DOING MORE WITH LESS: MONITORING GENERALIZATION DURING PRETRAINING VIA VIRTUAL MOE

Anonymous authors
Paper under double-blind review

ABSTRACT

How to monitor the generalization of dense LLMs as "generalist solvers" during their pretraining? A typical strategy is to evaluate each intermediate checkpoint on benchmarks of downstream tasks, which is costly and offers limited insights into the memorization-to-generalization transition. We propose to track an LLM's generalization by its capability of "doing more with less", i.e., *generating diverse, rich outputs by fewer, highly reusable programs.* We further discover that such a property can be captured by the pretraining dynamics of model internal states without additional finetuning or benchmark evaluation. Recent studies reveal a strong correlation between the generalization of Mixture-of-Experts (MoE) LLMs and its pathway (expert choices across layers) reusability across samples. Motivated by this, we develop a "virtual MoE" view of layers in dense LLMs by training a CrossCoder as a routing proxy, which enables us to compute and compare pathways between samples. Across the pretraining checkpoints of `Pythia` (160M-1B), `OLMo-1B`, and `OLMo2-1B`, we observe that (1) the edit distance between "virtual" pathways of relevant samples consistently decreases, which indicates their improved reusability; (2) within shared "virtual" pathways, latent representations become more discriminatively structured, suggesting that reusable pathways preserve input-specific distinctions needed for downstream generalization. Moreover, these two trends strongly correlate with generalization performance observed on downstream tasks. Hence, the two metrics form an efficient monitor, which tracks generalization capability and provides a mechanistic interpretation of pretraining.

1 INTRODUCTION

Generalist models such as large language models (LLMs) are normally pretrained using tasks different from the downstream tasks in the final deployment (Brown et al., 2020). This causes difficulty of tracking the generalization performance during pretraining as the dynamics of pretraining loss does not necessarily reflect the downstream performance (Liu et al., 2023; Isik et al., 2024). Moreover, pretraining checkpoints usually have to be finetuned before being tested on standard benchmarks as they struggle with following instructions or domain-specific prompts (Chung et al., 2024; Raffel et al., 2020). However, finetuning does not only require additional cost but also makes extra changes to the pretraining checkpoints, undermining the fidelity of benchmark scores in directly assessing the pretraining quality (Sun & Dredze, 2025). On the other hand, how LLMs' generalization capabilities "emerge" from the pretraining process remains mysterious and lacks a transparent theory (Wei et al., 2022; Schaeffer et al., 2023; Elhage et al., 2021). Hence, how to monitor, measure, and interpret the dynamics of LLMs' generalization capability during pretraining remains an open challenge.

To avoid extra finetuning and benchmark tests, can we establish a mapping between pretrained checkpoints' internal characteristics and their empirical generalization performance? Better generalization usually refers to the capability of leveraging limited internal capacity, e.g., skills, functions, tools, or modules, to address more tasks and generate diverse outcomes, i.e., "doing more with less." In contrast, memorization tends to require more internal capacity to achieve the output diversity, in other words, "doing more with more." Inspired by this concept, we study how to measure and track internal programs' complexity and output discriminativity, and their correlation with the generalization performance evaluated on benchmarks. Recent work (Li et al., 2025) on mixture-of-experts (MoE) LLMs offers a mechanistic measurement of the "internal complexity": as pretraining proceeds, inputs may route through increasingly structured expert pathways, and pathway statistics can reveal transitions

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.

between memorization and generalization. However, this metric is not applicable to dense LLMs, which lack explicit routers and therefore expose no native notion of pathways.

In this paper, we show that dense LLMs' pretraining can still be understood and monitored through a lens of "virtual MoE". Our key idea is to train a CrossCoder (Bricken et al., 2023; Lindsey et al., 2024; Cunningham et al., 2023) across checkpoints and use its sparse routing structure as a *learned proxy* for latent computation pathways in the underlying dense model. Once this proxy is fixed, each input induces a layerwise routing pattern, allowing us to compare samples in terms of the pathways they traverse during pretraining.

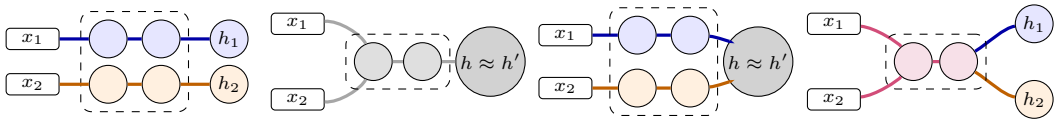
This virtual pathway view reveals a consistent empirical picture across model families and scales. On held-out benchmark prompts, routing patterns across samples become increasingly similar over training, suggesting improved *pathway reuse* through condensation of shared computational subroutines. At the same time, within pretraining-corpus domains, samples that follow similar pathways develop more discriminatively structured representations, preserving input-specific distinctions within shared computational pathways rather than collapsing to uniform states. Together, these trends suggest that dense pretraining develops increasingly reusable yet flexible computational organization.

Our goal is not to claim that CrossCoder pathways uniquely decompose dense transformers. Rather, we ask whether they provide a stable and informative coordinate system for monitoring training dynamics. Empirically, we find that they do: simple routing-based statistics track downstream accuracy trajectories and expose structure that is invisible to loss alone. We further combine pretraining-time pathway statistics with historical evaluations to predict changes in benchmark performance. The Adaptive Reuse Ratio predicts the direction of benchmark-score changes with 72.9% accuracy, compared with 65.7% for training-loss controls.

Our main contributions can be summarized as follows:

- We introduce a "virtual MoE" based monitor for dense LLM pretraining using a CrossCoder as a learned routing proxy across checkpoints.
- We identify two recurring dynamics across model families and scales: *pathway reuse* on held-out benchmark prompts, and *discriminative structure within shared computational pathways* in pretraining domains.
- Our study reveals a correlation between these two signals and downstream performance, providing a lightweight and more mechanistic complement to loss and periodic evaluation. We combine these signals with historical evaluations to predict changes in benchmark performance.

2 Motivation: From Memorization to Reusable Computation



(A) Memorization *distinct pathways, distinct states* | **(B) Collapse** *shared pathway, undifferentiated states* | **(C) Degenerate** *distinct pathways, undifferentiated states* | **(D) Reusable computation** *shared pathway, distinct states*

Figure 1: **Computation regimes from pathway complexity and state diversity.** Memorization preserves distinct states through fragmented pathways; collapse shares pathways but erases input-specific distinctions; degenerate computation remains fragmented without useful state diversity; reusable computation shares pathways while preserving discriminative states.

What does it mean for a pretrained LLM to generalize as a *generalist* model? Classical supervised learning studies generalization on a fixed task: examples share one target rule, and the main question is whether a learned hypothesis transfers from training samples to test samples. Generalist models face a different problem. They must transfer pretraining computation to many latent downstream tasks, often without being told which task an input instantiates. Thus, generalization is not only about fitting a fixed input-label relation; it also requires inferring the intended task, selecting an appropriate computation, and producing a representation from which many compatible tasks can be read out.

We hypothesize that this ability emerges as *reusable computation*. A memorizing model can preserve input-specific behavior by assigning samples to distinct pathways. A collapsed model can share pathways only by erasing useful distinctions. The desired regime is different: related inputs reuse a

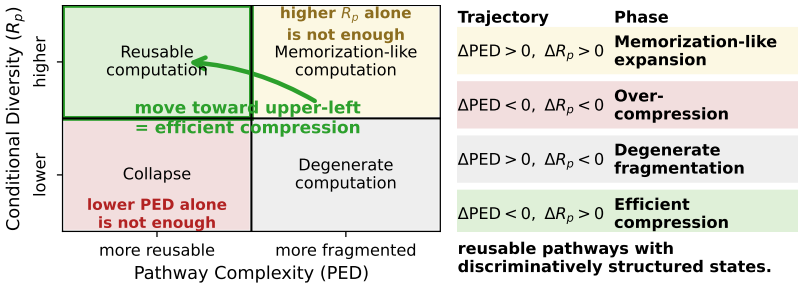


Figure 2: **Computation regimes and training transitions.** The P_t-R_t plane separates computation organization from training direction. Reusable computation is the desired regime: low pathway complexity with high conditional state diversity. Efficient compression is the desired transition toward it, reducing pathway complexity while strengthening discriminative structure within shared pathways.

compact family of computational pathways, while their representations remain discriminative. In short, generalist generalization requires doing more with less: fewer pathways, richer representations.

To depict this idea, we view a checkpoint θ as inducing a latent pathway Π and a representation Z:

$$\Pi \sim p_\theta(\pi \mid X), \qquad Z \sim p_\theta(z \mid X, \Pi).$$

Here, Π denotes the internal computational route selected for input X, and Z denotes the representation produced by executing that route. The pathway captures *what computation is selected*; the representation captures *what task-relevant information remains available after that computation*. Because the task is latent, pathway selection is itself part of the generalization problem.

A theoretical view of reusable computation. A generalist model must generalize across an open family of downstream tasks: given an input, it must infer an appropriate computational pathway Π and produce a representation Z that supports downstream readout. Let $R_{\text{open}}(\theta)$ denote the expected risk over this task family. We view this risk as governed by four factors: (1) *pretraining error*, measuring fit to the pretraining distribution; (2) *pathway complexity*, measuring the description length of reusable computational programs; (3) *representation separability*, measuring how well pathway-conditioned representations support the considered family of downstream tasks; and (4) *task-inference error*, measuring the cost of selecting a pathway inconsistent with the task implied by the input. The middle two factors define the two structural axes of the computation regimes in Figure 1. Under the sufficient assumptions stated in Appendix B, these four factors yield the following conditional generalization bound with probability at least $1 - \delta$.

$$R_{\text{open}}(\theta) \lesssim \underbrace{R_{\text{pre}}(\theta)}_{\substack{\text{pretraining} \\ \text{error}}} + \underbrace{C_1 \sqrt{\frac{\mathcal{L}_{\text{path}}(\Pi_{1:n}) + \log(1/\delta)}{n}}}_{\text{pathway complexity}} + \underbrace{\frac{C_2}{\sqrt{m\, \mathcal{D}_{\text{open}}(Z \mid \Pi)}}}_{\text{representation separability}} + \underbrace{\mathcal{E}_{\text{task}}}_{\substack{\text{task} \\ \text{inference}}}. \tag{1}$$

Here $\mathcal{D}_{\text{open}}$ measures task-relevant representation separability through normalized downstream margin; greater separability reduces the readout-complexity term. The bound provides a theoretical view of "doing more with less": lower pretraining error captures better fit to the pretraining distribution, reusable pathways reduce pathway complexity, and representations that preserve task-relevant distinctions reduce the readout-complexity penalty. Compression alone is insufficient; pathway reuse is useful only when shared computation preserves representations that support downstream tasks. The final term captures errors in inferring an appropriate pathway from the input.

The middle two terms organize the regimes in Figure 1. Memorization-like computation preserves useful distinctions but incurs high pathway complexity; collapse achieves pathway reuse at the cost of representation separability; degenerate computation is unfavorable on both axes. Reusable computation combines low pathway complexity with high task-relevant representation separability.

The theoretical quantities in Eq. 1 are not directly observable in dense LLMs. We therefore use pathway edit distance P_t as an empirical proxy for pathway-family compressibility, and conditional state diversity R_t as a task-agnostic proxy for preserved within-pathway representation structure. These proxies are not identified with the theoretical quantities themselves; Appendix B states the assumptions and geometric motivation underlying these connections.

