Learning to Read Out: Unembedding Dynamics in Language Model Pretraining

Matteo He[1] **William F. Shen**[1] **Alex Iacob**[1,2]
Andrej Jovanovic[1] **Xinchi Qiu**[1,3] **Nicholas D. Lane**[1,2]

Abstract

When does a language model acquire a capability? Developmental studies usually answer this by tracking when models start producing the correct token. But that prediction passes through the unembedding matrix, or readout, which is itself still being learned during pretraining. We show that a model's hidden states can encode task information before its own readout expresses it, so timelines read from outputs conflate what the model encodes with what its readout expresses. To track the readout directly, we introduce parameter-trajectory crosscoding, which fits one sparse dictionary to every token's readout row at every checkpoint. Unlike crosscoders on activations, it needs only checkpoint weights. We use it to locate a narrow early window, 0.4–1.1% of the way through Pythia's pretraining, in which the readout reorganizes and WordNet word categories become decodable from its rows. By then, linear probes on the hidden states already decode task information that the model's own outputs do not yet show. To test whether such gaps depend on how fast the readout learns, we pretrain small models from scratch that differ only in the readout's learning rate. We find that raising it makes the readout develop earlier and the gap close sooner, with little effect on final loss. Together, these results imply that early in training, where many developmental transitions are dated, a behavioral date can reflect when the readout learns to express task information and not when the model learns it.

1 Introduction

Developmental studies of language models ask when a model learns a capability, and they often date it through the model's outputs. Surprisal curves across checkpoints date when individual words are acquired [Chang and Bergen, 2022], and translation accuracy dates a shift from copying to translation [Körner et al., 2026]. The same study traces this shift across layers with the logit lens [nostalgebraist, 2020], which reads hidden states through the unembedding.

All of these measurements read the model through its *readout*, the unembedding matrix that maps the last-layer hidden state to vocabulary *logits*. The readout is itself learned during pretraining. For the prompt `The keys to the cabinet`, an early hidden state of Pythia-1B [Biderman et al., 2023] prefers the incorrect `is` under its own readout but the correct `are` under the final checkpoint's readout, although the hidden state is the same (Figure 1a).

The source of this discrepancy is the logit computation. For a token v with readout row w_v and last-layer hidden state h, its logit is $\ell_v = h \cdot w_v$, and between two checkpoints

$$\Delta\ell_v = \Delta h \cdot w_v + h \cdot \Delta w_v + \Delta h \cdot \Delta w_v. \tag{1}$$

[1]Department of Computer Science and Technology, University of Cambridge. [2]Flower Labs. [3]Isomorphic Labs.
Correspondence: `mh2274@cantab.ac.uk`. Code: `https://github.com/hematteo/learning-to-read-out`.
Dictionaries and checkpoints: `https://huggingface.co/hematteo`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

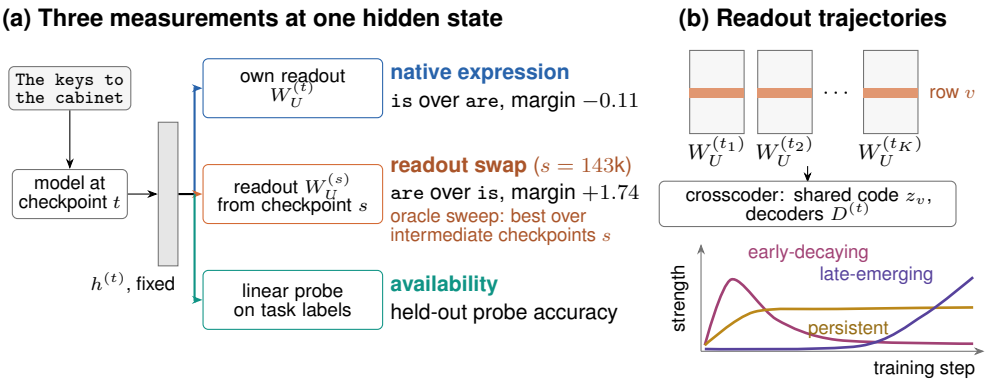

Figure 1: Overview. (a) The same Pythia-1B step-1000 hidden state prefers `is` under its own readout and `are` under the final readout, while a linear probe measures what the state encodes without passing through any readout. (b) Parameter-trajectory crosscoding fits one sparse dictionary to the trajectories of all readout rows. Feature strengths are schematic and use the profile colors of Figure 2.

A logit change can thus come from what the hidden state encodes, from the readout, or from the two changing together, and the logits alone cannot tell these apart. Checkpoint comparisons with the logit lens [Wang et al., 2024, Körner et al., 2026], vocabulary projections [Wiegreffe et al., 2025], or attention head scores computed through the readout [Tigges et al., 2024] share this ambiguity.

The information a task depends on, such as subject number when choosing `are` over `is`, is *available* once a linear probe can decode it from the hidden state, and *natively expressed* once the model's own readout prefers the correct answer. Probing studies measure the first [Tenney et al., 2019], and behavioral studies measure the second. Prior work has found related gaps between what a model encodes and what it outputs. Trained models can encode more than their outputs express [Orgad et al., 2025, Gekhman et al., 2025], and during training, internal structure can form before the behavior it supports appears in the outputs [Chen et al., 2024, Nanda et al., 2023].

We show that the readout has its own developmental timeline, which shapes when available information becomes natively expressed. This matters most early in training, where developmental transitions are dated [Olsson et al., 2022, Chen et al., 2024, Körner et al., 2026], and where the readout reorganizes, consistent with the early organization of vocabulary embeddings [Papadimitriou and Prince, 2026].

To track the readout itself, we introduce *parameter-trajectory crosscoding*, a crosscoder fit to readout weights rather than activations. Like a sparse autoencoder [Bricken et al., 2023, Huben et al., 2024], it learns a dictionary of directions, called features, and rebuilds every token's row at every checkpoint from a small fraction of them (Figure 1b). We also decode fixed hidden states with the readouts of other checkpoints (Figure 1a).

We make the following contributions:

1. **Problem.** We formalize a confound in behavioral dating that also affects evaluations of early checkpoints, with diagnostics that separate hidden state and readout contributions (Section 2.3).
2. **Method.** We introduce parameter-trajectory crosscoding, which decomposes the readout's development into sparse features that can be followed across checkpoints. Unlike crosscoders on activations [Bayazit et al., 2026, Ge et al., 2026], it needs only checkpoint weights and covers the full vocabulary (Section 2.1).
3. **Readout dynamics.** We identify a narrow early window, 0.4–1.1% of the way through training, in which the Pythia readout reorganizes and WordNet categories become decodable from its rows (Sections 4.1 and 4.2).
4. **Availability and expression.** Hidden states of Pythia-1B and 6.9B encode subject number before the model's own readout or any intermediate readout expresses it (Section 4.3).
5. **Training intervention.** We show that raising only the readout's learning rate in small controlled pretraining runs makes the readout develop earlier and native expression reach availability sooner,

with little effect on final loss. Early comparisons of training recipes can therefore reflect how fast the readout develops (Section 4.5).

2 Method

We track the readout's development with parameter-trajectory crosscoding (Sections 2.1 and 2.2) and separate hidden state and readout contributions with probes and readout swaps (Section 2.3).

2.1 Parameter-Trajectory Crosscoding

Dating the readout's development requires knowing which groups of tokens change, along which directions, and when. As training proceeds, each next-token loss pulls the target's readout row toward the hidden state and pushes every other token's row away from it, so the rows of many tokens drift along shared directions [Gao et al., 2019, Biś et al., 2021]. We describe these shared motions with a sparse dictionary, which rebuilds each token's row at every checkpoint from a small fraction of its learned directions, called features. Each feature is active for only some tokens, so unlike a principal component it identifies a group of tokens as well as a direction, and a token can belong to several groups. Sparse autoencoders fitted separately per checkpoint number their features independently, and matching them to follow a feature through training is error-prone because dictionaries trained with different seeds learn partly different features [Paulo and Belrose, 2026]. We therefore adopt the crosscoder approach [Lindsey et al., 2024, Ge et al., 2026] and fit one dictionary to all checkpoints at once. Each token has one code per feature that is shared across checkpoints, and each feature has one decoder direction per checkpoint. A feature's direction, norm, and active tokens can therefore change over training while its index stays fixed. We fit the dictionary to readout parameters rather than activations, so it needs only checkpoint weights, without labels or an activation corpus.

We center and scale each checkpoint's rows, so decoder norms exclude the growth of the mean row and of the readout's overall scale. For normalized rows $x_v^{(t)}$, checkpoint set $\mathcal{T}$, and $F = d_{\mathrm{SAE}}$ features, the dictionary encodes a token's rows at all checkpoints into one shared code z_v, gates it at each checkpoint into the features active there, $a_v^{(t)}$, and decodes these with the checkpoint's own decoder into a reconstructed row $\widehat{x}_v^{(t)}$,

$$z_v = \sum_{t\in\mathcal{T}} \left(x_v^{(t)} E^{(t)} + b_E^{(t)}\right), \quad a_{v,f}^{(t)} = z_{v,f}\, \mathbf{1}\Big\{z_{v,f}\|D_f^{(t)}\|_2 > \theta_f\Big\}, \quad \widehat{x}_v^{(t)} = a_v^{(t)} D^{(t)} + b_D^{(t)}. \tag{2}$$

Here $E^{(t)} \in \mathbb{R}^{d_{\mathrm{model}}\times F}$ and $D^{(t)} \in \mathbb{R}^{F\times d_{\mathrm{model}}}$ are encoder and decoder matrices with rows $D_f^{(t)}$, and the b terms are biases. The gate is the JumpReLU variant of Ge et al. [2026], with learned thresholds θ_f [Rajamanoharan et al., 2024b]. An active feature adds $z_{v,f} D_f^{(t)}$ to the token's reconstructed row at checkpoint t, and the gate compares the norm of this contribution, $z_{v,f}\|D_f^{(t)}\|_2$, with θ_f. Because the code is shared, a gate on the code alone would make a feature active for a token at every checkpoint or at none. Including the decoder norm lets a feature emerge or decay over training while keeping its index. The contribution norm also stays the same when a feature's codes are multiplied by a positive constant and its decoders divided by it, a rescaling that leaves every reconstruction unchanged, so the gate does not depend on this arbitrary scale.

Training fits the encoders, decoders, biases, and thresholds by gradient descent on

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\mathrm{rec}} + \lambda_1 \mathcal{L}_s, \qquad \mathcal{L}_{\mathrm{rec}} = \frac{1}{V|\mathcal{T}|}\sum\nolimits_{v,t} \big\|x_v^{(t)} - \widehat{x}_v^{(t)}\big\|_2^2, \\ \mathcal{L}_s &= \sum\nolimits_{f=1}^{F} \bar{q}_f\Big(1 + \frac{\bar{q}_f}{\alpha}\Big), \qquad \bar{q}_f = \frac{1}{V|\mathcal{T}|}\sum\nolimits_{v,t} \tanh\big(c\, a_{v,f}^{(t)}\|D_f^{(t)}\|_2\big), \end{aligned} \tag{3}$$

where V is the vocabulary size and c, α, and λ_1 are constants (Appendix A). Here $\mathcal{L}_{\mathrm{rec}}$ is the reconstruction error, and $\mathcal{L}_s$ is the saturating firing rate penalty of Ge et al. [2026], which encourages each row to be rebuilt from fewer features each making a substantial contribution. Each $\tanh$ term rises from 0 toward 1 as feature f's contribution to token v's row at checkpoint t grows, so $\bar{q}_f$ is a soft count of how often the feature is used, its firing rate. Because $\tanh$ levels off at 1, a large contribution costs no more than a moderate one, so the penalty does not push active contributions toward zero as an L_1 penalty would. The factor $1 + \bar{q}_f/\alpha$ makes each use more expensive the more

often the feature fires, which discourages broadly used features. Like the gate, the penalty depends only on contribution norms, so it is unchanged by the rescaling of codes and decoders above, and the dictionary cannot lower it by shrinking codes and inflating decoders.

A token's logit is the dot product of the hidden state with its row, so each decoder direction is also a direction in hidden-state space. A hidden state's component along this direction shifts the logits of all the feature's tokens together, in proportion to the decoder norm and each token's code. For a given hidden state, the decoder norm thus sets how far the feature moves its tokens' logits at each checkpoint, which is why we track decoder norms across checkpoints and leave them unconstrained. The rescaling above can multiply all of a feature's decoder norms by the same factor without changing the fit, so the shape of a feature's decoder norm trajectory is meaningful but its height is not. The lifecycle profiles below therefore divide each trajectory by its maximum. Normalization, training details, and a diagram of the forward pass (Figure A.1) are in Appendix A.

2.2 Measuring Readout Development

We describe each feature by its decoder norm trajectory $\|D_f^{(t)}\|_2$ across checkpoints. To compare when features grow and decay rather than how large they are, we divide each trajectory by its maximum and classify it with fixed rules into one *lifecycle profile*, such as early-decaying, persistent, or late-emerging (Appendix D).

To locate when the readout changes most, we measure six kinds of change between adjacent checkpoints, such as how much decoder norms change and how far decoder directions rotate. The *reorganization score* of a window of training steps is the geometric mean of the six measures, each summed over the window and normalized, so it is high only where several kinds of change coincide. The *reorganization window* is the window whose score most exceeds circular-shift nulls, which shift each measure in time independently, keeping its own profile but breaking the alignment between measures (Appendix D.4). For the controlled runs of Section 4.5, we also report two simpler statistics. The *median feature peak step* is the median over features of the checkpoint at which each decoder norm peaks, and the *peak turnover step* is the checkpoint at which normalized decoder norms change most from the previous checkpoint.

2.3 Distinguishing Hidden States and Readout Expression

We separate the hidden state's contribution to a logit change from the readout's with two measurements. First, a *readout swap* holds the hidden state fixed and decodes it with another checkpoint's readout. Let $W_U^{(t)} \in \mathbb{R}^{V \times d_{\text{model}}}$ be the readout at training checkpoint t. Let $h_p^{(t)} \in \mathbb{R}^{d_{\text{model}}}$ be the final hidden state at prompt position p, the residual stream vector (the sum of the embedding and all layer outputs) after the checkpoint's final normalization, immediately before the readout. Decoding this state with the readout of checkpoint s gives the logits

$$\ell_p^{(t,s)} = h_p^{(t)} \left(W_U^{(s)}\right)^\top. \tag{4}$$

Native decoding uses $s = t$, and a readout swap uses $s \neq t$ while keeping the state, including its own final normalization. A swap therefore isolates the readout term of Equation (1) at a fixed state.

For a correct answer y^+ and an incorrect alternative y^- (the foil), we score the margin at the final prompt position,

$$m^{(t,s)} = h_p^{(t)} \cdot \left(W_U^{(s)}[y^+, :] - W_U^{(s)}[y^-, :]\right). \tag{5}$$

The *contrast accuracy* of readout s is the fraction of items with $m^{(t,s)} > 0$, and *native expression* E is the contrast accuracy of the checkpoint's own readout, $s = t$. The *oracle readout sweep* scores the readouts of all intermediate checkpoints, excluding the native readout and the final readout, whose effect on early states we report separately (Section 4.3), and reports the highest contrast accuracy, reached by the *best swept readout*. Because the best readout is chosen after scoring, the sweep is an upper bound on the contrast accuracy of any single intermediate readout.

Second, a linear probe reads the hidden state directly, without passing it through any readout. *Availability* A is the probe's accuracy at predicting the task label from the hidden states that the swaps decode. Comparing A with E separates what a state encodes from what its own readout expresses.

Table 1: Task examples and item counts for subject–verb agreement (SVA), indirect object identification (IOI), and number comparison. Underlined nouns are attractors, which stand between the subject and the verb. n_A counts items in the broader task suite used for availability, and n_L counts the separately built localization items before the selection and scoring split. Simple SVA is the set plotted in Figure 4a. Templates and word lists are in Appendix F.1.

Task		Example prompt	Correct	Foil	n_A	n_L
SVA	simple	The keys to the cabinet	`are`	`is`	320	320
	across PP	The judge by the street	`appears`	`appear`	2,000	–
	subject RC	The author that admires the senators	`likes`	`like`	1,200	–
	object RC	The author that the senators admire	`likes`	`like`	1,200	–
IOI	availability	Lisa and Jane went to the store. Jane waited. Lisa gave the ring to	`Jane`	`Lisa`	2,000	–
	localization	Ryan and Adam went to the store. Ryan gave the pen to	`Adam`	`Ryan`	–	462
Number	comparison	Which is greater, 64 or 84? Answer:	`84`	`64`	2,000	–
	greater-than	84 is greater than	`64`	`84`	–	2,000

3 Experimental Setup

Models and dictionaries. We study readout development in Pythia-160M, 1B, and 6.9B [Biderman et al., 2023] and in OLMo-2-7B, which differs from Pythia in architecture, tokenizer, and training recipe [Team OLMo et al., 2025]. The dictionaries for all four models cover the full vocabulary across 32 public checkpoints (Table C.1). To test how the readout's own optimization shapes its development, we train four additional 31M-parameter models on the first 10B tokens of Pythia's preshuffled training data, in Pythia's order, from a shared initialization (Section 4.5). Three runs differ only in the readout's learning rate multiplier ($0.25\times$, $1\times$, $4\times$), and the fourth keeps the $1\times$ multiplier and halves the warmup from 1430 to 715 steps. Each 31M model is trained once, and three dictionaries with one shared width and sparsity weight are fitted to 16 of its checkpoints.

Tasks and scoring. We compare readouts on binary tasks, each with a correct answer token and a foil (Table 1). Subject–verb agreement (SVA) [Linzen et al., 2016, Marvin and Linzen, 2018] contrasts the agreeing verb form with the non-agreeing one, indirect object identification (IOI) [Wang et al., 2023] contrasts the recipient with the giver, and number comparison contrasts the correct number with the other number in the prompt. Labels are balanced by subject number, by whether the recipient is the first or second name, or by number order, and each answer is a single token that includes its leading space. The native readout and each swapped readout are scored at the final prompt position with the margin of Equation (5). Contrast accuracy compares only the two answers and ignores every other token in the vocabulary. With balanced labels, its chance level is 50%.

Probes and controls. We measure availability with linear probes on the final hidden state. For simple SVA and IOI in Pythia-6.9B, the probe is logistic regression, fit at each of the 32 checkpoints (Section 4.3). For a broader suite of 20 task families, the probe is a single direction from shrinkage linear discriminant analysis (LDA), fit on the families with a shared binary label, and we call its accuracy A_1 (Appendix F). Prompts that share a noun, a name pair, or a number pair always fall in the same cross-validation fold. We compare each probe with three controls: a probe at step 0 shows what is decodable before training, a probe on prompt tokens alone tests whether surface features suffice, and a probe on random Gaussian features tests whether the classifier fits unrelated inputs. Probes fit to shuffled labels stay below 0.56 on the availability sets of Table 1. Appendix A.3 gives fit details and maps controls to confounds.

Sparse localization. To find which readout features carry a task margin, we decode step-1000 Pythia-1B states with the dictionary's reconstruction of the final readout, which is how a logit lens with the final readout reads an early checkpoint. The reconstructed readout is a sum of feature contributions, so each feature contributes a separate term to the margin, and a feature's attribution is the mean of this term over items. He et al. [2026] explain logit lens scores this way for a single readout. In this work, the features come from a dictionary fitted across all 32 checkpoints, so each feature keeps the same index throughout training. For SVA, IOI, and greater-than completion (Table 1), we keep only answer pairs whose rows in the final readout have similar norms, $|\log(\|w_{y^+}\|_2/\|w_{y^-}\|_2)| \leq 0.5$. This filter limits the effect of row norm on the margin. We split each task's items into two equal

parts at random and select the eight features with the largest attribution on the first part. On the second part, we measure contrast accuracy under two edits to the reconstruction: ablation, which removes these features, and isolation, which keeps only them. We compare these edits with the full reconstruction and with the same edits applied to random features matched on decoder norm, firing rate, and attribution sign (Appendix G.4).

Lexical organization. The dictionary dates when the readout reorganizes but not what its rows learn. To check this without the dictionary, we probe the readout rows directly for WordNet supersenses, broad categories such as nouns for animals [Miller, 1995, Fellbaum, 1998]. For each of the 35 categories with at least 20 whole-word tokens, we train a one-vs-rest logistic probe on two-thirds of the centered readout rows and report its balanced accuracy (the mean recall over the two classes) on the remaining third. Three controls test whether token metadata (frequency, length, row norm, spacing, and token class) explains this accuracy. The first probes the metadata alone, the second probes matched random token sets, and the third shuffles labels within metadata bins (Appendix E).

4 Results

We first ask when and how the readout's structure changes (Sections 4.1 and 4.2), then whether hidden states encode task information before the model's own readout expresses it (Sections 4.3 and 4.4), and finally whether the readout's learning rate sets the timing of its development (Section 4.5).

We first check how well the dictionaries reconstruct the readout, since the lifecycle and localization analyses read it through them. Across Pythia-160M, 1B, and 6.9B and OLMo-2-7B, the dictionaries explain 0.808–0.920 of the variance in the readout trajectories. On average, each readout row uses 1.17–2.26% of the features at each checkpoint, and no feature is dead (Figure C.1). They also reconstruct Pythia-160M checkpoints held out from fitting nearly as well as the fitted ones (Appendix C.5). They nearly match the fit of sparse autoencoders fitted separately per checkpoint with far fewer active features, and unlike those autoencoders they keep one feature index across checkpoints. In Pythia, dense PCA needs far more components to reach the same fit (Appendix C).

4.1 The Readout Reorganizes Early

When does the readout change, and do its features share one growth pattern? In all four models, the reorganization score peaks within the first few percent of training. In Pythia, it peaks between step 512 and step 1.6k in all three models, 0.4% to 1.1% of the way through training, a window that contains the end of the learning rate warmup at step 1430. This window stands out against circular-shift nulls, with $p = 0.005$ at 160M and 1B and $p = 0.04$ at 6.9B (Figure 2, bottom; Figure D.2 and Table D.2). Of the 21 Pythia dictionaries we fitted, 20 select the same window and the remaining one selects the adjacent window (Appendix D.5). Measures of raw readout movement that use no dictionary, such as the drift of its top subspace, also put the largest movement early, at the checkpoint pair ending at step 256 or 512, just before this window (Figure C.6). In OLMo-2-7B, the score peaks at a similar fraction of training, between step 9k and step 28k. This window lies 1–3% of the way through OLMo's first training stage, the main pretraining phase before mid-training. It does not separate from the nulls ($p = 0.12$, Appendix D).

Features do not share one growth pattern. We follow each feature by its decoder norm, which sets how far the feature moves its tokens' logits. In Pythia-160M and 1B, some features peak early and decay, others persist, and others emerge late, while nearly all Pythia-6.9B features persist and most OLMo-2-7B features emerge late (Figures 2 and D.1). OLMo-2-7B's readout also grows more gradually, with its total decoder norm rising through most of training. These profile shares vary with dictionary width and sparsity (Appendix D.5). Early decay also depends on the checkpoints nearest initialization. A Pythia-1B dictionary fitted only from step 256 onward assigns 0.02% of its features to the early-decaying profile, against 24% in the main Pythia-1B dictionary (Appendix D.3).

The readout therefore changes most early on, in one window that clears the nulls in Pythia and at a similar fraction of training in OLMo-2-7B, and the mix of feature trajectories differs by model.

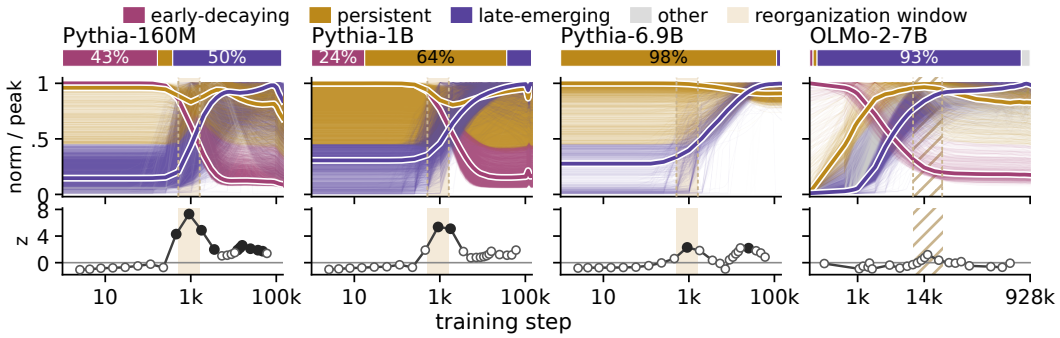

Figure 2: Pythia's readout reorganizes in one early window, and the mix of feature trajectories differs by model. Top: share of features in each lifecycle profile. Middle: each line is one active feature's decoder norm divided by its own maximum, so heights do not compare features; thick lines are profile medians. Bottom: reorganization score as a z-score against circular-shift nulls, with filled markers at $p \leq 0.05$. Bands mark the selected windows, hatched for OLMo-2-7B ($p = 0.12$).

