FUNCTION COORDINATES ARE PREDICTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Measuring a model's predictions over data offers a lens into understanding the functions it learns. Recent work places models in a shared coordinate system based on their predictions on a fixed dataset, providing a lens into the functions they learn. Yet, existing systems fall short in two ways. Some position models by their similarity to a specified set of reference models, describing each new model only through the information captured by the set, which can misorder its relationships to other models. And all existing systems are temperature-sensitive: two models with identical decisions but different temperatures can lie farther apart than two models that disagree with the same temperature. We propose a simple coordinate system that addresses both shortcomings by construction: models are placed by their own predictions alone, with temperature removed. Across 1,000+ full training runs spanning 15.6K checkpoints probed on ImageNet-1K, we ask what a checkpoint's position reveals about its training beyond how well it performs: dataset and patch size are linearly decodable at any training stage, model size only weakly, and optimizer settings only through accuracy and loss. Moreover, trajectories are predictive: a non-parametric estimator recovers a run's final and per-class accuracies to within a few points from its first 10-25% of training, while its per-input class predictions and confusion matrix become predictable in the second half of training.

1 INTRODUCTION

When we say a neural network has *learned a function*, we make a claim about its predictions over data (Mitchell, 1997; Vapnik, 1995). Understanding what a network has learned is both a foundational scientific question (Zhang et al., 2017) and a practical one: it informs how we train, select, merge, and deploy models (Hinton et al., 2015; Wortsman et al., 2022; Stoica et al., 2024). Most efforts have approached network understanding through the lens of (1) model parameters (Garipov et al., 2018; Frankle et al., 2020), (2) the training dynamics that produce them (Jacot et al., 2018; Saxe et al., 2014; Nanda et al., 2023), or (3) the relationships between their internal representations (Kornblith et al., 2019; Raghu et al., 2017). Yet none of these directly captures or reasons over the predictions themselves, limiting what they can say about how and what a model learns (Klabunde et al., 2025).

Recently, a more direct approach has emerged: reasoning over the predictions themselves. Many such methods perform this in classification settings, as is the focus of our work. Given a fixed probe set, these methods record a model's outputs as a *function fingerprint*, which determines its position in a shared coordinate space with other models (Mao et al., 2024; Quinn et al., 2019; Hoogland et al., 2025; Oyama et al., 2025). Such spaces provide a direct way to ask which models behave similarly, how behavior changes during training, and which training choices shape the learned function. Such coordinate spaces are broadly categorized into two groups: *relative-coordinates* and *absolute-coordinates*.

However, existing coordinate systems have two important limitations. *Relative-coordinate* methods compress each model through its similarity to a fixed set of anchors, which can discard behavioral differences not

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.

represented by those anchors (Mao et al., 2024; Quinn et al., 2019; Ramesh et al., 2023; Teoh et al., 2020; Helm et al., 2024; 2025). *Absolute-coordinate* methods preserve the predictions themselves, but, like relative methods, are sensitive to *temperature*: uniformly scaling a model's logits changes its position even though none of its decisions change (Hoogland et al., 2025; Carroll et al., 2025; Fort et al., 2020; Oyama et al., 2025; Quinn et al., 2019). As a result, existing spaces can confound differences in predictive behavior with differences in confidence.

We introduce an *absolute, temperature-invariant* coordinate system for model behavior. For each prediction, we first center log-probabilities across classes using the centered log-ratio transform (Aitchison, 1982), making temperature act as a uniform scaling of the coordinate. We then unit-normalize to remove this scaling and concatenate predictions across the probe set. The resulting fingerprint preserves the information of an absolute coordinate system while remaining invariant to temperature.

We use this space to study what predictive behavior reveals about training. Drawing on 1,060 controlled ViT runs comprising 15.6K checkpoints (Nezhurina et al., 2025), we first ask which training choices organize function space. In raw coordinates, dataset, patch size, and optimization configuration are readily decodable, while model size is more weakly represented. But much of this structure can arise simply because models differ in competence. After regressing out accuracy and loss, *optimization configuration explains little of the remaining position*: its apparent effect is carried almost entirely by competence, while the effects of dataset, patch size, and model size persist or strengthen. To our knowledge, this is the first explicit study of what drives predictive behavior beyond competence.

We next show that function space captures not only where a model is, but where its training trajectory is going. A non-parametric estimator that borrows trajectory structure from completed runs predicts a held-out model's final accuracy and per-class accuracies to within approximately 5% and 5–7%, respectively, from only the first 10–25% of training. Later in training, the same structure predicts the model's final confusion matrix and per-image decisions. Because the estimator has no learned parameters, these results reflect structure already present in the coordinate space itself.

Together, these results show that predictive behavior contains structure obscured by both confidence and competence. Removing these confounds separates training choices that genuinely change *what* a model predicts from those that primarily change *how well* or *how confidently* it predicts. More broadly, a temperature-invariant function space provides a direct way to study what neural networks learn, how that behavior emerges during training, and where a training run is headed.

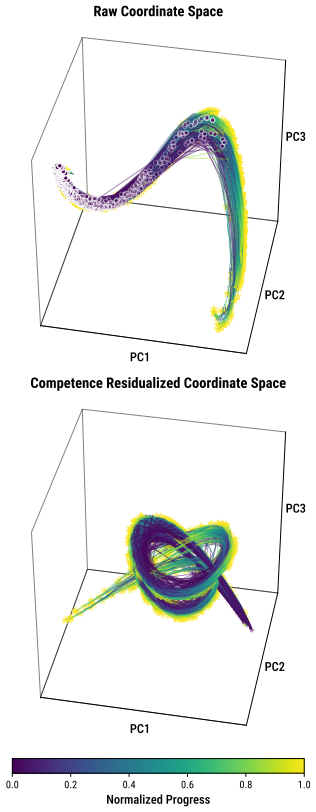


Figure 1: **Training trajectories of full runs visualized along their top-3 Principal Components in our coordinate system.** Top: in the raw coordinate space. Bottom: linearly regressing out competence (accuracy and loss information). Color indicates trajectory training progress, circles (○) mark starting locations and stars (⋆) denote endpoints.

2 Related Work

Function-space representations. Recent work has begun to compare neural networks directly through their predictions rather than their parameters or internal representations (Mao et al., 2024; Quinn et al., 2019;

Hoogland et al., 2025; Oyama et al., 2025). Given a fixed probe set, these methods construct a fingerprint from a model's outputs and use it to place many models in a shared coordinate space. Existing constructions broadly fall into two groups: *relative* and *absolute* coordinates.

Relative coordinates. Relative-coordinate methods place a model according to the similarity between its predictions and those of a predefined set of anchor models (Mao et al., 2024; Quinn et al., 2019; Ramesh et al., 2023; Teoh et al., 2020; Helm et al., 2024; 2025). These constructions are useful when the relevant similarity measure is known in advance, since the space is built directly from it. They also yield compact fingerprints whose dimensionality scales with the number of anchors rather than the total prediction length over the probe set. Their expressivity, however, depends on the anchors: behavior not represented by the anchor set can be invisible to the fingerprint (Gower, 1968; Bengio et al., 2003). We show in Section 4.1 that this information loss can reverse 4–12% of the true pairwise distance orderings between models and can become substantially worse for poorly chosen anchor sets. Adding new anchors mitigates the problem, but requires recomputing the coordinates of every model and all analyses derived from them.

