_{pre} on SupViT, MAE, and DINO, so we use it by default. Importantly, Table 3 shows that W_{mean} is not consistently a better fixed reference than W_{pre}, while it provides a better initialization for learning W_*.

One possible explanation is that centering by W_{mean} removes the mean displacement component $\mathbb{E}_l[\widetilde{\Delta}W]$ in addition to the shared correction C, which may still contain predictive information. Thus, W_{mean} can provide a favorable starting point for learning the shared reference, while being overly aggressive when used directly as a fixed reference. Consistent with this interpretation, Table 16 shows that the learned reference moves closer to W_{pre} only for ResNet, the only architecture where W_{pre} is also a better initialization than W_{mean}. Here, $d_{pre}(W')$ denotes relative Frobenius distance from the pretrained weight of an arbitrary weight matrix W':

$$d_{pre}(W') = \frac{\|W' - W_{pre}\|_F}{\|W_{pre}\|_F} \times 100 \tag{33}$$

In contrast, it moves farther away for SupViT, MAE, and DINO, suggesting that learning W_* adapts the reference toward a tree-specific shared component.

Having established W_{mean} as a suitable initialization, we next examine the optimization of the reference itself. We observed that W_* can undergo overly large updates during training, potentially destabilizing the joint optimization of the reference and the probing network. To stabilize this refinement, we cap the Frobenius norm of each post-Adam reference update at $\rho\|W_{\text{mean}}\|_F$. This limits individual update steps without constraining the cumulative distance from the initialization. As shown in Table 15, this simple constraint improves performance across all four architectures. We further sweep the cap value in Table 17. A cap of 10^{-3} provides the most consistent performance across architectures, achieving the best accuracy on ResNet, SupViT, MAE, and DINO. We therefore use 10^{-3} as the default cap throughout our experiments.

Table 15: Ablation on reference initialization and training configuration of R-DSProbe. Values are reported as mean ± population standard deviation over five seeds. Best results are shown in bold and second-best results are underlined.

Model	MVProbe	Reference initialization of **R-DSProbe**				
		W_{mean}	W_{pre}	W_{rand}	W_{rand} **+ Norm-matched**	W_{mean} **+ Cap (ours)**
ResNet	91.089 ± 0.329	97.315 ± 0.054	$\mathbf{97.402 \pm 0.183}$	89.326 ± 0.741	92.771 ± 1.914	$\underline{97.330 \pm 0.051}$
SupViT	92.413 ± 0.398	$\underline{95.952 \pm 0.072}$	95.917 ± 0.177	92.094 ± 0.573	91.632 ± 0.149	$\mathbf{96.511 \pm 0.187}$
MAE	81.238 ± 0.195	85.028 ± 0.267	84.713 ± 0.251	82.799 ± 0.350	83.002 ± 0.314	$\mathbf{85.243 \pm 0.265}$
DINO	77.879 ± 0.144	$\underline{79.011 \pm 0.227}$	78.910 ± 0.224	75.169 ± 0.244	76.959 ± 0.312	$\mathbf{79.113 \pm 0.132}$

Table 16: Relative Frobenius distance of the initial W_{mean} and the learned reference W_* from the pretrained weight W_{pre}. Values for W_* are reported as mean ± population standard deviation over five seeds.

Model	$d_{pre}(W_{mean})$	$d_{\text{pre}}(W_*)$	**Change in** d_{pre}**.**
ResNet	1.591%	$1.489 \pm 0.081\%$	$\mathbf{-0.102}\%$
SupViT	1.813%	$3.750 \pm 0.241\%$	$\mathbf{+1.937}\%$
MAE	2.069%	$2.416 \pm 0.205\%$	$\mathbf{+0.347}\%$
DINO	3.971%	$4.363 \pm 0.034\%$	$\mathbf{+0.392}\%$

K Layer Ordering

We treat each model as a sequence of layer weight matrices, and the order in which these matrices are enumerated defines the horizontal axis of our layer-wise analysis. The Model Jungle benchmark (Horwitz et al., 2025) enumerates the weight matrices of each model in lexicographic order of their parameter names, and prior probing work (Horwitz et al., 2025; Heo et al., 2026) together with the main text refers to layers by this index (e.g., "layer 59"). This enumeration, however, does not follow the network's forward pass: because multi-digit block indices are sorted lexicographically, the enumeration proceeds through blocks $0, 1, 10, 11, 2, \ldots, 9$, so that block 10 immediately follows block 1. The classifier head and the position embedding are placed at indices 0 and 1, putting the layers that come last and first in the forward pass at the front of the sequence.