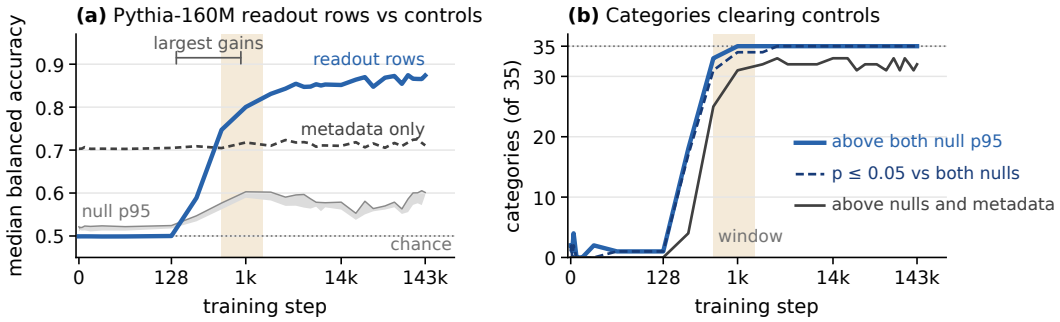

Figure 3: Pythia-160M readout rows come to separate WordNet categories beyond matched controls, with the largest gains between steps 128 and 1000, which overlap the start of the reorganization window. (a) Median held-out balanced accuracy over the 35 categories, against a probe on token metadata alone (dashed) and the 95th percentiles (p95) of the matched-random and shuffled-label nulls (gray band). The bracket marks the largest gains and the shaded band the reorganization window. (b) Number of categories above each control.

4.2 Lexical Organization Develops Alongside Structural Change

Do the readout rows acquire recognizable lexical organization while their structure changes? In Pythia-160M, the median probe accuracy across the 35 WordNet categories rises from 0.499 at initialization to 0.874 at the final checkpoint. We call a category decodable once its accuracy stays at or above $\max(0.55, \text{acc}_0 + 0.05)$ for three successive checkpoints, which counts only gains above its accuracy at initialization. All 35 categories become decodable by step 1000, and 33 have their largest gain between steps 128 and 1000 (Figure 3). In Pythia-1B, the median category becomes decodable at step 512, with onsets from step 128 to step 2000 (Appendix E).

We compare the probes with the three controls of Section 3. The first control, a probe on token metadata alone, predicts some categories well early in training. From step 512 onward, median readout accuracy exceeds it. Against the second and third, by step 512, 33 of 35 categories exceed the 95th percentile of both the matched-random and the shuffled-label nulls, and by step 1000 all 35 do. Splitting the tokens into training and test sets by lemma, so that variants such as `dog` and `Dog` never appear in both, leaves the early rise unchanged and lowers late accuracy, to a final median of 0.841 with 34 of 35 categories decodable by step 1000 (Appendix E).

Lexical organization therefore emerges in the readout rows before and during the 512–1.6k window.

4.3 Hidden State Availability Precedes Native Readout Expression

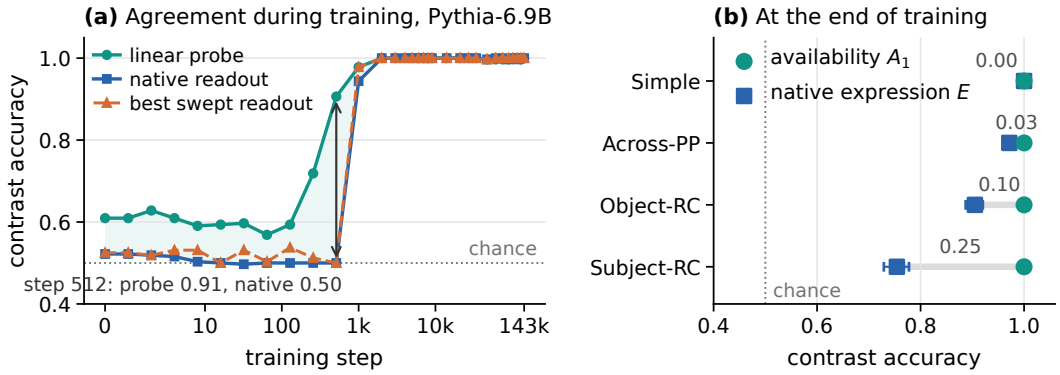

Figure 4: Pythia-6.9B hidden states encode subject number before the model's own readout or any intermediate readout expresses it. (a) Accuracy on simple SVA of the logistic probe, the native readout, and the best swept readout over intermediate checkpoints; the arrow marks the gap at step 512. (b) Single-direction probe A_1 and native expression E on the four agreement sets of Table 1 at the final checkpoint; annotations give $A_1 - E$, and bars are 95% Wilson intervals for E.

Is task information decodable from the hidden state before the model's own readout expresses it? A probe and the native readout can disagree because they read the hidden state along different directions. The probe learns its direction from the task labels, while the native readout can only use the difference between its two answer rows. If the state separates the answers along a direction that this difference does not yet point in, the information is available but not expressed. Expression begins once the hidden state or the readout changes so that the two directions line up. We track the probe, the native readout, and the oracle readout sweep on the same items at every checkpoint. On Pythia-6.9B agreement, the probe approaches ceiling by step 512 while native expression and even the best swept readout are still at chance (Figure 4a). Probes on prompt tokens and on random Gaussian features stay near chance, so neither surface tokens nor classifier fitting explains the probe's accuracy (Figure F.1). The same early gap appears in every subject–verb agreement set of the broader 20-family suite whose probe starts near chance, in both Pythia-1B and Pythia-6.9B (Appendix F).

Can a later readout decode the current hidden state better than its own? For next-token prediction, it can. On Wikipedia text in nine languages, the step-1000 readout lowers the aggregate next-token loss of Pythia-1B step-256 and step-512 states by 1.50 and 0.15 nats, and every language prefers an intermediate readout, even after aligning the two readouts' frames (Figures G.1 and G.3 and Table G.1). Early in training, the readout therefore lags behind the hidden states. The final readout also raises early task accuracy, from 0.800 to 0.925 on Pythia-1B step-1000 SVA states, as in Figure 1a, but this gain depends on its frame and falls to 0.566 after an orthogonal Procrustes alignment [Schönemann, 1966], which rotates the final readout to best match the native one (Appendix G.4).

Availability therefore precedes native expression. Section 4.4 asks which readout directions carry these margins, and Section 4.5 tests whether the readout's own optimization sets their timing.

4.4 A Few Readout Directions Carry the Agreement Margin

Which parts of the readout carry a task margin? In the localization setup of Section 3, eight features carry the SVA margin. Ablating them lowers reconstructed SVA accuracy from 0.900 to 0.487, keeping only them gives 0.938, and ablating matched random features leaves 0.931. These interventions are causal evidence that the reconstructed final readout's SVA margin on step-1000 states runs through these eight directions. Appendix G.4 repeats the analysis for IOI and greater-than completion. This analysis decomposes the logit lens margins of Section 4.3 into readout features.

4.5 Readout Learning Rate Shifts Readout Development

Early in training, the readout lags behind the hidden states, and in Pythia its reorganization window contains the end of warmup. Does the readout's timing come from how fast the readout itself learns,

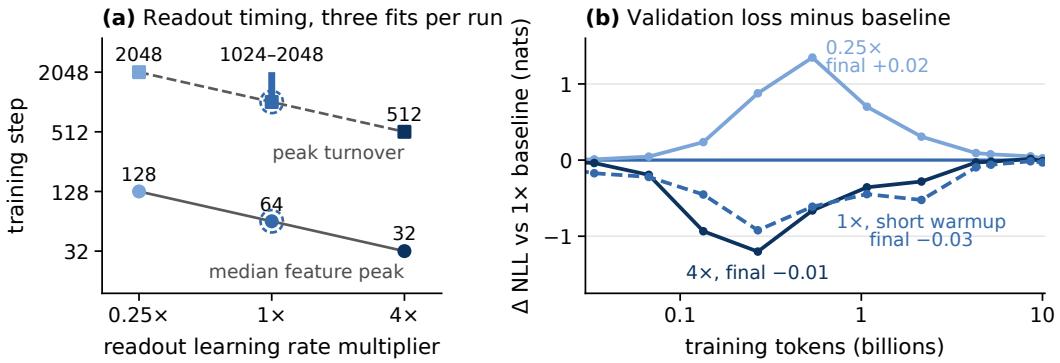

Figure 5: A faster readout develops earlier and lowers early validation loss, while final losses agree within 0.06 nats. (a) Median feature peak step and peak turnover step against the readout learning rate multiplier; bars span the three fits where they differ, and dashed circles mark the shorter warmup. (b) Validation loss of each 31M run minus that of the baseline run.

or from its warmup? We find that changing the readout's learning rate shifts when its features peak, while halving the warmup leaves this timing unchanged on our checkpoint grid. In the 31M runs of Section 3, raising the multiplier moves the median feature peak step from 128 to 64 to 32 in every fit (Figure 5), one checkpoint per fourfold increase. Halving the warmup leaves the median feature peak at its baseline value in every fit.

A faster readout also lets native expression reach availability sooner. In these runs, availability is a probe's accuracy on the frequency class of the next token, and native expression is the model's accuracy on the same classes. Along the multiplier axis, a slower readout gives a larger early gap between this probe and native expression, and the gaps close by step 1024 in every run except $0.25\times$, which closes by step 2048. Readout swaps show the same ordering. In the $0.25\times$ run, step-256 hidden states are decoded best by the step-512 readout, which lowers their next-token loss by 0.91 nats and keeps this gain after orthogonal Procrustes alignment, whereas in the $4\times$ run each checkpoint's own readout is best (Appendix H.4).

Does developing earlier leave the model better off at the end of training? It lowers early loss but not final loss. At 0.27B tokens, the validation loss of the $4\times$ run is 1.20 nats below the baseline and 2.08 nats below the $0.25\times$ run, whereas final validation losses agree within 0.06 nats (Figure 5b).

We therefore identify the readout's learning rate as one factor that shifts when the readout develops and when native expression reaches availability. Unlike the fixed-state swaps of Section 4.3, this intervention acts during training and changes the hidden states as well as the readout.

5 Related Work

Reading development through the readout. Developmental analyses track linguistic representations, circuits, and concepts across checkpoints [Olsson et al., 2022, Nanda et al., 2023, Inaba et al., 2025, Xu et al., 2024, Sawmya et al., 2025], and the logit lens and tuned lens project hidden states into vocabulary scores [nostalgebraist, 2020, Belrose et al., 2023]. Analyses that read development through the outputs depend on a readout that is itself still being learned. We add the readout to these comparisons, tracking its rows with shared sparse features and testing how changing it alters what fixed hidden states express (Table I.1 compares these approaches).

Weight-space analyses. Analyses of output embeddings study weight tying [Inan et al., 2017, Press and Wolf, 2017], the softmax bottleneck [Yang et al., 2018], and embedding geometry and its evolution [Gao et al., 2019, Machina and Mercer, 2024]. Closest to our lexical analysis, Papadimitriou and Prince [2026] correlate the geometry of input and output embeddings with semantic, syntactic, and frequency structure over training. Parameter decomposition and weight-sparse models analyze mechanisms in weights, typically at one model state [Braun et al., 2025, Gao et al., 2026]. As an alternative to reading logit lens scores in token units, He et al. [2026] explain them as sums of sparse feature contributions, decomposing a single readout from its weights alone.

Shared dictionaries over checkpoints. Sparse autoencoders decompose activations into sparse features [Bricken et al., 2023, Huben et al., 2024, Templeton et al., 2024, Gao et al., 2025, Rajamanoharan et al., 2024a], and crosscoders share feature indices across layers or models [Lindsey et al., 2024]. The closest methods are the checkpoint crosscoders of Bayazit et al. [2026] and Ge et al. [2026], which learn shared features over activations sampled from a text corpus (400M and 800M tokens) and relate their evolution to behavior through interventions or attribution. We adopt their shared codes and checkpoint-specific decoders, with the gate and sparsity penalty of Ge et al. [2026], both of which use decoder norms, and apply them to token-aligned readout parameters. Fitting then needs only checkpoint weights and covers every readout row rather than only the tokens a corpus elicits, while the behavioral tests still need text, forward passes, and task labels.

6 Discussion

Taking our findings together, we argue that the readout is part of the measurement in any analysis that reads development off logits. A rise in output accuracy or in a logit margin can come from the hidden state or from the readout, even for continuous scores that avoid metric-induced emergence [Schaeffer et al., 2023]. Reporting native expression, a probe on the same task with its initialization baseline, and a readout sweep on the same states would help distinguish the two.

Readout change concentrates in the same early period in which many developmental transitions are dated [Chen et al., 2024]. The induction head phase change, for example, occurs roughly 1–2% of the way through training in large models [Olsson et al., 2022], close to the 0.4–1.1% window in Pythia. A behavioral date in this period can therefore reflect readout development as well as hidden state development. What drives the different feature trajectories within this period remains open. Language models first predict from unigram frequencies and then loosely from bigram probabilities, before converging on richer predictions [Chang and Bergen, 2022], and early-decaying features may record the first of these stages, with late-emerging features taking over as context use develops.

The same confound affects evaluations of early checkpoints and of small, briefly trained models, which inform forecasts of how larger runs will perform [OpenAI, 2023, Snell et al., 2024]. Such forecasts already contend with capabilities that appear abruptly with scale [Wei et al., 2022, Ganguli et al., 2022]. While the readout still lags, outputs can underreport what the hidden states already encode, whereas a probe alone would overpredict expression. Internal progress measures can anticipate a behavioral transition [Nanda et al., 2023]. Testing whether native expression, probes, and readout sweeps sharpen such forecasts, also for safety-relevant capabilities, is a natural next step.

The lag is largest early in training. Once the readout has reorganized, each hidden state is decoded best by its own readout or a neighbor, and native expression catches up with availability on several tasks, so most implications concern analyses of early checkpoints. In the 31M runs, the readout's learning rate changes early loss and when a distinction becomes expressible, but leaves final losses within 0.06 nats (Section 4.5). Recipe comparisons made on partial runs can therefore reflect how fast the readout develops rather than where training ends. Overall, learning a representation and learning to express it can happen at different times in pretraining, and measuring both separates what a model encodes from what its logits express.

Limitations. First, limited compute restricted the evidence that the readout's learning rate shifts its timing to one 31M setting, with one run per condition stopped at 10B tokens. The readout holds about 42% of the parameters there, against about 10% in Pythia-1B, so larger models remain untested. Second, availability is measured with probes on two-token contrasts, so the gap describes when a binary distinction becomes expressible, not when an open-ended capability appears. Appendix I.2 covers further limits, such as checkpoint density and dictionary settings.

Future work. We suggest three useful directions. First, recipe comparisons on early checkpoints can favor a faster readout even when final losses agree (Section 4.5). Checking whether recipe rankings made with probes or swapped readouts match the final rankings would test a fix. Second, extending a pretrained model's vocabulary adds untrained readout rows, so training these rows with their own learning rate may speed up adaptation. Third, measuring weight drift in every layer (Figure C.6) would show whether each component, like the readout, settles on its own schedule.

Acknowledgments and Disclosure of Funding

Funding and competing interests. The authors report no third-party funding or in-kind support for this work and no additional competing interests beyond the affiliations listed above.

References

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2016. URL https://arxiv.org/abs/1610.01644.

Allen Institute for AI. OLMo-2-1124-7B model card. https://huggingface.co/allenai/OLMo-2-1124-7B, 2025.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Boaz Barak, Benjamin L. Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. In *Advances in Neural Information Processing Systems*, volume 35, pages 21750–21764, 2022. doi: 10.52202/068431-1581. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/884baf65392170763b27c914087bde01-Abstract-Conference.html.

Deniz Bayazit, Aaron Mueller, and Antoine Bosselut. Crosscoding through time: Tracking emergence & consolidation of linguistic representations throughout LLM pretraining. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1353–1377, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.60. URL https://aclanthology.org/2026.acl-long.60/.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022. doi: 10.1162/coli_a_00422. URL https://doi.org/10.1162/coli_a_00422.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the Tuned Lens, 2023. URL https://arxiv.org/abs/2303.08112.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. Too much in common: Shifting of embeddings in transformer language models and its implications. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.403. URL https://aclanthology.org/2021.naacl-main.403/.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An

open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL https://aclanthology.org/2022.bigscience-1.9/.

Terra Blevins and Luke Zettlemoyer. Language contamination helps explain the cross-lingual capabilities of English pretrained models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3563–3574, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.233. URL https://aclanthology.org/2022.emnlp-main.233/.

Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Jake Mendel, and Lee Sharkey. Interpretability in parameter space: Minimizing mechanistic description length with attribution-based parameter decomposition, 2025. URL https://arxiv.org/abs/2501.14926.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. 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. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Tyler A. Chang and Benjamin K. Bergen. Word acquisition in neural language models. *Transactions of the Association for Computational Linguistics*, 10:1–16, 2022. doi: 10.1162/tacl_a_00444. URL https://aclanthology.org/2022.tacl-1.1/.

Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L. Leavitt, and Naomi Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in MLMs. In *International Conference on Learning Representations*, pages 14479–14510, 2024. URL https://openreview.net/forum?id=MO5PiKHELW.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1198. URL https://aclanthology.org/P18-1198/.

Nolan Dey, Gurpreet Gosal, Zhiming Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. Cerebras-GPT: Open compute-optimal language models trained on the Cerebras wafer-scale cluster, 2023. URL https://arxiv.org/abs/2304.03208.

EleutherAI. Pythia-160M model card. https://huggingface.co/EleutherAI/pythia-160m, 2023a.

EleutherAI. Pythia-1B model card. https://huggingface.co/EleutherAI/pythia-1b, 2023b.

EleutherAI. Pythia-6.9B model card. https://huggingface.co/EleutherAI/pythia-6.9b, 2023c.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Nelson Elhage, Sheer El Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Scott Johnston, Andy Jones, Nicholas Joseph, Jackson Kernion, Shauna Kravec, Ben Mann, Neel Nanda, Kamal Ndousse, Catherine Olsson, Daniela Amodei, Tom Brown, Jared Kaplan, Sam McCandlish, Chris Olah, Dario Amodei, and Jack Clark. Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM Conference on Fairness,*

Accountability, and Transparency, FAccT '22, pages 1747–1764, Seoul, Republic of Korea, 2022. Association for Computing Machinery. doi: 10.1145/3531146.3533229. URL https://doi.org/10.1145/3531146.3533229.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkEYojRqtm.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, pages 26721–26754, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V. Govande, Bowen Baker, and Daniel P. Mossing. Weight-sparse transformers have interpretable circuits. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 33483–33513. PMLR, 2026. URL https://proceedings.mlr.press/v306/gao26w.html.

Xuyang Ge, Wentao Shu, Jiaxing Wu, Yunhua Zhou, Zhengfu He, and Xipeng Qiu. Evolution of concepts in language model pre-training. In *International Conference on Learning Representations*, pages 41767–41801, 2026. URL https://openreview.net/forum?id=qKmXz1UR0Z.

Zorik Gekhman, Eyal Ben David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. Inside-out: Hidden factual knowledge in LLMs. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=f7GG1MbsSM.

Xinyang Geng and Hao Liu. OpenLLaMA: An open reproduction of LLaMA. https://github.com/openlm-research/open_llama, May 2023.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. OLMo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.841. URL https://aclanthology.org/2024.acl-long.841/.

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek Abdelzaher, and Heng Ji. Word embeddings are steers for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16410–16430, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.864. URL https://aclanthology.org/2024.acl-long.864/.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Matteo He, William F. Shen, Xinchi Qiu, and Nicholas D. Lane. Sparse Readout Prism: Explaining logit-lens scores in features instead of tokens, 2026. URL https://arxiv.org/abs/2609.01936.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. 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)*, pages 2733–2743, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275/.

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)*, pages 4129–4138, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL https://aclanthology.org/N19-1419/.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, pages 7827–7845, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

John D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55. URL https://doi.org/10.1109/MCSE.2007.55.

Tatsuro Inaba, Go Kamoda, Kentaro Inui, Masaru Isonuma, Yusuke Miyao, Yohei Oseki, Yu Takagi, and Benjamin Heinzerling. How a bilingual LM becomes bilingual: Tracing internal representations with sparse autoencoders. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13458–13470, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.725. URL https://aclanthology.org/2025.findings-emnlp.725/.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=r1aPbsFle.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Felicia Körner, Maria Matveev, Florian Eichin, Gitta Kutyniok, Barbara Plank, and Michael A. Hedderich. Copy first, translate later: Interpreting translation dynamics in multilingual pretraining. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, Budapest, Hungary, 2026. Association for Computational Linguistics. URL https://arxiv.org/abs/2604.17633.

Patrick Leask, Bart Bussmann, Michael T. Pearce, Joseph Isaac Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis. In *International Conference on Learning Representations*, pages 53617–53642, 2025. URL https://openreview.net/forum?id=9ca9eHNrdH.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-demo.21. URL https://aclanthology.org/2021.emnlp-demo.21/.

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*, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4: 521–535, 2016. doi: 10.1162/tacl_a_00115. URL https://aclanthology.org/Q16-1037/.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, Richard Fan, Yi Gu, Victor Miller, Yonghao Zhuang, Guowei He, Haonan Li, Fajri Koto, Liping Tang, Nikhil Ranjan, Zhiqiang Shen, Roberto Iriondo, Cun Mu, Zhiting Hu, Mark Schulze, Preslav Nakov, Timothy Baldwin, and Eric Xing. LLM360: Towards fully transparent open-source LLMs. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=QdWhj0QZFw.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Anemily Machina and Robert Mercer. Anisotropy is not inherent to transformers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4892–4907, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.274. URL https://aclanthology.org/2024.naacl-long.274/.

Rebecca Marvin and Tal Linzen. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1151. URL https://aclanthology.org/D18-1151/.

Wes McKinney. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a. URL https://doi.org/10.25080/Majora-92bf1922-00a.

George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. doi: 10.1145/219717.219748. URL https://doi.org/10.1145/219717.219748.

Julian Minder, Clément Dumas, Caden Juang, Bilal Chughtai, and Neel Nanda. Overcoming sparsity artifacts in crosscoders to interpret chat-tuning. In *Advances in Neural Information Processing Systems*, volume 38, pages 106423–106474, 2025. doi: 10.52202/085713-3551. URL https://openreview.net/forum?id=yFdNygEryH.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

nostalgebraist. Interpreting GPT: The Logit Lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *International Conference on Learning Representations*, pages 66880–66913, 2025. URL https://openreview.net/forum?id=KRnsX5Em3W.

Isabel Papadimitriou and Jacob Prince. Vocabulary embeddings organize linguistic structure early in language model training. In *Third Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2510.07613.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. In *International Conference on Learning Representations*, pages 35748–35760, 2026. URL https://openreview.net/forum?id=EjInprGpk9.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL https://jmlr.org/papers/v12/pedregosa11a.html.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain, 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025/.

Princeton University. WordNet 3.0 license. https://wordnet.princeton.edu/license-and-commercial-use, 2006.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving sparse decomposition of language model activations with Gated Sparse Autoencoders. In *Advances in Neural Information Processing Systems*, volume 37, pages 775–818, 2024a. doi: 10.52202/079017-0024. URL https://openreview.net/forum?id=zLBlin2zvW.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with JumpReLU sparse autoencoders, 2024b. URL https://arxiv.org/abs/2407.14435.

Shashata Sawmya, Micah Adler, and Nir Shavit. The birth of knowledge: Emergent features across time, space, and scale in large language models, 2025. URL https://arxiv.org/abs/2505.19440.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, volume 36, pages 55565–55581, 2023. doi: 10.52202/075280-2425. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/adc98a266f45005c403b8311ca7e8bd7-Abstract-Conference.html.

Peter H. Schönemann. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10, 1966. doi: 10.1007/BF02289451.