Absolute coordinates. Absolute-coordinate methods instead construct a fingerprint directly from a model's predictions over the probe set. Prior work has used raw outputs (Hoogland et al., 2025; Carroll et al., 2025), softmax probabilities (Fort et al., 2020), log-likelihoods (Oyama et al., 2025), and square-root probabilities (Quinn et al., 2019). Because they retain each model's outputs independently, absolute coordinates preserve the available information, require no anchor models, and support arbitrary similarity measures (Klabunde et al., 2025).

Both relative and absolute constructions, however, are sensitive to temperature. Two models may rank classes identically and separate them in the same proportions, yet differ only by a uniform scaling of their logits. This scaling changes confidence without changing any decisions (Guo et al., 2017), but nevertheless changes their position in existing coordinate spaces. We show in Section 4.2 that varying temperature alone can reverse approximately 20% of a model's distance orderings to other models. Temperature can be estimated through calibration (Guo et al., 2017), but such estimates require held-out data and depend on the data used. Our construction instead removes temperature directly from the coordinate representation.

Inferring training properties from behavior. A related line of work asks whether a trained model's behavior reveals properties of its training procedure (Oh et al., 2018; Li et al., 2024; Mao et al., 2024). Such analyses can identify associations between training choices and predictive behavior, but generally do not separate these effects from model competence. However, accuracy is itself a strong confounder of functional and representational similarity (Klabunde et al., 2025; Huh et al., 2024). We explicitly regress out accuracy and loss before asking which training attributes remain decodable, allowing us to distinguish properties associated with competence from those reflected in behavior beyond competence.

Predicting training trajectories. Prior work has forecast scalar learning curves from partial training runs (Leite & Brazdil, 2005) and transferred updates between trajectories in weight space (Knyazev et al., 2025). Our focus is instead on predicting the endpoint *function*. Using the geometry of completed trajectories in function space, we recover not only final accuracy but per-class accuracies, confusion structure, and eventually individual image-level decisions of a held-out run. This asks whether early predictive behavior contains enough structure to anticipate the detailed function a model will ultimately learn.

3 Problem Setting and Notation

Before describing our approach and experiments, we state the setting for all our analysis and experimentation. Everything in this paper uses one population of models and one probe set.

Models used. We analyze 1,060 CLIP (Radford et al., 2021) vision-encoder training runs (consisting of 15.6K total checkpoints) from the scaling-law suite of Nezhurina et al. (2025), which, to the best of our

knowledge, contains the widest selection of controlled vision encoder training runs [1]. The collection covers three pretraining datasets: DataComp-1B (Gadre et al., 2023), DFN-2B (Fang et al., 2023) and Re-LAION-2B (LAION, 2024) and includes runs over 13 ViT (Dosovitskiy et al., 2021) configurations drawn from sizes S/M/B/L/H and patch sizes 32/16/14 (not every pair is available), and 11 data budgets from 1.28M to 3B images, along with hyperparameter sweeps over learning rate, batch size and warmup steps.

Dataset used. We conduct all our studies on ImageNet-1K (Deng et al., 2009), as it has become the standard for evaluating general recognition capabilities of vision encoders, and is also the evaluation benchmark for the models used in our study (Nezhurina et al., 2025). For every checkpoint, we freeze its parameters and train a linear probe on its final-layer features.

Every probe is trained on the same random 100-per-class subset (100,000 images total) of the ImageNet training set. We minimize the mean cross-entropy plus an ℓ_2 penalty $\frac{\lambda}{2}\|W\|_F^2$ on the probe weights, with $\lambda = 10^{-4}$, using the PyTorch L-BFGS solver (Paszke et al., 2019; Liu & Nocedal, 1989) with a strong-Wolfe line search, a learning rate of 1.0, a history size of 10, a gradient tolerance of 10^{-4}, and at most 1,000 iterations. We conduct all our experiments on a node of 8 A100s with 80GB VRAM each. We feed the entire validation set (50,000 images over 1,000 classes) in a fixed order through each checkpoint and its probe, recording the output logits for every image.

Notation. Let $X = \{x_1, \ldots, x_N\}$ be a fixed probe set for a K-way classification task. Passing each x_i through a model-j yields logits $l_i^{(j)} \in \mathbb{R}^K$, and taking their softmax forms the probability distribution $p_i^{(j)} = \mathrm{softmax}(l_i^{(j)})$ for every input i. A *coordinate* maps a model to a vector computed from its outputs on X. We call that vector the model's *position* or *fingerprint*. A training *run* is a sequence of model checkpoints. Certain experiments investigate how the predictions in model training runs evolve over training by analyzing the coordinates of intermediate checkpoints in a run. We describe a checkpoint by the fraction $F \in [0, 1]$ of its run's sample budget consumed, and call the sequence of a run's positions under a coordinate its *trajectory*. Fractions are obtained by dividing the checkpoint's epoch number by the total epochs covered in the run, enabling us to compare runs with different lengths at the same relative progress.

4 Coordinate systems should be absolute and temperature-invariant

Existing coordinate systems are built from the predictions (e.g., $l_i^{(j)}$) of a model (e.g., model-j) over inputs from a fixed probe dataset (e.g., $x_i \in X$). These systems are broadly grouped into two classes: *relative* and *absolute* coordinates respectively, and both are sensitive to temperature. In this section, we argue that coordinate systems should be both absolute (Section 4.1) and temperature-invariant (Section 4.2), before proposing a novel coordinate system that is both (Section 4.3).

4.1 Relative coordinates misplace new models

We now show how relative coordinate systems incorrectly place new models that lie outside the anchor set, making any analysis in the system error prone unless the new models are added to the anchor set and the coordinate is recomputed for all models. We illustrate this limitation with a simple setting.

Setting. Assume an anchor set of 210 ViT-S runs (totaling 3,101 checkpoints) trained on the DataComp-1B dataset, each placed in the coordinate system constructed from the entire set. Suppose two held-out sets of runs: (1) an "in-distribution" set comprising of 30 other ViT-S runs (totaling 454 checkpoints) also trained on DataComp-1B, and (2) an "out-of-distribution" set consisting of 27 large ViT-L/H runs (spanning 424 checkpoints) trained on DFN-2B and Re-LAION-2B.

[1] We use every CLIP checkpoint that was publicly downloadable at the time of writing.

We are interested in understanding how the relative distance orderings between models in each held-out set changes, when they are placed on the coordinate space constructed from joining the anchor set alone, versus the space constructed with the anchor and held-out set together. If distance ordering in the space created from the joint set is not preserved in that of the anchor set, then its coordinate system *misplaces* models, thereby compromising downstream studies. We measure ordering retention via the Kendall-τ metric Kendall (1938), which is bounded between $[-1, 1]$: $\tau = 1$ indicates perfect ranking preservation while $\tau = -1$ is the reverse.