Each index still identifies a specific weight matrix; "layer 53," for instance, denotes the attention-output weight of block 6 and is used with this meaning throughout. What the index does not reflect is the network's depth. For every layer-wise analysis (Figure 1, Appendix I) we therefore keep the matrix that each index denotes but reorder the matrices by their forward-pass position, recovered from the parameter names (block/stage number and within-block position), and apply the same forward order to all three methods. Figure 3 illustrates the effect on SupViT: the same weight matrix (L53) sits at position 53 under the repository index but at position 40 in forward order, and reordering turns the jagged repository-index profile into a coherent depth trend.

L Proof of Higher-Order Separation Theorem

We provide the proof of Theorem 3. Recall the alternating probing hierarchy

$$\Phi_{2q}(X;U) = (X^\top X)^q U, \qquad \Phi_{2q+1}(X;U) = X(X^\top X)^q U, \qquad q \geq 0.$$

The proof proceeds in two steps. We first establish sufficiency by explicitly constructing two matrices whose probing responses coincide up to order $k-1$ and separate at order k. We then prove necessity using the growth of the corresponding block-Krylov reachable subspaces.

L.1 Preliminaries

We begin by converting the alternating multiplication by X and $X^\top$ into powers of a single symmetric operator.

Lemma 1 (Symmetric lifting)**.** For $X \in \mathbb{R}^{m\times n}$, define

$$B_X := \begin{bmatrix} 0 & X \\ X^\top & 0 \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}, \qquad \bar{U} := \begin{bmatrix} 0 \\ U \end{bmatrix}.$$

Table 17: Ablation on the reference update cap. Values are reported as mean ± population standard deviation over five seeds. Best results are shown in bold.

Cap	ResNet	SupViT	MAE	DINO
Uncap (0)	97.315 ± 0.054	95.952 ± 0.072	85.028 ± 0.267	79.011 ± 0.227
1×10^{-4}	97.294 ± 0.058	96.076 ± 0.159	85.240 ± 0.216	78.725 ± 0.292
1×10^{-3}	$\mathbf{97.330 \pm 0.051}$	$\mathbf{96.511 \pm 0.187}$	$\mathbf{85.243 \pm 0.265}$	$\mathbf{79.113 \pm 0.132}$
1×10^{-2}	97.315 ± 0.054	95.952 ± 0.072	85.028 ± 0.267	79.074 ± 0.192
1×10^{-1}	97.315 ± 0.054	95.952 ± 0.072	85.028 ± 0.267	79.011 ± 0.227

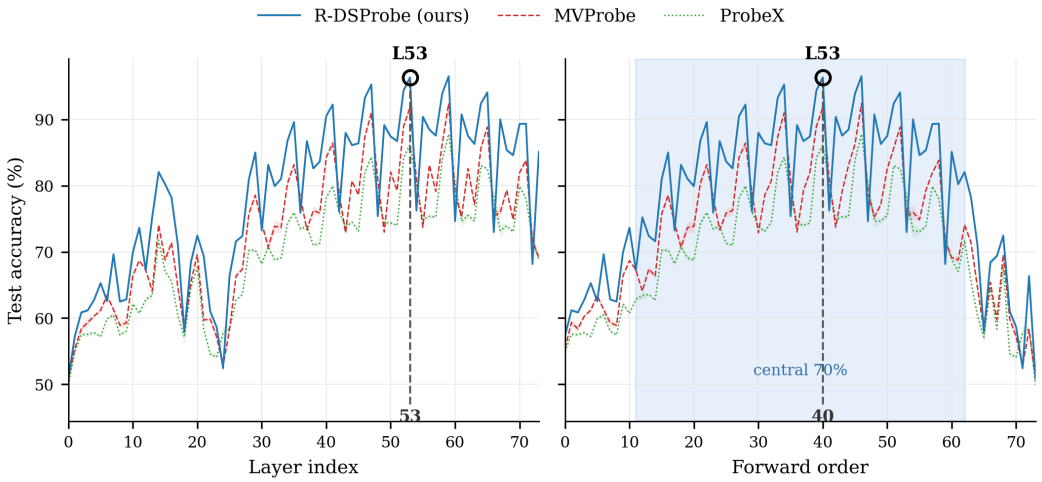


Figure 3: Effect of layer ordering on SupViT. Test accuracy (%) of the three methods plotted against the repository enumeration index (left) and the recovered forward-pass position (right). The same weight matrix (L53) moves from index 53 to forward position 40.