These empirical proxies yield the phase plane in Figure 2. The target regime is reusable computation: low pathway complexity with high conditional state diversity. The corresponding training transition

is efficient compression, where the model reduces unnecessary pathway variation while preserving or strengthening discriminative structure within shared pathways.

This motivation leads to two empirical predictions. First, as generalization improves, related inputs should increasingly share latent computation, reflected by lower pathway-level complexity. Second, this sharing should not arise from representational collapse; inputs that follow similar pathways should still retain discriminative internal representations. The next section develops a virtual-pathway construction for dense LLMs and uses it to test these predictions across pretraining checkpoints.

3 METHOD OVERVIEW

The reusable-computation hypothesis motivates two observable signals: pathway-level complexity and state diversity within shared pathways. While pathways are explicit in mixture-of-experts (MoE) models, dense transformers lack explicit routing mechanisms. We construct a virtual routing space from CrossCoder latents, assigning each input a discrete pathway through the model. We then quantify pathway complexity with *pathway edit distance*, measure state diversity within shared pathways, and combine both signals into a unified monitor of generalization.

3.1 CONSTRUCTING VIRTUAL EXPERTS FOR DENSE MODELS

Dense transformers do not contain native experts or routers. For each model family, we train a single CrossCoder on transformer-layer activations from multiple checkpoints, placing all checkpoints in a shared latent coordinate system. We then construct *virtual experts* from CrossCoder latents. Virtual experts are defined separately for each model and layer $\ell \in \{1, \ldots, L\}$. This preserves checkpoint-level comparability while avoiding artificial alignment across unrelated layers.

Within each (model, ℓ) pair, virtual experts are formed by clustering latents according to their co-activation patterns. With the CrossCoder router frozen, each latent k induces a binary firing pattern over calibration samples:

$$X_k = [b_{1,k}, \ldots, b_{S,k}] \in \{0,1\}^S,$$

where $b_{s,k} = 1$ if the router selects latent k at any valid token of sample s, and 0 otherwise. We retain latents with large decoder-vector norms to filter weak features, then apply k-means to the retained firing-pattern vectors. Each cluster defines one virtual expert:

$$\mathcal{C}_{\ell,e} = \{k : \text{cluster}(X_k) = e\}, \qquad e = 1, \ldots, E, \;\; E \ll S.$$

Thus, a virtual expert groups latent features that tend to fire on similar inputs. Each expert therefore represents a recurring co-activation pattern in CrossCoder latent space rather than a cluster based on activation magnitude alone. Given an input x_i at checkpoint θ_t, we aggregate its selected latents into expert-level usage weights. Let $m^{(t)}_{i,\ell,k}$ indicate whether latent k is selected for x_i at layer ℓ after aggregating binary router masks over valid, non-special tokens. The usage weight of virtual expert e is

$$r^{(t)}_{i,\ell,e} = \textstyle\sum_{k \in \mathcal{C}_{\ell,e}} \mathbf{1}\{m^{(t)}_{i,\ell,k} > 0\}.$$

Thus, $r^{(t)}_{i,\ell,e}$ counts the retained latent features from expert e selected for sample x_i at layer ℓ. The resulting matrix $r^{(t)}_i \in \mathbb{R}^{L \times E}$ defines a dense model's virtual routing profile.

3.2 PATHWAY COMPLEXITY

We convert each virtual routing profile into a discrete pathway and use it to estimate pathway complexity. For each layer ℓ, we select the top-K_{route} virtual experts according to $r^{(t)}_{i,\ell,e}$. This expert-selection budget is distinct from the CrossCoder's latent-selection top-k. Experts are sorted by decreasing usage weight, with ties broken by increasing expert id for determinism:

$$\mathcal{E}^{(t)}_{i,\ell} = \text{TopK}^{K_{\text{route}}}_e \left(r^{(t)}_{i,\ell,e} \right).$$

The virtual pathway of input x_i is the depth-ordered sequence of these layerwise expert lists:

$$\pi_t(x_i) = \left[\mathcal{E}^{(t)}_{i,1}, \mathcal{E}^{(t)}_{i,2}, \ldots, \mathcal{E}^{(t)}_{i,L} \right].$$

This pathway records the top-ranked virtual experts for the input at each layer, together with their within-layer order. To compare two pathways, we serialize $\pi_t(x_i)$ into a layer-structured string that preserves both within-layer order and cross-layer depth order. Each expert identifier is encoded as a single character. Within each layer, selected experts form a comma-separated list; layerwise lists are joined by hyphens in depth order. We compute raw, unnormalized character-level Levenshtein distance over the resulting strings, including separators. For samples x_i and x_j, their pathway distance is

$$D_{ij}^{\text{path}}(t) = d_{\text{edit}}(\pi_t(x_i), \pi_t(x_j)),$$

where d_{edit} denotes Levenshtein edit distance over serialized pathways. This distance is order-aware and captures changes in the serialized layerwise expert sequence.

For a domain $\mathcal{D}$, we define pathway complexity as the average pairwise pathway distance:

$$P_t(\mathcal{D}) = \mathbb{E}_{x_i, x_j \in \mathcal{D}} \left[D_{ij}^{\text{path}}(t) \right].$$

In experiments, we refer to $P_t(\mathcal{D})$ as pathway edit distance, or PED. High P_t indicates that samples use heterogeneous virtual routes, suggesting high pathway-level complexity. Low P_t indicates more similar virtual pathways across samples, suggesting stronger computation reuse.

3.3 Conditional State Diversity under Shared Pathways

Pathway reuse alone is ambiguous. A decrease in pathway complexity may indicate productive reuse, but it may also indicate collapse if the resulting representations become indistinguishable. We therefore measure whether samples with similar pathways remain distinct in representation space.

For each input x_i, we extract a sequence-level representation $z_i^{(t)}$ from the final transformer layer and compute pairwise representation distance:

$$D_{ij}^{\text{state}}(t) = d_{\text{repr}}\big(z_i^{(t)}, z_j^{(t)}\big),$$

where d_{repr} is cosine distance unless otherwise specified. Let $\tau_p^{(t)}(\mathcal{D})$ be the p-th percentile of pathway distances over distinct unordered pairs ($i < j$) in domain $\mathcal{D}$, including pairs at the threshold. We define conditional state diversity as

$$R_t(\mathcal{D}; p) = \frac{\mathbb{E}_{x_i, x_j \in \mathcal{D}} \left[D_{ij}^{\text{state}}(t) \mid D_{ij}^{\text{path}}(t) \le \tau_p^{(t)}(\mathcal{D}) \right]}{\mathbb{E}_{x_i, x_j \in \mathcal{D}} \left[D_{ij}^{\text{state}}(t) \right]}.$$

This ratio measures representation diversity among pathway-similar samples relative to the domain baseline. Normalizing by the domain-wide distance controls for checkpoint-level changes in the scale of representation dispersion, isolating diversity specifically within shared pathways. Higher R_t indicates flexible reuse: shared computation without erasing input-specific representational distinctions.

3.4 Adaptive Reuse Ratio

Pathway complexity and conditional state diversity capture complementary aspects of reusable computation. Decreasing P_t alone may reflect either useful reuse or collapse. High R_t alone does not penalize fragmented pathways. We therefore define the *Adaptive Reuse Ratio*:

$$A_t(\mathcal{D}; p) = \log \tfrac{R_t(\mathcal{D}; p) + \epsilon}{P_t(\mathcal{D}) + \epsilon}.$$

Higher A_t indicates greater conditional state diversity relative to pathway complexity. It therefore favors checkpoints that preserve richer state structure with fewer distinct pathways.

Interpreting computation organization. The joint behavior of P_t and R_t provides a qualitative map of how computation is organized during training. As shown in Figure 2, the desired regime is reusable computation: low pathway complexity with high conditional state diversity. This indicates that inputs share virtual pathways while preserving diverse internal states. In contrast, high P_t with high R_t suggests memorization-like computation, where diverse states rely on distinct pathways; low P_t with low R_t suggests collapse; and high P_t with low R_t suggests degenerate fragmentation.

Training transitions describe local movement in this plane. The desired transition is efficient compression, where P_t decreases while R_t increases. This moves the model toward reusable computation by reducing pathway-level complexity while increasing state diversity within shared pathways. Other directions correspond to over-compression, memorization-like expansion, or degenerate fragmentation. We use these terms only as qualitative descriptors of local metric trends; implementation details for slope-based transition detection are provided in Appendix C.4.

4 Experiments

We evaluate whether the virtual-pathway view provides an informative monitor of dense language model pretraining. Our motivation predicts two coupled structural signatures of improving generalization: increasing pathway reuse and preserved state diversity within shared pathways. We test these predictions by tracking the pathway complexity P_t, conditional state diversity R_t, and their combined Adaptive Reuse Ratio A_t across model families, scales, checkpoints, and data domains.

Models. We analyze public pretraining trajectories of dense decoder-only LLMs: Pythia-160M, -410M, and -1B (Biderman et al., 2023); OLMo-1B (Groeneveld et al., 2024) and OLMo2-1B (OLMo et al., 2024).

Virtual pathway construction. For each model, we train a single CrossCoder on transformer-layer activations from 10 checkpoints and use it to construct virtual experts as described in Section 3.1. This full-trajectory CrossCoder provides a fixed coordinate system across checkpoints, allowing us to test whether pathway statistics track generalization rather than changes in the proxy itself. For prospective monitoring, the proxy can instead be trained after an initial burn-in period, frozen, and applied to later checkpoints. Appendix D.2 shows that an early-only CrossCoder trained on the first four checkpoints transfers to six later checkpoints with only a 0.77% average reconstruction-loss gap relative to the full ten-checkpoint CrossCoder.

Evaluation domains. We analyze two types of domains. First, we use 20 held-out benchmarks as task-aligned domains spanning commonsense reasoning, question answering, reading comprehension, and language modeling. Because samples within each benchmark share an explicit downstream objective, this setting tests whether improved generalization is associated with greater pathway sharing among inputs requiring similar capabilities. Second, we analyze pretraining domains to study whether reusable computation emerges without explicit task labels. Pythia is trained on the Pile, a broad mixture of sources, while OLMo and OLMo2 use curated mixtures including web text, code, scientific papers, mathematical content, and encyclopedic text. Held-out benchmark details are provided in Appendix C.1, and pretraining-domain details are provided in Appendix C.2.

4.1 Pathway Reuse Tracks Generalization on Held-Out Benchmarks

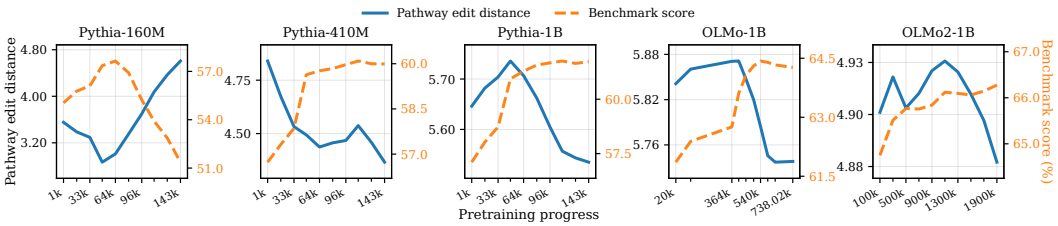


Figure 3: **Lower pathway complexity accompanies stronger downstream performance.** Across pretraining, higher held-out benchmark accuracy is generally associated with lower pathway complexity P_t, indicating greater pathway reuse among inputs from the same task domain.