Anchors consistently misplace models. We illustrate this phenomenon using the Bhattacharyya Distance coordinate space proposed by (Mao et al., 2024), and perform all model placement operations following the procedure outlined in their work (see Appendix A for more details). Figure 2 shows the results over both our held-out sets, across random subsets of the anchor sets at increasing size. Notably, the coordinate spaces built off the anchor sets alter the relative distance rankings of the held-out models by 4-12%, and plateau well below $\tau = 1.0$ despite the size of the anchor set increasing. Merely increasing the anchor set with other models is *not sufficient*. Moreover, Figure 2 shows that "out-of-distribution" models can collapse when placed in an anchor set (bottom right of the figure). In our study, this occurs because the coordinate system constructed from that anchor set introduces a new component in its vector space over which held-out models place 58% of their energy, thereby skewing the relative distance rankings under placement (further details are found in Appendix A). Together, these results highlight the risks of building coordinate systems from anchors: new models may not be placed well, and the coordinates of new models must be regenerated, along with any analysis performed. Note that while this analysis was performed using the coordinate system of (Mao et al., 2024), it affects any relative coordinate system of similar construction. Ideally, rankings should not change between coordinate spaces, mirroring the 1.0 line.

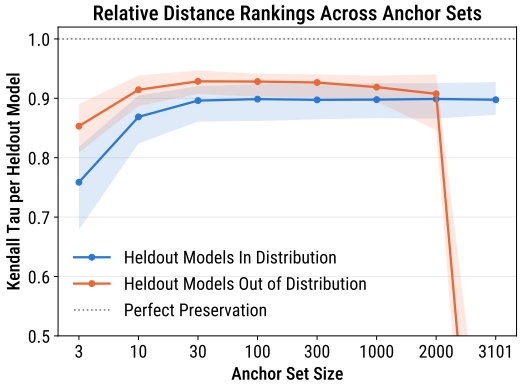


Figure 2: **Relative coordinates misplace new models outside the anchor set.** We place held-out models onto the coordinate system of a fixed anchor set constructed with (Mao et al., 2024), and compare their relative distance rankings with one another to their distance rankings when placed in the coordinate system created from joining the anchor and held-out sets together. Y-axis: Median Kendall τ over held-out models, τ=1 means identical rankings, τ=0.9 means 5% of pairs are reversed. X-axis: Anchor sets of increasing sizes. Blue: held-out models "in-distribution". Orange: held-out models "out-of-distribution". Shading: IQR over the held-out models.

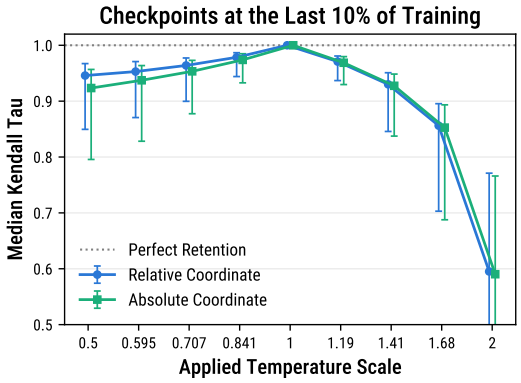


Figure 3: **Temperature changes distance rankings in existing coordinate spaces.** We compute the median (+IQR) Kendall Tau between the relative distance rankings of near final checkpoints with all others as their temperature changes. Blue: Effect of temperature scaling on relative coordinates such as that of (Mao et al., 2024). Green: Effect on absolute coordinates, illustrated with one built from (Quinn et al., 2019).

4.2 Temperature Sensitivity Matters

While absolute coordinates avoid these limitations, to the best of our knowledge every coordinate system—relative or absolute—is sensitive to temperature. Temperature is a uniform scaling applied to logits (e.g., $l_i^{(j)}$ from model-j) that sharpens or flattens the resultant softmax probability distribution (e.g., $p_i^{(j)}$) without changing the ordering be-

tween predicted classes for input x_i. Temperature *does not* drive a model's decision making, and is largely set by training calibration (Guo et al., 2017). Existing coordinate systems are *temperature-sensitive*, because changing the temperature within the predicted logits of a model, can not only change its position but also *alter its relative distances to others*. We demonstrate this empirically with a simple experiment.

Temperature arbitrarily alters model distances. Figure 3 shows how the median Kendall-τ (with IQR bands) changes between the distance rankings of the 579 checkpoints in the final 10% of training from the full anchor set described in Section 4.1 to all other checkpoints in the anchor set, as their temperature varies on a scale of $[0.5, 2.0]$. The figure shows the result using the relative coordinate system of (Mao et al., 2024) and an example absolute coordinate system created by taking the square root of the probabilities (Quinn et al., 2019), which the system of Mao et al. (2024) builds from. Overall, we observe that altering the temperature in either coordinate system changes the relative distance orderings of the underlying models, despite not changing the relative reasoning of each checkpoint on the same input. This is inherently problematic, as it confounds distance measures: two models with similar class reasoning for the same input but different temperatures can lie further apart than two models with dissimilar reasoning but similar temperatures. While we illustrate the issue with just two coordinate systems, to the best of our knowledge all are linked: none control for temperature in placement and are sensitive to it. Although it is possible to calibrate for temperature using data (Guo et al., 2017), the estimated temperature is data-dependent and thus prone to variability. Ideally, a coordinate system should be temperature-invariant, achieving 1.0 for all temperature ranges.

4.3 We propose an absolute and temperature-invariant coordinate

We propose a simple coordinate that is both absolute and temperature-invariant. Similar to (Hoogland et al., 2025; Carroll et al., 2025), we concatenate the output predictions from a model-j across all $x_i \in X$ after applying a simple two-step transformation on its probability distributions.

Step one: logit centering. Let $p_i^{(j)}$ denote the probability distribution model-j assigns to x_i, with unknown temperature $T > 0$. Using our notation from Section 3, $p_i^{(j)} = \mathrm{softmax}(\tilde{l}_i^{(j)}/T)$, where $\tilde{l}_i^{(j)}$ are the "untempered" logits of model-j. We first perform the centered log-ratio transform (Aitchison, 1982): take the log-probabilities of each class in $p_i^{(j)}$ and subtract their mean. Because $\log p_i^{(j)} = \tilde{l}_i^{(j)}/T - \left(\log \sum_{k=1}^{K} \exp\left([\tilde{l}_i^{(j)}]_k/T\right)\right)\mathbf{1}$—where $\mathbf{1}$ is a K-dimensional ones-vector—and the subtracted term is constant, it cancels out under centering. Writing $z_i^{(j)} = \tilde{l}_i^{(j)} - \left(\frac{1}{K}\sum_{k=1}^{K}[\tilde{l}_i^{(j)}]_k\right)\mathbf{1}$ for the centered untempered logits, the result is $\log p_i^{(j)} - \left(\frac{1}{K}\sum_{k=1}^{K}[\log p_i^{(j)}]_k\right)\mathbf{1} = z_i^{(j)}/T$. Overall, this centers every logit vector at a common origin and makes T a pure uniform scaling of it.