Then, for every $q \geq 0$,

$$B_X^{2q}\bar{U} = \begin{bmatrix} 0 \\ \Phi_{2q}(X;U) \end{bmatrix}, \qquad B_X^{2q+1}\bar{U} = \begin{bmatrix} \Phi_{2q+1}(X;U) \\ 0 \end{bmatrix}.$$

Proof. Direct multiplication gives

$$B_X^2 = \begin{bmatrix} XX^\top & 0 \\ 0 & X^\top X \end{bmatrix}.$$

Hence

$$B_X^{2q} = \begin{bmatrix} (XX^\top)^q & 0 \\ 0 & (X^\top X)^q \end{bmatrix},$$

which yields

$$B_X^{2q}\bar{U} = \begin{bmatrix} 0 \\ (X^\top X)^q U \end{bmatrix}.$$

Multiplying once more by B_X gives

$$B_X^{2q+1}\bar{U} = \begin{bmatrix} X(X^\top X)^q U \\ 0 \end{bmatrix}.$$

□

We next record a perturbation identity that will allow us to characterize the first order at which two probing sequences differ.

Lemma 2 (Perturbation recurrence)**.** Let

$$B_1 := B_{X_1}, \qquad B_2 := B_{X_2}, \qquad E := B_2 - B_1,$$

and define

$$D_j := B_2^j \bar{U} - B_1^j \bar{U}.$$

Then

$$D_{j+1} = B_2 D_j + E B_1^j \bar{U}.$$

Consequently, if

$$D_j = 0, \qquad 0 \le j < k,$$

then

$$E B_1^j \bar{U} = 0, \qquad 0 \le j \le k-2,$$

and

$$D_k = E B_1^{k-1} \bar{U}.$$

Proof. Using $B_2 = B_1 + E$,

$$D_{j+1} = B_2^{j+1} \bar{U} - B_1^{j+1} \bar{U} \tag{34}$$

$$= B_2 \left(B_2^j \bar{U} - B_1^j \bar{U} \right) + (B_2 - B_1) B_1^j \bar{U} \tag{35}$$

$$= B_2 D_j + E B_1^j \bar{U}. \tag{36}$$

If $D_j = D_{j+1} = 0$, the recurrence implies $E B_1^j \bar{U} = 0$. Applying this for $j = 0, \dots, k-2$ gives the first claim, while setting $D_{k-1} = 0$ gives

$$D_k = E B_1^{k-1} \bar{U}.$$

□

L.2 SUFFICIENCY

Suppose

$$n - r \ge \left\lfloor \frac{k}{2} \right\rfloor, \qquad m \ge \left\lceil \frac{k}{2} \right\rceil.$$

Let

$$s := \left\lfloor \frac{k}{2} \right\rfloor, \qquad \ell := \left\lceil \frac{k}{2} \right\rceil.$$

Step 1: Construct an alternating orthonormal chain. Choose a unit vector

$$p_0 \in \mathrm{range}(U).$$

Then

$$p_0^\top U \neq 0.$$

Since

$$\dim \ker(U^\top) = n - r \ge s,$$

there exist orthonormal vectors

$$p_1, \dots, p_s \in \ker(U^\top),$$

so that

$$p_i^\top U = 0, \qquad i = 1, \dots, s.$$

Similarly, since $m \ge \ell$, choose orthonormal vectors

$$q_1, \dots, q_\ell \in \mathbb{R}^m.$$

Lift these vectors to $\mathbb{R}^{m+n}$ by defining

$$z_{2i} := \begin{bmatrix} 0 \\ p_i \end{bmatrix}, \qquad z_{2i-1} := \begin{bmatrix} q_i \\ 0 \end{bmatrix}.$$

Then

$$z_0, z_1, \dots, z_k$$

form an orthonormal sequence alternating between the input and output spaces.

Step 2: Construct X_1 and X_2. For $j = 1, \dots, k$, define

$$M_j := \begin{cases} q_{i+1} p_i^\top, & j = 2i+1, \\ q_i p_i^\top, & j = 2i. \end{cases}$$

Set

$$X_1 := \sum_{j=1}^{k-1} M_j, \qquad X_2 := \sum_{j=1}^{k} M_j.$$

Hence

$$X_2 - X_1 = M_k,$$

which is rank one, and therefore $X_1 \neq X_2$.