We first ask whether generalization is accompanied by increased pathway reuse. For each held-out benchmark domain $\mathcal{D}$, we compute $P_t(\mathcal{D})$, the mean pairwise pathway edit distance among benchmark inputs at checkpoint t. Lower P_t indicates that samples are processed through more similar virtual pathways, reflecting stronger pathway-level reuse.

Figure 3 shows that improved downstream performance is associated with lower pathway complexity. This is consistent with the hypothesis that improved generalization is accompanied by a reorganization of internal computation toward shared pathways. However, the path toward reuse is scale-dependent.

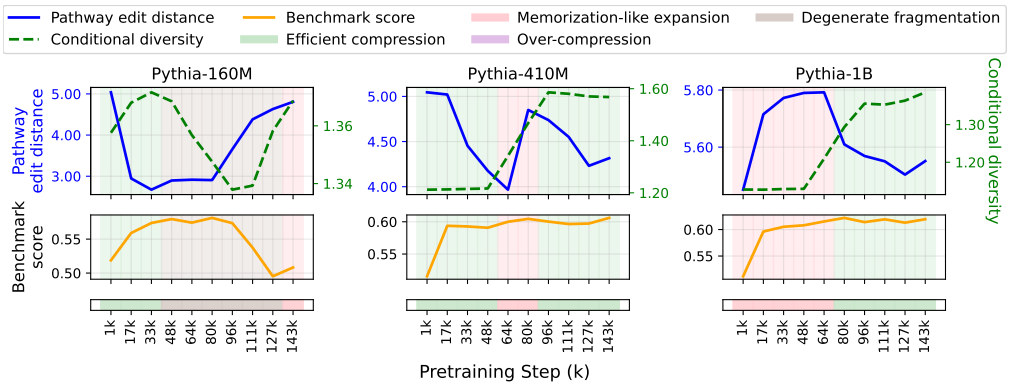


Figure 4: **Training phase transitions across model scales.** Larger benchmark gains coincide with intervals of efficient compression, characterized by decreasing P_t and increasing R_t.

Early compression. `Pythia-410M` shows a relatively direct trajectory: pathway complexity decreases as benchmark accuracy improves.

Expansion before compression. `Pythia-1B`, `OLMo-1B`, and `OLMo2-1B` exhibit a different pattern. Pathway complexity first increases as virtual routes become more differentiated, then decreases as performance continues to improve or reaches its peak. Within these evaluated trajectories, this pattern is consistent with expansion followed by compression.

Generalization failure with less pathway reuse. `Pythia-160M` reveals the opposite failure mode. After an initial phase where decreasing P_t aligns with improving accuracy, pathway complexity rises again while performance declines. This divergence suggests a loss of shared computational structure, with increasing pathway fragmentation coinciding with degraded generalization.

Together, these results show that high-performing checkpoints are characterized by stronger pathway reuse, but reuse need not emerge monotonically. High-performing trajectories can exhibit expansion followed by compression: computation first differentiates, then consolidates into shared pathways.

4.2 Efficient Compression Characterizes Scale-Dependent Training Transitions

The held-out results show that pathway reuse tracks generalization, but they do not determine whether reuse is productive or collapsed. We therefore examine the joint dynamics of pathway complexity P_t and conditional state diversity R_t. In our framework, reusable computation requires not only lower P_t but also preserved or increasing R_t among pathway-similar pairs. The desired transition is *efficient compression*: P_t decreases while R_t increases.

We first study this transition across the Pythia family. This isolates a scale axis: all three models are trained on the same Pile mixture, but differ in capacity. Figure 4 tracks P_t, R_t, and benchmark performance across checkpoints. Across scales, favorable performance dynamics tend to coincide with efficient compression. During these intervals, pathway complexity decreases while input-conditioned state diversity increases within shared computation. This is the empirical signature of "doing more with less."

The route to this transition changes qualitatively with scale. `Pythia-160M` exhibits early efficient compression, but this organization later breaks down: pathway distances increase again, conditional diversity fails to recover, and performance drops. The smallest model therefore exhibits only transient reuse in this trajectory, with pathway organization deteriorating later in training. `Pythia-410M` shows a more recoverable trajectory. After an initial compression phase, it undergoes memorization-like expansion, but later returns to efficient compression and reaches its best performance near the end of training. `Pythia-1B` follows the clearest expansion-compression pattern: its pathways first expand, then enter a sustained efficient-compression phase in which increasing reuse and R_t coincide with the most stable performance across the three scales.

Under a shared pretraining mixture, reusable computation emerges through scale-dependent training transitions. Smaller models may compress early but fail to sustain reuse, whereas larger models can first differentiate their computation and later consolidate it into shared pathways while preserving state diversity. These trajectories suggest that successful pretraining is better characterized by efficient compression than by monotonic pathway compression alone.

4.3 Reusable Computation Emerges Through Domain-Dependent Reorganization

The Pythia analysis reveals scale-dependent differences in how pretraining dynamics transition from computational expansion to efficient compression. We next ask whether the same reusable-computation principle holds across heterogeneous data sources. This probes a different axis: instead of comparing model scales under one mixture, we compare pretraining domains whose internal structure, style, and semantic regularity differ substantially.

For each pretraining domain $\mathcal{D}$, we track pathway complexity $P_t(\mathcal{D})$ together with conditional state diversity $R_t(\mathcal{D}; p)$. Across OLMo and OLMo2 domains, R_t generally increases even when P_t follows markedly different trajectories across data sources. Full domain-wise trajectories are provided in Appendix D.1. The preserved or increasing R_t suggests that these pathway dynamics do not simply reflect representational collapse. Instead, many domains preserve route-conditional state diversity: samples can share more virtual computation while remaining distinguishable in representation space. Conditioning on pathway similarity remains meaningful despite the heterogeneity of pretraining data. Because samples within a domain can differ substantially, pairwise PED values are naturally broad rather than concentrated at a single distance. Appendix D.5 verifies this empirically and shows that the lower tail of the PED distribution provides a stable notion of relative pathway similarity within each checkpoint. Thus, R_t compares representations among samples that are pathway-similar relative to the current domain and checkpoint, rather than assuming that all samples in a pretraining domain should follow the same route.

The path toward reusable computation is domain-dependent. Some domains exhibit relatively direct compression, with decreasing P_t and increasing R_t, whereas others follow expansion-compression trajectories. Across heterogeneous pretraining sources, reusable computation does not follow a single universal trajectory: pathway organization evolves differently across domains while preserving state diversity within shared pathways. This domain dependence motivates monitoring the joint organization of computation rather than expecting a single monotonic pathway statistic to characterize all pretraining data.

4.4 From Reusable Computation to Generalization Monitoring

Table 1: Pearson correlation with downstream accuracy. Background colors indicate nominal significance: $p < 0.1$, $p < 0.05$, $p < 0.01$, $p > 0.1$, $p > 0.5$.

Pythia			
Scale	P_t	R_t	A_t
160M	**-0.94**	-0.09	**0.93**
410M	-0.47	**0.68**	0.62
1B	-0.60	0.77	**0.89**
OLMo-1B			
Domain	P_t	R_t	A_t
Alg. Stack	**-0.84**	0.79	0.80
arXiv	-0.71	0.83	**0.83**
peS2o	**-0.89**	0.71	0.73
StarCoder	**-0.89**	0.82	0.84
OWM	0.10	**1.00**	0.82
DCLM	-0.38	0.71	**0.74**
Wiki	-0.51	0.71	**0.78**
OLMo2-1B			
Domain	P_t	R_t	A_t
Alg. Stack	**-0.90**	0.56	0.89
arXiv	-0.90	**0.98**	0.91
peS2o	-0.98	0.98	**0.98**
StarCoder	-0.97	0.59	**0.97**
OWM	-0.92	0.98	**0.99**
DCLM	**-1.00**	0.69	0.98
Wiki	-0.68	0.71	**0.74**

We next ask whether the structural signatures of reusable computation can provide a useful monitor of downstream generalization.

Adaptive Reuse Ratio. Table 1 compares how pathway complexity P_t, conditional state diversity R_t, and their combination A_t are associated with downstream accuracy. Pathway complexity and state diversity provide complementary signals: P_t captures computation reuse but cannot distinguish productive sharing from collapse, whereas R_t captures preserved within-pathway structure without penalizing fragmented computation. Their combination therefore distinguishes the target regime—shared computation with preserved state diversity—from either compression or diversity alone. A_t combines these complementary signals and shows consistently strong associations with downstream accuracy across model families and domains. It is strongest in two of three Pythia scales and remains strong across OLMo and OLMo2 domains, including settings where either P_t or R_t alone is less informative.

Predicting benchmark performance. Beyond retrospective association, we test whether pathway statistics from a checkpoint help predict its downstream benchmark performance. This provides a stronger monitoring test than trajectory-level correlation: the benchmark result for the checkpoint being evaluated is withheld from the predictor, while its pathway statistics remain available. On Pythia-410M, we combine pathway statistics from the checkpoint being evaluated with benchmark scores from earlier checkpoints to predict task-level performance changes. We evaluate the final four checkpoints across 20 tasks; prediction protocols are provided in Appendix C.5.

Table 2 shows that A_t achieves 72.9% direction accuracy, compared with 68.6% for PED and 65.7% for loss-based controls. MAE differ-

Table 2: **Benchmark-score prediction.** Higher direction accuracy and lower MAE are better.

Monitor	Direction acc.	MAE (pp)
A_t	72.9%	1.39
P_t / PED	68.6%	1.40
Loss-based controls	65.7%	1.42

Table 3: **Optimization-progress controls.** Residual Pearson correlations between PED and mean downstream performance.

Control	Residual r
$\log(\text{step})$	$+0.925$
Training loss	$+0.760$

ences are smaller (1.39, 1.40, and 1.42 pp, respectively), so the observed difference is larger for predicting the direction of benchmark-score changes than for their magnitude.

Optimization-progress controls. On a ten-checkpoint Pythia-160M analysis, PED remains associated with downstream accuracy after separately controlling for training loss ($r = +0.760$) or log training step ($r = +0.925$). Thus, the PED-accuracy association is not explained by either linear progress indicator alone. Table 3 summarizes the results, with protocol details in Appendix C.6.

Metric sensitivity. Conditional diversity preserves its qualitative training trajectory across alternative representation distances and pathway-similarity thresholds, although its correlation with downstream accuracy varies with the threshold. Full sensitivity results are provided in Appendices D.3.1 and D.3.2.

Alternative monitors. Pathway-based monitors are slightly more strongly correlated with downstream accuracy than conventional optimization signals such as training loss and perplexity (Appendix D.4.1). A representation-only linear CKA baseline shows substantially weaker association in the tested settings, with a maximum absolute correlation of 0.325 (Appendix D.4.2). Together, these comparisons suggest that pathway statistics provide structural information complementary to both optimization fit and global representation similarity.