Step two: unit normalization. We then divide $z_i^{(j)}/T$ by its norm, which removes any positive scalar, temperature included: $(z_i^{(j)}/T)/(\|z_i^{(j)}\|_2/T) = z_i^{(j)}/\|z_i^{(j)}\|_2$, so every tempered version of $p_i^{(j)}$ collapses to the same coordinate. Call the result $\tilde{z}_i^{(j)}$.

Discussion. These two steps are equivalent to taking the z-score over each logit vector and dividing by $\sqrt{K}$. Thus, the transformation we use is standard and we only propose to use it as a coordinate. By construction, our coordinate system achieves Kendall-$\tau = 1.0$ for both studies performed in Sections 4.1 and 4.2. *The order of operations matters in our coordinate system*: normalizing the log-probabilities before centering does not remove temperature because the subtraction term enters the norm, nor does only normalizing the probabilities or transformations of them because temperature is not a pure scaling of those vectors.

5 ISOLATING BEHAVIOR CHANGES WHAT MODEL ASPECTS ARE PREDICTABLE

Using our newfound absolute and temperature-invariant coordinate, we ask what a model's behavior on data alone reveals about its configuration: *"does a checkpoint's position decode its training dataset, architecture size and patch size, or its learning rate, batch size, warmup steps and samples seen?"*

Analysis setup and objective. We use the full run pool described in Section 3, placed in our coordinate system (Section 4.3). For categorical attributes (dataset, model size, patch size) we train a linear decoder of the form $f(\tilde{z}^{(j)}) = \mathrm{softmax}\big(\frac{1}{N}\sum_{i=1}^{N} W_i\, \tilde{z}_i^{(j)}\big)$, where W_i maps the coordinate's block for input i to the class dimension. For continuous attributes (learning rate, batch size, warmup steps and samples seen) we estimate their log and drop the softmax, leaving a single linear map, and record R^2. The decoder can thus be applied iteratively over each input $\tilde{z}_i^{(j)}$ independently, with their transforms accumulated to obtain the result. All experiments use 5-fold cross validation over *entire runs*: each fold holds out 20% of runs, and no checkpoint of a held-out run appears in training. We train *one* decoder per attribute *across all checkpoints* of the training folds, and report held-out balanced accuracy for categorical attributes and R^2 for continuous ones, averaged over folds with standard errors, across binned training stages.

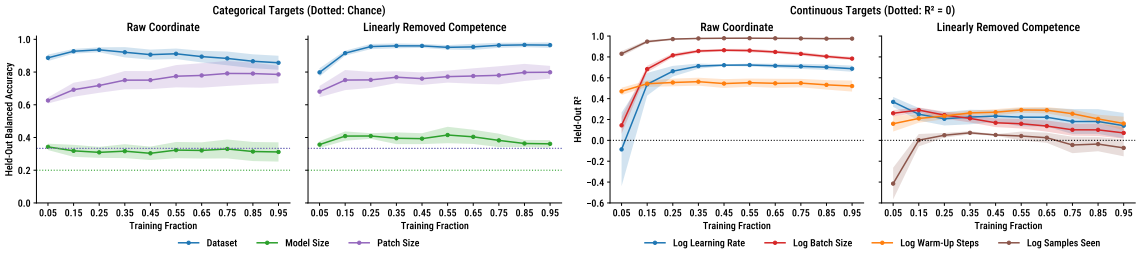


Figure 4: **Nearly any model configuration is linearly decodable from raw coordinates regardless of training amount, though not all are after competence is removed.** We show the average results with standard errors for five-fold cross validation. First: We visualize the average balanced accuracy from predicting dataset, model size and patch size from checkpoint throughout training in our raw coordinates. Second: We visualize the same metric for the same targets and checkpoints, after linearly regressing out competence. Third: We plot the average R^2 across varying training features for the same checkpoints in our raw coordinate system. Fourth: We show the same metric for the same targets and models after linearly regressing out competence.

The raw coordinate nearly decodes the full training recipe. The first and third plots in Figure 4 show the results at every training stage. Dataset is decoded at 0.86-0.93 balanced accuracy (chance 0.33) and patch size at 0.63-0.79 (chance 0.33) at every stage, samples seen at $R^2 \geq 0.95$ from $F = 0.15$, batch size at $\approx$0.8, learning rate at $\approx$0.7 and warmup at $\approx$0.55 from $F = 0.25$. Only model size is weakly predictable, staying near chance (0.30-0.35 against 0.20). Nearly every attribute is predictable *with a linear decoder*, from a single checkpoint at any point in its run, similar to the findings of prior work (Oh et al., 2018; Li et al., 2024)

Yet, this *does not* mean that behavior alone tracks these attributes. A checkpoint's coordinate is a function of its outputs and inherently encodes its accuracy or competence on the task. Consequently, a decoder may read an attribute through accuracy rather than through decisions (Klabunde et al., 2025). Thus, we linearly regress accuracy and loss out of the coordinate in two steps. First, we fit an affine map from a checkpoint's accuracy and loss to its coordinate across all checkpoints. Then, we subtract its prediction from every coordinate across all checkpoints, using their own accuracy and loss, leaving representations that are not linearly explained by competence. The choice of linear removal is deliberate: it is closed-form, and its low capacity limits what it can take out to the part of position that varies linearly with competence.

Training configuration predictability collapses with competence removed. We retrain the decoders on the same folds after residualization (second and fourth plots in Figure 4). Dataset, patch size and model size are unchanged or improve with accuracy and loss removed (dataset 0.95-0.96 from $F = 0.25$, patch size 0.75-0.80, model size 0.36-0.41), while the training attributes collapse: samples seen to $R^2 \approx 0$, batch size and warmup to at most 0.3, and learning rate to 0.13-0.23 past $F = 0.25$. Thus, the decoding quality of optimization from the raw coordinate is almost entirely built from competence information—accuracy and loss—rather than behavior, *yet behavior does measure a checkpoint's training structure* (dataset, architecture).

6 Coordinates can estimate convergence behaviors

Section 5 shows that a checkpoint's raw position records its training configuration, and Figure 1-top suggests that trajectories occupy a shared structure throughout the training process. In this section, we examine whether this observed structure is predictive: *"Is it possible to estimate the behavior a model reaches at the end of its training from just a fraction of its run?"*

Experiment Setup. We form nine groups from the runs of Section 3, each comprised of a dataset-architecture pair in $\{\text{DataComp-1B}, \text{DFN-2B}, \text{Re-LAION-2B}\} \times \{\text{ViT-S/16}, \text{ViT-S/32}, \text{ViT-M/32}\}$, and randomly split the full runs of each group 80%/20% into a retrieval pool of completed runs and a held-out set respectively, so that no checkpoint of a held-out run exists in the completed run pool. We choose these architectures as they have the most runs across our study's datasets. Given a held-out run observed up to a fraction F of its training, we predict the decisions of its final checkpoint on every image from the ImageNet validation set and compare them to the decisions of the actual endpoint across four metrics: (1) the absolute error of the predicted top-1 accuracy, (2) the mean absolute error of the predicted per-class accuracies, (3) the total variation between the predicted and actual confusion matrices, and (4) the top-1 disagreement between the predicted and actual labels per-image.