Step 2 (continued): Lifted operators. Recall the lift $A \mapsto B_A := \begin{bmatrix} 0 & A \\ A^\top & 0 \end{bmatrix}$ of Lemma 1, which is linear in A. Hence

$$B_{X_1} = B_{\sum_{j=1}^{k-1} M_j} = \begin{bmatrix} 0 & \sum_{j=1}^{k-1} M_j \\ \sum_{j=1}^{k-1} M_j^\top & 0 \end{bmatrix} = \sum_{j=1}^{k-1} B_{M_j}, \qquad B_{X_2} = \sum_{j=1}^{k} B_{M_j}.$$

For each j, the term M_j couples the consecutive chain vectors z_{j-1} and z_j, which lie in complementary blocks; writing $z_{j-1} = \begin{bmatrix} q \\ 0 \end{bmatrix}$ and $z_j = \begin{bmatrix} 0 \\ p \end{bmatrix}$ with $M_j = qp^\top$,

$$z_{j-1} z_j^\top = \begin{bmatrix} 0 & qp^\top \\ 0 & 0 \end{bmatrix}, \qquad z_j z_{j-1}^\top = \begin{bmatrix} 0 & 0 \\ pq^\top & 0 \end{bmatrix},$$

so that

$$B_{M_j} = \begin{bmatrix} 0 & M_j \\ M_j^\top & 0 \end{bmatrix} = z_j z_{j-1}^\top + z_{j-1} z_j^\top.$$

Summing over j,

$$B_1 := B_{X_1} = \sum_{j=1}^{k-1} \big(z_j z_{j-1}^\top + z_{j-1} z_j^\top \big), \qquad B_2 := B_{X_2} = B_1 + z_k z_{k-1}^\top + z_{k-1} z_k^\top,$$

and therefore the perturbation is the single terminal coupling

$$E := B_2 - B_1 = z_k z_{k-1}^\top + z_{k-1} z_k^\top,$$

which acts only on $\operatorname{span}\{z_{k-1}, z_k\}$.

Step 3: Track the reachable directions. By construction,

$$B_1 z_0 = z_1,$$

and, for $1 \leq j \leq k-2$,

$$B_1 z_j = z_{j-1} + z_{j+1}.$$

It follows inductively that

$$B_1^t z_0 = z_t + v_t, \qquad v_t \in \operatorname{span}\{z_0, \dots, z_{t-1}\}, \qquad 0 \leq t \leq k-1.$$

In particular, the newly reached direction z_t appears with nonzero coefficient at every step.

Let

$$c := p_0^\top U \in \mathbb{R}^{1 \times r}, \qquad c \neq 0.$$

Since $p_i^\top U = 0$ for $i \geq 1$, we may write

$$\bar{U} = z_0 c + \bar{U}_\perp,$$

where every column of $\bar{U}_\perp$ is orthogonal to

$$\operatorname{span}\{z_0, \dots, z_k\}.$$

Since both B_1 and E act only on this span,

$$B_1 \bar{U}_\perp = 0, \qquad E \bar{U}_\perp = 0.$$

Step 4: All lower-order responses coincide. For $0 \le t \le k-2$,

$$B_1^t \bar{U} \in \operatorname{span}\{z_0, \dots, z_{k-2}\}.$$

On the other hand, E acts only through the final edge between z_{k-1} and z_k. Therefore,

$$EB_1^t \bar{U} = 0, \qquad 0 \le t \le k-2.$$

Lemma 2 then gives

$$B_2^j \bar{U} = B_1^j \bar{U}, \qquad 0 \le j < k.$$

By Lemma 1,

$$\Phi_j(X_1; U) = \Phi_j(X_2; U), \qquad 1 \le j < k.$$

Step 5: The k-th response separates. At $t = k-1$,

$$B_1^{k-1} \bar{U} = z_{k-1} c + v, \qquad v \in \operatorname{span}\{z_0, \dots, z_{k-2}\}.$$

Therefore,

$$EB_1^{k-1} \bar{U} = z_k c \neq 0.$$

By Lemma 2,

$$B_2^k \bar{U} - B_1^k \bar{U} = z_k c \neq 0.$$

Applying Lemma 1 yields

$$\Phi_k(X_1; U) \neq \Phi_k(X_2; U).$$

This proves sufficiency.