5 Related Work

Monitoring pretraining dynamics. Pretraining is monitored using optimization metrics such as training loss and perplexity (Kaplan et al., 2020; Hoffmann et al., 2022). While effective for tracking convergence, these metrics provide limited insight into the emergence of generalization. Prior work on grokking and delayed generalization shows that models can achieve low training loss long before exhibiting generalization, highlighting a mismatch between optimization progress and functional capability (Power et al., 2022; Nanda et al., 2023). Alternative approaches based on representation analysis characterize how representations evolve across models or training stages (Raghu et al., 2017; Kornblith et al., 2019), but do not model the organization and reuse of computational pathways.

Routing and modular computation. Mixture-of-Experts models provide routing mechanisms that define computational pathways and enable specialization across inputs (Shazeer et al., 2017; Fedus et al., 2022). Prior work suggests that neural networks implicitly organize computation into reusable substructures, as evidenced by lottery ticket subnetworks and modular architectures (Frankle & Carbin, 2018; Kirsch et al., 2018). However, such structure is not observable in dense models.

We bridge these directions with a virtual pathway framework for dense models, enabling pathway-level analysis without architectural changes. Pathway reuse and representation diversity together yield a domain-agnostic signal that tracks generalization during pretraining.

6 Conclusion

We introduce a virtual-pathway framework for monitoring generalization during dense LLM pretraining. A learned routing proxy exposes checkpoint-comparable pathways, revealing how computation reorganizes over training. Across Pythia, OLMo, and OLMo2, improved generalization consistently coincides with *reusable computation*: inputs increasingly share pathways while preserving discriminative internal states. This points to *efficient compression*, reducing pathway complexity without collapsing input-conditioned representations, as a structural signature of successful training. The Adaptive Reuse Ratio captures this tradeoff, while pathway statistics combined with historical evaluations improve prediction of benchmark-score changes over loss-based controls. Overall, our results support "doing more with less" as a structural signature of generalization during dense LLM pretraining, providing a practical and mechanistically motivated complement to loss-based monitoring.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and writing polish. The authors reviewed all AI-assisted edits and take full responsibility for the accuracy and content of the paper.

REPRODUCIBILITY STATEMENT

We provide detailed theoretical derivations and experimental protocols in the appendices to support reproducibility of our results. All evaluated model checkpoints and benchmark datasets are publicly available.

REFERENCES

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2357–2367, 2019.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukherjee, and Alkis Gotovos. On the existence of universal lottery tickets. *arXiv preprint arXiv:2111.11146*, 2021.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. *arXiv preprint arXiv:2010.02066*, 2020.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stablemoe: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, 2022.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15789–15809, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance of large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Louis Kirsch, Julius Kunze, and David Barber. Modular networks: Learning to decompose neural computation. *Advances in neural information processing systems*, 31, 2018.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 785–794, 2017.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Ziyue Li, Chenrui Fan, and Tianyi Zhou. Grokking in llm pretraining? monitor memorization-to-generalization without test. *arXiv preprint arXiv:2506.21551*, 2025.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, pp. 3982–3992, 2024.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *International Conference on Machine Learning*, pp. 22188–22214. PMLR, 2023.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663, 2022.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2024. URL https://arxiv.org/abs/2501.00656.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pp. 90–95, 2011.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581, 2023.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Kaiser Sun and Mark Dredze. Amuro & char: Analyzing the relationship between pre-training and fine-tuning of large language models. In *Proceedings of the 10th Workshop on Representation Learning for NLP (RepL4NLP-2025)*, pp. 131–151, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 93–104, 2018.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A Additional Related Work

This section expands the discussion in Section 5, focusing on work most closely related to pretraining dynamics, MoE routing, and modular computation.

Monitoring pretraining dynamics. Pretraining is typically monitored using optimization metrics such as training loss and perplexity, or through periodic downstream evaluation (Kaplan et al., 2020; Hoffmann et al., 2022; Brown et al., 2020). While these metrics are effective for tracking convergence and scaling behavior, they provide limited insight into the internal computational changes associated with generalization.

Recent work on grokking and delayed generalization demonstrates that models can achieve low training loss long before exhibiting strong generalization, revealing a mismatch between optimization progress and functional capability (Power et al., 2022; Liu et al., 2022; Nanda et al., 2023). Similar phenomena have been observed in studies of double descent and training dynamics, further suggesting that loss alone is insufficient to characterize generalization (Belkin et al., 2019; Nakkiran et al., 2021).

Beyond loss-based metrics, several works have explored alternative signals based on internal representations and training dynamics. These include representation similarity measures (Raghu et al., 2017; Kornblith et al., 2019), probing-based analyses (Hewitt & Manning, 2019), and studies of feature evolution and emergence during training (Elhage et al., 2021; Olah et al., 2020). These approaches characterize representation similarity, feature structure, and training dynamics, but do not explicitly model the organization and reuse of computational pathways studied here.

Overall, despite extensive study of training dynamics, a consistent, interpretable, and domain-agnostic monitor of generalization remains an open problem.

Mixture-of-Experts and routing. In MoE models, routing decisions provide an explicit notion of computational pathways by dynamically selecting a subset of experts for each input (Shazeer et al., 2017; Fedus et al., 2022). Prior work shows that routing patterns can reflect both specialization and generalization behavior, as different experts learn to capture distinct structures in the data (Lepikhin et al., 2020; Fedus et al., 2022; Zoph et al., 2022).

Recent studies further investigate the role of routing design in training dynamics, including improved routing strategies (Zhou et al., 2022), stability and load balancing (Dai et al., 2022), and their impact on learning efficiency and generalization. These works suggest that routing structure plays a central role in organizing computation in sparse models.

However, dense models lack such explicit routing mechanisms, making it difficult to directly observe or analyze pathway-level computation. Our work bridges this gap by introducing a virtual routing perspective that enables pathway-level analysis in dense architectures without modifying the model.

Pathways and modular computation. Recent work suggests that neural networks may implicitly organize computation into reusable substructures or modules, even in the absence of explicit architectural constraints. The lottery ticket hypothesis shows that dense networks contain sparse subnetworks that can be trained in isolation while achieving comparable performance, indicating the existence of functionally meaningful substructures (Frankle & Carbin, 2018). Subsequent work further explores the role of such subnetworks in generalization and training dynamics (Burkholz et al., 2021).

Complementary lines of research study modularity more directly, showing that neural networks can decompose computation into specialized components that capture distinct functions (Kirsch et al., 2018; Csordás et al., 2020). These findings suggest that reusable computational structure emerges naturally during training and may play a key role in generalization.

Unlike prior work, we connect pathway reuse with representation diversity and show that their combination provides a cross-domain structural signal for monitoring pretraining dynamics.

B A CONDITIONAL GENERALIZATION BOUND FOR REUSABLE COMPUTATION

This appendix provides a learning-theoretic justification for the four-term view in Eq. 1. The four terms arise from distinct sources: pretraining fit, description complexity of the pathway system, downstream readout complexity induced by representation separability, and latent task-inference error. To keep the connection to standard generalization theory explicit, we define open-task representation separability through downstream classification margin. Pathway edit distance and conditional state diversity are introduced subsequently as empirical proxies for the corresponding theoretical quantities.

B.1 SETUP

Let $S_{\mathrm{pre}} = \{(x_i, y_i^{\mathrm{pre}})\}_{i=1}^n$ denote an i.i.d. pretraining sample. For simplicity, we state the bound for deterministic pathway selection:

$$\Pi = r_\theta(X), \qquad Z = f_\theta(X; \Pi),$$

where Π is the latent computational pathway selected for the input and Z is the resulting representation. The stochastic-routing case can be handled by conditioning on router randomness.

Let $\mathcal{H}_{\mathrm{natural}}$ denote a family of natural downstream tasks. A task $h \in \mathcal{H}_{\mathrm{natural}}$ induces a target

$$Y_h = h(X).$$

For a new downstream task, assume that m labeled examples are available to fit a simple readout g_h on top of Z. The task identity is not provided explicitly to the router and must be inferred from the input.

We assume that the losses entering the generalization analysis are bounded,

$$0 \le \ell(\hat{y}, y) \le 1,$$

and write $R_{\mathrm{open}}(\theta)$ for the expected downstream risk over the open task family.

B.2 FOUR SOURCES OF ERROR

We assume the following four-term upper decomposition of open-task risk:

$$R_{\mathrm{open}}(\theta) \le \mathcal{E}_{\mathrm{fit}} + \mathcal{E}_{\mathrm{path}} + \mathcal{E}_{\mathrm{readout}} + \mathcal{E}_{\mathrm{task}},$$

corresponding respectively to pretraining fit, pathway description complexity, downstream readout complexity, and latent task inference. We bound these four contributions separately below.

B.3 TERM I: PRETRAINING FIT

Let

$$R_{\mathrm{pre}}(\theta) = \mathbb{E}[\ell_{\mathrm{pre}}(\theta; X, Y^{\mathrm{pre}})]$$

denote the population pretraining risk.

Assumption 1 (Pretraining approximation)**.** *The residual approximation error inherited from pretraining is bounded, up to a universal constant absorbed into notation, by the pretraining risk:*

$$\mathcal{E}_{\mathrm{fit}} \le R_{\mathrm{pre}}(\theta).$$

This assumption states only that failure to model the pretraining distribution can contribute to downstream error. Low R_{pre} alone does not imply generalization: low pretraining error may coexist with sample-specific or otherwise non-reusable computation.

B.4 TERM II: PATHWAY DESCRIPTION COMPLEXITY

For the pretraining samples, let

$$\Pi_i = r_\theta(x_i), \qquad \Pi_{1:n} = (\Pi_1, \ldots, \Pi_n).$$

Condition on a fixed library $\mathcal{E}$ of reusable computational primitives (virtual experts in our construction). We describe the induced pathway system using a two-part prefix code:

$$\mathcal{L}_{\mathrm{path}}(\Pi_{1:n}) = L(r_\theta \mid \mathcal{E}) + \sum_{i=1}^{n} L(\Pi_i \mid r_\theta, \mathcal{E}) \,.$$

The first term describes the shared routing program, while the second captures the sample-specific description cost of the pathway family induced over the pretraining sample. In the ideal algorithmic-information limit, this corresponds to

$$\mathcal{L}_{\mathrm{path}}(\Pi_{1:n}) \simeq K(r_\theta \mid \mathcal{E}) + \sum_{i=1}^{n} K(\Pi_i \mid r_\theta, \mathcal{E}) \,,$$

where K denotes prefix Kolmogorov complexity.