Predictors Employed. We estimate endpoint behavior using two simple algorithms. The first—termed "mean of trained runs"—predicts its behavior to be the coordinate mean of all the final checkpoints in the completed run pool. This method gauges the degree to which runs within each pair converge to similar predictions. The second is a very simple "non-parametric estimator": a nearest-neighbor extrapolation of the trajectory using the retrieval pool. For a held-out run at fraction $F \in [0, 1]$, we take each pool run's checkpoint nearest to F and weigh pool runs by a softmax over their negative distances to the held-out run's checkpoint at F. We then calculate the displacement from the nearest to F checkpoint in each pool run and its final checkpoint, and take the weighted average over the respective displacements of all pool runs using the computed softmax weights. We finally add the result to the held-out checkpoint. Formally, writing $\tilde{z}^{(h)}_{F,i}$ for the held-out run's coordinate on input i at fraction F, and $\tilde{z}^{(j)}_{F,i}$, $\tilde{z}^{(j)}_{1,i}$ for pool run j's blocks at F and at its endpoint, we predict the held-out endpoint as follows:

$$\hat{z}^{(h)}_{1,i} = \tilde{z}^{(h)}_{F,i} + \sum_j w_{ij}\big(\tilde{z}^{(j)}_{1,i} - \tilde{z}^{(j)}_{F,i}\big), \qquad w_{ij} \propto \exp\big(-\|\tilde{z}^{(h)}_{F,i} - \tilde{z}^{(j)}_{F,i}\|_2^2/\beta\big), \tag{1}$$

where $\hat{z}^{(h)}_{1,i}$ is then unit-normalized and β is found apriori and fixed via a grid search over $\{0.003, 0.01, 0.03, 0.1, 0.3, 1.0, 10.0\}$ using leave-one-out cross validation over the retrieval pool. We perform this process *iteratively over the individual representations corresponding to each input in our coordinate at a time*, never holding the full coordinate in memory.

Accuracy and per-class accuracies are recoverable from just 10-25% of training. Figure 5 reports all four metrics against the fraction observed, per group. It reveals that from the first 10-25% of a run, the predicted endpoint accuracy is within $\approx$0.05 of the true one and the per-class accuracies within 0.05-0.07 on average, against 0.15-0.29 and 0.16-0.29 for the mean of trained runs. Both errors fall monotonically, to $\leq$0.01 and 0.02-0.03 by $F = 0.9$. This holds in every group, without a single trained parameter.

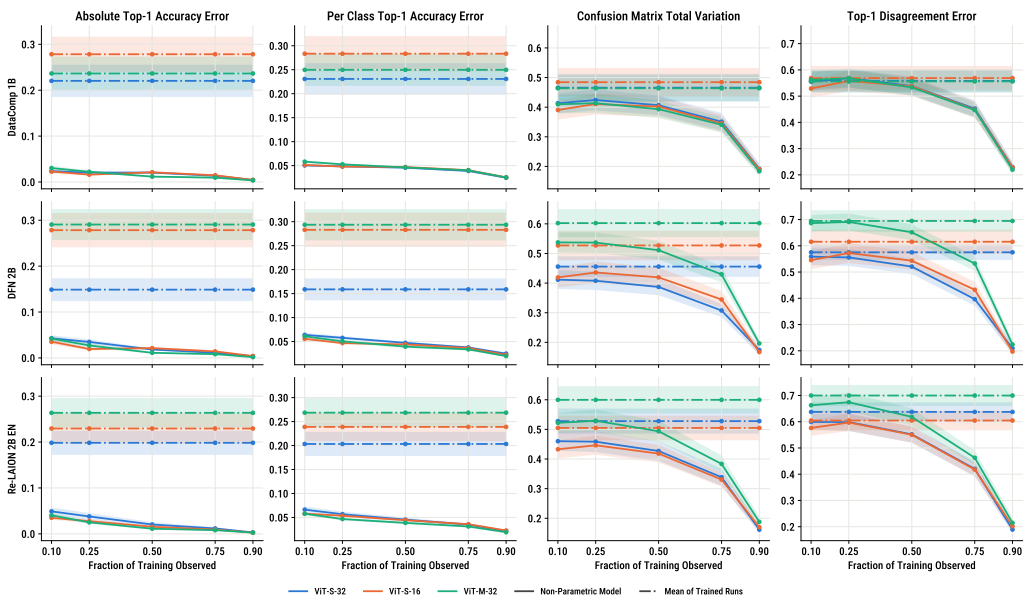


Figure 5: **Endpoint behavior of held-out runs, predicted from a fraction of their training.** Nine dataset-architecture groups (rows: pretraining dataset, colors: architecture). Columns: the absolute error of the predicted top-1 accuracy, the mean absolute error of the per-class top-1 accuracies, the total variation between predicted and actual confusion matrices, and the top-1 class disagreement per-image with the true endpoint. Solid line: the non-parametric extrapolator of Equation 1. Dotted line: the mean of the retrieval pool's completed runs. Bands: standard error over held-out runs. Accuracy and per-class accuracies are recovered from 10% of training, while per-input decisions and the confusion matrix are in the second half of training.

Decisions become predictable from about half of training. Estimating per-image decisions is a significantly more challenging task. Despite this, Figure 5 shows that top-1 disagreement with the true endpoint is competitive with the mean of trained runs up to $F = 0.25$ across all groups, begins to separate from it at $F = 0.5$, and falls to 0.39-0.53 by $F = 0.75$, eventually obtaining ≈80% agreement by F=0.9. The confusion matrix follows the same course.

Results demonstrate significant structure in the coordinate space. To our knowledge, no prior work recovers a training run's endpoint at this granularity. Scalar learning curves have been forecast from other runs (Leite & Brazdil, 2005) and weight-space updates transferred between them (Knyazev et al., 2025), yet neither forecasts the per-class accuracies and confusion matrix a run will reach. A non-parametric estimator alone is *sufficient* to perform well across each metric, highlighting the intrinsic shared structure our coordinate systems captures across trajectories and a learned predictor can only improve predictions.

7 Conclusion

We address limitations in prior work by proposing absolute and temperature-invariant coordinate system for placing and reasoning over the functions neural networks learn. We use the largest open-source repository for full vision encoder training runs—1000+ trajectories spanning 15.6K checkpoints—and find that position in our coordinate system predicts a checkpoint's specification without knowing its accuracy or loss, while its training configuration requires these for prediction. Trajectories in our coordinate system are also highly structured: granular performance metrics for a fully trained model are predictable from its partial trajectory.