Charlie Snell, Eric Wallace, Dan Klein, and Sergey Levine. Predicting emergent capabilities by finetuning. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=vL8BIGuFTF.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari

Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

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, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, 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, 2025. URL https://arxiv.org/abs/2501.00656. Shorter version accepted to COLM 2025.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL https://aclanthology.org/P19-1452/.

Curt Tigges, Michael Hanna, Qinan Yu, and Stella Biderman. LLM circuit analyses are consistent across training and scale. In *Advances in Neural Information Processing Systems*, volume 37, pages 40699–40731, 2024. doi: 10.52202/079017-1287. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/47c7edadfee365b394b2a3bd416048da-Abstract-Conference.html.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *International Conference on Learning Representations*, pages 17282–17333, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Trillion Labs. Tri series intermediate checkpoints release. https://trillionlabs.co/research/tri-series-intermediate-checkpoints-release, September 2025. Archived at https://web.archive.org/web/20260419025910/https://trillionlabs.co/research/tri-series-intermediate-checkpoints-release.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020. doi: 10.1038/s41592-019-0686-2. URL https://doi.org/10.1038/s41592-019-0686-2.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In *Advances in Neural Information Processing Systems*, volume 37, pages 95238–95265, 2024. doi: 10.52202/079017-3017. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad217e0c7fecc71bdf48660ad6714b07-Abstract-Conference.html.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

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. doi: 10.1162/tacl_a_00321. URL https://aclanthology.org/2020.tacl-1.25/.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=yzkSU5zdwD.

Sarah Wiegreffe, Oyvind Tafjord, Yonatan Belinkov, Hannaneh Hajishirzi, and Ashish Sabharwal. Answer, assemble, ace: Understanding how LMs answer multiple choice questions. In *International Conference on Learning Representations*, pages 78293–78318, 2025. URL https://openreview.net/forum?id=6NNA0MxhCH.

Wikimedia Foundation. Wikimedia downloads: Wikipedia dumps. 2023-11-01 Wikipedia database dumps, 2023a. URL https://dumps.wikimedia.org/.

Wikimedia Foundation. Wikimedia Foundation terms of use. https://foundation.wikimedia.org/wiki/Policy:Terms_of_Use, 2023b. Effective June 7, 2023.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clément Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Yang Xu, Yi Wang, Hengguan Huang, and Hao Wang. Tracking the feature dynamics in LLM training: A mechanistic study, 2024. URL https://arxiv.org/abs/2412.17626.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkwZSG-CZ.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An open-source small language model, 2024. URL https://arxiv.org/abs/2401.02385.

Appendix Contents

A Dictionary Normalization, Objective, and Evaluation Details **20**
A.1 Normalization and Reconstruction . . . 20
A.2 Gate and Training Objective . . . 20
A.3 Behavioral Probe Fitting and Reporting . . . 21

B Measurement Setup and Reproducibility **22**
B.1 Code and Data Release . . . 22
B.2 Models, Tokenizers, and Checkpoint Schedules . . . 22
B.3 Checkpoint-Suite Selection . . . 23
B.4 Unembedding Convention and Tied Embeddings . . . 23
B.5 Sparse Dictionary Architecture and Loss . . . 23
B.6 Existing Assets, Licenses, and Terms . . . 24

C Dictionary Validation and Quality **25**
C.1 Mean-Direction and Preprocessing Check . . . 25
C.2 Next-Token Distribution Fidelity of the Reconstructed Readout . . . 26
C.3 Sparse Autoencoder Baselines . . . 27
C.4 Linear and Trajectory Baselines . . . 27
C.5 Held-out Checkpoint Reconstruction . . . 29
C.6 Sparsity-Penalty Sensitivity . . . 29
C.7 Full Dictionary Inventory . . . 30

D Sparse Lifecycle and Reorganization Timing **31**
D.1 Rule-Based Lifecycle Profiles . . . 31
D.2 Aggregate Timing Summaries . . . 31
D.3 Late-Start Control for OLMo . . . 32
D.4 Temporal Localization and Null Models . . . 33
D.5 Stability Across Dictionary Fits . . . 34

E WordNet Supersense Probe Analyses **35**
E.1 Probe Hyperparameters . . . 35
E.2 WordNet Supersense Inventory . . . 35
E.3 Matched-Control Analysis . . . 36
E.4 Probe Timing Trajectories . . . 37
E.5 Pythia-1B Replication . . . 37
E.6 Lemma-Grouped Splits . . . 37

F Availability and Expression Across Contrastive Task Families **37**
F.1 Prompt Construction . . . 38
F.2 Task Items and Final Checkpoint Results . . . 39
F.3 Initialization, Probe, and Answer-Token Controls . . . 40
F.4 Transient and Persistent Gaps . . . 40
F.5 Contrast Accuracy and Full-Vocabulary Prediction . . . 41

G Behavioral and Intervention Readout Checks **42**
G.1 Swap Grid Setup and Aggregate Measures . . . 42
G.2 Corpus Composition and Per-Language Swaps . . . 42
G.3 Step-1000 Swap Optimum and Late Checkpoint Anomaly . . . 44
G.4 Contrastive Task Margin Localization . . . 45

H Recipe Control: Readout Learning Rate and Developmental Timing **47**
H.1 Setup and Conditions . . . 47
H.2 Lifecycle Response to the Readout Learning Rate . . . 47
H.3 Warmup Length Leaves the Crosscoder Timing Statistics Unchanged . . . 48
H.4 Agreement Across Views on the Same Checkpoints . . . 49
H.5 Early Loss Differences and Similar Final Loss . . . 50

I Related Work, Limitations, and Broader Impact **50**
I.1 Extended Related Work . . . 50
I.2 Extended Limitations . . . 51
I.3 Implications . . . 51
I.4 Social Impact . . . 51

A Dictionary Normalization, Objective, and Evaluation Details

A.1 Normalization and Reconstruction

At checkpoint t, we subtract the mean vocabulary row and scale the centered entries by their root mean square:

$$\mu^{(t)} = \frac{1}{V}\sum_{v=1}^{V} W_U^{(t)}[v,:],$$
$$s^{(t)} = \left[\frac{1}{V d_{\text{model}}}\sum_{v=1}^{V}\left\|W_U^{(t)}[v,:]-\mu^{(t)}\right\|_2^2\right]^{1/2}, \qquad x_v^{(t)} = \frac{W_U^{(t)}[v,:]-\mu^{(t)}}{s^{(t)}}. \tag{6}$$

The quantity inside the square root is floored at 10^{-8}. Apart from this floor, the normalized matrix has zero mean row and average squared row norm d_{model}. Centering removes a vocabulary-wide offset, and scaling keeps a checkpoint with larger entries from dominating the loss. Appendix C.1 measures these global effects.

Dictionary inputs keep every output row, including padding and special-token rows. GPT-NeoX uses `<|endoftext|>` (id 0) as both BOS and EOS and `<|padding|>` (id 1) as the pad token, and the probe predicates exclude both. The mean and scale are stored with the dictionary, so raw W_U reconstructions can be recovered. Training uses no external token frequency table. The configuration name `frequency_scale` is historical, and its value, 0.01, is α in Equation (3). Because the dictionaries are fit to every vocabulary row, EV is an in-sample metric and does not measure generalization to unseen rows. We therefore report it with mean L_0 (active features per row and checkpoint pair), the dead feature rate, per-checkpoint fidelity, stability checks, and baselines.

Given a normalized reconstruction $\widehat{x}_v^{(t)}$, its original-space reconstruction is

$$\widehat{W_U}^{(t)}[v,:] = s^{(t)}\widehat{x}_v^{(t)} + \mu^{(t)}.$$

Feature f therefore contributes $s^{(t)}a_{v,f}^{(t)}D_f^{(t)}$ in the original readout space. Readout interventions keep the stored mean and decoder bias unless stated otherwise.

The final normalization and the readout share one exact symmetry. For the learned gain γ, the LayerNorm bias β, and any invertible diagonal matrix Λ, replacing (γ, β, W_U) with $(\Lambda\gamma, \Lambda\beta, W_U\Lambda^{-1})$ leaves every logit unchanged. Native expression, next-token loss, and the standardized logistic probes of Appendix A.3 are invariant to it. Readout rows, their norms and directions, the dictionary features, and readout swaps between checkpoints are not, because they depend on how each coordinate's scale is split between the gain and the readout. We measure readout geometry in the split that each released checkpoint stores, and centering and scaling remove only a vocabulary-wide offset and one global scale. Appendix H.5 discusses the scalar case in the 31M runs.

A.2 Gate and Training Objective

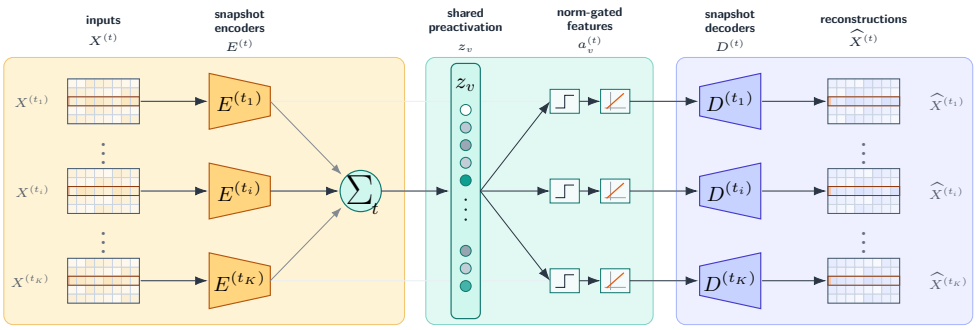

Figure A.1: The parameter-trajectory crosscoding forward pass (Equation (2)) retains one feature index across checkpoints while allowing its contribution to change through training. One vocabulary row is collected across centered and scaled checkpoints and encoded into a shared preactivation, and a checkpoint-specific gate and decoder reconstruct each view.

Training minimizes the objective of Equation (3). The reconstruction term sums squared errors over hidden dimensions and averages over vocabulary rows and checkpoints, without dividing by d_{model}. Minibatch training uses the same reductions with the batch size in place of V (`src/readout/crosscoder/wu_adapter.py`). The separately logged per-entry reconstruction MSE also averages over hidden dimensions, so it is d_{model} times smaller and cannot replace this term without rescaling λ_1.

The constant c sets the saturation scale of the $\tanh$, α the extra cost of broad firing, and λ_1 the tradeoff with reconstruction. The effective λ_1 is warmed up linearly from zero over the first 5% or 10% of optimizer steps (Table B.2). The gate and the penalty are unchanged by the positive featurewise rescaling $z_{v,f} \mapsto z_{v,f}/r$ and $D_f^{(t)} \mapsto rD_f^{(t)}$ of Section 2.1, but this invariance does not make the sparse factorization unique.

A.3 Behavioral Probe Fitting and Reporting

Hidden-state convention. The hidden state $h_p^{(t)}$ of Section 2.3 is the output of the final normalization, captured with a forward pre-hook on Pythia's `embed_out` module. Native and swapped logits are computed directly as $h_p^{(t)}(W_U^{(s)})^\top$, without normalizing a second time.

Logistic probes for the 32-checkpoint SVA and IOI trajectories. The SVA target is subject number, and the role-balanced IOI target is whether the recipient is the first or second name. In each of five folds, a standardization fit on the training examples is followed by L_2-regularized logistic regression ($C = 1$, `lbfgs`, at most $10{,}000$ iterations), with no tuning on test folds. We report mean held-out accuracy; fold standard deviations describe split variation, not independent training runs. Folds group SVA items by head noun lemma and IOI items by unordered name pair, so individual names can recur in other pairs across folds. The unbalanced recipient identity diagnostic is a separate multiclass probe, not the role-balanced binary result.

The prompt-token controls fit the same classifier to binary indicators or counts of prompt tokens. The random-feature control fits it to a fixed Gaussian matrix of the same shape as the hidden states, on the same folds. These controls test surface token statistics and fitting to unrelated features.

Broader binary task suite. The broader task suite reports a single shrinkage LDA direction, A_1, and a logistic probe, A_{full}, both with cross-validation grouped by the family's lemma or pair key. Number pair groups keep both orderings of a pair together, but individual numbers can recur in other pairs. The shrinkage coefficient, fold count, and selection rule of the shrinkage LDA probe were not recovered from the archived runs. Both classifiers project onto one direction, but their supervision and optimization differ from the model's pretrained answer row difference, so they measure task-supervised decodability.

Readout sweeps and candidate scores. For the 32-checkpoint SVA and IOI trajectories and Figure 4, the sweep B_{env} picks one readout per task and hidden state checkpoint after scoring all items, so it is a post hoc upper envelope over the evaluated readouts. In the broader task suite, the sweep B_{ho} picks the readout on a random half of the items and reports its accuracy on the other half (Appendix F). Neither sweep picks a readout per item. Both exclude the native and terminal readouts, while the corpus negative log-likelihood (NLL) grid uses the full readout schedule. Contrast accuracy counts strictly positive margins. Full-vocabulary rank-1 accuracy, whether the gold answer is the argmax over all tokens, is reported separately.

Swap effects cannot come from drift in the mean readout row. Adding a vector u to every row of $W_U^{(s)}$ shifts every logit by $h_p^{(t)} \cdot u$, which cancels in each contrast margin and in the softmax, so contrast accuracy and next-token NLL are unchanged. Multiplying $W_U^{(s)}$ by a positive scalar scales every margin, which leaves contrast accuracy unchanged but changes NLL.

Tokenizer filters and localization splits. Both answers are single tokens with a leading space (templates in Appendix F.1). Only the localization items pass the terminal row norm filter of Section 3, which bounds norm differences but does not equalize norms or remove directional effects. Selecting features on half the items and scoring the other half leaves 160 SVA, 1000 greater-than completion, 231 IOI, and 13 relational-fact scoring items (Tables 1 and F.1).

31M recipe probe. The availability probe of the 31M runs predicts one of four next-token frequency classes, each holding about equal token mass in the evaluation slice. It uses standardized logistic regression ($C = 1$, `lbfgs`, at most $10{,}000$ iterations) over five shuffled folds of sampled positions. The folds are stratified by frequency class and are not grouped by document, so positions from one 2048-token block can fall in both training and test folds. Native expression maps the model's full-vocabulary argmax to its frequency class. Label-shuffle and random-label controls use the same folds, and the readout sweep covers each condition's reported checkpoints (Appendix H).

Table A.1: Controls and the comparison each supports. Each control addresses one alternative explanation.

Control	Comparison
Step-0 hidden-state probe	Distinctions decodable before training versus gains in decodability during training.
Shuffled/random labels	Probe predictability under targets unrelated to the task assignment.
Prompt-token probe	Hidden-state predictions versus information in token presence/counts.
Gaussian-feature probe	Hidden-state predictions versus fitting an equally sized unrelated feature matrix.
WordNet matched controls	Lexical labels versus frequency, length, row norm, spacing, and token-class metadata or matched token sets.
Answer-row geometry	Differences in contrast scores attributable to candidate norms and token-frequency patterns.
Matched-feature ablations	Selected reconstructed contributions versus edits matched on decoder norm and firing rate, with an additional attribution-sign control.

B Measurement Setup and Reproducibility

B.1 Code and Data Release

Code is available at `https://github.com/hematteo/learning-to-read-out`, the trained dictionaries for Pythia-160M, 1B, 6.9B, and OLMo-2-7B with derived aggregates and evaluation outputs at `https://huggingface.co/hematteo/parameter-trajectory-crosscoders`, and the checkpoints of the controlled 31M runs at `https://huggingface.co/hematteo/readout-recipe-control`. The code repository provides training and evaluation implementations and instructions for external model and data inputs.

B.2 Models, Tokenizers, and Checkpoint Schedules

The primary trajectory is `EleutherAI/pythia-1b` ($d_{\text{model}} = 2048$), with `EleutherAI/pythia-160m` ($d_{\text{model}} = 768$) as the smaller setting, `EleutherAI/pythia-6.9b` ($d_{\text{model}} = 4096$) as a larger-scale check, and `allenai/OLMo-2-1124-7B` ($d_{\text{model}} = 4096$) as a cross-architecture check [Biderman et al., 2023, Team OLMo et al., 2025].

The Pythia models share the GPT-NeoX BPE tokenizer, fixed across checkpoints, with $V = 50{,}304$ readout rows for Pythia-160M and Pythia-1B and $V = 50{,}432$ padded rows for Pythia-6.9B. OLMo-2-7B has its own tokenizer and $V = 100{,}352$ readout rows.

Checkpoint selection is fixed before any dictionary is fit. Pythia runs use $K = 32$ checkpoints, loaded from Hugging Face revisions `step{N}`:

$$0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512,$$
$$1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000,$$
$$14000, 21000, 27000, 34000,$$
$$47000, 61000, 75000, 89000, 102000, 116000, 130000, 143000.$$

These are 20 checkpoints before step 10k, 4 from step 14k to 34k, and 8 from step 47k to 143k. OLMo-2-7B runs use 32 stage-1 checkpoints, loaded from revisions named `stage1-stepXXX-tokensYYYB`:

$$150, 600, 700, 850, 900, 1000, 2000, 3000, 4000, 5000,$$
$$6000, 7000, 8000, 9000, 14000, 21000, 27000, 34000,$$
$$47000, 110000, 173000, 236000, 299000, 362000,$$
$$425000, 488000, 614000, 677000, 740000, 803000,$$
$$866000, 928000.$$

B.3 Checkpoint-Suite Selection

Tracking token rows needs a fixed tokenizer, accessible W_U weights, and enough checkpoints around step 1000 to resolve the early readout changes. Pythia is the controlled family because each model has 154 public checkpoints (step 0, log-spaced steps 1 to 512, then every 1000 steps to 143k) and all scales share one tokenizer, one data order, and an untied readout. OLMo-2 is the cross-architecture check, with a public stage-1 trajectory that starts before step 1000 and shares many later steps with the Pythia grid. Its architecture, tokenizer, data, optimizer, and schedule all differ from Pythia's, so it is not a point on the Pythia scaling axis. Table B.1 lists the other public suites we considered and why we did not use each.

Table B.1: Other public checkpoint suites considered for W_U trajectory analysis, and why each was not used.

Suite	Reason not used
OLMo 7B, April 2024 [Groeneveld et al., 2024]	Sparser coverage before step 1000 than OLMo-2
Olmo 3 [Team Olmo et al., 2025]	No stage-1 checkpoint between steps 0 and 1000
LLM360 Amber [Liu et al., 2024]	Checkpoints indexed `ckpt_000` to `ckpt_359` instead of by training step
LLM360 Crystal [Liu et al., 2024]	Phase-specific revisions, custom loading, and a code and data mixture
Cerebras-GPT Intermediate [Dey et al., 2023]	No documented early step schedule, and some maximal update parametrization (μP) variants need custom loading
OpenLLaMA previews [Geng and Liu, 2023]	Sparse previews that begin after step 1000 (3B from step 2.5k)
TinyLlama [Zhang et al., 2024]	First intermediate checkpoint at step 5k (10B tokens)
SmolLM3 [Bakouch et al., 2025]	Checkpoints every 40k steps (about 94.4B tokens)
Tri family [Trillion Labs, 2025]	Coarse token intervals, and a multilingual mix that needs new lexical controls

B.4 Unembedding Convention and Tied Embeddings

The Pythia unembedding is `model.embed_out.weight`, a layer without bias that Hugging Face GPT-NeoX stores with shape (V, d_{model}), so row v is the output direction for token v. Pythia's input embedding W_E (`gpt_neox.embed_in.weight`) is a separate, untied parameter, and the readout analyses use only `embed_out.weight`. The OLMo-2-7B unembedding is `model.lm_head.weight`. Extracted unembeddings are stored as fp32 CPU tensors, one per checkpoint.

B.5 Sparse Dictionary Architecture and Loss

The dictionaries follow the checkpoint-spanning crosscoder recipe of Ge et al. [2026], applied to aligned W_U rows instead of activations from a text corpus. Each has $F = d_{\text{SAE}}$ features, $K = 32$ checkpoint-specific encoder and decoder heads with a decoder bias, and the JumpReLU gate of Equation (2) [Rajamanoharan et al., 2024b]. Each is trained on the objective of Equation (3) (Appendix A.2) with $\alpha = 0.01$ and $c = 1.0$. The inputs are already centered and scaled (Appendix A.1), so the dictionary library applies no further normalization to them. The decoder is initialized uniformly on $[-1/\sqrt{d_{\text{SAE}}}, 1/\sqrt{d_{\text{SAE}}}]$, and the encoder is then set to the decoder transpose, which replaces its initial uniform draw on $[-1/\sqrt{d_{\text{model}}}, 1/\sqrt{d_{\text{model}}}]$. JumpReLU thresholds are learned in log space, starting from the initial values in Table B.2. The table gives the remaining settings of each

main dictionary. Near-inactive features can keep small nonzero decoder norms, so decoder norm trajectories are read together with firing rate and reconstruction checks.

Table B.2: Sparse-dictionary training hyperparameters for every main dictionary setting. Each column is a complete recipe for one model family and matrix setting. All dictionaries train in fp32 on centered-and-scaled vocabulary rows (`center_scale`) for 300 epochs with batches of 1024 rows and no auxiliary loss on dead features (AuxK). Adam-family optimizers use the adaptive-moment method of Kingma and Ba [2015].

Hyperparameter	Pythia-1B	Pythia-160M	Pythia-6.9B $\lambda_1 = 0.6$	Pythia-6.9B $\lambda_1 = 0.3$	OLMo-2-7B
d_{SAE}	24,576	24,576	32,768	32,768	32,768
Expansion factor	12.0	32.0	8.0	8.0	8.0
Sparsity coefficient λ_1	0.3	0.3	0.6	0.3	0.3
Optimizer	Adam	Adam	Adam	Adam	SparseAdam
Learning rate	2×10^{-5}	5×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
JumpReLU LR factor	0.2	0.1	0.4	0.3	0.3
JumpReLU init threshold	0.1	0.1	0.15	0.1	0.1
Optimizer steps	15,000	15,000	15,000	15,000	29,400
Sparsity warmup fraction	0.05	0.05	0.05	0.10	0.10
LR warmup / decay	0 / 0	0 / 0	0 / 0	0.10 / 0.20	0.10 / 0.20
Training strategy	single process	single process	4-GPU head parallel	4-GPU head parallel	4-GPU head parallel
Compute	A100-class GPU(s)	A100-class GPU(s)	A100/H100-class GPUs	A100/H100-class GPUs	A100/H100-class GPUs
Seeds	0	0, 1, 2	0	0	0

Each epoch visits all vocabulary rows in a fresh permutation from a CPU `torch.Generator` seeded by the run seed, so a run takes $\lceil V/1024 \rceil \times 300$ optimizer steps. We keep the final epoch without early stopping, and rolling checkpoints save the generator state. EV, mean L_0, and dead feature counts (Appendix C) are computed afterwards over the full vocabulary and play no role in selecting checkpoints. The head-parallel strategy follows Ge et al. [2026], and the Pythia-6.9B $\lambda_1 = 0.3$ run is a denser capacity point.

Table B.3: Compute for reproducing the main dictionary fits and the controlled 31M pretraining runs, by hardware class rather than specific cloud offers. The 31M wall clock is estimated from the 10B-token budget and the recorded throughput of about 131k tokens/s. The 31M row covers the four compared runs and leaves out the excluded fifth run (Appendix H.1).

Run family	Worker class	GPU memory	Wall clock	GPU-hours
Pythia-160M W_U/W_E checkpoint-spanning fits	1–3 A100-class GPUs	40–80 GB/GPU	4–8 h	4–24
Pythia-1B W_U capacity sweep	1 A100-80GB GPU	80 GB	up to 12 h	up to 12
Pythia-6.9B W_U checkpoint-spanning fit, per λ_1 setting	4 A100/H100-class GPUs	40–80 GB/GPU	4–8 h	16–32
OLMo-2-7B W_U checkpoint-spanning fit	4 A100/H100-class GPUs	40–80 GB/GPU	4–8 h	16–32
31M pretraining runs (four compared)	1 A40 GPU per run	48 GB	about 21 h per run	about 85 in total
Probe, null, baseline, and figure jobs	CPU workstation or 1 GPU	24–80 GB if GPU	minutes–4 h	up to 4

The four-GPU fits need batch size 1024 on 40 GB GPUs and can use 2048 on 80 GB GPUs. The wall-clock ranges exclude checkpoint extraction. The Pythia-1B width sweep uses $d_{\mathrm{SAE}} \in \{8192, 16384, 24576\}$, expansion factors of about $\{4.0, 8.0, 12.0\}$.