Lemma 1 (Conditional Occam bound for pathway programs)**.** *Assume that the effective predictors belong to a countable, prefix-coded hypothesis class fixed independently of the realized pretraining sample, and that the predictor induced by $\Pi_{1:n}$ has code length $\mathcal{L}_{\mathrm{path}}(\Pi_{1:n})$. Then, for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\mathcal{E}_{\mathrm{path}} \le \sqrt{\frac{\mathcal{L}_{\mathrm{path}}(\Pi_{1:n}) \ln 2 + \ln(1/\delta)}{2n}}.$$

Proof. For a fixed description-coded predictor q, Hoeffding's inequality gives

$$\Pr\Big[R(q) - \widehat{R}_S(q) > \epsilon\Big] \le e^{-2n\epsilon^2}.$$

If $L(q)$ is a prefix-code length, Kraft's inequality gives

$$\sum_q 2^{-L(q)} \le 1.$$

Allocate failure probability

$$\delta_q = \delta\, 2^{-L(q)}.$$

Then $\sum_q \delta_q \le \delta$, so a union bound makes the guarantee simultaneous over all coded predictors. Solving

$$e^{-2n\epsilon_q^2} = \delta\, 2^{-L(q)}$$

gives

$$\epsilon_q = \sqrt{\frac{L(q) \ln 2 + \ln(1/\delta)}{2n}}.$$

Taking $L(q) = \mathcal{L}_{\mathrm{path}}(\Pi_{1:n})$ proves the result. □

Ignoring universal constants and the base of the logarithm,

$$\mathcal{E}_{\mathrm{path}} \lesssim \sqrt{\frac{\mathcal{L}_{\mathrm{path}}(\Pi_{1:n}) + \log(1/\delta)}{n}}.$$

This term formalizes the role of pathway reuse. Sample-specific pathways generally require a longer joint description, whereas reusable pathways can share a compact routing rule and computational primitives. Thus, greater pathway reuse can reduce $\mathcal{L}_{\mathrm{path}}(\Pi_{1:n})$ and the corresponding complexity penalty.

B.5 Term III: Margin-Based Open-Task Separability

We next condition on an appropriate pathway having been selected. For clarity, consider binary downstream tasks with linear readouts

$$g_h(z) = \mathrm{sign}(w_h^\top z);$$

the same argument extends to standard multiclass margin bounds. For this binary case, let $Y_h \in \{-1, +1\}$.

Assume that representations are uniformly bounded:

$$\|Z\|_2 \le B.$$

For task h and pathway π, define the best population margin available to the readout class:

$$\gamma_h(\pi) = \sup_{\|w\|_2 \le 1} \inf_{(X,Y_h)|\Pi=\pi} Y_h \, w^\top Z.$$

Definition 1 (Open-task representation separability)**.** *For compatible task–pathway pairs, define*

$$\mathcal{D}_{\text{open}}(Z \mid \Pi) = \inf_{\pi \in \text{supp}(\Pi)} \inf_{h \in \mathcal{H}_{\text{natural}}(\pi)} \frac{\gamma_h^2(\pi)}{B^2}.$$

The normalization by B^2 makes $\mathcal{D}_{\text{open}}$ invariant to a global rescaling of the representation.

Lemma 2 (Readout generalization)**.** *Suppose m i.i.d. labeled examples are available for a compatible downstream task h, and the readout class contains a separator with margin $\gamma_h(\pi) > 0$. A standard linear margin bound gives, with high probability,*

$$R_h(g_h \mid \pi) \le \widehat{R}_h^{\gamma}(g_h \mid \pi) + C \frac{B}{\gamma_h(\pi)\sqrt{m}} + O\left(\sqrt{\frac{\log(1/\delta_h)}{m}}\right),$$

for a universal constant C.

Under the simplifying assumption of zero empirical margin loss for compatible downstream tasks, Definition 1 gives

$$\frac{B}{\gamma_h(\pi)} \le \frac{1}{\sqrt{\mathcal{D}_{\text{open}}(Z \mid \Pi)}}.$$

Suppressing standard confidence terms,

$$\mathcal{E}_{\text{readout}} \lesssim \frac{C_2}{\sqrt{m\, \mathcal{D}_{\text{open}}(Z \mid \Pi)}}.$$

Thus, greater task-relevant representation separability corresponds to a larger downstream margin and a smaller readout-complexity penalty. This definition avoids assuming a direct theoretical equivalence between the task-agnostic conditional state diversity metric R_t and downstream separability; their connection is discussed below through an explicit task-alignment assumption and a Fisher-style geometric proxy.

B.6 Term IV: Task-Inference Regret

The previous term assumes that the selected pathway is compatible with the intended task. A generalist model must additionally infer an appropriate computation from the input.

Let $R^*(x, h; \pi)$ denote the minimum achievable conditional risk for task h after using pathway π, and define

$$\pi^*(x, h) \in \arg\min_{\pi} R^*(x, h; \pi).$$

The model instead selects

$$\Pi = r_\theta(X).$$

Definition 2 (Task-inference regret)**.**

$$\mathcal{E}_{\text{task}} = \mathbb{E}_{X,h}\left[R^*(X, h; \Pi) - R^*(X, h; \pi^*(X, h))\right].$$

This term follows from the exact add-and-subtract decomposition

$$R^*(X, h; \Pi) = R^*(X, h; \pi^*(X, h)) + \left[R^*(X, h; \Pi) - R^*(X, h; \pi^*(X, h))\right].$$

Taking expectations isolates the additional risk due to pathway mismatch as $\mathcal{E}_{\text{task}}$.

B.7 Combined Bound

Theorem 1 (Four-term open-task generalization bound)**.** *Assume:*

- *(A1) open-task risk admits the four-term upper decomposition above;*
- *(A2) the residual pretraining error satisfies $\mathcal{E}_{\mathrm{fit}} \le R_{\mathrm{pre}}(\theta)$;*
- *(A3) the induced effective predictor belongs to a sample-independent prefix-coded hypothesis class and has pathway description length $\mathcal{L}_{\mathrm{path}}(\Pi_{1:n})$;*
- *(A4) the relevant losses are bounded and the pretraining sample is i.i.d.;*
- *(A5) compatible downstream tasks admit bounded-norm linear readouts with positive margin, zero empirical margin loss, and $\mathcal{D}_{\mathrm{open}}(Z \mid \Pi) > 0$;*
- *(A6) task mismatch is measured by Definition 2.*

Then, up to universal constants and standard confidence terms, with probability at least $1-\delta$,

$$R_{\mathrm{open}}(\theta) \lesssim R_{\mathrm{pre}}(\theta) + C_1\sqrt{\frac{\mathcal{L}_{\mathrm{path}}(\Pi_{1:n}) + \log(1/\delta)}{n}} + \frac{C_2}{\sqrt{m\,\mathcal{D}_{\mathrm{open}}(Z \mid \Pi)}} + \mathcal{E}_{\mathrm{task}}.$$

Proof. The pretraining approximation assumption gives

$$\mathcal{E}_{\mathrm{fit}} \le R_{\mathrm{pre}}(\theta).$$

Lemma 1 gives

$$\mathcal{E}_{\mathrm{path}} \lesssim \sqrt{\frac{\mathcal{L}_{\mathrm{path}}(\Pi_{1:n}) + \log(1/\delta)}{n}}.$$

Lemma 2, together with Definition 1, gives

$$\mathcal{E}_{\mathrm{readout}} \lesssim \frac{1}{\sqrt{m\,\mathcal{D}_{\mathrm{open}}(Z \mid \Pi)}}.$$

Finally, Definition 2 measures the excess risk due to selecting a nonoptimal pathway. Combining the four contributions yields the stated bound. □

B.8 Why Edit Distance Is a Proxy for Pathway Description Length

The pathway description length in the bound is not directly observable in dense LLMs. We therefore use pathway edit distance as an empirical proxy for the compressibility of the induced pathway family. For a domain $\mathcal{D}$, we define

$$P_t(\mathcal{D}) = \mathbb{E}_{x_i, x_j \in \mathcal{D}}\left[D_{ij}^{\mathrm{path}}(t)\right],$$

where $D_{ij}^{\mathrm{path}}(t)$ is the edit distance between the virtual pathways of x_i and x_j at checkpoint t.

The connection to $\mathcal{L}_{\mathrm{path}}(\Pi_{1:n})$ is through a prototype-plus-edit code, which can be made prefix-free by using a self-delimiting encoding of the prototype and edit sequence. Suppose the pathways in a domain can be represented by a prototype pathway $\Pi_{\mathcal{D}}^*$ and edit operations from that prototype:

$$\Pi_i = \Pi_{\mathcal{D}}^* + \mathrm{edits}_i.$$

If each edit sequence admits a code whose length is bounded by $c\,d_{\mathrm{edit}}(\Pi_i, \Pi_{\mathcal{D}}^*)$ apart from delimiting overhead, then

$$\mathcal{L}_{\mathrm{path}}(\Pi_{1:n_{\mathcal{D}}}) \le L(\Pi_{\mathcal{D}}^*) + c\sum_{i=1}^{n_{\mathcal{D}}} d_{\mathrm{edit}}(\Pi_i, \Pi_{\mathcal{D}}^*) + L_{\mathrm{overhead}},$$

where L_{overhead} denotes the cost of delimiting the prototype and edit sequences. For a metric edit distance, choosing an appropriate medoid among the observed pathways gives a prototype whose average distance is controlled by the average pairwise distance. Thus, lower average pairwise edit distance indicates the existence of a shorter prototype-plus-edit description under this coding scheme.

This does not mean that $P_t(\mathcal{D})$ equals the pathway description length. Rather, $P_t(\mathcal{D})$ is a computable proxy for the edit-cost component of a prototype-plus-edit upper bound on pathway description length. A decrease in $P_t(\mathcal{D})$ suggests that pathways within the domain have become more reusable and more compressible.

B.9 Conditional State Diversity as a Separability Proxy

Pathway reuse alone is insufficient for generalization. If samples share a pathway but their representations collapse, then downstream tasks cannot recover the distinctions needed for correct prediction. This motivates the separability term $\mathcal{D}_{\text{open}}(Z \mid \Pi)$ in Eq. 1.

Our empirical metric $R_t(\mathcal{D}; p)$ does not directly observe downstream labels for pretraining domains. Instead, it measures whether samples with similar pathways remain separated in representation space relative to the domain baseline:

$$R_t(\mathcal{D}; p) = \frac{\mathbb{E}\left[D_{ij}^{\text{state}}(t) \mid D_{ij}^{\text{path}}(t) \le \tau_p^{(t)}(\mathcal{D})\right]}{\mathbb{E}\left[D_{ij}^{\text{state}}(t)\right]}.$$

Here $D_{ij}^{\text{state}}(t)$ is a representation-space distance, such as cosine distance, and $\tau_p^{(t)}(\mathcal{D})$ is the p-th percentile of pairwise pathway distances within domain $\mathcal{D}$ at checkpoint t.

A larger $R_t(\mathcal{D}; p)$ means that pathway-similar samples remain diverse in representation space rather than collapsing to a common representation. The theoretical quantity $\mathcal{D}_{\text{open}}$, however, is defined through downstream classification margin, whereas R_t is task-agnostic and does not use downstream labels.

To make this proxy relation explicit, consider a compatible binary downstream task h within pathway π. Let

$$\mu_y^{(\pi,h)} = \mathbb{E}[Z \mid Y_h = y, \Pi = \pi],$$

and let $\Sigma_w^{(\pi,h)}$ denote the corresponding within-class covariance. A Fisher-style separability measure is

$$F_h(\pi) = \left(\mu_+^{(\pi,h)} - \mu_-^{(\pi,h)}\right)^\top \left(\Sigma_w^{(\pi,h)} + \lambda I\right)^{-1} \left(\mu_+^{(\pi,h)} - \mu_-^{(\pi,h)}\right).$$

Under an additional geometric alignment assumption relating class-conditional Fisher separation to the achievable population margin, greater Fisher separation corresponds to a larger margin. Schematically,

$$F_h(\pi) \uparrow \quad \Longrightarrow \quad \gamma_h^2(\pi) \uparrow,$$

which, through Definition 1, reduces the readout-complexity term in the bound.