8 AI USE STATEMENT

In this work, we used a mix of generative AI tools (we used Claude for all) that partly intersect a few of the required disclosure categories. We state the broader categories first, and provide more detail afterwards. The broad required categories we used AI tools for are "design or providing feedback on research methodology or experiments", "method implementation", and "support qualitative and thematic data analysis". We did not use generative AI tools for any of the other tasks listed (most are not applicable to this paper): "generating synthetic datasets", "helping develop theoretical models or conceptual frameworks", "formulating mathematical claims", "provide critical ingredients for proving mathematical claims", "assist in the writing of proofs", "propose or refine hypotheses", "assist with translation", "clean and reformat datasets", or "interpret results". We have manually reviewed all AI-assisted work for correctness. Regarding "design or providing feedback on research methodology or experiments": we used AI-tools to aid in our checking for *novelty* in methodology and experimentation. Coupled with a manual cross-check (and literature review where applicable) for each of its responses, it has aided us in crediting prior work for similar results. "Method implementation" and "support qualitative and thematic data analysis" are combined in our case: we used AI-tools to help implement code for our figure visualizations. We manually reviewed every line of generated code to ensure that it exactly mirrored our intentions. We did not use AI-tools for ideation, or design new experiments based off its suggestions. AI also aided our search for related literature, and it helped place finishing touches on the writing of our submission (e.g., syntax and grammatical prose). All its proposed edits were manually reviewed.

9 REPRODUCIBILITY STATEMENT

We commit to releasing our code upon publication. We describe all our methods, the data used, and training procedures in each applicable section.

REFERENCES

J. Aitchison. The statistical analysis of compositional data. Journal of the Royal Statistical Society: Series B (Methodological), 1982. 2, 6

Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In NeurIPS, 2003. 3, 13

Liam Carroll, Jesse Hoogland, Max Farrugia-Roberts, and Daniel Murfet. Dynamics of transient structure in in-context linear regression transformers, 2025. 2, 3, 6

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In CVPR, 2009. 4

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 ICML, 2021. 4

Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. Data filtering networks. arXiv preprint arXiv:2309.17425, 2023. 4

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2020. URL https://arxiv.org/abs/1912.02757. 2, 3

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In ICML, 2020. 1

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets. In NeurIPS, 2023. 4

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. NeurIPS, 2018. 1

John C. Gower. Adding a point to vector diagrams in multivariate analysis. Biometrika, 1968. 3, 13

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. 3, 6

Hayden Helm, Brandon Duderstadt, Youngser Park, and Carey Priebe. Tracking the perspectives of interacting language models. ACL, 2024. 2, 3

Hayden Helm, Aranyak Acharyya, Youngser Park, Brandon Duderstadt, and Carey Priebe. Statistical inference on black-box generative models in the data kernel perspective space. ACL, 2025. 2, 3

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 1

Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. The developmental landscape of in-context learning, 2025. 1, 2, 3, 6

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. In ICML, 2024. 3

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In NeurIPS, 2018. 1

Maurice G. Kendall. A new measure of rank correlation. Biometrika, 1938. 5

Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. ACM, 2025. 1, 3, 7

Boris Knyazev, Abhinav Moudgil, Guillaume Lajoie, Eugene Belilovsky, and Simon Lacoste-Julien. Accelerating training with neuron interaction and nowcasting networks. In ICLR, 2025. 3, 9

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In ICML, 2019. 1

LAION. Releasing re-laion-5b: transparent iteration on laion-5b with additional safety fixes. https://laion.ai/blog/relaion-5b/, 2024. 4

Rui Leite and Pavel Brazdil. Predicting relative performance of classifiers from samples. In ICML, 2005. 3, 9

Rongqing Li, Jiaqi Yu, Changsheng Li, Wenhan Luo, Ye Yuan, and Guoren Wang. DREAM: Domain-free reverse engineering attributes of black-box model. IEEE Transactions on Knowledge and Data Engineering, 2024. 3, 7

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. Mathematical Programming, 45(1):503–528, 1989. 4

Jialin Mao, Itay Griniasty, Han Kheng Teoh, Rahul Ramesh, Rubing Yang, Mark K. Transtrum, James P. Sethna, and Pratik Chaudhari. The training process of many deep networks explores the same low-dimensional manifold. PNAS, 2024. 1, 2, 3, 5, 6, 13

Tom M. Mitchell. Machine Learning. McGraw-Hill, New York, 1997. ISBN 0070428077. 1

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In ICLR, 2023. 1

Marianna Nezhurina, Tomer Porian, Giovanni Pucceti, Tommie Kerssies, Romain Beaumont, Mehdi Cherti, and Jenia Jitsev. Scaling laws for robust comparison of open foundation language-vision models and datasets. arXiv:2506.04598, 2025. URL https://arxiv.org/abs/2506.04598. 2, 3, 4

Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In ICLR, 2018. 3, 7

Momose Oyama, Hiroaki Yamagiwa, Yusuke Takase, and Hidetoshi Shimodaira. Mapping 1,000+ language models via the log-likelihood vector. ACL, 2025. 1, 2, 3

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, volume 32, 2019. 4

Katherine N. Quinn, Colin B. Clement, Francesco De Bernardis, Michael D. Niemack, and James P. Sethna. Visualizing probabilistic models and data with intensive principal component analysis. PNAS, 2019. 1, 2, 3, 5, 6, 13

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In PMLR, 2021. 3

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In NeurIPS. 2017. 1

Rahul Ramesh, Jialin Mao, Itay Griniasty, Rubing Yang, Han Kheng Teoh, Mark Transtrum, James Sethna, and Pratik Chaudhari. A picture of the space of typical learnable tasks. In ICML, 2023. 2, 3, 13

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014. URL https://arxiv.org/abs/1312.6120. 1

George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In ICLR, 2024. 1

Han Kheng Teoh, Katherine N. Quinn, Jaron Kent-Dobias, Colin B. Clement, Qingyang Xu, and James P. Sethna. Visualizing probabilistic models in minkowski space with intensive symmetrized kullback-leibler embedding. Phys. Rev. Res., Aug 2020. 2, 3

Vladimir N. Vapnik. The nature of statistical learning theory. Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8. 1

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In ICML, 2022. 1

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In ICLR, 2017. 1

A RELATIVE COORDINATES MISPLACE MODELS OUTSIDE THE ANCHOR SET

This section provides a detailed description for how relative coordinates may misplace models that lie outside the anchor set, making re-embedding a *requirement* for any downstream analysis. We illustrate this limitation using the established InPCA embedded Bhattacharyya Distance (BD) coordinate system used by (Mao et al., 2024; Quinn et al., 2019; Ramesh et al., 2023), and note that conclusions drawn from our study extend to other relative systems by nature.

For each model-pair, the BD is computed via $d_{\mathrm{B}}(p^{(u)}, p^{(v)}) = -N^{-1} \sum_{i=1}^{N} \log \sum_{k=1}^{K} \sqrt{[p_i^{(u)}]_k} \sqrt{[p_i^{(v)}]_k}$, where $\sqrt{[p_i^{(u)}]_k}$ is the k^{th} class index of $p_i^{(u)}$. The resultant matrix composed of all pairwise-distances, D, then double-centered, obtaining matrix W and finally eigen-decomposed: $W = U\Lambda U^T$. The j^{th} model's position V_j is then given by $[U|\Lambda|^{1/2}]_j$. Following (Mao et al., 2024), a model w outside the set is placed via the formulation,

$$V_{wk} = |\lambda_{kk}|^{-1/2} \sum_{u} U_{uk} W_{wu} \tag{2}$$

where U_u denotes the u^{th} column of U and is the standard out-of-sample extension of classical scaling (Gower, 1968; Bengio et al., 2003). We now show how this standard procedure alone can compromise the placement of such out-of-set models.