B.6 Existing Assets, Licenses, and Terms

The Pythia checkpoints and tokenizers are released by EleutherAI under Apache-2.0, as stated on the Hugging Face model cards for the Pythia suite [EleutherAI, 2023a,b,c]. The OLMo-2-1124-7B checkpoints and tokenizer are released by Ai2 under Apache-2.0 [Allen Institute for AI, 2025]. The controlled 31M runs train on Pythia's preshuffled Pile [Gao et al., 2020, Biderman et al., 2023], distributed by EleutherAI on Hugging Face as `EleutherAI/pile-standard-pythia-preshuffled`, and use it for research; we do not redistribute the data. WordNet is used under the Princeton WordNet 3.0 license, including its notice, disclaimer, and non-endorsement terms [Princeton University, 2006]. The BLiMP families are used under CC BY 4.0 [Warstadt et al., 2020], and the function vector datasets under the MIT license [Todd et al., 2024]. The Wikipedia slices used for the multilingual readout-swap corpus come from Wikimedia database dumps; Wikimedia text contributions are available under CC BY-SA and GFDL terms under the Wikimedia Terms of Use [Wikimedia Foundation, 2023a,b]. The experiments use standard Python scientific and machine-learning dependencies, including PyTorch, Transformers, Datasets, NumPy, SciPy, Pandas, Matplotlib, and scikit-learn, under their respective open-source licenses [Paszke et al., 2019, Wolf et al., 2020, Lhoest et al., 2021, Harris et al., 2020, Virtanen et al., 2020, McKinney, 2010, Hunter, 2007, Pedregosa et al., 2011]. The code repository, the dictionaries, and the 31M checkpoints are released under the MIT license. We do

not redistribute the original Pythia or OLMo checkpoints or raw Wikipedia dumps; the released dictionaries and derived measurements identify the upstream assets and should be used subject to the original licenses and terms.

C Dictionary Validation and Quality

This appendix validates the checkpoint-spanning sparse dictionaries of Section 2, with the Pythia-1B dictionary of the main analyses as the primary case. Figure C.1 compares the selected dictionaries across models and, on Pythia-160M, compares a checkpoint-spanning dictionary with independent per-snapshot SAEs; Table C.1 lists every fit. Explained variance (EV) is one minus the mean squared reconstruction error divided by the variance of the centered and scaled rows, computed in sample on the rows the dictionary was fit to (Appendix A.1). Crosscoder L_0 is the number of features with nonzero activation at one row and checkpoint pair, averaged over all pairs, and active density is $100 \cdot L_0/d_{\mathrm{SAE}}$. A feature is active if its maximum decoder norm over checkpoints, $\max_t \|D_f^{(t)}\|_2$, exceeds 0.01, and dead if its firing rate over all row and checkpoint pairs is at most 10^{-6}. Several checks use the Pythia-160M $d_{\mathrm{SAE}} = 8{,}192$ dictionary, a narrower fit than the selected Pythia-160M $d_{\mathrm{SAE}} = 24{,}576$ dictionary.

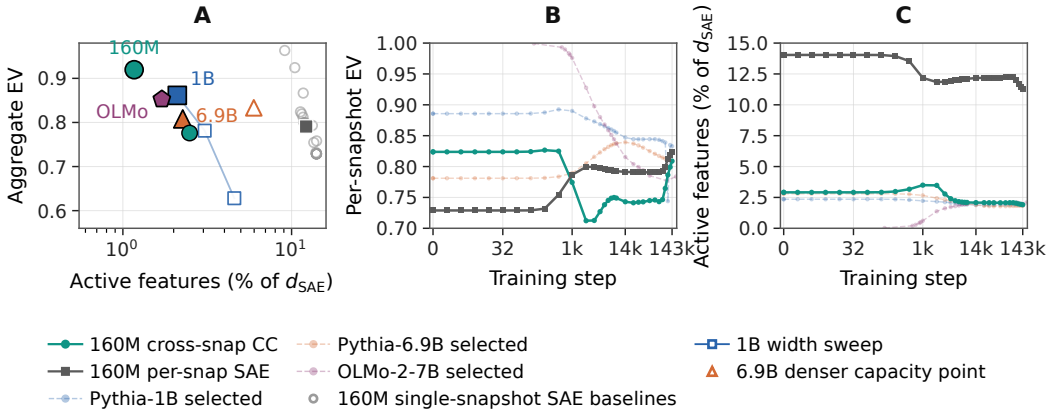

Figure C.1: The selected dictionaries reconstruct the readout trajectories with aggregate EV 0.808–0.920 while activating 1.17–2.26% of their features per row, and on Pythia-160M a checkpoint-spanning dictionary stays within 0.1 EV of independent per-snapshot SAEs at every checkpoint while activating 1.9–3.5% of its features against their 11–14%. (A) Aggregate EV against active density $100 \cdot L_0/d_{\mathrm{SAE}}$ for the four selected dictionaries (large labeled markers), the Pythia-1B width sweep, the denser Pythia-6.9B $\lambda_1 = 0.3$ fit, the Pythia-160M $d_{\mathrm{SAE}} = 8{,}192$ dictionary, and the Pythia-160M single-snapshot SAE baselines (Table C.1). (B) Per-checkpoint EV and (C) per-checkpoint active density. Solid lines compare the Pythia-160M $d_{\mathrm{SAE}} = 8{,}192$ seed-0 dictionary ("160M cross-snap CC"), not the selected $d_{\mathrm{SAE}} = 24{,}576$ dictionary, with the 32 per-snapshot SAEs of the same width; dashed lines show the selected Pythia-1B, Pythia-6.9B, and OLMo-2-7B dictionaries for reference. OLMo-2-7B is plotted on its own stage-1 step axis, which panels B and C cut at step 180k, so its last 11 checkpoints are not shown.

C.1 Mean-Direction and Preprocessing Check

Raw W_U geometry is dominated by a global offset shared across vocabulary rows. After early training, the mean row $\mu^{(t)} = V^{-1} \sum_{v=1}^{V} W_U^{(t)}[v,:]$ aligns with the top right singular vector of raw W_U, with absolute cosine close to 1 at the final checkpoint of Pythia-1B, Pythia-6.9B, and OLMo-2-7B (Figure C.2), and it inflates raw spectral and cosine geometry. Centering removes the positive floor in mean pairwise row cosine and lowers the top spectral gap σ_1/σ_2, estimated by block power iteration, at late checkpoints in all three models, so all sparse dictionaries and dense baselines use centered rows with a per-snapshot scalar rescaling (`center_scale`).

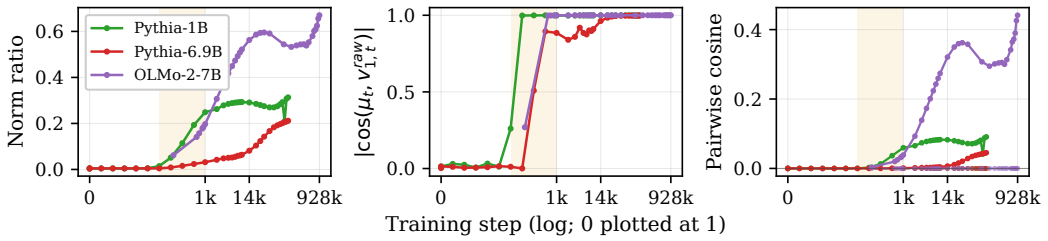

Figure C.2: The mean row of checkpointed W_U grows and aligns with the raw top right singular vector after early training, inducing a positive row-cosine floor, so later analyses use centered rows. The norm ratio is the mean-row norm relative to the median row norm, and the right panel compares raw and centered mean pairwise row cosine. The shaded band marks steps 64–1k; OLMo-2-7B is plotted on its own stage-1 schedule. Pythia-160M is not shown for readability.

C.2 Next-Token Distribution Fidelity of the Reconstructed Readout

Matrix EV measures agreement row by row, so we also decode the hidden states of a fixed held-out shard through the original and the reconstructed W_U at each checkpoint. The shard is the first 32 sequences of 512 tokens (16,352 predictions in English, Russian, and Chinese) from the multilingual corpus of the readout swaps (Appendix G), and ΔNLL is the mean increase in target NLL when the reconstructed readout replaces the original.

Fidelity varies non-monotonically over training and is least stable before step 2k (Figure C.3). At the terminal checkpoint, the selected Pythia-1B and Pythia-6.9B reconstructions agree with the original readout on 0.70 and 0.69 of top-1 predictions, with ΔNLL of 0.23 and 0.35 nats. The narrower Pythia-160M $d_{\mathrm{SAE}} = 8{,}192$ dictionary keeps logit shape at the final checkpoint but loses scale calibration (ΔNLL of 3.758 nats/token), so matrix EV alone is not sufficient evidence at small widths.

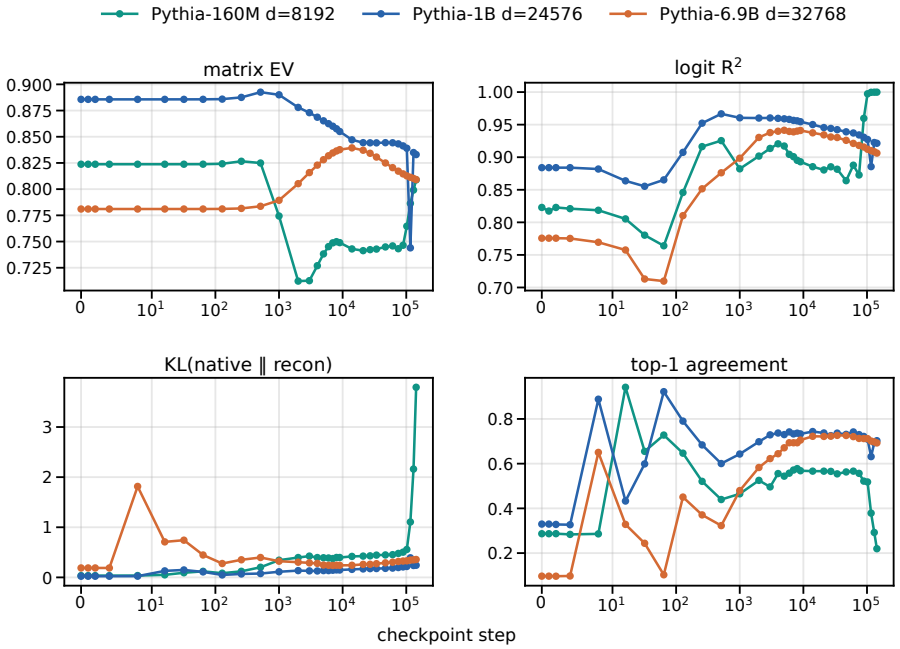

Figure C.3: Next-token fidelity of the reconstructed readouts varies over training, and the Pythia-160M $d_{\mathrm{SAE}} = 8{,}192$ dictionary keeps logit shape but loses scale calibration late in training. At each of the 32 checkpoints, hidden states from the fixed held-out shard are decoded through both the original (native) and reconstructed readouts. The third panel plots KL(native ‖ reconstructed), whereas the text reports ΔNLL.

Token-stratified check. Within token-frequency deciles, visible-length bins, and coarse token classes, the primary Pythia-1B reconstruction stays faithful at step 1000, with matrix EV of at least 0.868 in every stratum and target-conditioned ΔNLL of at most 0.42 nats (for ASCII punctuation). No single stratum explains the step-116k dip in aggregate EV (Appendix G.3).

C.3 Sparse Autoencoder Baselines

On Pythia-160M, we compare checkpoint-spanning W_U dictionaries with final-snapshot SAEs and 32 independent per-snapshot SAEs. Reconstruction barely changes across fit seeds and rises with width (Figure C.1 and Table C.1). Per-snapshot SAEs reconstruct individual checkpoints but are much denser, and the final-snapshot SAE fits the terminal readout but fails on early checkpoints (Figures C.1 and C.4). Neither gives a feature index shared across training at low L_0. Checkpoint-spanning dictionaries are sparse in active-code density but not parameter-efficient, because they spend parameters on checkpoint-specific encoder and decoder heads to preserve a shared feature index (Params column of Table C.1).

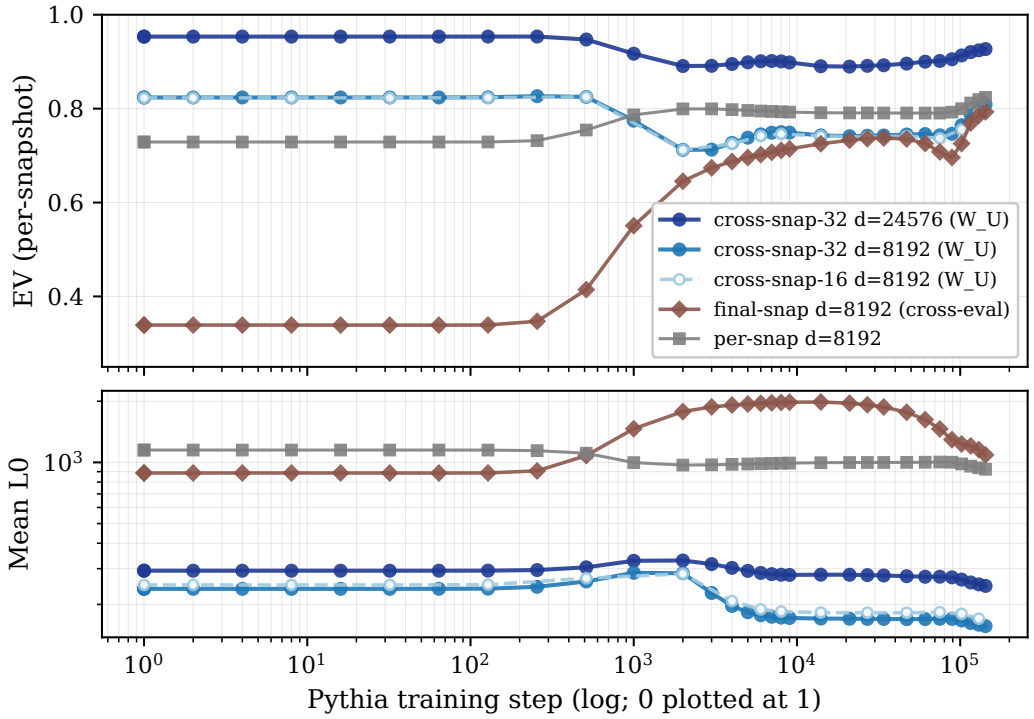

Figure C.4: On Pythia-160M, the checkpoint-spanning dictionaries ($d_{\mathrm{SAE}} = 8{,}192$ and the selected $d_{\mathrm{SAE}} = 24{,}576$) reconstruct early and late checkpoints with one feature index, while the SAE baselines either fail early or require far higher L_0. The final-snapshot SAE (cross-eval) is trained on the final checkpoint and evaluated backward; per-snapshot SAEs are trained independently at each checkpoint.

C.4 Linear and Trajectory Baselines

The dense baselines ask how much of the centered W_U trajectory can be explained without a sparse shared feature basis, at ranks and active-code budgets that differ from the dictionaries', so they do not rank the methods at equal budget.

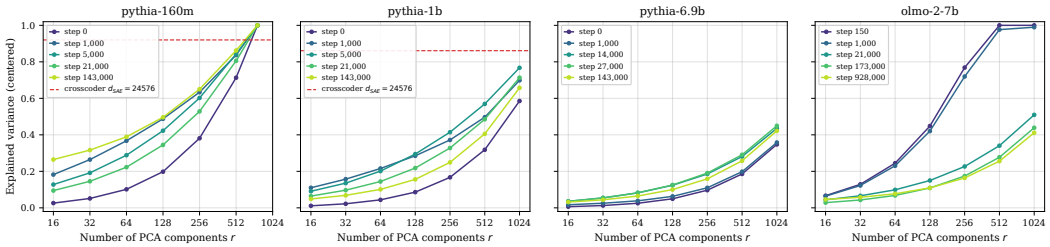

Figure C.5: Per-snapshot PCA of centered W_U needs many dense components to approach the EV of the selected sparse dictionaries, and at rank 1024 it falls short of them at every plotted Pythia-1B and Pythia-6.9B checkpoint and at the plotted OLMo-2-7B checkpoints from step 21k onward. Each checkpoint is approximated independently by its best rank-r PCA reconstruction. The dictionary reference line is drawn only for Pythia-160M and Pythia-1B; the selected Pythia-6.9B and OLMo-2-7B dictionaries reach aggregate EV 0.808 and 0.853 (Table C.1). Pythia-160M has $d_{\text{model}} = 768$, so its PCA is exact at rank 768.

Per-snapshot PCA needs many dense components to approach the selected dictionaries. At the final checkpoint, rank 512 explains 0.256–0.405 of the centered variance, and rank 1024 still falls short of the dictionaries at every Pythia-1B and Pythia-6.9B checkpoint and at OLMo-2-7B checkpoints from step 4k onward (Figure C.5). Pythia-1B would need roughly 1.36k–1.66k components to match its dictionary. The exception is OLMo-2-7B's earliest checkpoints (steps 150–3k), where rank-1024 PCA exceeds the dictionary's aggregate EV and, at steps 150 and 1k, rank 512 already explains nearly all centered variance. A single dense subspace fitted to each model's centered token rows concatenated across checkpoints (45 for Pythia-160M, 32 for Pythia-1B and OLMo-2-7B, and 15 for Pythia-6.9B) does worse at every tested rank from 16 to 256, reaching EV 0.41 in Pythia-160M and 0.23 in Pythia-1B at rank 256, against 0.92 and 0.86 for their dictionaries.

In Pythia, the dominant centered subspaces move most just before the 512–1.6k reorganization window of Appendix D.4 and then stabilize (Figure C.6). Top-10 drift peaks on the checkpoint pair ending at step 256 in Pythia-160M and 1B and at step 512 in Pythia-6.9B, one pair later at $k = 50$. This dense check locates change in time without identifying sparse features or token supports.

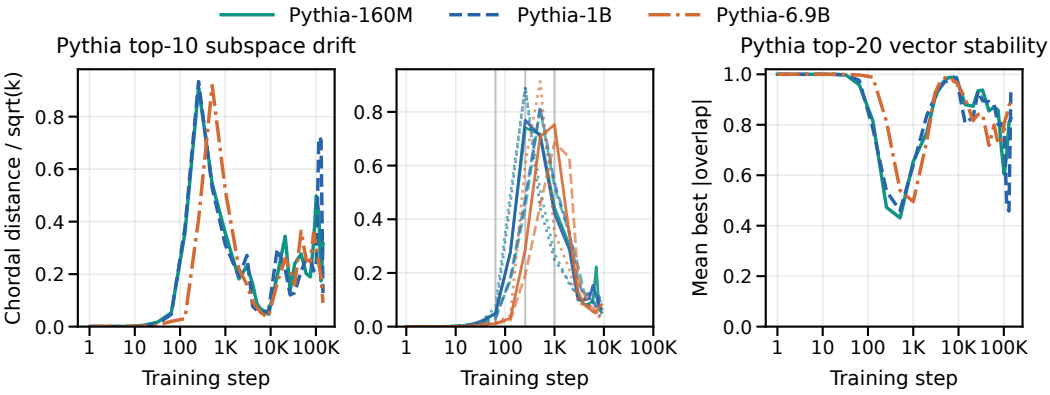

Figure C.6: Dominant centered W_U subspaces in Pythia move most during early training, with the largest top-10 movement between steps 128 and 512; the later Pythia-1B step-$116{,}000$ anomaly is examined in Appendix G.3. For each checkpoint, we center W_U, estimate top right-singular directions, and track adjacent-checkpoint chordal Grassmann distance (divided by $\sqrt{k}$ for a rank-k subspace) and top-vector stability, plotted at the later checkpoint of each pair. In the left and right panels, color and line style mark the model. The middle panel restricts the left panel's distance to steps up to 9k for subspace ranks $k = 5$ (dotted), 20 (solid), and 50 (dashed), with colors as in the left panel; gray verticals mark steps 64, 256, and 1k.

C.5 Held-out Checkpoint Reconstruction

Checkpoint-spanning dictionaries are trained on 32 anchor checkpoints, so held-out reconstruction tests whether the fitted heads merely memorize those anchors. Pythia has 154 released checkpoints; the 122 absent from the anchor schedule are unseen by construction. We evaluate eight of them, at steps 10k, 23k, 33k, 43k, 73k, 93k, 113k, and 123k, so held-out coverage starts at step 10k and misses the 512–1.6k reorganization window and all earlier steps. For a held-out step $t^\star$ with trained neighbors t_-, t_+ and log-spaced interpolation weight $\eta = (\log(1{+}t^\star){-}\log(1{+}t_-))/(\log(1{+}t_+){-}\log(1{+}t_-))$, we build an interpolated synthetic head from the neighboring encoder and decoder parameters,

$$E^\eta = (1-\eta)\,E^{(t_-)} + \eta\,E^{(t_+)}, \qquad b^\eta_{\text{enc}} = (1-\eta)\,b^{(t_-)}_{\text{enc}} + \eta\,b^{(t_+)}_{\text{enc}},$$

and analogously for the decoder parameters, leaving the shared JumpReLU threshold unchanged. The shared code z_v of token v keeps the 32 anchor terms of Equation (2), computed from v's rows at the anchor checkpoints, and adds the synthetic head's term $x_v^{(t^\star)} E^\eta + b^\eta_{\text{enc}}$ on its held-out row; the gate then uses the synthetic decoder norms. Only the reconstruction of the held-out row is scored, after applying the held-out checkpoint's `center_scale` statistics, with nearest-head, raw-space interpolation, endpoint-line interpolation, and rank-50 PCA as references.

From step 10k onward, neither tested dictionary shows a sharp anchor-memorization failure, although smoother trajectory interpolation remains possible. For the Pythia-160M d_{SAE}=8,192 seed-0 crosscoder, synthetic heads track the in-sample ceiling within 0.011 EV at all eight held-out checkpoints, well above rank-50 PCA and endpoint interpolation and below direct raw-space interpolation, which has no sparsity bottleneck. For the primary Pythia-1B crosscoder, the only deviations bracket the step-116k anomaly, which the in-sample ceiling and direct W_U interpolation also show (Figure C.7).

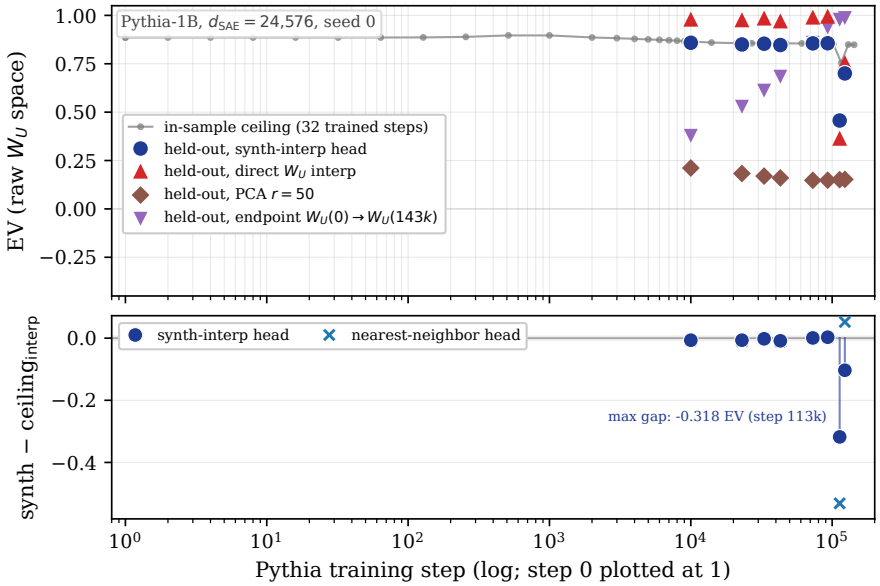

Figure C.7: Interpolated heads of the Pythia-1B d_{SAE}=24,576 seed-0 crosscoder track the in-sample ceiling within 0.01 EV at six held-out steps (10k–93k) and deviate only at 113k and 123k. These deviations bracket the step-116k anomaly (Appendix G.3), where the in-sample ceiling dips to 0.755 EV and direct W_U interpolation drops to 0.36 EV at 113k.

C.6 Sparsity-Penalty Sensitivity

Retraining the Pythia-160M $d_{\text{SAE}} = 8{,}192$ dictionary at five sparsity weights from $\lambda_1 = 0.4$ to 1.8 (seven fits, three of them at $\lambda_1 = 1.35$) traces a single reconstruction–sparsity tradeoff at fixed width, and the $\lambda_1 = 0.3$ fits of the same width lie at its high-EV end (Figure C.8). The sweep keeps the rest of the recipe but adds an auxiliary loss on dead features, which the $\lambda_1 = 0.3$ fits do not use.