Our empirical metric does not measure $F_h(\pi)$ directly. For cosine distance, let

$$\bar{z}_i = \frac{z_i}{\|z_i\|_2}.$$

Then

$$1 - \cos(z_i, z_j) = \frac{1}{2}\|\bar{z}_i - \bar{z}_j\|_2^2,$$

so larger conditional cosine distance indicates greater normalized representation spread among samples following similar pathways. Within a fixed pathway π and task h, the representation covariance admits the standard between-/within-class decomposition

$$\Sigma_Z^{(\pi,h)} = \Sigma_b^{(\pi,h)} + \Sigma_w^{(\pi,h)}.$$

Consequently, increased conditional state diversity alone does not imply greater downstream separability: the additional variation could instead reflect task-irrelevant within-class noise.

We therefore require a task-alignment assumption: during successful pretraining, increases in the normalized pathway-conditioned diversity captured by R_t predominantly reflect preserved or enlarged task-relevant between-class structure rather than merely increased within-class noise. Under this assumption, the intended proxy relation is

$$R_t \uparrow \quad \Longrightarrow \quad F_h(\pi) \uparrow \quad \Longrightarrow \quad \gamma_h^2(\pi) \uparrow \quad \Longrightarrow \quad \mathcal{E}_{\text{readout}} \downarrow .$$

Thus, conditional state diversity is not identified with the theoretical margin. Rather, it serves as an observable, task-agnostic proxy for the task-aligned representation separability that controls the third term of Eq. 1.

Table 4: Held-out benchmarks and links to their corresponding Hugging Face dataset pages.

Benchmark	HF Dataset ID	Link
BLiMP (SVA subset)	nyu-mll/blimp	https://huggingface.co/datasets/nyu-mll/blimp
COPA	aps/super_glue (copa)	https://huggingface.co/datasets/aps/super_glue
Winogrande	allenai/winogrande	https://huggingface.co/datasets/allenai/winogrande
PIQA	ybisk/piqa	https://huggingface.co/datasets/ybisk/piqa
HellaSwag	Rowan/hellaswag	https://huggingface.co/datasets/Rowan/hellaswag
CommonsenseQA	tau/commonsense_qa	https://huggingface.co/datasets/tau/commonsense_qa
ARC-Easy/-Challenge	allenai/ai2_arc	https://huggingface.co/datasets/allenai/ai2_arc
LAMBADA	EleutherAI/lambada_openai	https://huggingface.co/datasets/EleutherAI/lambada_openai
BoolQ	aps/super_glue (boolq)	https://huggingface.co/datasets/aps/super_glue
SciQ	allenai/sciq	https://huggingface.co/datasets/allenai/sciq
OpenBookQA	allenai/openbookqa	https://huggingface.co/datasets/allenai/openbookqa
PubMedQA	qiaojin/PubMedQA	https://huggingface.co/datasets/qiaojin/PubMedQA
MathQA	allenai/math_qa	https://huggingface.co/datasets/allenai/math_qa
LogiQA	lucasmccabe/logiqa	https://huggingface.co/datasets/lucasmccabe/logiqa
Social IQA	allenai/social_i_qa	https://huggingface.co/datasets/allenai/social_i_qa
MMLU	cais/mmlu	https://huggingface.co/datasets/cais/mmlu
TriviaQA	mandarjoshi/trivia_qa	https://huggingface.co/datasets/mandarjoshi/trivia_qa
RACE	ehovy/race	https://huggingface.co/datasets/ehovy/race
SWAG	allenai/swag	https://huggingface.co/datasets/allenai/swag

C Experimental Setup and Evaluation Protocols

C.1 Held-Out Evaluation Suite

Our held-out evaluation suite spans BLiMP (Warstadt et al., 2020), COPA (Roemmele et al., 2011), Winogrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), CommonsenseQA (Talmor et al., 2019), ARC (easy/challenge) (Clark et al., 2018), LAMBADA (Paperno et al., 2016), BoolQ (Clark et al., 2019), SciQ (Welbl et al., 2017), OpenBookQA (Mihaylov et al., 2018), PubMedQA (Jin et al., 2019), MathQA (Amini et al., 2019), LogiQA (Liu et al., 2020), Social IQA (Sap et al., 2019), MMLU (Hendrycks et al., 2020), TriviaQA (Joshi et al., 2017), RACE (Lai et al., 2017), and SWAG (Zellers et al., 2018). Table 4 lists the corresponding Hugging Face dataset pages used in our pipeline.

The suite contains 20 task scores, counting ARC-Easy and ARC-Challenge separately. The benchmark-score series in Figure 3 is the unweighted mean of these 20 task scores at each checkpoint.

C.2 Pretraining Domains

In addition to held-out benchmarks, we analyze pretraining-domain samples to study how reusable computation emerges within the model's original training distribution. Unlike held-out benchmarks, which provide task-aligned groups with explicit downstream objectives, pretraining domains reflect naturally occurring data sources with different styles, structures, and semantic regularities. This

allows us to test whether pathway reuse and conditional state diversity arise beyond curated evaluation tasks.

For `Pythia`, we sample from the Pile mixture (Gao et al., 2020), which contains diverse web, academic, code, and reference sources. Because the public Pythia checkpoints are trained on the full Pile mixture, we primarily use these samples to analyze scale-dependent dynamics under a shared pretraining distribution.

For `OLMo-1B` and `OLMo2-1B`, we analyze domain-labeled samples from their corresponding pretraining mixtures. The domains include scientific text, code, web text, encyclopedic text, and mathematical or structured sources, such as `arXiv`, `StarCoder`, `peS2o`, `DCLM`, `OWM`, `Wiki`, and `Algebraic Stack`. These heterogeneous sources allow us to test whether reusable computation follows a common trajectory or reorganizes differently across pretraining domains.

For each pretraining domain $\mathcal{D}$ and checkpoint t, we compute pathway complexity $P_t(\mathcal{D})$, conditional state diversity $R_t(\mathcal{D}; p)$, and the Adaptive Reuse Ratio $A_t(\mathcal{D}; p)$ using the same definitions as in Section 3. All samples are processed without downstream labels or task-specific prompting. Thus, the pretraining-domain analysis evaluates whether the proposed pathway-based signals capture intrinsic organization in the pretrained model, rather than artifacts of benchmark formatting or supervised task structure.

C.3 CrossCoder Training

Overview. We train a CrossCoder on cached transformer-layer activations extracted from $K = 10$ base-model checkpoints. For Pythia, we collect post-block residual-stream activations (`hook_resid_post`). For OLMo and OLMo2, we use the layer-indexed hidden states returned by the model implementation. Training follows a six-stage procedure that transitions from dense reconstruction to sparse routing and finally to a stabilized routing regime. The stage-specific settings are summarized below and in Table 5.

Unless otherwise specified, optimization uses Adam with $(\beta_1, \beta_2) = (0.9, 0.999)$ and learning rate 10^{-4}, with 10% linear warmup and 60% decay. We use bfloat16 precision for CrossCoder parameters and normalize reconstruction loss using statistics computed over 10^5 tokens. Training uses sequence length 48 and micro-batch size 8. All experiments use text from the Pile (uncopyrighted subset), with activations cached in float16 without compression.

Multi-checkpoint inputs. For each model family, we construct CrossCoder inputs from $K = 10$ checkpoints sampled at uniform intervals from the ordered list of released checkpoints. This exposes the CrossCoder to activations from different stages of pretraining, yielding a shared latent space that captures temporally diverse training dynamics.

Model size and routing configuration. The CrossCoder dictionary size is set to $6 \times d_{\text{model}}$, and routing selects approximately 1% of features via top-k selection. Concretely, we use $(4608, k{=}48)$ for Pythia-160M, $(6144, k{=}64)$ for Pythia-410M, and $(12288, k{=}128)$ for Pythia-1B, OLMo-1B, and OLMo2-1B.

Training procedure. Training proceeds in six stages as summarized in Table 5:

- **Stage 1 (dense baseline):** ReLU activations without sparsity regularization.
- **Stage 2 (sparse ReLU):** L1 regularization is introduced with a final weight of 5×10^{-4} and linearly increased over $80{,}000$ steps.
- **Stage 3 (routing initialization):** hard top-k routing is enabled using a straight-through estimator.
- **Stage 4-5 (routing refinement):** leaky ReLU activations are applied with slopes 0.1 and 0.01, respectively.
- **Stage 6 (stabilization):** the router is frozen and training continues with a reduced learning rate (3×10^{-5}).

Stages 1-2 run for $30{,}001$ steps, while later stages run for 2001 steps.

Table 5: Summary of CrossCoder training stages. The procedure transitions from dense reconstruction to sparse routing and finally to a stabilized routing regime.

Stage	Description	Key settings
1	Dense baseline	ReLU, no sparsity
2	Sparse encoding	L1 regularization (5×10^{-4})
3	Routing initialization	Hard top-k routing
4	Routing refinement	Leaky ReLU (slope 0.1)
5	Further refinement	Leaky ReLU (slope 0.01)
6	Stabilization	Frozen router, reduced LR

Implementation details. Activation caches are capped at 100-300 GB depending on model size. Distributed training uses two processes per node for larger models to maintain consistent checkpoint sharding. All experiments use deterministic checkpointing for reproducibility.

C.4 Phase Detection Protocol

We assign descriptive phase labels from local changes in pathway complexity P_t and conditional state diversity R_t. These labels summarize trajectory patterns for visualization and do not identify distinct causal training mechanisms.

Given a trajectory, we first work in log-space to capture relative changes:

$$p_t = \log P_t, \qquad r_t = \log R_t.$$

We then apply a centered moving average with window size w:

$$\bar{p}_t = \mathrm{MA}_w(p_t), \qquad \bar{r}_t = \mathrm{MA}_w(r_t),$$

and estimate local slopes using finite differences:

$$s_t^P = \bar{p}_{t+1} - \bar{p}_t, \qquad s_t^R = \bar{r}_{t+1} - \bar{r}_t.$$

To ensure comparable magnitudes, we normalize each slope using a robust global scale estimate:

$$\tilde{s}_t^P = \frac{s_t^P}{\sigma_{\mathrm{P}} + \epsilon}, \qquad \tilde{s}_t^R = \frac{s_t^R}{\sigma_R + \epsilon},$$

where σ_{P} and σ_R are computed from the corresponding slope distributions.

Lower P_t indicates stronger pathway reuse, while higher R_t indicates greater conditional state diversity. Based on the signs of the normalized slopes, we define four directional scores:

$$S_{\mathrm{eff}}(t) = \mathrm{ReLU}(-\tilde{s}_t^P)\mathrm{ReLU}(\tilde{s}_t^R),$$
$$S_{\mathrm{over-compression}}(t) = \mathrm{ReLU}(-\tilde{s}_t^P)\mathrm{ReLU}(-\tilde{s}_t^R),$$
$$S_{\mathrm{mem}}(t) = \mathrm{ReLU}(\tilde{s}_t^P)\mathrm{ReLU}(\tilde{s}_t^R),$$
$$S_{\mathrm{deg}}(t) = \mathrm{ReLU}(\tilde{s}_t^P)\mathrm{ReLU}(-\tilde{s}_t^R),$$

where $\mathrm{ReLU}(x) = \max(x, 0)$.

These scores correspond to efficient compression, over-compression, memorization-like expansion, and degenerate dynamics, respectively. We normalize them into phase weights:

$$W_k(t) = \frac{S_k(t)}{\sum_j S_j(t) + \epsilon},$$

and define phase strength as $\sum_j S_j(t)$.