Experiment setup in more detail. Using the sets introduced in Section 4.1, we further create sub-anchor-sets of sizes 3 to 3,101 by randomly sub-sampling from the anchor set without replacement. For a given set, we compute its D and take the eigen-decomposition to embed its models. We then project all held-out models from both sets independently onto the resultant frame following (Gower, 1968; Mao et al., 2024). Separately, for each sub-sampled set, we join it with a full held-out set – either the "in-distribution" or "out-of-distribution" one – and embed all models in the joined space.

Out of distribution models are degenerately placed. Zooming into the bottom right of Figure 2, we observe that the distance rankings of models in the held-out set collapse when viewed in the full sized anchor set of 3,101 models—their $\tau = -0.2$. The collapse has a single cause. The full anchor set contains a single eigenvalue whose value is near zero, making its factor $|\lambda_k|^{-1/2}$ large, and thus amplifies any component a model has lying along its corresponding eigen-direction when projected via Equation 2. Models in the anchor set and those in-distribution barely lie along this component—placing just 0.4% of their total energy on it, and hence remain unaffected from the projection. However, out-of-distribution models place 58% of their total energy on this singular direction, indicating their behavior is poorly captured by the set. This dramatically alters their positions and corresponding relative distances after projection, and exemplifies the risk with embedding new models into a coordinate system composed of a set of anchors.

B COMPETENCE REMOVED TRAJECTORY VISUALIZATIONS IN MORE DETAIL

Figures 6, 7 and 8 show the trajectories of all the runs used in this work, grouped according to their dataset (columns) and budget (rows), then colored according to their architecture size. The initialization of each run is denoted by a circle, while its endpoint is given a star. The right-most column, termed 'combined", plots the trajectories with the row's budget from any dataset together. Each column traces the trajectories in the top-3 principal components, computed by taking the PCA over the checkpoints' linearly residualized coordinates (described in Section 5) *only trained on that dataset*, such that *all scales share the same PCA*. In the case of the "combined" column, the PCA is taken jointly across all datasets. The figures are split across budgets as there are too many to fit in one figure.

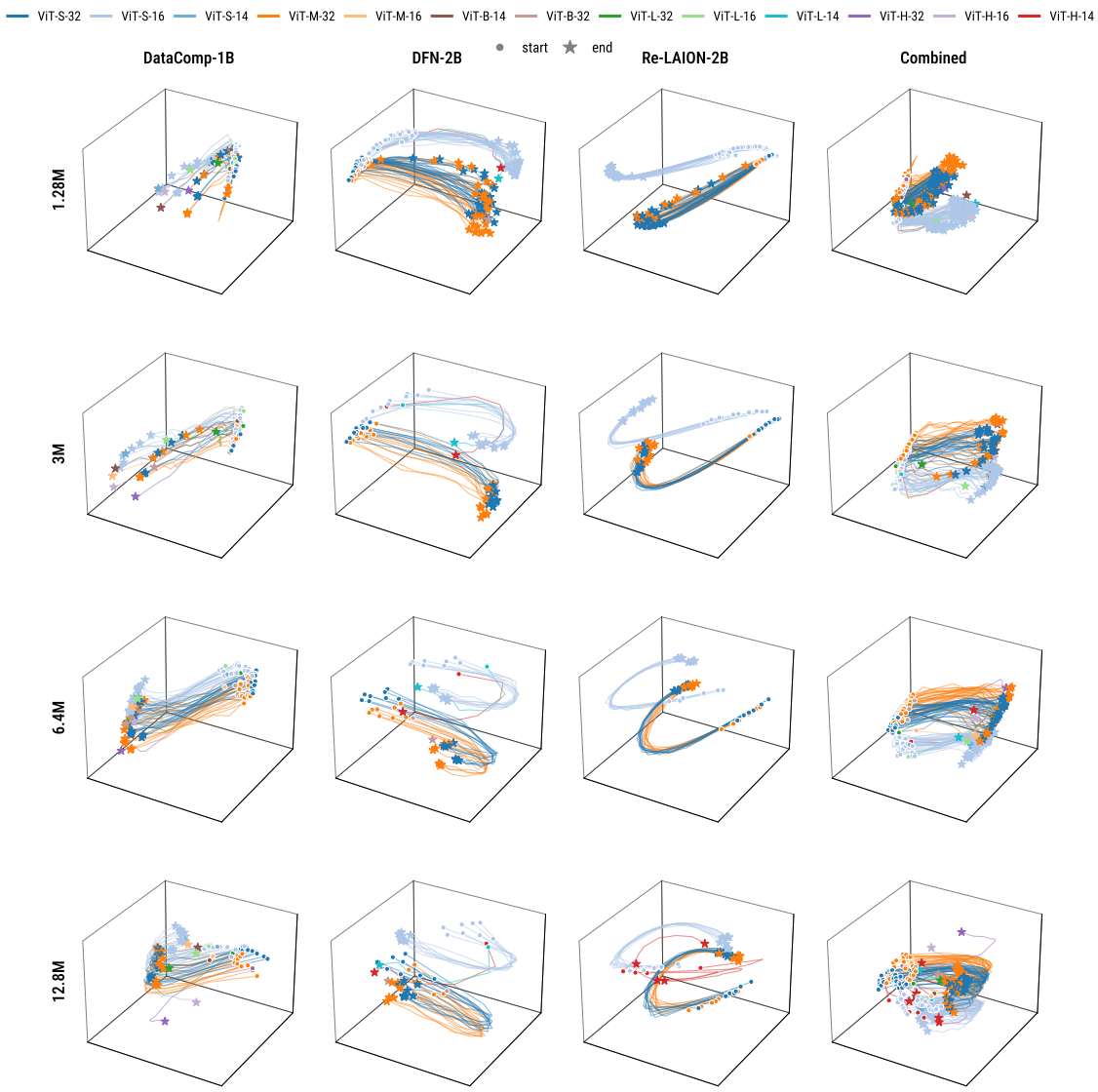


Figure 6: **First set of training trajectories across budget (rows) and datasets (columns) in the competence removed coordinate space.** Each panel shows 3D trajectories of CLIP model training runs projected onto the top-3 principal components of our linearly residualized coordinate space using the approach described in Section 5. Rows correspond to data budgets ranging from 1.28M–12.8M samples seen, while columns correspond to training datasets, with the right-most column projecting all runs onto a joint PC space regardless of dataset. Color indicates architecture; circles (○) and stars (⋆) mark trajectory start and end points, respectively. Across scales and datasets, runs of the same architecture tend to trace similar paths through function space.