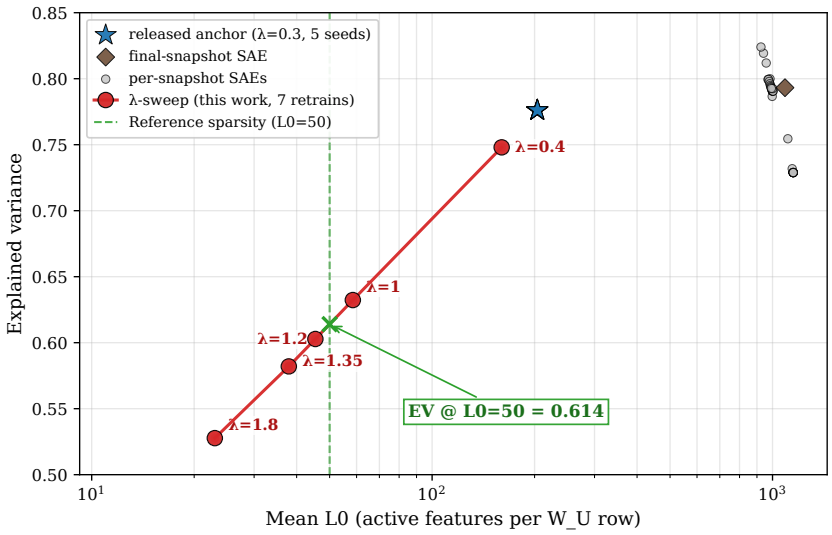

Figure C.8: For the Pythia-160M $d_{\text{SAE}} = 8{,}192$ dictionary, raising the sparsity weight λ_1 from 0.4 to 1.8 lowers mean L_0 from about 160 to about 23 and EV from 0.75 to 0.53 along a single tradeoff curve. Red points are the sweep fits (Table D.3). The star ("released anchor") marks the five-seed Pythia-160M $d_{\text{SAE}} = 8{,}192$ fits at $\lambda_1 = 0.3$, and the gray and brown markers are the same-width SAE baselines of Table C.1. The dashed green line marks $L_0 = 50$, near the low end of the sparsities of the Pythia-160M activation crosscoders of Ge et al. [2026] ($L_0 \approx 47$–64 in their Figure 2), and "EV @ L0 = 50" is the sweep's EV at that L_0, interpolated linearly in $\log L_0$ between adjacent sweep points.

C.7 Full Dictionary Inventory

Table C.1 lists every dictionary and baseline behind a validation figure or numeric result, with selected dictionaries marked by ⋆. The 160M, 1B, and OLMo checkpoint-spanning rows use the main $\lambda = 0.3$ recipe; final-snapshot SAE baselines use $\lambda = 0.05$ and 100 epochs; per-snapshot SAE baselines use $\lambda = 0.3$ and 100 epochs. W_U crosscoders use penalty saturation scale 1.0, except the Pythia-160M width-16 384 fit marked [d], which uses 4.0. The final-snapshot SAE baselines also use saturation scale 4.0. The init threshold is 0.1 except for the selected Pythia-6.9B $\lambda_1 = 0.6$ retrain.

Raw mean L_0 is not comparable across dictionary widths, so the table also reports active density $100 \cdot L_0/d_{\text{SAE}}$, on which the selected dictionaries are similarly sparse and the single-snapshot SAE baselines several times denser. In the selected Pythia-6.9B and OLMo-2-7B dictionaries, per-token active counts cluster near their medians (734 and 599), so a small dense-token tail does not drive the averages. Every feature of the four selected dictionaries, and of every Pythia fit in Table D.3, is active, so the lifecycle analyses use all features of each dictionary.

Table C.1: Full dictionary inventory used across the validation figures. ⋆ marks the selected dictionaries analyzed in Sections 3 and 4, each the seed-0 fit of its row, and rows with several fit seeds give ranges over seeds. The **Exp.** column reports the expansion ratio $d_{\text{SAE}}/d_{\text{model}}$ (Pythia-160M: d_{model}=768; Pythia-1B: 2 048; Pythia-6.9B and OLMo-2-7B: 4 096). The **% Act.** column reports $100 \cdot L_0/d_{\text{SAE}}$, i.e. the fraction of the dictionary active per token row. For the distributed 6.9B and OLMo fits, EV and mean L_0 are recomputed over the full vocabulary after training. [a] per-snap SAE family aggregates 32 per-checkpoint SAEs; [b] Pythia-6.9B retrain with $\lambda_1 = 0.6$ (selected dictionary; recipe in Table B.2); [c] Pythia-6.9B run trained at $\lambda_1 = 0.3$ (denser capacity-point); [d] fit with saturation scale 4.0 instead of 1.0.

⋆	Model	Matrix / Family	d_{SAE}	Exp.	Seed	Params	EV	Mean L_0	% Act.	Dead
	160M	W_U cross-snap-32	8 192	10.7×	0–4	403 M	0.776	203.6–203.9	2.49	0.000
	160M	W_U cross-snap-32[d]	16 384	21.3×	0	806 M	0.780	104.9	0.64	0.000
⋆	160M	W_U cross-snap-32	24 576	32×	0–2	1.21 B	0.920	286.4–287.2	1.17	0.000
	160M	W_U final-snap SAE	6 144	8×	—	9.5 M	0.758	864.2	14.07	0.000

continued on next page

⋆	Model	Matrix / Family	d_{SAE}	Exp.	Seed	Params	EV	Mean L_0	% Act.	Dead
	160M	W_U final-snap SAE	8 192	10.7×	—	12.6 M	0.793	1 088	13.28	0.000
	160M	W_U final-snap SAE	16 384	21.3×	—	25.2 M	0.867	1 921	11.72	0.000
	160M	W_U final-snap SAE	32 768	42.7×	—	50.4 M	0.924	3 424	10.45	0.000
	160M	W_U final-snap SAE	65 536	85.3×	—	100.8 M	0.963	5 967	9.10	0.000
	160M	W_U per-snap SAE[a]	8 192	10.7×	per-step	12.6 M	0.729–0.824	925–1 150	11.29–14.04	0.000
	1B	W_U cross-snap-32	8 192	4×	0	1.07 B	0.628	374.8	4.58	0.000
	1B	W_U cross-snap-32	16 384	8×	0	2.15 B	0.781	500.2	3.05	0.000
⋆	1B	W_U cross-snap-32	24 576	12×	0	3.22 B	0.861	516.6	2.10	0.000
⋆	6.9B	W_U cross-snap-32[b]	32 768	8×	0	8.59 B	0.808	741.8	2.26	0.000
	6.9B	W_U cross-snap-32[c]	32 768	8×	0	8.59 B	0.833	1 956.6	5.97	0.000
⋆	OLMo-2-7B	W_U cross-snap-32	32 768	8×	0	8.59 B	0.853	557.7	1.70	0.000

D Sparse Lifecycle and Reorganization Timing

This appendix supports Section 4.1 with the rule-based lifecycle profiles, aggregate timing summaries, the reorganization score, and the stability of these statistics across dictionary fits.

D.1 Rule-Based Lifecycle Profiles

For each feature, let $\rho_f(t) = \|D_f^{(t)}\|_2 / \max_{t'} \|D_f^{(t')}\|_2$ be its decoder norm relative to its own maximum, and map checkpoints to normalized log-step time $\tau = (\log_{10}(t+1) - \log_{10}(t_{\min}+1))/(\log_{10}(t_{\max}+1) - \log_{10}(t_{\min}+1)) \in [0,1]$ over each dictionary's checkpoint range $[t_{\min}, t_{\max}]$. The $+1$ offset keeps step 0 finite, so Pythia's step 0 sits at $\tau = 0$. In Pythia, whose range is steps 0–143k, the early bin $\tau \le 0.35$ covers steps up to 62 (checkpoints 0–32), and the late bin $\tau \ge 0.65$ covers steps from 2,243 (checkpoints 3k–143k). In OLMo-2-7B, whose range starts at step 150 and ends at step 928k, the early bin covers steps up to 3,197 (checkpoints 150–3k), and the late bin covers steps from 43,807 (checkpoints 47k–928k). We summarize each trajectory by early strength $r_f^{\mathrm{early}} = \mathrm{mean}_{\tau \le 0.35} \rho_f(t)$, late strength $r_f^{\mathrm{late}} = \mathrm{mean}_{\tau \ge 0.65} \rho_f(t)$, mid-training peak $r_f^{\mathrm{mid,max}} = \max_{0.35 < \tau < 0.65} \rho_f(t)$, mid-training mean $r_f^{\mathrm{mid,mean}} = \mathrm{mean}_{0.35 < \tau < 0.65} \rho_f(t)$, peak time τ_f^{peak}, alive fraction $\phi_f^{\mathrm{alive}} = \Pr[\rho_f(t) \ge 0.5]$, alive span $\Delta\tau_f^{\mathrm{alive}} = \tau_{\text{last alive}} - \tau_{\text{first alive}}$, and trajectory range $r_f^{\mathrm{range}} = \max_t \rho_f(t) - \min_t \rho_f(t)$. Features are assigned in the order shown in Table D.1; later rows apply only if no earlier row matched, and Figure D.1 aggregates the assignments.

Table D.1: Rule-based lifecycle profile assignment used in Figure D.1. Rules are applied top to bottom, so each active feature receives exactly one descriptive label. The labels "persistent", "early-decaying", "late-emerging", "transitional", and "mixed/ambiguous" are thresholded trajectory descriptions.

Profile	Rule	Meaning
persistent	$(\phi_f^{\mathrm{alive}} \ge 0.70$ and $r_f^{\mathrm{range}} \le 0.35)$ or $\min(r_f^{\mathrm{early}}, r_f^{\mathrm{mid,mean}}, r_f^{\mathrm{late}}) \ge 0.45$	sustained across training
transitional	$r_f^{\mathrm{mid,max}} \ge 0.75$, $r_f^{\mathrm{mid,max}} - \max(r_f^{\mathrm{early}}, r_f^{\mathrm{late}}) \ge 0.35$, $r_f^{\mathrm{late}} \le 0.30$, $\Delta\tau_f^{\mathrm{alive}} \le 0.35$, and $0.35 < \tau_f^{\mathrm{peak}} < 0.65$	rare localized mid-training peak
early-decaying	$r_f^{\mathrm{early}} \ge 0.50$, $r_f^{\mathrm{early}} - r_f^{\mathrm{late}} \ge 0.30$, and $\tau_f^{\mathrm{peak}} \le 0.45$	strong early, weaker late
late-emerging	$r_f^{\mathrm{late}} \ge 0.50$, $r_f^{\mathrm{late}} - r_f^{\mathrm{early}} \ge 0.30$, and $\tau_f^{\mathrm{peak}} \ge 0.55$	weak early, stronger late
mixed/ambiguous	no previous rule matched	residual category

D.2 Aggregate Timing Summaries

In the Pythia dictionaries, the 512–1.6k reorganization window is where early-peaking features decay rather than where features peak. Peaks cluster at steps 16–128 and, in Pythia-160M and 1B, again

late in training, with at most 2% inside the window. Features that peak by step 128 keep a mean ρ_f of at least 0.98 through step 256, which falls by step 3k to 0.29 in Pythia-160M and 0.63 in Pythia-1B as total decoder norm mass also falls.

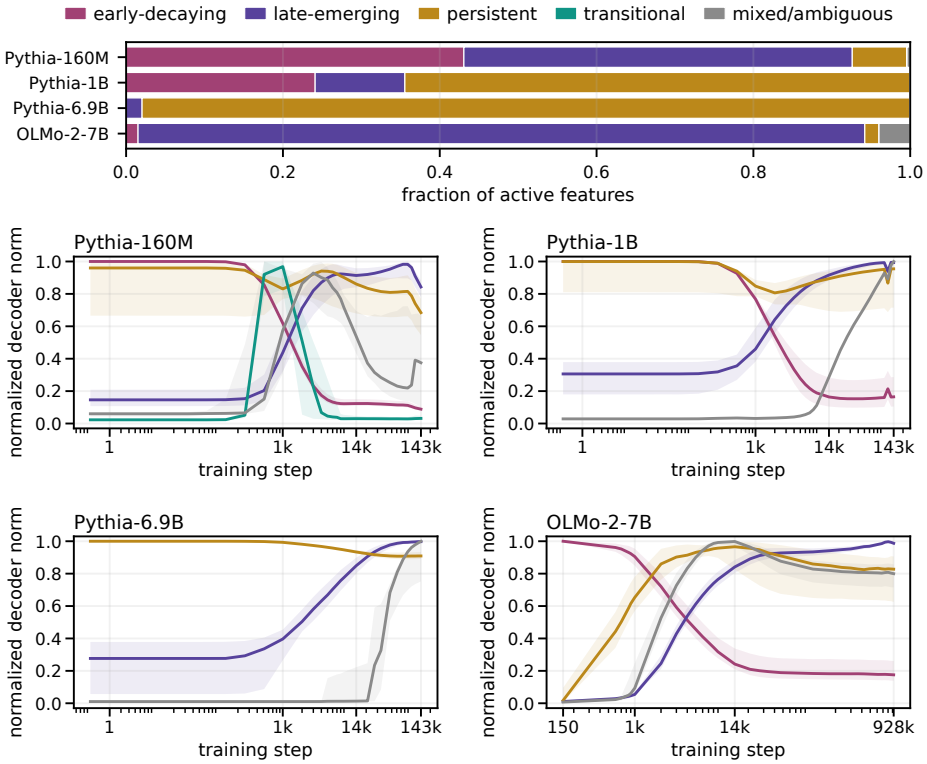

Figure D.1: The four selected cross-snapshot sparse dictionaries contain persistent and late-emerging features, with substantial early-decaying shares in Pythia-160M and Pythia-1B. Sharply transitional mid-training peaks are rare, so readout features do not follow a single monotone growth pattern. Bars give the share of active features in each rule-based profile, and lines give each profile's median trajectory with interquartile bands.

D.3 Late-Start Control for OLMo

Few OLMo-2-7B features peak early and then decay, unlike those of the selected Pythia-160M and Pythia-1B dictionaries. Possible explanations include the missing OLMo-2-7B checkpoints before step 150, more monotone readout development, underrepresentation of weak early-only directions by a checkpoint-spanning dictionary, and differences in tokenizer, data mixture, optimizer, and checkpoint spacing. To test the first, we fit a Pythia-1B W_U crosscoder on a late-starting 32-checkpoint schedule beginning at step 256, which removes the very early checkpoints but none of the other Pythia–OLMo differences.

Without the early checkpoints, the early-decaying profile almost vanishes, at 0.02% of late-start features (τ normalized over steps 256–143k) against 24.1% in the selected Pythia-1B dictionary, whose own trajectories restricted to steps from 256 onward still give 25.4%. The profile therefore arises when the dictionary is fitted together with checkpoints near initialization, with the underlying Pythia model unchanged. Activation-rate definitions agree. Features whose activation rate peaks by step $1{,}000$ and ends below half of its peak make up 0.012% of late-start and 0.015% of OLMo-2-7B features against 6.5% in the selected dictionary, and a looser early-dominant score, $\mathrm{mean}_{t\le 1\mathrm{k}} r_f(t) - \mathrm{mean}_{t\ge 14\mathrm{k}} r_f(t) > 0.2$ for peak-normalized activation rate r_f, gives 0.057% and 0.595% against 41.0%. OLMo's missing early-decaying features are therefore better read as an effect of the checkpoint range than as evidence for a qualitatively different readout lifecycle. The two

trajectory distributions still differ, since 91% of the late-start features peak at the first checkpoint without decaying below half of their peak, whereas OLMo-2-7B features mostly peak late.

D.4 Temporal Localization and Null Models

This subsection defines the reorganization score and window of Section 4.1, which ask whether several non-identical views of readout change in sparse coordinates concentrate in the same training window. The window is a multi-metric co-occurrence peak, and individual change measures can reach their maxima outside it.

For each adjacent checkpoint pair, we define six structural components over the active features (Appendix C): decoder norm turnover, activation rate turnover, weighted adjacent decoder rotation, movement toward the terminal decoder direction, instability of the top feature set, and peak time density. Norm and activation turnover are weighted mean absolute changes in max-normalized trajectories, with pair weight $\max(\|D_f^{(t_i)}\|_2, \|D_f^{(t_{i+1})}\|_2)$. Adjacent rotation and movement toward the terminal decoder direction use weight $\sqrt{\|D_f^{(t_i)}\|_2 \|D_f^{(t_{i+1})}\|_2}$. Movement toward the terminal decoder direction is the absolute change in cosine between $D_f^{(t)}$ and the feature's final decoder direction. Instability of the top feature set is one minus the Jaccard overlap of the top features by decoder norm, using the larger of 10% of active features or 256 features, and peak time density assigns each feature's peak mass to the adjacent pair ending at its peak checkpoint.

We integrate each component over candidate windows of width 0.5 in $\log_{10}(\text{step} + 1)$ by converting adjacent-pair values to rates per log-step interval and summing the overlap with each window. Candidate windows start at each checkpoint whose window ends by the final checkpoint, so their stride follows the checkpoint schedule; the Pythia window starting at step 512 ends at step $1{,}621$. Within each model, each integrated component is min–max normalized, and we define the reorganization score as the geometric mean of the six normalized components. Aggregate decoder-norm mass growth is integrated and plotted as a control, but is excluded from the score.

For nulls, each model uses 200 circular-shift draws. In each draw, the six adjacent-pair component trajectories are independently shifted by nonzero offsets before integration and aggregation. The null-standardized score subtracts the mean of the null scores for each window and divides by their standard deviation, and the selected window maximizes it; empirical one-sided p values use a plus-one correction. To test the choice of window, we resample active features with replacement 400 times per model, recompute the metrics, use 50 circular-shift null draws per resample, and count how often each window is selected.

Table D.2 gives the selected windows, their null probabilities, and how often feature resamples select each window, which shows the Pythia-6.9B window as the least stable of the three.

Table D.2: Windows selected by the reorganization score, in training steps. p is the one-sided circular-shift probability with a plus-one correction over 200 draws, so 0.005 is the smallest attainable value. The last two columns count how many of 400 resamples of active features select the same window and each other window.

Model	Selected window	p	Same window	Other windows (resamples)		
Pythia-160M	512–1.6k	0.005	370	1k–3.2k (30)		
Pythia-1B	512–1.6k	0.005	283	1k–3.2k (117)		
Pythia-6.9B	512–1.6k	0.04	222	9k–28k (134), 8k–25k (1)	14k–44k (33),	1k–3.2k (10),
OLMo-2-7B	9k–28k	0.12	399	900–2.8k (1)		

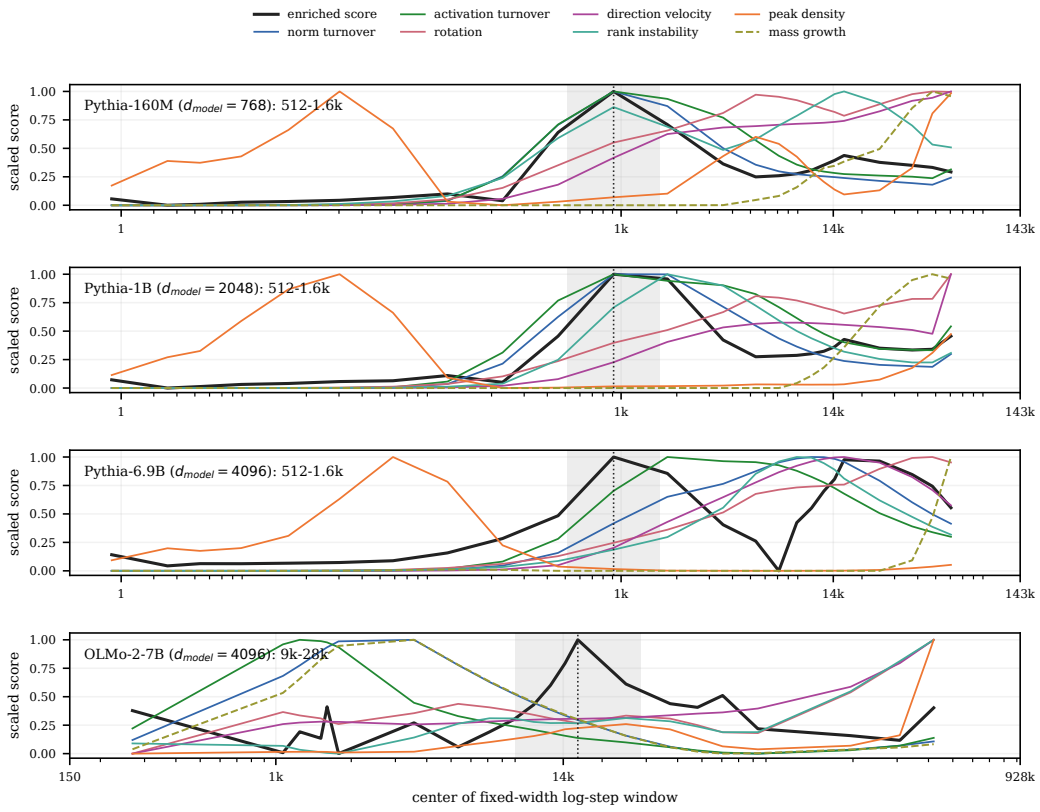

Figure D.2: The three Pythia dictionaries select the same reorganization window, steps 512–1.6k, where several feature change metrics co-occur more than expected under shifted-component nulls. OLMo-2-7B's highest-scoring window, 9k–28k stage-1 steps, is not separated from the nulls (Table D.2). Each curve is plotted at its window's center and min–max scaled within each model. Black curves ("enriched score") show the null-standardized reorganization score, gray bands mark the selected windows, and dotted verticals mark their centers. In the legend, "direction velocity" is movement toward the terminal decoder direction and "rank instability" is instability of the top feature set; the other four components match the text. Mass growth is plotted as a control and is excluded from the score.

D.5 Stability Across Dictionary Fits

To test whether the lifecycle statistics describe the readout or one dictionary fit, we recompute the profile fractions of Table D.1 and the reorganization window, with the same code, rules, and null draws, for all 21 Pythia dictionaries fitted with this architecture on the 32-checkpoint schedule (Table D.3); OLMo-2-7B has one fit.

The reorganization window is stable across fits. Twenty of the 21 fits, covering 12 of the 13 configurations, select steps 512–1.6k, and the exception, Pythia-1B at width 16,384, selects steps 1k–3.2k, the same alternative that feature resamples of the selected Pythia-1B dictionary pick (Table D.2).

The profile fractions and the median feature peak step agree across fit seeds, within about one percentage point and one checkpoint, but move with dictionary settings. Larger sparsity weights, and in Pythia-1B larger widths, assign more features to the persistent profile, and across settings the median peak step ranges from step 0 to step 8k. Every Pythia-160M and Pythia-1B dictionary contains all three profiles, although one of them can fall below 1%, and Pythia-6.9B is almost entirely persistent at both sparsity weights. We therefore read the profile fractions qualitatively, do not attribute differences in them to the models alone, and compare the four 31M dictionaries of Section 4.5 only at their shared width and sparsity weight.

Table D.3: Lifecycle statistics across dictionary fits. Profile shares are percentages of active features; rows with several fit seeds give the range over seeds. The window is the one selected by the reorganization score, and p is its circular-shift probability (200 draws). *Selected dictionaries of the main text. †Fit with penalty saturation scale $c = 4$ instead of $c = 1$. The rows with λ_1 from 0.4 to 1.8 come from a separate sweep that also used an auxiliary loss on dead features.