Each checkpoint receives the highest-scoring label when both the total strength and maximum phase weight exceed fixed thresholds; otherwise, it is labeled as plateau or transition. In Figure 4, we use $w = 4$ to smooth local variation while retaining the broad trajectory changes used for visualization.

C.5 Benchmark Prediction Protocol

The Pythia-410M prediction experiment tests whether pathway statistics from a pretrained checkpoint can help predict its benchmark performance. We use the same ten pretrained checkpoints and 20-task evaluation suite as in the main analysis. Pathway features are computed from the pretrained checkpoint using a fixed CrossCoder trained across all ten checkpoints. At each prediction point, that checkpoint's benchmark score is withheld from the predictor.

For pathway extraction, both texts from each available benchmark pair are pooled without their correct/incorrect identities. Benchmark labels determine the paired evaluation examples, but these identities are removed before pathway extraction, and the current-checkpoint benchmark score is never used as a predictor input.

Let m_t denote a pathway monitor computed from pretrained checkpoint t, and let $a_{d,t}$ denote its benchmark score on downstream task d. We define

$$\Delta m_t = m_t - m_{t-1}, \qquad \Delta a_{d,t} = a_{d,t} - a_{d,t-1}.$$

The nine-dimensional input contains the current monitor change together with four preceding monitor and benchmark-score changes:

$$\mathbf{x}_{d,t} = [\Delta m_t, \Delta m_{t-4}, \ldots, \Delta m_{t-1}, \Delta a_{d,t-4}, \ldots, \Delta a_{d,t-1}].$$

The training set expands with t, pooling all earlier task-checkpoint pairs with a complete four-transition history and finite inputs. Changes with $|\Delta a_{d,j}| \leq 10^{-12}$ are excluded when fitting the direction target. For each monitor and prediction point, we standardize features using the training-set mean and standard deviation and fit ridge regression to $\mathrm{sign}(\Delta a_{d,j})$, leaving the intercept unpenalized. All monitors use $\alpha = 10$.

Let f_t denote the fitted direction score and s_t the mean absolute accuracy change among its training targets. The accuracy-change estimate is

$$\widehat{\Delta a}_{d,t} = s_t \tanh(f_t(\mathbf{x}_{d,t}))\,.$$

Each prediction assumes the required earlier benchmark scores are available; it does not recursively replace them with previous predictions. The current score $a_{d,t}$ is used only to evaluate that prediction, not to fit or construct its inputs.

Prediction metrics. Direction accuracy is the fraction of predictions with the correct sign, excluding pairs where the observed or predicted change has magnitude below 10^{-12}. Mean absolute error (MAE), in percentage points, averages absolute error.

Evaluation scope. Task-checkpoint predictions share tasks and training history, so the 80 predictions should not be interpreted as independent runs. The experiment tests whether pathway statistics computed from a pretrained checkpoint provide information about its downstream benchmark performance under a fixed trajectory-level CrossCoder. It does not evaluate forecasting of unseen future checkpoints or establish a causal effect of pathway organization on downstream performance.

C.6 Optimization Progress Control Protocol

This supporting analysis uses ten checkpoints from one Pythia-160M training trajectory. PED is computed with mean token pooling over full paths from the saved partial-domain output, and the accuracy target is the 20-task macro-average. This protocol differs from the selected-layer, last-token Pythia-160M analysis in Figure 3.

We control separately for training loss and $\log(\mathrm{step})$. For each control, we regress PED and accuracy separately on the control variable and compute the Pearson correlation between the resulting residual series. Step denotes the actual training step, and training loss is the token-averaged NLL from the conventional-metric evaluation.

C.7 Representation-Similarity Baseline Protocol

We compute linear centered kernel alignment (CKA) (Kornblith et al., 2019) on a fixed set of 512 texts across the original ten Pythia-160M checkpoints. Each input is truncated to at most 128 tokens, and hidden states are mean-pooled over non-padding tokens. We compare each checkpoint's representations with the same texts at either the earliest or final checkpoint.

D SUPPORTING EXPERIMENTAL RESULTS

D.1 DOMAIN-WISE PATHWAY DYNAMICS

Figure 5 and Figure 6 show the full domain-wise trajectories underlying the analysis in Section 4.3. Across domains, pathway complexity P_t follows heterogeneous trajectories, while conditional state diversity R_t generally increases over training. Some domains exhibit relatively direct compression, whereas others undergo expansion followed by consolidation. Thus, the observed transition toward reusable computation follows different trajectories across domains rather than a single universal compression path.

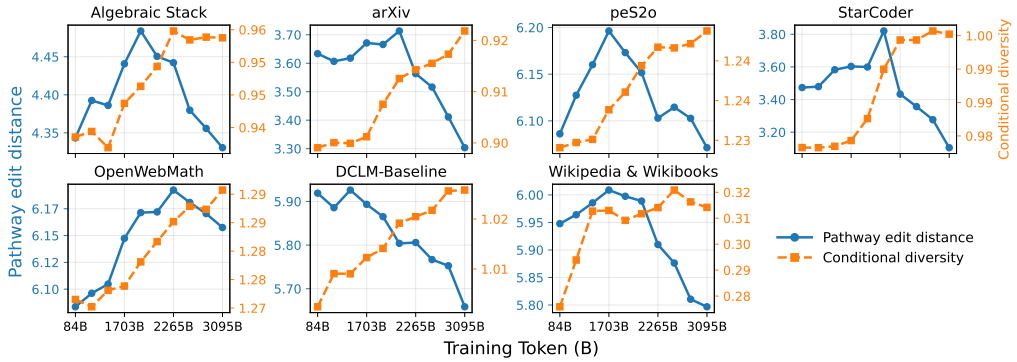


Figure 5: **Domain-wise pathway complexity and conditional state diversity for `OLMo2-1B`.** Trajectories show pathway complexity P_t and conditional state diversity R_t across pretraining domains.

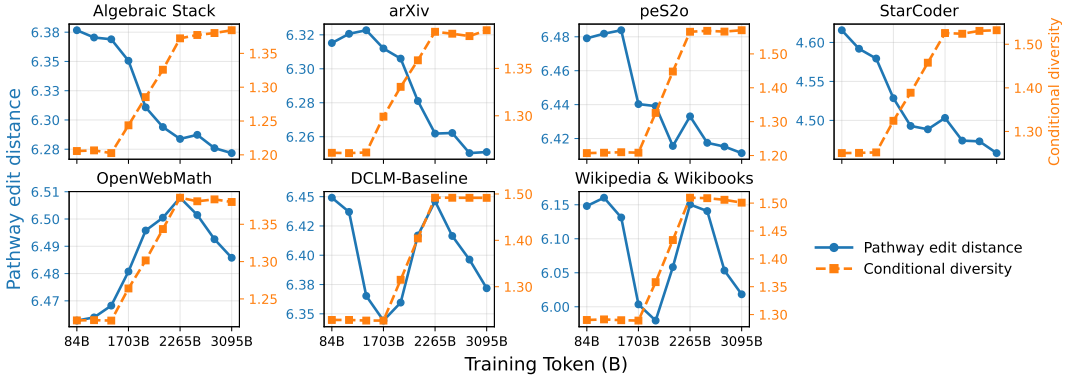


Figure 6: **Domain-wise pathway complexity and conditional state diversity for `OLMo-1B`.** Trajectories show pathway complexity P_t (left axis; pathway edit distance) and conditional state diversity R_t (right axis) across pretraining domains.

D.2 EARLY-CHECKPOINT CROSSCODER GENERALIZATION

The main experiments train a single CrossCoder over the full set of saved checkpoints, providing a stable longitudinal coordinate system for analyzing completed training trajectories. A practical question is whether this proxy must be trained with access to all future checkpoints, or whether a proxy learned from an initial burn-in period can be frozen and applied prospectively.

To evaluate this, we conduct an early-checkpoint ablation on `Pythia-410M`. We train CrossCoders using only the first $N \in \{2, 4, 6\}$ checkpoints and evaluate them on six later checkpoints through a zero-shot *swap-last-slot* protocol: for each future checkpoint, we measure reconstruction quality without retraining. As a reference, we compare against a Full-10 CrossCoder trained on all ten checkpoints and evaluated at the corresponding future model checkpoint.

Table 6 reports future-checkpoint reconstruction NMSE, and Figure 7 shows the gap relative to Full-10. Early-2 degrades substantially, with an average relative gap of $+19.83\%$, whereas Early-4

Table 6: Future-checkpoint reconstruction for early-only CrossCoders trained on the first $N \in \{2, 4, 6\}$ checkpoints versus a full-checkpoint CrossCoder. Each early-N model is evaluated on later checkpoints using swap-last-slot reconstruction. Full-10 is trained on all ten checkpoints and evaluated at the corresponding future model index. "Rel." is the relative gap to Full-10, $(\text{NMSE}_{\text{Early-}N} - \text{NMSE}_{\text{Full-10}})/\text{NMSE}_{\text{Full-10}}$. Lower NMSE / smaller positive Rel. is better; negative Rel. means the early-only model beats Full-10.

	Early-2		Early-4		Early-6		Full-10
Revision	NMSE	Rel.	NMSE	Rel.	NMSE	Rel.	NMSE
step64000	0.5457	+10.8%	0.5115	+3.8%	0.4723	−4.1%	0.4927
step80000	0.5713	+8.2%	0.4747	−10.1%	0.4679	−11.4%	0.5281
step96000	0.6143	+13.3%	0.5085	−6.2%	0.4915	−9.4%	0.5423
step111000	0.6776	+29.0%	0.5593	+6.5%	0.5276	+0.5%	0.5252
step127000	0.6826	+28.6%	0.5596	+5.4%	0.5238	−1.3%	0.5307
step143000	0.6927	+28.6%	0.5686	+5.5%	0.5292	−1.8%	0.5388
Average	0.6307	+19.83%	0.5304	+0.77%	0.5020	−4.61%	0.5263

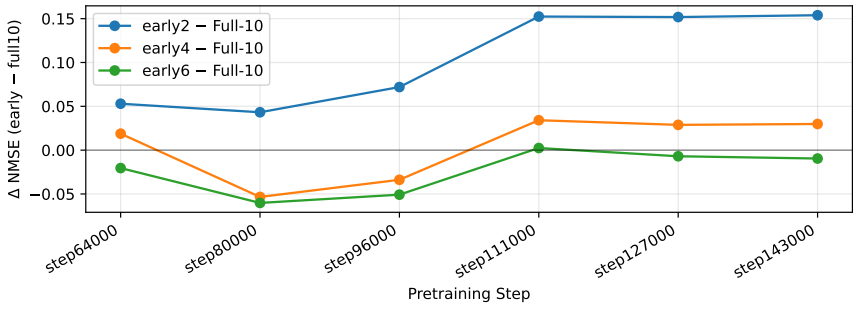


Figure 7: **Early-trained CrossCoder reconstruction transfers to later checkpoints.** We compare CrossCoders trained on early checkpoints (Early-2/4/6) with a Full-10 CrossCoder and report NMSE differences on held-out later checkpoints. The smaller gaps for Early-4 and Early-6 indicate substantially better reconstruction-level transfer than Early-2.

closely matches Full-10 (+0.77%). Early-6 obtains slightly lower NMSE on average (−4.61%), which may reflect optimization-level variation rather than a uniformly better routing proxy.