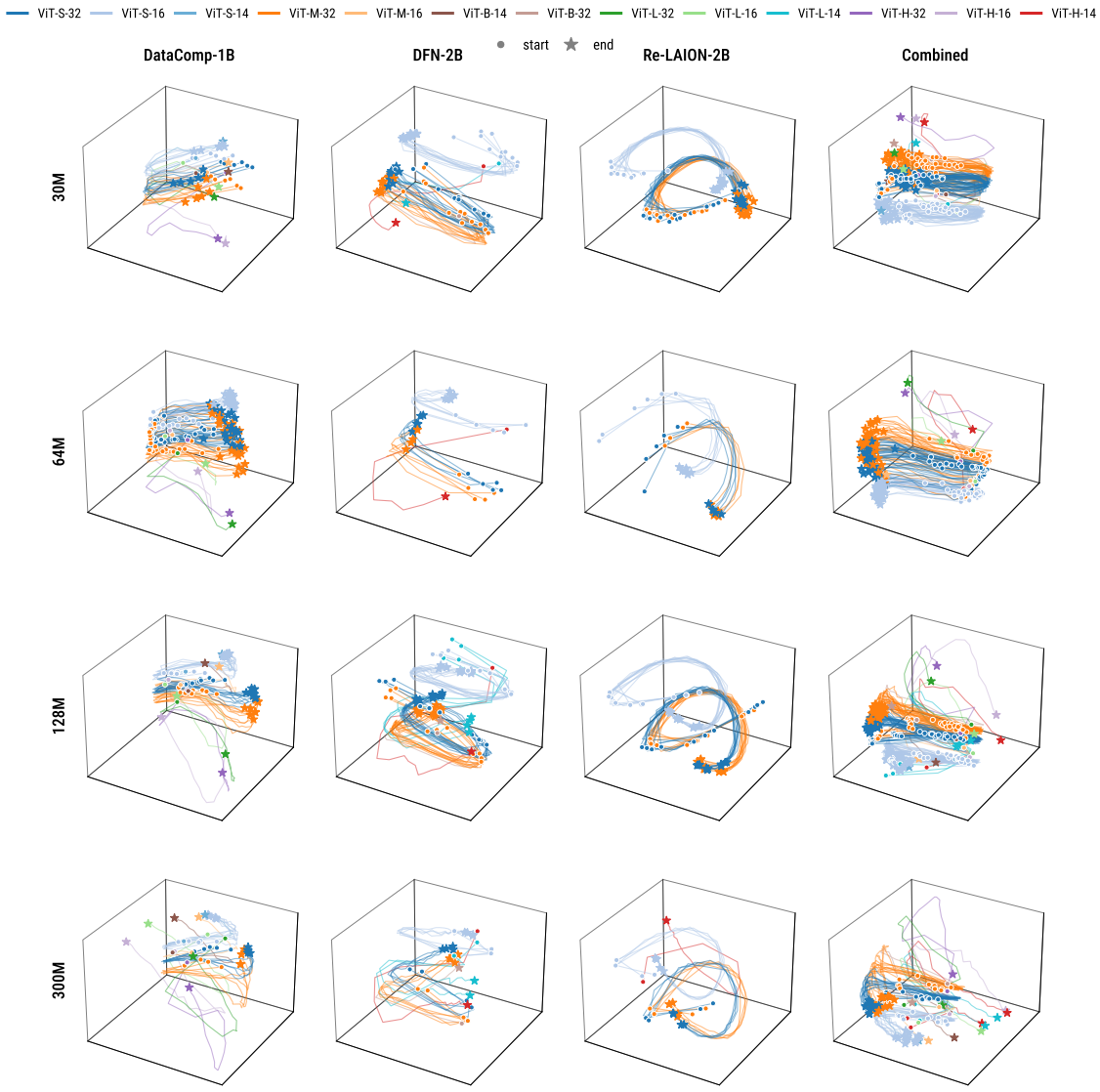


Figure 7: **Second set of training trajectories across budget (rows) and datasets (columns) in the competence removed coordinate space.** Refer to Figure 6 for detailed explanation of each facet, we show trajectories for data scales ranging from 30M to 300M.

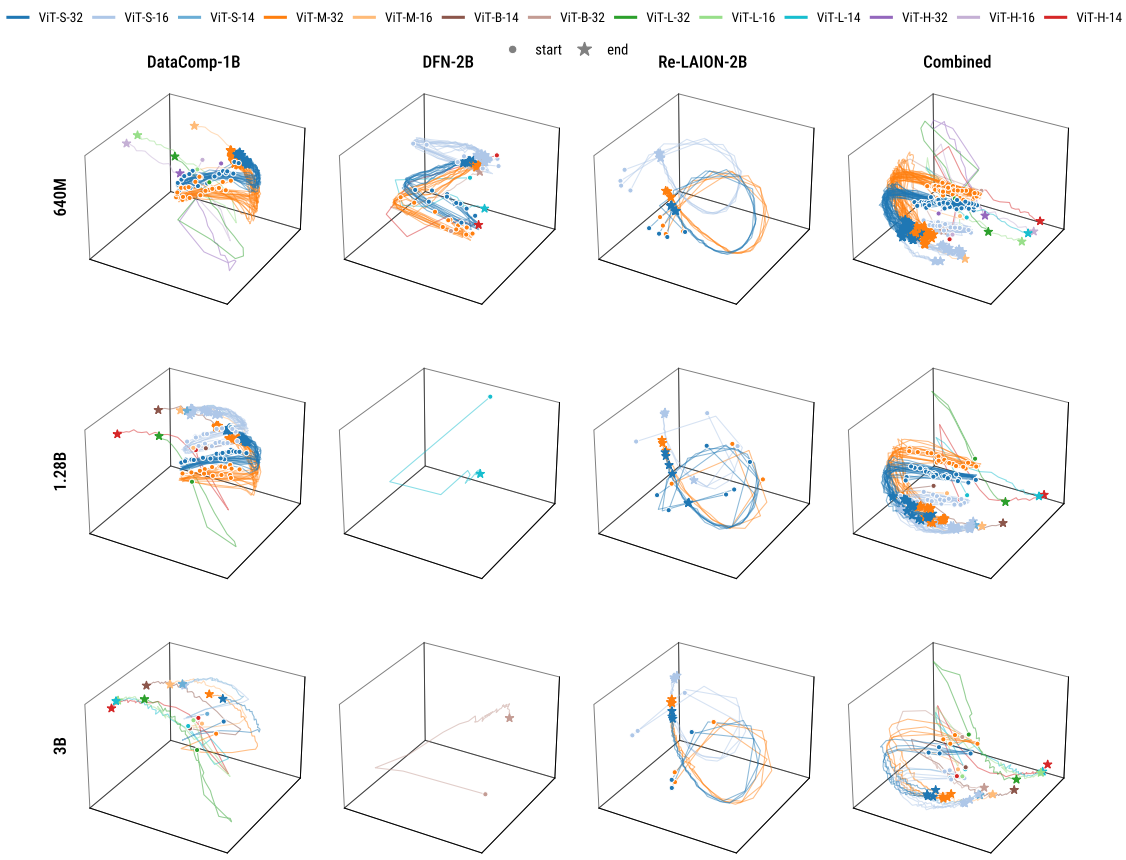


Figure 8: **Third set of training trajectories across budget (rows) and datasets (columns) in the competence removed coordinate space.** Refer to Figure 6 for detailed explanation of each facet, we show trajectories for data scales ranging from 640M to 3B.