Model	Width	λ_1	Seeds	Early-decaying	Persistent	Late-emerging	Window	p
Pythia-160M	8,192	0.3	5	45.6–46.1	0.5–0.9	52.5–53.0	512–1.6k	0.005
Pythia-160M†	16,384	0.3	1	7.8	80.9	11.2	512–1.6k	0.005
Pythia-160M*	24,576	0.3	3	42.9–43.1	6.9	49.5–49.7	512–1.6k	0.005
Pythia-160M	8,192	0.4	1	43.2	1.2	55.0	512–1.6k	0.005
Pythia-160M	8,192	1.0	1	27.8	43.4	28.3	512–1.6k	0.005
Pythia-160M	8,192	1.2	1	22.3	57.5	19.7	512–1.6k	0.005
Pythia-160M	8,192	1.35	3	17.8–18.1	66.2–67.3	14.4–15.2	512–1.6k	0.005
Pythia-160M	8,192	1.8	1	6.8	82.7	10.0	512–1.6k	0.005
Pythia-1B	8,192	0.3	1	41.4	21.7	36.9	512–1.6k	0.010
Pythia-1B	16,384	0.3	1	34.0	44.8	21.2	1k–3.2k	0.005
Pythia-1B*	24,576	0.3	1	24.1	64.4	11.4	512–1.6k	0.005
Pythia-6.9B	32,768	0.3	1	0.0	98.1	1.9	512–1.6k	0.04
Pythia-6.9B*	32,768	0.6	1	0.0	98.0	2.0	512–1.6k	0.04

E WordNet Supersense Probe Analyses

This appendix gives the protocol and results of the Pythia-160M WordNet analysis of Section 4.2 and its Pythia-1B replication.

E.1 Probe Hyperparameters

Each WordNet category is probed one-vs-rest on the mean-centered W_U rows. Its positives are the category's clean token ids, and its negatives are all other vocabulary rows, including the tokens of other categories, with class weights that balance the two classes. Each category uses one stratified split of vocabulary rows into two-thirds train and one-third held-out test, sampled with `StratifiedShuffleSplit` and run seed 0 [Pedregosa et al., 2011]. The split depends only on the labels and the seed, so each category keeps the same train and test rows at every checkpoint. This row split assigns token ids independently, so case and spacing variants of one lemma (␣`dog`, `dog`, ␣`Dog`) can fall on both sides of it. Appendix E.6 repeats the probe with the split grouped by lemma. All other probe results in this appendix, including every figure and table, use the row split. Within the training split, an inner 3-fold stratified cross-validation selects $C \in \{0.1, 1, 10\}$. The selected classifier is refit on the full training split and evaluated once on the held-out split. The reported WordNet analysis uses a batched GPU implementation that matches `LogisticRegression(solver='liblinear', class_weight='balanced')` [Pedregosa et al., 2011] to within a few thousandths in agreement checks, with `max_iter=80`. All reported probe values are held-out balanced accuracy.

E.2 WordNet Supersense Inventory

For each WordNet synset [Miller, 1995, Fellbaum, 1998], each one-word ASCII alphabetic lemma with length at least three is assigned to a lexicographer file only if that file accounts for at least 60% of the lemma's synset occurrences. Each surviving lemma is mapped through five surface variants (lowercase and capitalized, each with and without a leading space, and all capitals), and a variant is kept only if it is a single Pythia token. A kept token is clean if, after its leading space is stripped, it decodes to at least three ASCII alphabetic characters and it has a leading space, an initial capital, or all capitals, which excludes word-internal fragments. A category is probed only if at least 20 clean token ids survive. This leaves 35 probed categories out of 45 WordNet lexicographer files. The skipped support-limited files are: `adj.ppl` (15 clean token ids), `noun.Tops` (13), `noun.motive` (3), `noun.shape` (9), `verb.body` (8), `verb.competition` (8), `verb.consumption` (11), `verb.emotion` (12), `verb.perception` (16), and `verb.weather` (1).

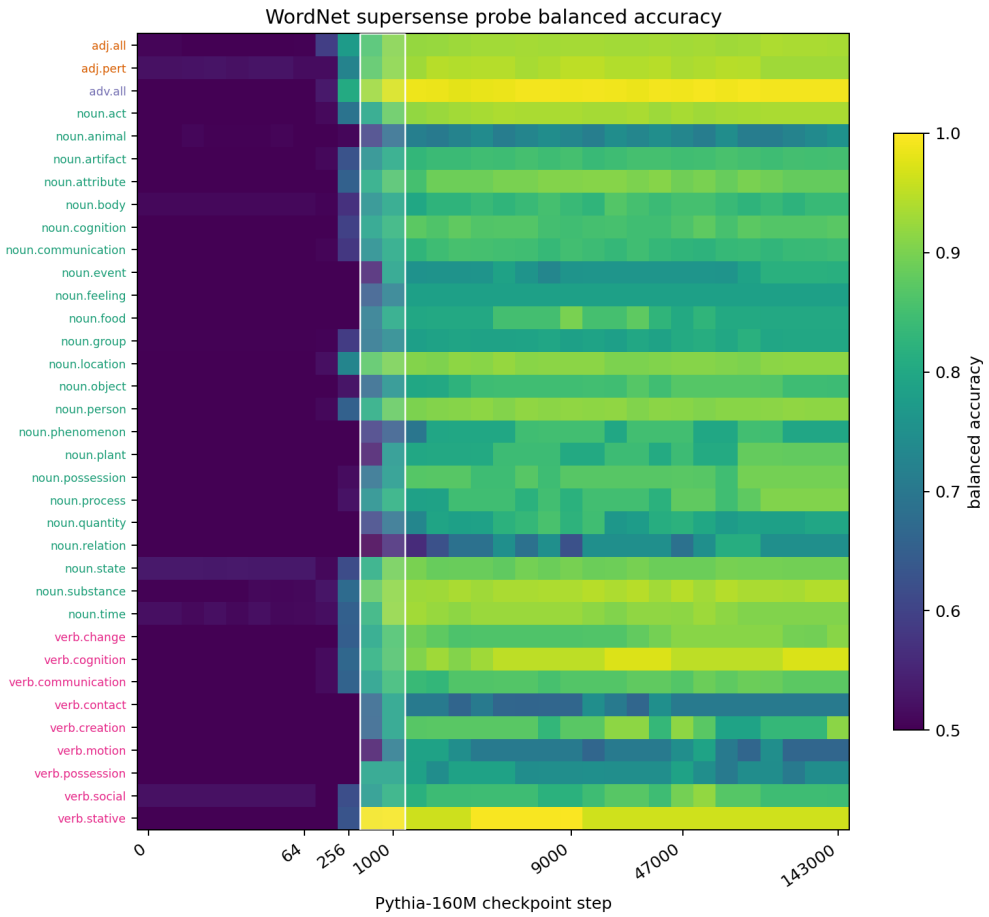

Figure E.1: In Pythia-160M, all 35 WordNet supersense categories reach sustained decodability in the readout rows between steps 128 and 1000, and 33 have their largest adjacent-checkpoint gain in that interval. Rows are the 35 lexicographer-file categories with at least 20 clean token ids, in alphabetical order, and color is the held-out balanced accuracy of a one-vs-rest linear probe on centered W_U rows under the row split of Appendix E.1. Columns are the 32 Pythia checkpoints, equally spaced by index. The white box marks steps 512 and 1000, the checkpoints inside the 512–1.6k reorganization window.

E.3 Matched-Control Analysis

A matched-control analysis repeats the same probe procedure at all 32 Pythia-160M checkpoints. It compares observed WordNet labels with three controls. The metadata-only baseline is a probe on token features alone. These are the log count of the token in a held-out evaluation token set (not its training-corpus frequency), its character length, the norm of its centered row at the same checkpoint, spacing (whether the token begins with a space), special-token status, and a coarse token class (special, whitespace, lowercase, uppercase, or mixed-case alphabetic, digit, punctuation, other ASCII, or non-ASCII). The matched-random control replaces the category with tokens drawn from the same frequency, length, row norm, spacing, special-token, and token class bins, relaxing the match when a bin is empty, and the label-shuffle control shuffles labels within these bins. The null probes draw new token sets and splits at every checkpoint. The matched-random and label-shuffle controls use 20 permutations per category and null type, so they are interpreted as a confound check for this analysis. With 20 permutations the smallest attainable permutation p-value is $1/21 \approx 0.048$, so $p \leq 0.05$ means that the observed accuracy beats all 20 null values. The 95th-percentile threshold used below interpolates between the two largest null values and is slightly less strict.

E.4 Probe Timing Trajectories

A category's onset is the first checkpoint at which it reaches sustained decodability, meaning balanced accuracy of at least $\max(0.55, \mathrm{acc}_0 + 0.05)$ at that checkpoint and the next two. In Pythia-160M, lexical organization appears within a short interval (Figure E.1). Onsets fall between steps 128 and 1000, 19 of them at step 256, and 33 of 35 categories have their largest adjacent-checkpoint gain in that interval, whose three adjacent pairs a category's largest gain would hit with probability 0.10 if it fell on a uniformly random pair of the grid. Median probe accuracy starts near the matched-random and label-shuffle nulls at step 0, and with each category's threshold set to the larger of its two 95th-percentile nulls, it exceeds the median threshold by 0.170 at step 512 and 0.272 at the final checkpoint.

E.5 Pythia-1B Replication

The timing replicates on the Pythia-1B W_U trajectory, the model of the localization analysis in Section 4.4, with the same 35 categories and probe settings under the row split. Onsets range from step 128 to step 2000 with a median of 512, and 33 of 35 categories have their largest gain in the four adjacent pairs from $128 \to 256$ through $1000 \to 2000$, against 0.13 expected if each category's largest gain fell on a uniformly random pair. Neither the metadata-only baseline nor the matched-random and label-shuffle nulls were run on Pythia-1B, so these are observed probe accuracies only.

E.6 Lemma-Grouped Splits

Under the row split, 1199 of the 3249 positive test tokens (37%) have a variant of the same lemma in the training split. We therefore rerun the probe with every vocabulary row grouped by its lowercased, space-stripped decoded string. The outer split is `StratifiedGroupKFold` with three folds and seed 0, holding out one fold as the test set, and the inner selection of C uses the same grouped scheme on the training split. No positive lemma then appears on both sides. The categories, token ids, probe settings, and 32 checkpoints are unchanged, and each category keeps a test support within one token of the row split.

In Pythia-160M, median balanced accuracy rises from 0.500 at step 0 to 0.841 at the final checkpoint, against 0.874 under the row split. Of the 35 categories, 34 reach sustained decodability by step 1000, 33 still have their largest adjacent-checkpoint gain between steps 128 and 1000, and the median onset stays at step 256. The category that never separates is `noun.relation`, whose 24 token ids come from 14 lemmas. It stays at 0.500 at every checkpoint under the grouped split, so its accuracy under the row split came from variants of the same lemma falling on both sides of the split. Across split seeds 1 to 4, the median per-category difference between accuracy under the lemma-grouped split and under the row split is about -0.04 at the final checkpoint and lies within ± 0.005 at steps 256 and 512. In Pythia-1B, the median onset stays at step 512, onsets range from step 128 to step 1000, and one category never separates. The final median is 0.060 lower than under the row split. The matched-random and label-shuffle nulls and the metadata-only baseline were not rerun under the grouped split. Against the Pythia-160M nulls computed under the row split, 32 of 35 categories under the lemma-grouped split exceed both 95th percentiles at step 512, and 34 do from step 1000 to the final checkpoint.

F Availability and Expression Across Contrastive Task Families

This appendix extends the probe and readout comparison to a broader suite of 20 task families over 24 checkpoints of Pythia-1B and Pythia-6.9B. The 32-checkpoint logistic probe trajectories for SVA and IOI are in Figure F.1.

Measurements. All measures read the final normalized hidden state. The 32-checkpoint SVA and IOI trajectories use the sweep B_{env} and the broader task suite uses B_{ho} (Appendix A.3), and both exclude the native and terminal readouts, so neither need exceed E. Most open-vocabulary families have no shared binary label, so Table F.3 reports them by expression only. The BLiMP (Benchmark of Linguistic Minimal Pairs) anaphor and determiner–noun agreement families do have one, and Appendix F.4 uses the anaphor family as a contrast case.

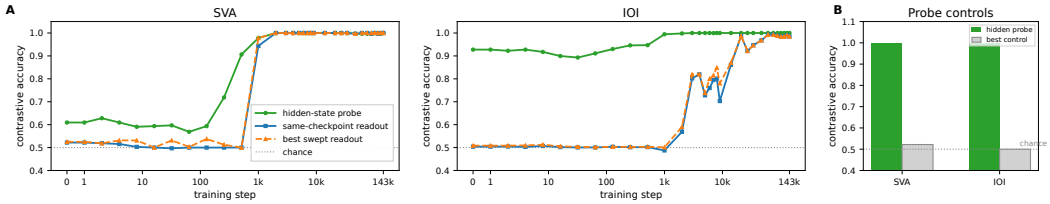

Figure F.1: In the 32-checkpoint Pythia-6.9B trajectories for SVA ($n = 320$) and role-balanced IOI ($n = 2{,}000$), the logistic hidden state probe reaches high accuracy before native contrast accuracy rises, and on SVA the best swept readout tracks the native one. The best swept readout is the post hoc envelope B_{env}. IOI probe accuracy is already 0.928 at initialization, so the entire early probe–expression gap cannot be attributed to learning. All three curves in panel A use the same task inventory, and panel B compares the probe with the strongest prompt-token or Gaussian-feature control.

F.1 Prompt Construction

The tasks of Table 1 and the other binary families of the broader task suite are built from fixed templates and word lists, and every item has a correct answer and a foil that are both single tokens with a leading space. The prompt stops where the answer would begin, and we score the margin at its last token. Table F.1 gives the templates. The availability task suite and the localization items were built separately, with different templates for IOI and numeric comparison.

Availability task suite. Simple agreement crosses 20 noun lemmas (`key`, `book`, `student`, and others) in singular and plural form with eight fixed prepositional phrases whose nouns are singular (`to the cabinet`, `near the wall`, and six more), so every plural subject is followed by a singular attractor. The across-PP set pairs each singular item with a plural item that has the same head lemma and verb pair. Its heads are 30 person nouns, its prepositions are `by`, `near`, `beside`, `behind`, `next to`, `across from`, and `in front of`, its attractors are singular place and object nouns or plural person nouns, and its answers come from 12 verb pairs; head and attractor number differ in 966 of 2,000 items. The relative clause sets follow the constructions of Marvin and Linzen [2018]. They use 20 person nouns as heads and attractors (never the same noun twice in one item), six relative clause verbs, and five main verb pairs (`likes/like`, `runs/run`, `knows/know`, `writes/write`, `reads/read`). The two sets contain the same 1,200 combinations of head, attractor, and verbs and differ in clause structure, so the clause verb agrees with the head in a subject relative clause and with the attractor in an object relative clause. Role-balanced IOI uses two of 22 single-token first names and one of eight objects. For each name pair and object it produces both role orders, with a filler sentence naming the recipient (`Jane waited.`), so the two prompts of a pair contain the same tokens and only their order marks the answer. The three numeric sets use the same 1,000 unordered pairs of the numbers 2 to 99, each in both orders, so the correct answer comes first in half of the comparison items and `True` is correct in half of the true/false items.

Localization items. A separate builder produced the localization items with the Pythia tokenizer and the terminal row norm filter of Appendix A.3. Simple agreement uses the same 320 prompts. IOI takes every ordered pair of the 22 names once, with a random object, in the template without the filler sentence, so the recipient is always the second name, and all 462 pairs pass the norm filter. In place of numeric comparison, localization uses a greater-than completion task that asks for a number smaller than the one in the prompt (`84 is greater than`, with answers `64`/`84`), so it contrasts a smaller number with the prompt's own number. The builder draws 2,000 ordered pairs of numbers from 2 to 99 without repeating a draw, but both orders of a pair give the same item, so an item can occur twice, and each of the at most 97 prompts recurs with different smaller numbers. Relational facts keep the 26 of 30 country and capital pairs that pass the single-token and norm filters. The foil is the first other capital in a fixed list that passes the filters, which makes `Paris` the foil for most items, so this family largely contrasts each capital with one token.

Splits. Availability probes use the grouped folds of Appendix A.3. Localization splits each task's items into halves at random at the item level (split seed 0), so the selection and scoring halves share answer tokens and, for greater-than completion, prompts and occasionally whole items, since that builder can produce an item twice. The held-out half therefore tests whether the selected features carry the margin on other items built from the same answer tokens, and it does not test transfer to new tokens.

Table F.1: Prompt templates. Braces mark slots filled from the word lists in the text, and answers are the correct token and the foil, each with a leading space. In role-balanced IOI, `{R}` is the recipient and `{S}` the giver, one each of `{A}` and `{B}`. Set A is the availability task suite and set L the localization items.

Task	Template	Answers	Set	n
Simple SVA	`The {noun} {prepositional phrase}`	`are/is`	A, L	320
Across-PP SVA	`The {head} {preposition} the {attractor}`	agreeing verb / other form	A	2000
Subject-RC SVA	`The {head} that {clause verb} the {attractor}`	agreeing main verb / other form	A	1200
Object-RC SVA	`The {head} that the {attractor} {clause verb}`	agreeing main verb / other form	A	1200
IOI	`{A} and {B} went to the store. {R} waited. {S} gave the {object} to`	`{R}/{S}`	A	2000
IOI	`{A} and {B} went to the store. {A} gave the {object} to`	`{B}/{A}`	L	462
Numeric $>$	`Which is greater, {x} or {y}? Answer:`	larger / smaller number	A	2000
Numeric $<$	`Which is smaller, {x} or {y}? Answer:`	smaller / larger number	A	2000
Numeric T/F	`{x} is greater than {y}. True or False? Answer:`	`True/False` by truth value	A	2000
$>$ completion	`{x} is greater than`	smaller number / `{x}`	L	2000
Relational facts	`The capital of {country} is`	capital / another capital	L	26

F.2 Task Items and Final Checkpoint Results

Table F.2 reports the binary families at the final checkpoint, and Table F.3 reports the open-vocabulary and control families, which reach high contrast accuracy. The BLiMP anaphor family combines the BLiMP anaphor gender and anaphor number agreement sets, 1,000 items each. Its probe label is the gender of the reflexive, so A_1 is fit only on the 1,455 items whose correct token is `herself` or `himself`, while E, B_{ho}, and the rank-1 rate use all 2,000 items.

Table F.2: Binary families at the final checkpoint (Pythia-6.9B), with the 1B gap and regime. $A_1@0$ is the step-0 initialization baseline, and the across-PP value of 0.43 lies below chance for a reason the stored runs do not record. B is B_{ho}, and "unreach" $= (A_1 - B)/(A_1 - E)$ is the fraction of the gap that the evaluated intermediate readout does not reach, computed from unrounded values. The Pythia-1B numeric probes use stratified folds without number pair grouping. A gap is persistent when $A_1 - E$ exceeds 0.05 at the final checkpoint and transient when it ends at or below 0.05 after exceeding 0.30 while E was below 0.7.

family	n	$A_1@0$	A_1	E	B	$A_1 - E$	unreach	$A_1 - E$(1B)	regime
SVA simple	320	0.60	1.00	1.00	1.00	0.00	–	0.00	transient
SVA across-PP	2000	0.43	1.00	0.97	0.98	0.03	63%	0.03	transient
SVA subject-RC	1200	0.73	1.00	0.75	0.79	0.25	85%	0.21	persistent
SVA object-RC	1200	0.73	1.00	0.91	0.93	0.10	72%	0.13	persistent
IOI	2000	0.92	1.00	1.00	1.00	0.00	–	0.00	transient
numeric $>$	2000	0.86	1.00	0.63	0.82	0.37	47%	0.55	persistent
numeric $<$	2000	0.86	1.00	0.45	0.66	0.55	62%	0.40	persistent
numeric T/F	2000	0.89	0.99	0.51	0.66	0.48	70%	0.48	persistent

Table F.3: Open-vocabulary and control families (Pythia-6.9B), with a truncated example prompt and the contrast tokens y^+/y^- (leading space shown). The BLiMP families come from Warstadt et al. [2020] and the function vector (FV) families, which are in-context prompts, from Todd et al. [2024]; BLiMP probe folds are grouped by antecedent name for anaphor agreement and by singular noun lemma for determiner–noun agreement. Apart from these two BLiMP agreement families, these carry no binary feature, so A_1 is undefined; expression is reported by contrast accuracy E, the swept accuracy $B = B_{\mathrm{ho}}$, and the full vocabulary rank-1 rate of y^+. NPI is negative polarity item licensing, and for BLiMP anaphor n counts the gender-labeled items of Appendix F.2. The control families are built from single-token nouns drawn at random. Fixed-token control prompts list two to five distinct nouns, and the answer is always `the` against `,`, so the context carries no information about the answer. Copy control prompts have the form $a\,a\,b\,b\,c$, with answer c and a foil absent from the prompt. Induction prompts have the form $a\,b\,a\,b\,a$, with answer b and a foil absent from the prompt. Relational (bal.) gives each of 24 countries five foils drawn at random from the other capitals, whereas relational facts uses the fixed foil list of the localization items (Appendix F.1).

family	example prompt (truncated)	y^+/y^-	n	E	B	rank1
BLiMP anaphor	Katherine can't help	`herself/ himself`	1455	0.99	0.99	0.02
BLiMP det-noun	Raymond is selling this	`sketch/ sketches`	838	0.92	0.94	0.03
BLiMP NPI	Teresa had	`not/ probably`	656	0.98	0.97	0.15
copy control	valley valley anchor anchor zebra	`zebra/ jacket`	2000	0.98	0.98	0.01
fixed-token control	mirror thunder cookie music island	`the/ ,`	2000	0.98	0.90	0.00
FV antonym	selfish:altruistic / note:silence / known…	`perfect/ socialist`	2000	1.00	1.00	0.00
FV country-capital	Monaco:Monaco-Ville / Senegal:Dakar / Slo…	`Vienna/ Moscow`	36	1.00	0.94	0.00
FV en-fr	vector:vecteur / indonesia:Indonésie / ga…	`ici/ tous`	435	1.00	1.00	0.00
FV present-past	demonstrate:demonstrated / treat:treated …	`accelerated/ fastened`	288	1.00	1.00	0.00
induction	lemon burner lemon burner lemon	`burner/ basket`	2000	0.98	0.98	0.70
relational facts	The capital of France is	`Paris/ Berlin`	26	1.00	1.00	0.96
relational (bal.)	The capital of France is	`Paris/ Ottawa`	120	1.00	0.98	0.96

F.3 Initialization, Probe, and Answer-Token Controls

The step-0 probe ($A_1@0$ in Table F.2) separates information that becomes decodable during training from information already decodable at initialization, so each gap is read against it. On the three numeric families it already reaches 0.86 to 0.89, consistent with the probe learning a value for each number from its token identity, since folds grouped by number pair still share individual numbers. The numeric rows therefore measure a persistent expression gap and do not show that the model encodes the comparison.

Shuffled-label fits stay below 0.56 on the agreement, IOI, and numeric families of Table 1 and reach 0.585 on BLiMP anaphor agreement at Pythia-6.9B, and prompt-token and matched-dimension Gaussian-feature probes control for simple task cues in the 32-checkpoint trajectories. The availability result holds under both fitting choices, since at step 1000 in Pythia-6.9B the shrinkage-LDA fit A_1 and the logistic fit A_{full} (Appendix A.3) differ by at most 0.03 on the eight binary families. Both fit a single direction, so this comparison checks the fitting method, and neither is trained with the model's next-token objective.

Answer-token norm and rarity percentiles correlate with the final gap at $+0.23$ and $+0.36$ across the eight families, a sample too small to show that geometry or frequency is irrelevant. The subject and object relative clause sets give a more direct check, because they share their verb pairs. At the end of training, native expression is higher for object than for subject relative clauses while both probes are at ceiling, so the answer tokens alone do not set the gap.

F.4 Transient and Persistent Gaps

In Pythia-6.9B, simple and across-PP agreement and IOI close their gaps during training, while the relative clause and numeric gaps persist to the final checkpoint (Figure F.2).

Among the families whose step-0 A_1 lies within 0.1 of chance, so that any gap reflects information learned during training (Table F.4), every subject–verb agreement row has A_1 near ceiling at step 512 while native and swept expression are at or below chance, whereas in BLiMP anaphor agreement A_1 and native expression rise together. For simple and across-PP agreement, native expression has risen by step 1000, the next checkpoint of this grid, so the lead spans a single checkpoint that the grid cannot resolve.