These results support reconstruction-level transfer after a modest burn-in: Early-4 and Early-6 provide a substantially more stable reconstruction coordinate system for later checkpoints than Early-2. This does not establish that pathway statistics are identical across independently trained CrossCoders, whose decoder bases need not align feature-by-feature.

D.3 Metric Sensitivity

D.3.1 Representation Distance

We evaluate whether conditional diversity R_t depends on the choice of representation distance metric. On `Pythia-1B`, we compute R_t using three distance functions: cosine distance, Euclidean distance, and centered cosine distance. The pathway edit distance is kept fixed, and only the representation distance used in the numerator and denominator of R_t is varied.

Figure 8 shows that all distance metrics produce highly consistent temporal trajectories. In each case, R_t remains stable in early training and increases sharply in later stages, indicating the emergence of representation enrichment under shared pathways. While the absolute scale varies across metrics, particularly between cosine-based and Euclidean distances, the transition pattern and overall trajectory shape are nearly identical.

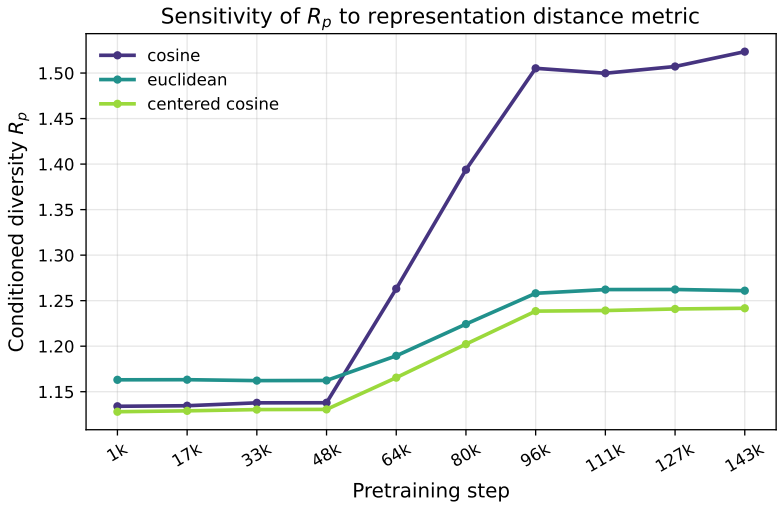


Figure 8: **Sensitivity of conditional diversity R_t to representation distance on `Pythia-1B`.** Across cosine, Euclidean, and centered cosine distances, R_t exhibits a similar qualitative trajectory, with a stable early period followed by a later increase.

Table 7: **Sensitivity of conditional diversity R_t to representation distance on `Pythia-1B`.** Pearson correlations with mean downstream accuracy are similar across the three tested distance metrics.

Distance metric	$\mathrm{corr}(R_t, \mathrm{Acc})$
Cosine	**0.76**
Euclidean	0.73
Centered cosine	0.75

To quantify this sensitivity, we compute the correlation between R_t and mean downstream accuracy (Table 7). All three distance metrics yield similar full-trajectory correlations ($r = 0.73$–0.76), indicating that the observed association is not specific to one representation distance.

D.3.2 Pathway-Similarity Threshold

We analyze the sensitivity of conditional diversity R_t to the pathway quantile p on `Pythia-1B`. We vary $p \in \{0.2, 0.3, 0.4, 0.5\}$ when defining pathway-similar pairs and compute R_t across pretraining checkpoints.

Figure 9 shows that larger p increases the absolute scale of R_t as progressively less similar pathway pairs are included. Despite this scale shift, the temporal trajectories remain highly consistent across the tested quantiles: the transition from an early stable regime to increasing conditional diversity later in training is preserved.

The relative ordering of checkpoints and the overall trajectory shape remain nearly invariant across the tested quantiles. Thus, the qualitative training signal captured by R_t is stable to moderate changes in the pathway-similarity threshold. The strength of its association with downstream accuracy nevertheless varies with p: Pearson correlation increases from 0.61 at $p = 0.2$ to 0.89 at $p = 0.5$ (Table 8).

D.4 Alternative Monitor Comparisons

D.4.1 Standard Pretraining Metrics

We compare pathway-based monitors with conventional pretraining metrics on `Pythia-160M`: token-level loss, perplexity, a bits-per-byte proxy, and total NLL. These lower-is-better metrics

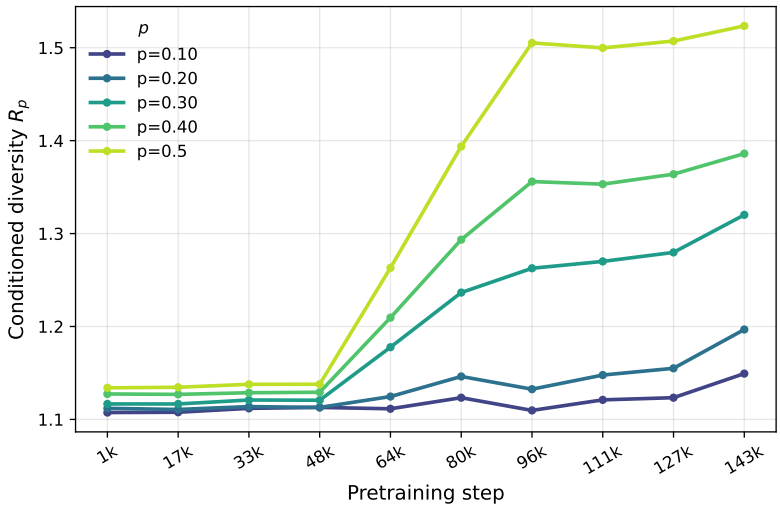


Figure 9: **Sensitivity of conditional diversity R_t to the pathway-similarity quantile p.** Although the absolute scale changes with p, the relative ordering of checkpoints and the overall temporal trajectory remain highly consistent across the tested thresholds.

Table 8: Sensitivity of conditional diversity R_t to the pathway-similarity quantile p on `Pythia-1B`. The qualitative trajectory is preserved across the tested thresholds, while the correlation with mean downstream accuracy varies with p.

p	$\mathrm{corr}(R_t, \mathrm{Acc})$
0.20	0.61
0.30	0.74
0.40	0.83
0.50	**0.89**

Table 9: **Comparison with standard pretraining metrics.** Pearson correlations with mean downstream accuracy across ten `Pythia-160M` checkpoints. Background color indicates nominal Pearson significance: $p < 0.1$, $p < 0.05$, $p > 0.5$.

Metric	r
Standard pretraining metrics	
Loss / token	-0.75
Perplexity	-0.63
Bits-per-byte proxy	-0.75
NLL	-0.75
Pathway-based monitors	
P_t	$\mathbf{-0.94}$
R_t	-0.09
A_t	$\mathbf{0.93}$

Table 10: **Linear CKA baseline on `Pythia-160M`.** Signed Pearson correlations with 20-task mean accuracy across ten checkpoints. Columns specify the reference checkpoint; layer numbers are one-based.

	Reference checkpoint	
CKA summary	Initial (1k)	Final (143k)
Layer 1	-0.050	$+0.140$
Layer 2	-0.006	$+0.180$
Layer 3	-0.286	$+0.310$
Layer 4	-0.315	$+0.312$
Layer 5	-0.314	$+0.313$
Layer 6	-0.311	$+0.312$
Layer 7	-0.316	$+0.312$
Layer 8	-0.315	$+0.311$
Layer 9	-0.319	$+0.310$
Layer 10	-0.315	$+0.309$
Layer 11	-0.325	$+0.294$
Layer 12	-0.315	-0.163
All-layer mean	-0.285	$+0.285$
Layers 1-4 mean	-0.190	$+0.256$
Layers 5-8 mean	-0.314	$+0.312$
Layers 9-12 mean	-0.319	$+0.274$

measure fit to the pretraining distribution and are therefore expected to correlate negatively with downstream accuracy.

Table 9 shows correlations of $r = -0.75$ for loss/token, bits-per-byte, and NLL, and $r = -0.63$ for perplexity. For comparison, P_t correlates at $r = -0.94$, A_t at $r = +0.93$, and R_t alone at $r = -0.09$.

These results distinguish optimization fit from computation organization. Loss-based metrics characterize fit to the pretraining distribution, whereas the pathway-based metrics characterize pathway reuse and within-pathway representation structure.

D.4.2 Representation-Similarity Baseline

Table 10 reports signed Pearson correlations for all tested layer and reference settings. The largest absolute correlation is $|r| = 0.325$ (layer 11, initial-checkpoint reference). These are retrospective within-trajectory associations, and the representation settings differ from the pathway analysis in Figure 3; the comparison should therefore be interpreted as a representation-similarity baseline rather than a fully matched benchmark.

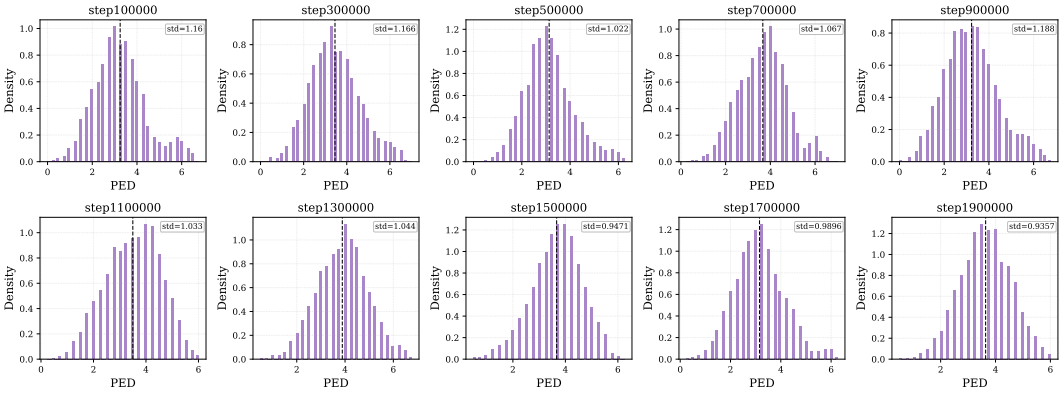


Figure 10: **Empirical PED distributions across checkpoints.** Pairwise pathway edit distance distributions for the `OLMo2-0425-1B` `StarCoder` domain across ten checkpoints. PED distributions remain broad and non-degenerate throughout pretraining, indicating that sample pairs span a meaningful continuum of pathway similarity. This supports the percentile-based conditioning used in R_t: pathway-similar pairs are selected from the lower tail of each checkpoint's empirical distribution, rather than by a fixed absolute threshold.

D.5 Empirical PED Distributions

Our conditional state diversity metric $R_t(\mathcal{D}; p)$ defines pathway-similar pairs using the lower p-th percentile of the checkpoint-specific PED distribution. Figure 10 shows these distributions for the `StarCoder` domain of `OLMo2-0425-1B`. They remain broad and non-degenerate across checkpoints, providing a continuum of relative pathway similarity rather than a concentration of identical routes.

Because the PED distribution shifts over training, R_t recomputes the percentile threshold $\tau_p^{(t)}(\mathcal{D})$ at each checkpoint. The conditioning therefore adapts to changes in the global PED scale and selects the relatively most pathway-similar pairs at each stage, rather than relying on a fixed absolute distance threshold.