L.3 Necessity

Suppose there exist distinct

$$X_1, X_2 \in \mathbb{R}^{m \times n}$$

such that

$$\Phi_j(X_1; U) = \Phi_j(X_2; U), \qquad 1 \le j < k,$$

while

$$\Phi_k(X_1; U) \neq \Phi_k(X_2; U).$$

Let

$$X := X_1, \qquad \Delta := X_2 - X_1, \qquad G := X^\top X.$$

Since the k-th responses differ,

$$\Delta \neq 0.$$

Define the right reachable subspaces

$$\mathcal{R}_t := \operatorname{col}\left[U,\, GU,\, \dots,\, G^t U\right] \subseteq \mathbb{R}^n$$

and the left reachable subspaces

$$\mathcal{L}_t := \operatorname{col}\left[XU,\, XGU,\, \dots,\, XG^t U\right] \subseteq \mathbb{R}^m,$$

with

$$\mathcal{L}_{-1} := \{0\}.$$

By construction,

$$\mathcal{L}_t = X\mathcal{R}_t.$$

Lemma 3 (No restart after stagnation)**.** If

$$\mathcal{R}_t = \mathcal{R}_{t-1}$$

for some $t \ge 1$, then

$$\mathcal{R}_{t+j} = \mathcal{R}_{t-1}, \qquad \forall j \ge 0.$$

The analogous statement holds for $\mathcal{L}_t$.

Proof. If

$$\mathcal{R}_t = \mathcal{R}_{t-1},$$

then

$$G^t U \in \mathcal{R}_{t-1},$$

and hence

$$G\mathcal{R}_{t-1} \subseteq \mathcal{R}_{t-1}.$$

Thus $\mathcal{R}_{t-1}$ is G-invariant, implying that all subsequent powers $G^{t+j}U$ remain in the same subspace. The argument for $\mathcal{L}_t$ is identical, replacing $G = X^\top X$ by $XX^\top$. □

Since

$$E = \begin{bmatrix} 0 & \Delta \\ \Delta^\top & 0 \end{bmatrix},$$

Lemmas 1 and 2 translate equality of the lower-order probing responses into explicit annihilation conditions. We consider even and odd k separately.

Case 1: $k = 2q$**.** Equality at all orders below $2q$ implies

$$\Delta G^t U = 0, \qquad 0 \le t \le q-1, \tag{A.1}$$

and

$$\Delta^\top X G^t U = 0, \qquad 0 \le t \le q-2. \tag{A.2}$$

The first differing response satisfies

$$\Delta^\top X G^{q-1} U \neq 0. \tag{A.3}$$

Equation A.3, together with equation A.2, implies

$$\mathcal{L}_{q-1} \supsetneq \mathcal{L}_{q-2}.$$

By Lemma 3, the sequence cannot have stagnated at any earlier stage. Hence

$$\mathcal{L}_{-1} \subsetneq \mathcal{L}_0 \subsetneq \cdots \subsetneq \mathcal{L}_{q-1},$$

and therefore

$$\dim(\mathcal{L}_{q-1}) \ge q.$$

Since

$$\mathcal{L}_{q-1} \subseteq \mathbb{R}^m,$$

we obtain

$$m \ge q.$$

Moreover, strict growth of $\mathcal{L}_t = X\mathcal{R}_t$ forces strict growth of the corresponding $\mathcal{R}_t$. Since

$$\dim(\mathcal{R}_0) = \operatorname{rank}(U) = r,$$

we have

$$\dim(\mathcal{R}_{q-1}) \ge r + q - 1.$$

From equation A.1,

$$\Delta\mathcal{R}_{q-1} = \{0\},$$

so that

$$\mathcal{R}_{q-1} \subseteq \ker(\Delta).$$

Because $\Delta \neq 0$,

$$\dim\ker(\Delta) \le n - 1.$$

Consequently,

$$r + q - 1 \le n - 1,$$

which gives

$$n - r \ge q.$$

Thus, when $k = 2q$,

$$n - r \ge \frac{k}{2}, \qquad m \ge \frac{k}{2}.$$

Case 2: $k = 2q + 1$**.** Equality at all lower orders implies

$$\Delta G^t U = 0, \qquad 0 \le t \le q - 1, \tag{A.4}$$

and

$$\Delta^\top X G^t U = 0, \qquad 0 \le t \le q - 1. \tag{A.5}$$

The first differing response satisfies

$$\Delta G^q U \neq 0. \tag{A.6}$$

Equation A.4 and equation A.6 imply

$$\mathcal{R}_q \supsetneq \mathcal{R}_{q-1}.$$

By Lemma 3,

$$\mathcal{R}_0 \subsetneq \mathcal{R}_1 \subsetneq \cdots \subsetneq \mathcal{R}_q.$$