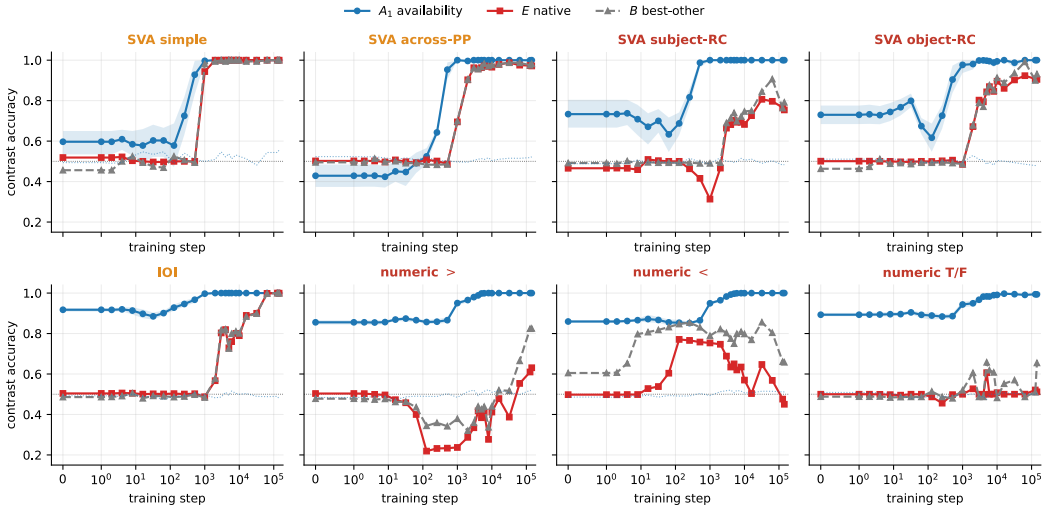

Figure F.2: Across the eight binary families in Pythia-6.9B, agreement and numeric tasks differ in both initialization baselines and final checkpoint gaps. Panel titles are colored by the regime of Table F.2, orange for a transient gap and red for a persistent one. B (best-other in the legend) is B_{ho}. Shaded bands are 95% percentile bootstrap intervals for A_1 over its five per-fold accuracies, and dotted curves show the label-shuffle control (blue) and chance (black). Native expression on numeric $<$ rises to about 0.75 between steps 128 and 2000 and falls to 0.45 at the final checkpoint, so this family's final gap is larger than its gap in mid-training.

Table F.4: Families whose probe starts within 0.1 of chance, from the broader task suite (24 checkpoints; A_1 is the shrinkage-LDA probe and B is B_{ho}). For BLiMP anaphor, n counts the items with a gender label used by A_1, and E and B use all $2{,}000$ items.

				step 512			step 2000		final
model	family	n	A_1@0	A_1	E	B	A_1	E	E
6.9B	SVA simple	320	0.60	0.93	0.50	0.49	1.00	1.00	1.00
6.9B	SVA across-PP	2000	0.43	0.95	0.49	0.48	1.00	0.90	0.97
6.9B	BLiMP anaphor	1455	0.60	0.63	0.48	0.48	0.85	0.90	0.99
1B	SVA simple	320	0.45	0.97	0.50	0.49	1.00	1.00	1.00
1B	SVA across-PP	2000	0.53	0.98	0.48	0.48	1.00	0.93	0.97
1B	SVA subject-RC	1200	0.56	1.00	0.22	0.39	1.00	0.55	0.79

F.5 Contrast Accuracy and Full-Vocabulary Prediction

A two-token contrast does not measure whether either candidate is the most likely token over the full vocabulary. At the final Pythia-6.9B checkpoint, native contrast accuracy on subject-relative-clause agreement is 0.75, but the gold verb's rank-1 rate is zero. Numeric greater-than, by contrast, has a rank-1 rate near its contrast accuracy. The rank-1 rates in Table F.3 likewise separate strong contrast scores from generation, most clearly in the function vector families, which reach $E = 1.00$ with rank-1 rate 0.00.

G Behavioral and Intervention Readout Checks

This appendix documents the corpus-level readout swaps and the sparse feature interventions behind Section 4.4. Swaps exchange W_U while keeping hidden states and targets fixed, and feature ablations remove selected reconstructed contributions to W_U rows; neither intervenes on upstream circuits.

G.1 Swap Grid Setup and Aggregate Measures

The readout-swap grid evaluates Pythia-1B on a fixed held-out corpus of 173,229 next-token predictions. The corpus consists of nine 20,000-character language slices from the 2023-11-01 Wikimedia Wikipedia dumps [Wikimedia Foundation, 2023a], tokenized with the Pythia tokenizer and covering English, Russian, Chinese, Japanese, Thai, Arabic, Hindi, Korean, and Bengali. Each slice concatenates articles of at least 200 characters in dump order until it reaches 20,000 characters, from at most 50 articles, and is then truncated to 20,000 characters; the number of articles in each slice was not recorded. For each hidden-state checkpoint $t \in \{256, 512, 1000, 2000, 3000, 14000, 47000, 143000\}$, we record the normalized input $h_p^{(t)}$ to the output layer at every held-out position p and score it through the readouts $W_U^{(s)}$ of 32 released checkpoints from step 0 through step 143k, with logits $h_p^{(t)} \left(W_U^{(s)}\right)^\top$ and no sparse dictionary or feature edit (Figure G.1). A replication on Pythia-160M uses the same corpus and hidden-state checkpoints $t \in \{256, 512, 1000, 2000, 143000\}$ with the full 32-step readout schedule. It shows the same qualitative pattern, with many later readouts decoding early hidden states without large NLL spikes and later hidden states preferring nearby readouts, and it lacks the step-116k spike (Appendix G.3).

G.2 Corpus Composition and Per-Language Swaps

The swap corpus is mostly non-English, whereas Pythia was trained mostly on English. English supplies only 2.3% of the scored targets (Table G.1), while the Pile is about 97.4% English by its authors' fastText estimate and includes a EuroParl component covering 21 European languages [Gao et al., 2020]. Pythia's BPE tokenizer, trained on the Pile [Black et al., 2022], operates on bytes and so can encode text in any script [Radford et al., 2019], and small amounts of non-English text in nominally English pretraining corpora correlate with cross-lingual ability, most strongly in RoBERTa [Blevins and Zettlemoyer, 2022], so next-token prediction on these languages is measurable in Pythia. Because the aggregate NLL in Figure G.1 describes mainly non-English text, we also report it per language.

Table G.1: Readout swap gains by language for Pythia-1B. For hidden states from checkpoint t, s^* is the readout checkpoint with the lowest NLL, and the gain is native NLL minus the NLL under $W_U^{(s^*)}$ in nats, so a gain of 0 means that the native readout is best. Targets are scored next-token positions and share is their percentage of all targets; the aggregate is the token-weighted mean over all targets, including 8 targets whose context token belongs to the preceding language slice. Brackets give 95% paired bootstrap intervals from 2,000 resamples of 256-token blocks within each slice, which do not measure variation across documents, and † marks a bootstrap probability below 0.95 that s^* is the minimum.

			$t = 256$		$t = 512$		$t = 1000$	
Language	Targets	Share (%)	s^*	Gain	s^*	Gain	s^*	Gain
Aggregate	173,229	100.0	1000	1.503 [1.455, 1.548]	1000	0.147 [0.141, 0.153]	1000	0
English	3,939	2.3	512	0.048 [0.026, 0.070]	1000	0.064 [0.039, 0.089]	2000†	0.002 [−0.021, 0.026]
Russian	10,639	6.1	512	1.033 [1.007, 1.060]	1000	0.114 [0.090, 0.139]	1000†	0
Chinese	25,330	14.6	1000	1.251 [1.176, 1.305]	1000	0.134 [0.123, 0.146]	1000	0
Japanese	20,698	11.9	2000	1.148 [1.090, 1.195]	1000	0.192 [0.178, 0.207]	2000	0.013 [0.002, 0.024]
Thai	22,192	12.8	2000	2.542 [2.492, 2.585]	1000	0.223 [0.212, 0.233]	1000	0
Arabic	14,477	8.4	1000	1.768 [1.732, 1.802]	1000	0.149 [0.139, 0.158]	1000	0
Hindi	19,853	11.5	1000	2.042 [2.008, 2.078]	1000	0.139 [0.124, 0.152]	2000†	0.003 [−0.009, 0.014]
Korean	29,118	16.8	512	1.474 [1.441, 1.505]	1000	0.189 [0.174, 0.203]	1000	0
Bengali	26,975	15.6	2000	1.976 [1.944, 2.007]	1000	0.045 [0.037, 0.054]	1000	0

English gains far less from a swap than the aggregate (Table G.1). For step-256 hidden states, English is best decoded by the step-512 readout, and the aggregate's best readout, from step 1000, raises English NLL by 0.726 nats $[0.677, 0.779]$ over the native readout. For step-512 states both reach their best at the step-1000 readout, and from step 1000 onward no swap improves English NLL beyond its interval of zero.

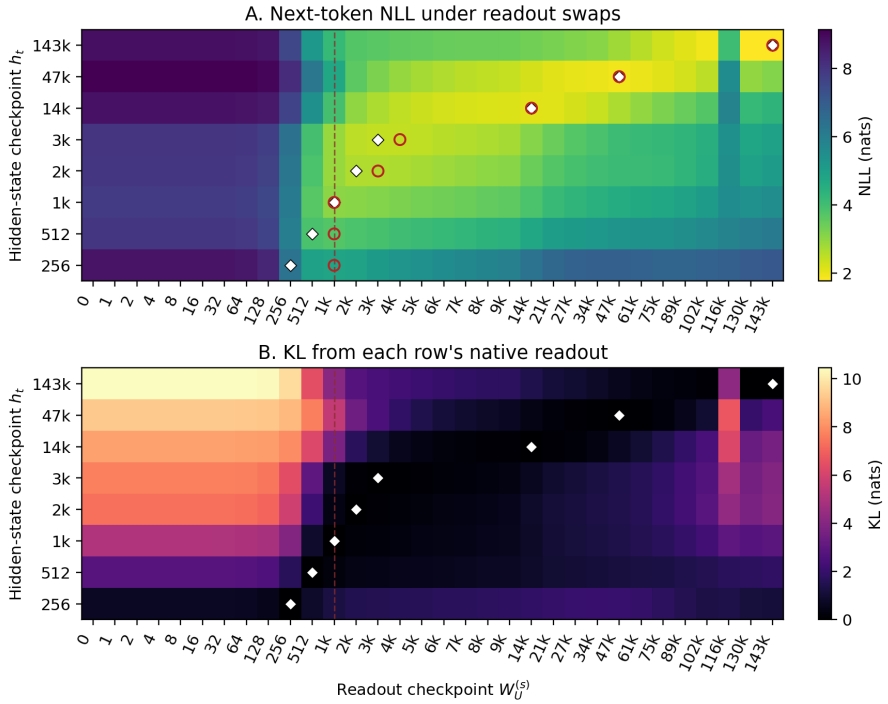

Figure G.1: In the token-weighted aggregate for Pythia-1B, the step-1000 readout is the NLL optimum for hidden states from steps 256, 512, and 1000, while later hidden states prefer nearby or same-checkpoint readouts and become more readout-specific. Per language, step 1000 is the shared optimum only for step-512 states (Appendix G.2). Rows are hidden state checkpoints t (the axis label h_t denotes $h^{(t)}$), and columns are readout checkpoints s. White diamonds mark same-checkpoint readouts, red circles mark the row-wise minimum-NLL readout, and the dashed vertical line marks the step-1000 readout column. Panel B shows $\mathrm{KL}(\mathrm{softmax}(h_p^{(t)} \left(W_U^{(s)}\right)^\top) \,\|\, \mathrm{softmax}(h_p^{(t)} \left(W_U^{(t)}\right)^\top))$, so dark cells are close to the same-checkpoint predictive distribution for that hidden-state row. The bright column near 116k is the released-checkpoint anomaly of Appendix G.3.

The size of the gain depends on the language (Table G.1 and Figure G.2). For step-256 hidden states, the seven languages written in neither Latin nor Cyrillic script supply 97.2% of the aggregate gain at the step-1000 readout. With one Latin-script language in the corpus, language, script, training share, and tokenization cannot be separated.

The location of the best readout agrees across languages only in a weaker form (Table G.1). Every language prefers an intermediate readout for step-256 and step-512 states, but the step-256 optima differ by language, each with bootstrap probability at least 0.997 of being the minimum, and from step 1000 onward each language prefers its native readout or the next checkpoint, with gains of at most 0.109 nats.

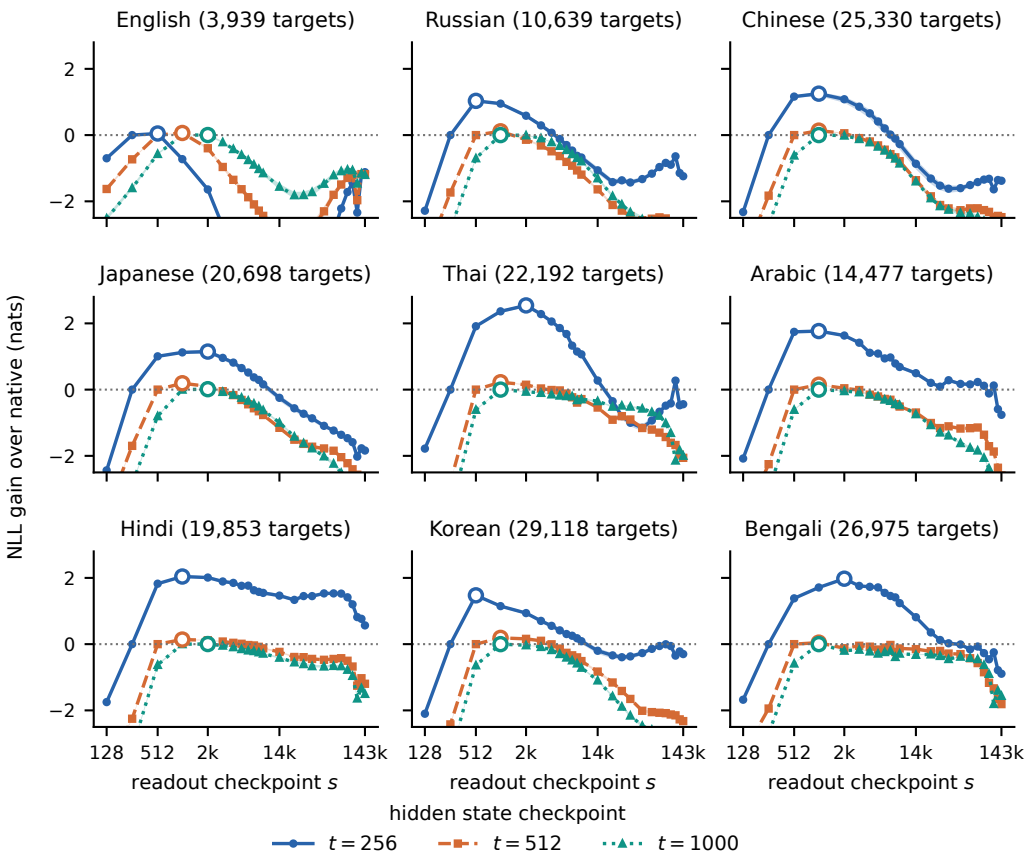

Figure G.2: In Pythia-1B, the readout swap NLL gain and the best readout checkpoint for step-256 hidden states vary by language, while every language prefers an intermediate readout to its native and terminal readouts; for step-512 states, all nine languages share the step-1000 optimum. The gain is native NLL minus swapped NLL, so positive values favor the swapped readout. Open circles mark each curve's best readout, bands give 95% block-bootstrap intervals, and values below -2.5 nats are clipped.

G.3 Step-1000 Swap Optimum and Late Checkpoint Anomaly

Later hidden states become readout-specific (Figure G.1). States from steps 2000 and 3000 narrowly prefer the next checkpoint's readout in the aggregate, and in three and five of the nine languages, and late states stay close to their same-checkpoint distribution only under nearby late readouts, whereas early states stay close under many later ones. Terminal W_U is therefore not a neutral logit lens decoder for early checkpoints.

The late spike is the Pythia-1B step-116k readout anomaly. For fixed step-47k hidden states, NLL rises from 2.50 nats under $W_U^{(102k)}$ to 5.55 under $W_U^{(116k)}$ and returns to 2.95 under $W_U^{(130k)}$; for step-143k states, the values are 1.93, 4.06, and 1.78. The spike therefore comes from the step-116k readout itself, since it appears for fixed hidden states from two checkpoints and disappears under the next readout, and the Pythia-160M replication lacks it, consistent with an anomaly specific to the released Pythia-1B checkpoint. The same checkpoint stands out in reconstruction. A baseline that interpolates each row along the line between its first and last checkpoints drops to EV 0.13 there, while the dictionary loses only about 0.10 EV. The held-out heads deviate only at the neighboring steps 113k and 123k (Figure C.7), and no single token stratum explains the dip in aggregate EV (Appendix C.2).

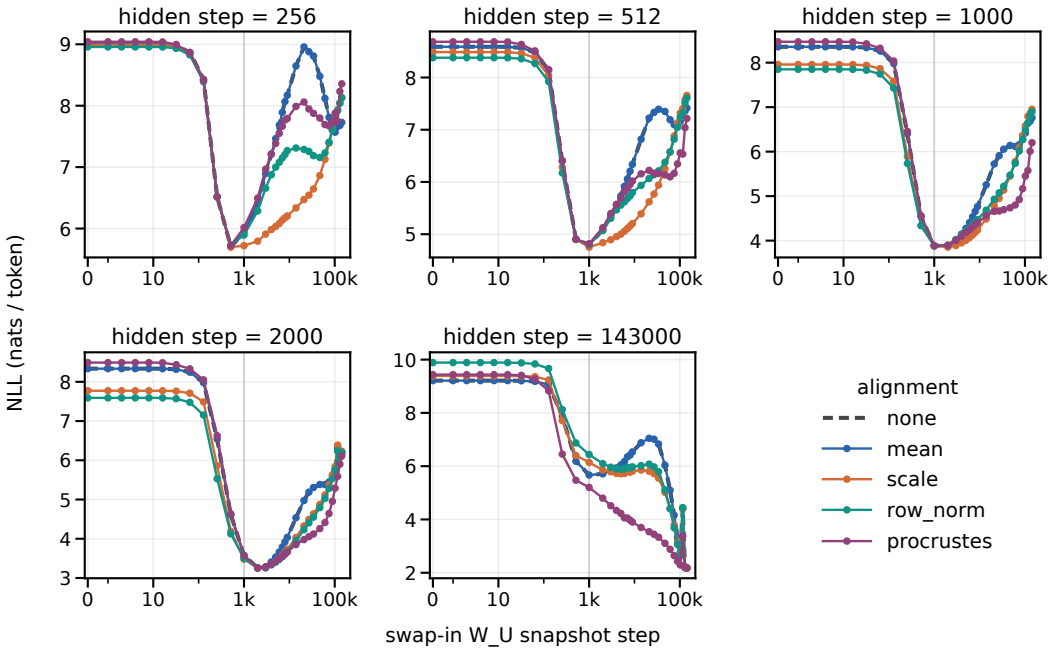

Figure G.3: In Pythia-1B, each hidden-state checkpoint keeps its minimum-NLL readout across four alignments of the swapped readout (step 512 for $t = 256$ and step 1000 for $t = 512$), except for a one-checkpoint shift under scale alignment for hidden states from checkpoint 1000 with a small NLL gap. Panels are hidden-state checkpoints, the horizontal axis is the readout checkpoint s, and the thin vertical line marks $s = 1000$. "None" is the raw swap. Its curve (dashed) lies under the "mean" curve, which is identical by construction.

Readout-alignment check. The location of the best readout for early hidden states does not depend on the swapped readout's mean, scale, row norms, or orthogonal frame. Scoring each alternate readout raw or after matching the native readout's column mean, centered Frobenius scale, per-row norm, or centered orthogonal Procrustes frame leaves the row-wise optimum unchanged, apart from a one-checkpoint shift under Frobenius scaling for step-1000 states with an NLL gap of 0.031 nats (Figure G.3). The check uses the first 16,352 targets of the corpus, mostly English and Russian (about 3.9k and 10.6k) with some Chinese, so its NLL values are not comparable with the nine-language aggregate. Its optimum for $t = 256$ is the step-512 readout rather than the aggregate's step 1000, consistent with both English and Russian preferring step 512 there (Table G.1).

G.4 Contrastive Task Margin Localization

The localization of the four task margins (Section 4.4, Table G.2) uses the released Pythia-1B d_{SAE}=24,576 seed-0 trajectory crosscoder, the norm-filtered localization items of Appendix F.1, and features selected by attribution to a *task* margin.

Evaluation configuration. All interventions on the readout side use one fixed configuration, step-1000 hidden states decoded through the reconstructed terminal readout $\widehat{W_U}^{(143k)}$. The swap grid above shows that the terminal readout is not a neutral decoder for early hidden states, so the localization results are specific to this configuration. The hidden state projection (below) re-scores under the native $W_U^{(1000)}$, so its baseline differs from the reconstruction baseline.

For greater-than completion, IOI, and relational facts, the recon accuracy is at or below chance (Table G.2), so the reconstructed terminal readout does not express these margins on step-1000 hidden states. Ablation can still lower accuracy when the selected features carry a margin that other features cancel, as for greater-than completion, where it drops accuracy from 0.512 to 0.084. Two caveats bound these three rows. First, the reconstruction agrees with the original terminal readout on 0.70 of held-out top-1 decisions (Appendix C.2), so a recon value measures the reconstructed

readout and may in part reflect reconstruction error. Second, a margin absent at this configuration can be present at the native terminal checkpoint, where the IOI gap closes by the final checkpoint (Table F.2). The IOI result here is therefore that no compact set carries the margin on early hidden states, and ablation where IOI is expressed remains to be run. We read the ablation and preserve-only deltas against each family's recon baseline, which replaces 0.5, and the sign-matched control gives the effect of a non-selective edit of the same size.

Terminal readout swap and alignment. The task sweeps of Section 4.3 use intermediate readouts; the terminal readout behaves differently on early states. On all 320 SVA items of the swap grid at this cell, Pythia-1B's native step-1000 readout scores 0.800, and the terminal readout scores 0.925 without alignment, 0.925 after scale alignment, and 0.922 after row norm alignment. After an orthogonal Procrustes alignment of the terminal readout to the native one, accuracy falls to 0.566. The terminal readout's gain on these early states therefore depends on its coordinate frame. The intervention run reports the same native and terminal accuracies on the 160 held-out items used in Table G.2. Figure 1 uses one of these items under the unaligned swap to show that the prediction depends on the readout.

Attribution and interventions. Features are attributed to the reconstructed contrastive margin on a random 50% split of the items (split A), and all interventions are scored on the held-out half (split B), which shares answer tokens with split A (Appendix F.1),

$$\mathrm{attr}_f = \mathbb{E}_{n\in A}\Big[\big(a^{(143\mathrm{k})}_{y^+,f} - a^{(143\mathrm{k})}_{y^-,f}\big)\,\big(h^{(1000)}_n \cdot D^{(143\mathrm{k})}_f\big)\,\sigma^{(143\mathrm{k})}\Big],$$

where $a^{(143\mathrm{k})}_{y^\pm,f}$ is feature f's reconstructed coefficient on the answer token row $y^\pm$, $D^{(143\mathrm{k})}_f$ is its decoder row at the terminal checkpoint, and $\sigma^{(143\mathrm{k})}$ is the terminal normalization scale of Appendix A.1 (written $s^{(t)}$ there), so that $\sigma^{(143\mathrm{k})}a^{(143\mathrm{k})}_{v,f}D^{(143\mathrm{k})}_f$ is the feature's contribution to row v in the original readout space. The *ablate* intervention removes the reconstructed contributions of the top eight positive attribution features, $\widehat{W_U} \leftarrow \widehat{W_U} - \sum_{f\in S}\sigma^{(143\mathrm{k})}a^{(143\mathrm{k})}_{:,f}D^{(143\mathrm{k})}_f$; *preserve-only* keeps only those eight; and the *hidden state projection* sets $h^{(1000)} \leftarrow (I - QQ^\top)h^{(1000)}$, where the columns of Q are an orthonormal basis, computed by QR, of the eight decoder directions $\{D^{(143\mathrm{k})}_f\}_{f\in S}$, and re-scores under the model's native readout $W^{(1000)}_U$. Each intervention is compared against the same edit applied to eight random features matched on decoder norm, firing rate, and attribution sign. Each control feature is the nearest unused neighbor of one selected feature in the plane of log decoder norm and log firing rate, the fraction of vocabulary rows on which the feature fires at the terminal snapshot. The sign-matched control draws only from features with positive attribution, falling back to all features if too few qualify. Random features matched only on decoder norm and firing rate leave accuracy at the recon value in 11 of the 12 stored family and model rows of this run, as expected if they are inactive on the answer rows, since an ablation changes only the rows where a feature is active; the table therefore reports the sign-matched control.

Table G.2: Per-family top eight intervention accuracies at the fixed evaluation cell (Pythia-1B, held-out half; relational facts, scored on 13 items, are exploratory). *orig.* is the original terminal readout and *recon* the full reconstructed terminal readout; *ablate* and *preserve-only* act on the top eight positive attribution features, and *sign* is the same edit applied to eight random features matched on decoder norm, firing rate, and attribution sign. The *hidden projection* columns score step-1000 states under the native $W^{(1000)}_U$ before (*native*) and after (*proj.*) the top eight decoder directions are projected out. *Scored on all localization items ($2{,}000$, 462, and 26), not the held-out half.

			ablate		preserve-only		hidden projection	
Family	orig.	recon	top-8	sign	top-8	sign	native	proj.
Subject–verb agreement	0.925	0.900	0.487	0.931	0.938	0.463	0.781	0.500
Greater-than completion	0.529*	0.512	0.084	0.500	0.894	0.507	0.601	0.497
Indirect object id. (IOI)	0.455*	0.429	0.394	0.424	0.481	0.420	0.377	0.338
Relational facts	0.269*	0.231	0.154	0.231	0.923	0.308	0.308	0.154

Reading the table. Subject–verb agreement and greater-than completion localize, with ablation far below the sign-matched control and preserve-only high on the eight directions alone, and relational

facts show a strong but exploratory preserve-only effect on 12 of 13 scoring items (Table G.2). Indirect object identification shows no selective positive-margin effect from a compact set at this configuration. Its ablation stays close to the sign-matched control, positive attribution is spread across 4,387 of 24,576 features, and projecting the eight directions out of $h^{(1000)}$ shifts the y^+ and y^- logits almost equally (-0.44 and -0.45), consistent with generic token suppression at the top eight budget.

Projecting the eight directions out of $h^{(1000)}$ removes at most 1% of the hidden state norm, yet it costs subject–verb agreement far more accuracy than greater-than completion. Because the projection is scored under the unmodified native readout $W_U^{(1000)}$, the agreement effect depends on hidden state directions spanned by the eight decoders, which the native readout uses to separate y^+ from y^-, so the change in the readout between steps 1000 and 143k is not its only source. The projection run's native SVA baseline of 0.781 differs from the 0.800 of the swap grid and the intervention run.

H Recipe Control: Readout Learning Rate and Developmental Timing

This appendix details the controlled training experiment summarized in Section 4.5.

H.1 Setup and Conditions

The control trains four Pythia-architecture models of 31M parameters that share initialization seed and sequential data order, on 10B tokens. The conditions differ only in one of two recipe factors. Three vary the learning rate multiplier applied to the output readout W_U (`embed_out.weight`): `wu_lr_0p25` ($0.25\times$), `baseline` ($1\times$), and `wu_lr_4x` ($4\times$). The fourth, `warmup_short`, shortens the learning rate warmup (from the baseline 1430 steps to 715) at the baseline $1\times$ readout learning rate. Each condition has one pretraining run and three crosscoder fits (seeds 0, 1, and 2). Crosscoder tables and figures show seed 0, and the timing statistics are reported for all three fits. A fifth run with a fourfold longer warmup (5720 steps) was also trained. Its warmup exceeds the 4769-step budget, so it never reaches the peak learning rate, and we leave it out of the comparison; its checkpoints are released with the others. The models use the Pythia architecture at $d_{\text{model}} = 256$ with six layers and the Pythia tokenizer. Training follows Pythia's optimizer recipe: AdamW [Loshchilov and Hutter, 2019] with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1 (excluding LayerNorm parameters and biases), gradient clipping at 1.0, fp16 mixed precision, and a global batch of 1024 sequences of 2048 tokens (2.1M tokens per step). The learning rate warms up linearly to a peak of 10^{-3} over 1430 steps and follows Pythia's cosine schedule over 143k steps, stopped at the 10B-token budget (4769 steps). At that point the learning rate is still within 0.2% of its peak, so the reported final losses precede any decay of the learning rate. The data are the first 10B tokens of Pythia's preshuffled, pretokenized Pile [Gao et al., 2020, Biderman et al., 2023] (`EleutherAI/pile-standard-pythia-preshuffled`), in 2048-token blocks read once, in Pythia's released order, by all conditions; the last 256 blocks (about 0.5M tokens) are held out for validation loss. The control keeps $K = 16$ checkpoints at steps $\{0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 2479, 4096, 4769\}$, that is, step 0, the powers of two up to 4096, the final step, and one checkpoint that the trainer saved when the first 11-hour job slot ended. That slot checkpoint falls at step 2479 for `baseline` and at steps 2464 to 2481 across the four conditions, and we write it as 2479 throughout. The hidden state views of Appendix H.4 use held-out Pile tokens that directly follow the 10B-token training prefix in the same preshuffled order. The readout swap scores all 118,726 next-token positions of a 120k-token slice, and the frequency class probe uses 20,000 positions sampled (seed 0) from a 400k-token slice.

Every W_U trajectory crosscoder is fitted using the main Pythia-160M recipe of Table B.2 (mapped to $d_{\text{model}} = 256$): JumpReLU, $d_{\text{SAE}} = 8192$, $\lambda_1 = 0.3$, learning rate 5×10^{-5}, centered and scaled inputs, 300 epochs. The four seed-0 crosscoders reconstruct their trajectories with comparable explained variance and have no dead features (Table H.1). Decoder norm trajectories $\rho_f(t)$, the alive threshold $\rho_f(t) \geq 0.5$, peak step, and the rule-based profile assignment are computed exactly as in Appendix D.

H.2 Lifecycle Response to the Readout Learning Rate

A faster readout shifts the per-feature trajectories from flat, persistent curves toward curves that rise and fall with earlier peaks, so the profile shares move from mostly persistent at $0.25\times$ to mostly early-decaying at $4\times$ (Table H.1).

Table H.1: Per-condition crosscoder quality and lifecycle summary for the recipe control ($d_{\mathrm{SAE}} = 8192$, $K = 16$, 8192 active features each, seed 0). EV is explained variance; L_0 is mean active features per row; "val. loss" is the condition's final validation loss in nats; the profile columns are fractions of active features; "med. peak" is the median peak step; "turnover step" is the peak turnover step. The validation loss at the peak turnover step is 3.72 nats for $0.25\times$ (step 2048) and 5.13 nats for $4\times$ (step 512). At $1\times$, turnover at steps 1024 and 2048 differs by less than 0.002 in every fit, and seed 1 selects step 2048. The two timing statistics are the primary measures; the rule-based profile fractions are discrete summaries sensitive to the profile rule boundaries (Appendix H.3).

Condition	EV	L_0	val. loss	persist.	early-dec.	late-em.	mixed	med. peak	turnover step
`wu_lr_0p25`	0.905	193.9	3.405	0.875	0.000	0.093	0.032	128	2048
`baseline`	0.891	177.2	3.381	0.403	0.219	0.125	0.252	64	1024
`warmup_short`	0.890	175.5	3.350	0.011	0.442	0.153	0.394	64	1024
`wu_lr_4x`	0.874	168.5	3.369	0.003	0.675	0.131	0.191	32	512

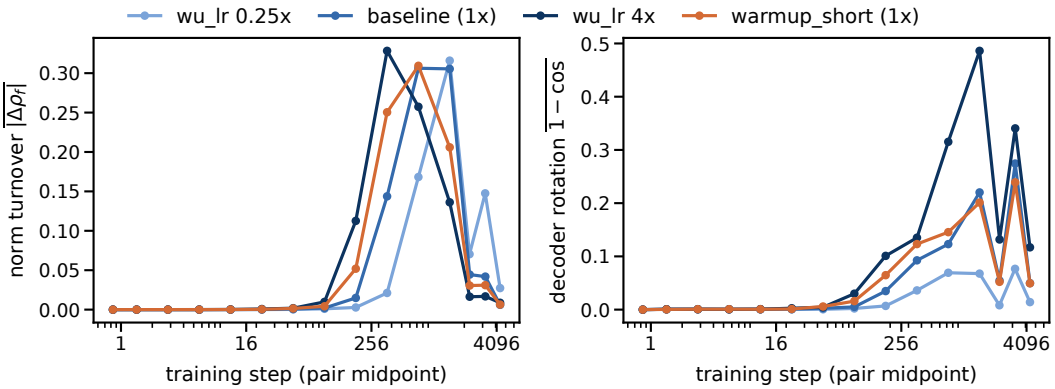

Figure H.1: Higher readout learning rate shifts the peak of decoder norm turnover (left) earlier, and `warmup_short` tracks `baseline`. Decoder rotation (right) peaks at $1024\to2048$ only for $4\times$. Its late spike, and the late rise in norm turnover for $0.25\times$, come from the pair from the slot checkpoint near step 2479 to step 4096, which spans about 1,600 steps, the longest interval of the schedule, against fewer than 450 steps for the pair before it. Each point averages the change over active features of the seed-0 crosscoder for one consecutive checkpoint pair, plotted at the pair midpoint.

Decoder norm turnover $\overline{|\Delta\rho_f|}$ peaks in a localized window that moves earlier as the readout learning rate rises, whereas decoder rotation $\overline{1-\cos(W_{\mathrm{dec},f}^{(t)}, W_{\mathrm{dec},f}^{(t+1)})}$ peaks late in three of the four conditions, at the long interval after the slot checkpoint, so the timing claim rests on norm turnover (Figure H.1). Figure 5 reports the peak turnover step and the median peak step for all three fits.

H.3 Warmup Length Leaves the Crosscoder Timing Statistics Unchanged

Halving the warmup (715 steps against 1430) at the baseline multiplier raises the effective readout learning rate early in training, and both schedules reach the same peak rate after step 1430. The `warmup_short` run matches `baseline` on both crosscoder timing statistics and on maximum turnover (Table H.1 and Figure H.1), although its validation loss at the peak turnover step is 0.52 nats lower. Its discrete profile fractions differ even though its peak-step distribution does not, and because these classifications are sensitive to the thresholds for near-flat trajectories in Table D.1, we read them separately from the timing statistics. On the hidden state views of Appendix H.4, it lies near `baseline` with a small shift toward the faster readout, with a smaller early frequency class gap and a swap optimum one checkpoint later than native at $t = 256$ and 512 whose gains stay within 0.05 nats, against 0.91 nats in the $0.25\times$ run. Warmup length therefore leaves the crosscoder timing unchanged on this grid and acts on the frequency class gap as a small partial step along the multiplier axis.

H.4 Agreement Across Views on the Same Checkpoints

We compute geometry, lifecycle, swap, and probe diagnostics on the same four runs, and they agree along the readout learning rate axis.

The aggregate W_U geometry shifts monotonically with the readout learning rate, with final mean row norms of 1.85 at $4\times$ and 0.93 at $0.25\times$ and the final LayerNorm gain moving in the opposite direction (Figure H.2).

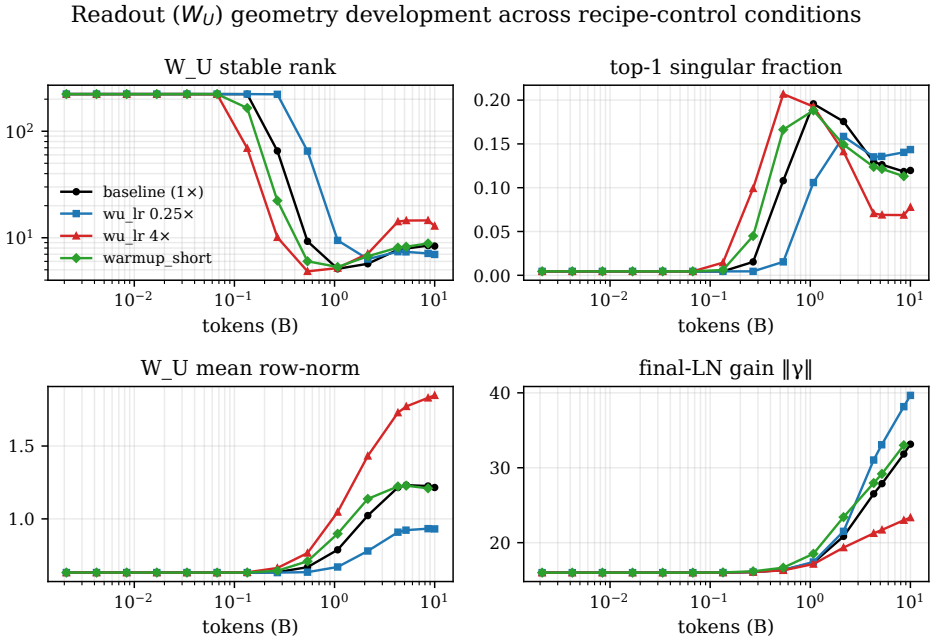

Figure H.2: By the end of training, higher readout learning rate gives larger W_U row norms, a less concentrated spectrum, and a smaller final LayerNorm gain, and `warmup_short` tracks `baseline`. The stable rank of W_U is $\|W_U\|_F^2/\sigma_1^2$, where σ_1 is its largest singular value. The top-1 singular fraction is $\sigma_1^2/\|W_U\|_F^2$, the share of squared Frobenius norm in the top singular direction. The final LayerNorm gain is the ℓ_2 norm of its gain vector γ, and the `warmup_short` curves end at step 4096 because its final geometry measurement is missing.

The readout swap of Appendix G, applied per condition, follows the same axis (Figure H.3). At $4\times$ each hidden state is best decoded by its own readout, whereas at $0.25\times$ early states are best decoded by a later readout, and for $t = 256$ the step-512 readout gives a gain of 0.91 nats that stays positive under the mean, scale, row norm, and Procrustes alignments (0.91 to 1.07 nats).

The probe-versus-native comparison of Appendix F, applied per condition, uses the frequency class probe of Appendix A.3 in place of a task label, so it measures availability of this coarse next-token property rather than of any task in Table 1. The early frequency class gap, probe minus native accuracy, is ordered $0.25\times > 1\times > 4\times$ at $t = 256$ and 512, and `warmup_short` lies between `baseline` and $4\times$ at $t = 256$ and below $4\times$ at $t = 512$ (Figure H.4). Because the native readout exceeds the probe in every condition from $t = 2048$ on, the early gap is read for its ordering across conditions rather than for its absolute size.

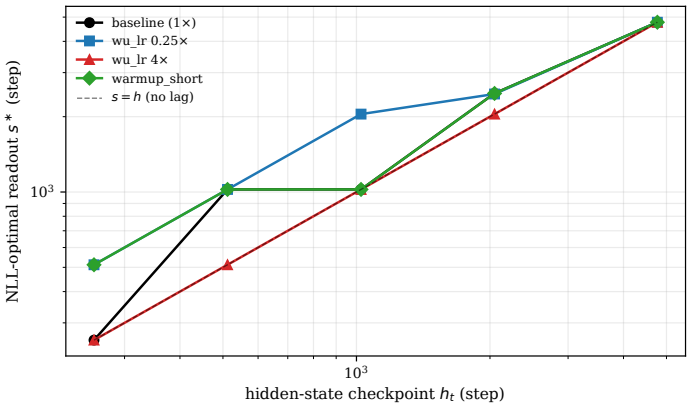

Figure H.3: The swap optimum s^*, the readout checkpoint with the lowest NLL for hidden states from checkpoint t, tracks the readout learning rate. Points above the diagonal are hidden states best decoded by a later readout. `wu_lr_4x` follows the diagonal and `wu_lr_0p25` lies above it, while for `baseline` and `warmup_short` the gain over the native readout is at most 0.05 nats.

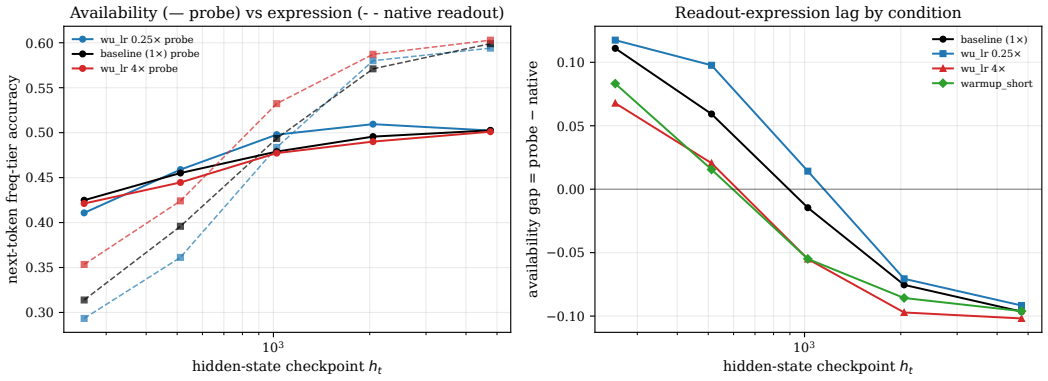

Figure H.4: The frequency class gap (probe accuracy minus native accuracy) in the early window is ordered $0.25\times > 1\times > 4\times$ and closes by step 1024 in every run except $0.25\times$, which closes by step 2048. Accuracy is on four next-token frequency classes of equal corpus occurrence mass. From step 2048 on the native readout exceeds the probe in every condition, so the probe is not a ceiling late in training.

H.5 Early Loss Differences and Similar Final Loss

The early loss comparison of Section 4.5 is made at step 128 ($268{,}435{,}456$ training tokens), where the validation loss of the $4\times$ run is 1.202 nats below baseline and 2.079 nats below $0.25\times$, and Table H.1 gives the final losses.

Similar final losses need not imply identical models. Because of the scalar symmetry between the final normalization and the readout (Appendix A.1), readout row norms cannot be interpreted independently of the final normalization. The symmetry does not establish that the independently trained models differ only along it.

I Related Work, Limitations, and Broader Impact

I.1 Extended Related Work

Encoded but unexpressed information. Probes test what is decodable from representations [Alain and Bengio, 2016, Conneau et al., 2018, Hewitt and Manning, 2019, Tenney et al., 2019], and control tasks separate what the probe itself learns [Hewitt and Liang, 2019, Belinkov, 2022]. Trained models can encode answers that their outputs do not express [Burns et al., 2023, Orgad et al., 2025, Gekhman

et al., 2025], and training can make progress that loss and error metrics do not reveal [Barak et al., 2022]. We follow this gap across pretraining checkpoints and use readout swaps on fixed hidden states to attribute its expression side to the readout.

Feature correspondence. Within a fixed tokenizer, token identity gives our dictionary its row correspondence, but it does not remove hidden space reparameterizations or establish semantic identity across independently trained models [Ainsworth et al., 2023], and shared indices do not establish that the features are canonical units [Leask et al., 2025, Minder et al., 2025].

Table I.1: Analysis families and the comparisons they support. Temporal correspondence describes the unit compared across checkpoints. The last column distinguishes dictionary fitting from the interventions used to assess behavior.

Family	Analyzed object	Input data	Temporal correspondence	What the comparison isolates
Lens methods	Hidden states in vocabulary space	Text and forward passes	Vocabulary coordinates; native or learned readout	Predictions under the chosen readout
Vocabulary geometry	Input/output embedding rows	Weights; lexical annotations	Token identities or geometric statistics	Changes in geometry and lexical association
Independent SAEs	Activations per checkpoint	Activation corpus	Separate indices; post-hoc matching	Features within each fit
Activation crosscoders	Activations across checkpoints	Activation corpus	Shared sparse indices	Representation change; feature use via interventions
Parameter methods	Decomposed or sparse weights	Weights and inputs	Typically one model state	Parameter components and circuits
This work	Readout row trajectories	Weights; text for tests	Token identities and shared sparse indices	Readout change; swaps and ablations at fixed states

I.2 Extended Limitations

Beyond those of Section 6, four limits apply. The checkpoint schedules bound temporal resolution, since they sample Pythia densely only before step 10k and each 31M run at 16 checkpoints. Profile shares and the median feature peak step depend on dictionary width and sparsity (Appendix D.5), so we read the shares qualitatively and compare timelines only at matched settings, whereas the reorganization window holds in 20 of 21 fits. The probes are linear and read only the final hidden state at the answer position, so they can miss distinctions decodable only nonlinearly or at other positions and layers. The comparisons have narrow scope. OLMo-2-7B is the only cross-family check, the sparse interventions use one configuration (Section 4.4), and tokenization, preprocessing, and optimization shape all comparisons.

I.3 Implications

Open questions. It remains open how task directions become coupled to the readout late in training, since relative clause agreement and numeric comparison keep gaps that no evaluated readout closes, and how hidden state and readout updates are coordinated, since a readout swap isolates only one of the three terms of Equation (1).

Control through the readout. The directions that carry the agreement margin at one configuration (Section 4.4) make the readout a compact intervention site, as steering generation through linear transformations of the output embeddings suggests [Han et al., 2024], and the side effects and persistence of such edits remain to be measured (Appendix I.4).

I.4 Social Impact

The work deploys no language model and does not propose a validated safety evaluation procedure. The main positive impact is methodological transparency, because treating the learned readout as part of the measured system makes developmental claims in logit space easier to audit, and the analysis code is released to support auditing of the reported checkpoint comparisons. The main negative-use concern is more targeted readout manipulation, since sparse readout features can identify token-family directions that affect logits.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the research question, the confound in behavioral dating, the method and how it differs from prior crosscoders, and three empirical claims with the models and runs that support them (Section 1), and the results sections give the tasks and statistics; Section 6 states their limits, including that the task results measure two-token contrasts rather than general capability acquisition.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main text has a limitations paragraph (Section 6) covering the single 31M setting with one run per condition and the two-token task scope, Appendix I.2 adds checkpoint density, dictionary settings, and probe scope, and the appendix states the scope of each remaining result with that result.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is empirical and methodological; it does not state formal theoretical results or proofs.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 2 and 3 and Appendices A, B, B.1 and H specify checkpoints, preprocessing, architecture, losses, hyperparameters, task items, probes, baselines, swaps, intervention protocols, and the training protocol of the controlled runs.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is available at `https://github.com/hematteo/learning-to-read-out`, with instructions to extract public checkpoints, fit the dictionaries, train the controlled runs, and compute the reported metrics. The inputs are public (Pythia and OLMo checkpoints, WordNet, Wikipedia dumps). The trained dictionaries with derived measurements and the checkpoints of the controlled runs are released at `https://huggingface.co/hematteo/parameter-trajectory-crosscoders` and `https://huggingface.co/hematteo/readout-recipe-control`, and regenerating them takes the compute in Table B.3 (Appendix B.1).

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3 and Appendices A.3, B, F.1 and H specify model checkpoints, tokenizer conventions, preprocessing, dictionary hyperparameters, optimizer settings, prompt templates, probe splits, and evaluation protocols.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Probe accuracies are averaged over grouped cross-validation folds (Appendix A.3), and the per-language readout swaps report 95% block-bootstrap intervals (Table G.1). Structural results come with matched controls, circular-shift nulls with their p values and feature bootstrap counts (Table D.2), dictionary fit-seed, width, and sparsity checks of reconstruction and of the lifecycle statistics (Table D.3), and held-out checkpoint checks. Many reconstruction and swap measurements are deterministic given the released checkpoints, and the controlled training comparisons use one pretraining run per condition, which Sections 3 and 6 state.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Table B.3 reports worker class, GPU memory, wall-clock ranges, and GPU-hour ranges for the main dictionary fits, the controlled 31M pretraining runs, and the probe and figure jobs, as hardware-class requirements rather than provider-specific records.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work analyzes public model checkpoints and public text resources, trains small 31M-parameter models only for a controlled comparison, does not involve human subjects, and does not deploy a system.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The appendix social-impact statement in Appendix I.4 notes that the work is analytic, identifies likely transparency-auditing benefits, and flags possible misuse through more targeted readout manipulation.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases code, sparse dictionaries fitted to public checkpoints, and 31M-parameter models trained for a controlled comparison; none is a capable pretrained language model, image generator, or scraped dataset with high misuse risk.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Appendix B.6 explicitly enumerates licenses and terms for the public Pythia and OLMo checkpoints, WordNet, the BLiMP and function vector datasets, Wikimedia Wikipedia dumps, and the major open-source software dependencies, and notes that original model checkpoints and raw Wikipedia dumps are not redistributed.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset is the code repository at `https://github.com/hematteo/learning-to-read-out`, released with a README, a reproduction guide that maps each paper figure to the scripts that compute its metrics, a license, and a citation file. The trained dictionaries with derived measurements (`https://huggingface.co/hematteo/parameter-trajectory-crosscoders`) and the checkpoints of the controlled runs (`https://huggingface.co/hematteo/readout-recipe-control`) are released with model cards under the MIT license.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not a component of the research methodology.