Hence

$$\dim(\mathcal{R}_q) \ge r + q.$$

Since

$$\mathcal{R}_q \subseteq \mathbb{R}^n,$$

we obtain

$$n - r \ge q.$$

We next show that

$$\mathcal{L}_{q-1} \supsetneq \mathcal{L}_{q-2}.$$

Suppose, to the contrary, that

$$\mathcal{L}_{q-1} = \mathcal{L}_{q-2}.$$

Then

$$X G^{q-1} U \in \mathcal{L}_{q-2},$$

and therefore

$$G^q U = X^\top (X G^{q-1} U) \in X^\top \mathcal{L}_{q-2} \subseteq \mathcal{R}_{q-1}.$$

Equation A.4 would then imply

$$\Delta G^q U = 0,$$

contradicting equation A.6. Thus

$$\mathcal{L}_{q-1} \supsetneq \mathcal{L}_{q-2}.$$

Lemma 3 therefore gives

$$\dim(\mathcal{L}_{q-1}) \ge q.$$

Moreover, equation A.5 implies

$$\Delta^\top \mathcal{L}_{q-1} = \{0\},$$

and hence

$$\mathcal{L}_{q-1} \subseteq \ker(\Delta^\top).$$

Since $\Delta \neq 0$,

$$\dim \ker(\Delta^\top) \le m - 1.$$

Therefore

$$q \le m - 1,$$

or equivalently,

$$m \ge q + 1.$$

Thus, when $k = 2q + 1$,

$$n - r \ge q, \qquad m \ge q + 1.$$

Combining the even and odd cases yields

$$n - r \ge \left\lfloor \frac{k}{2} \right\rfloor, \qquad m \ge \left\lceil \frac{k}{2} \right\rceil.$$

This proves necessity and completes the proof of Theorem 3. □

L.4 Maximum First-Separation Order

Theorem 3 immediately implies the maximum order at which a pair of matrices can become distinguishable for the first time.

Corollary 1 (Maximum first-separation order)**.** Let

$$U \in \mathbb{R}^{n\times r}, \qquad \operatorname{rank}(U) = r < n.$$

The maximum feasible first-separation order is

$$k_{\max} = \min\left\{2m,\, 2(n-r)+1\right\}.$$

Equivalently, for every integer

$$2 \le k \le \min\{2m,\, 2(n-r)+1\},$$

there exist $X_1 \neq X_2 \in \mathbb{R}^{m\times n}$ such that

$$\Phi_j(X_1;U) = \Phi_j(X_2;U), \qquad \forall j < k,$$

while

$$\Phi_k(X_1;U) \neq \Phi_k(X_2;U).$$

Proof. By Theorem 3, order k is feasible if and only if

$$\left\lfloor \frac{k}{2} \right\rfloor \le n-r, \qquad \left\lceil \frac{k}{2} \right\rceil \le m.$$

The largest integer satisfying both inequalities is

$$k_{\max} = \min\left\{2m,\, 2(n-r)+1\right\}.$$

□

L.5 Special Cases

For $k = 2$, the dimension condition reduces to

$$n - r \ge 1, \qquad m \ge 1.$$

In this case, the construction admits

$$X_1 = q_1 p_0^\top, \qquad X_2 = q_1 (p_0 + p_1)^\top,$$

where

$$p_0^\top U \neq 0, \qquad p_1^\top U = 0.$$

Hence

$$X_1 U = X_2 U,$$

while

$$X_1^\top X_1 U \neq X_2^\top X_2 U.$$

Thus, the second-order response resolves a collision that is invisible to the first-order response.

For $k = 3$, the required condition is

$$n - r \ge 1, \qquad m \ge 2.$$

Accordingly, there exist $X_1 \neq X_2$ satisfying

$$X_1 U = X_2 U,$$

and

$$X_1^\top X_1 U = X_2^\top X_2 U,$$

but

$$X_1 X_1^\top X_1 U \neq X_2 X_2^\top X_2 U.$$

Therefore, the third-order response can strictly refine the distinguishability induced jointly by the first- and second-order responses.