LET'S INTERPRET STEP-BY-STEP

Anonymous authors
Paper under double-blind review

ABSTRACT

Interpretability research often seeks to explain model computation using ordered sequences of discrete steps, particularly when describing latent reasoning pipelines or circuits. When selecting features for circuits, we should therefore prefer features that actually capture discrete steps in the model's forward pass. In practice, however, this criterion is rarely verified or explicitly targeted when building feature dictionaries. We propose a new metric, *punctuality*, to measure if a feature was computed in a step, rather than a gradual refinement process. As a case study, we analyze a popular sparse autoencoder (SAE) feature circuit and find that many features are punctual and represent a shared step in computation. Next, we assess residual stream SAE features and find them to be less punctual than neuron basis features—and furthermore, that more punctual SAE features are less causally important, limiting their value for interpreting latent reasoning mechanisms. Finally, we show that crosscoders, which are designed to merge features that persist across layers, outperform SAEs in finding features that are both punctual and causally important. By explicitly measuring punctuality along with other desiderata like causal importance and interpretability, we can assess which features can be used in circuits or explanations of latent reasoning.

1 INTRODUCTION

Interpretability researchers often aim to explain model processing as algorithms composed of ordered steps. These algorithms might be precise circuits like induction heads (Olsson et al., 2022), which first search for a prefix and then copy the next token, or more high-level pipelines, as in multi-hop question-answering (Yang et al., 2025). In practice, the steps in these algorithmic explanations are often linear features, typically selected from sparse dictionaries (Marks et al., 2025). Researchers typically evaluate these candidate features on criteria such as *interpretability* (Cunningham et al., 2023; Karvonen et al., 2025) or *faithfulness* (Marks et al., 2025; Heap et al., 2026; Paulo & Belrose, 2025)? In this paper, we propose a new criterion which we argue is a prerequisite to compose features into an ordered sequence of steps: Are these features even *steps* in the first place?

We call a linear feature *punctual* if it becomes available suddenly in a discrete step—that is, the feature is defined by just a few layers, while most layers change it very little. The circuit paradigm describes features as corresponding to discrete steps, so this is a necessary criterion for circuit features. This insight is not limited to circuits: we cannot explain latent reasoning without decomposing the model's processing into discrete steps. The main contribution of this work is to highlight this criterion for feature quality and suitability for interpreting latent reasoning as circuits, algorithms, or pipelines. Do our feature dictionaries actually capture meaningful steps of computation or are they marking arbitrary points along a smooth, gradual refinement process?

Interpretable features might be computed punctually or gradually, but only punctual features can illuminate latent reasoning. Consider the prompt `Fact: the capital of the state containing Dallas is:`. Lindsey et al. (2025) claim that a language model answers this prompt by accomplishing two subtasks: one, determine that the state is `Texas` and that it must retrieve a `capital`; two, select `Austin`. Figure 1 depicts two simplified pipelines, *both of which are compatible with Lindsey et al. (2025)'s posited circuit*. We can think of these as a reasoning pipeline (Model A) and a non-reasoning pipeline (Model B)—the difference between them is how they compose `Texas` and `capital`. Model A first identifies Texas and then retrieves Texas's capital `Austin`, while Model B is effectively linear: it sees the tokens `capital` and `Dallas` in the prompt and gradually promotes the related tokens in parallel. Model A's proposed pipeline assumes

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

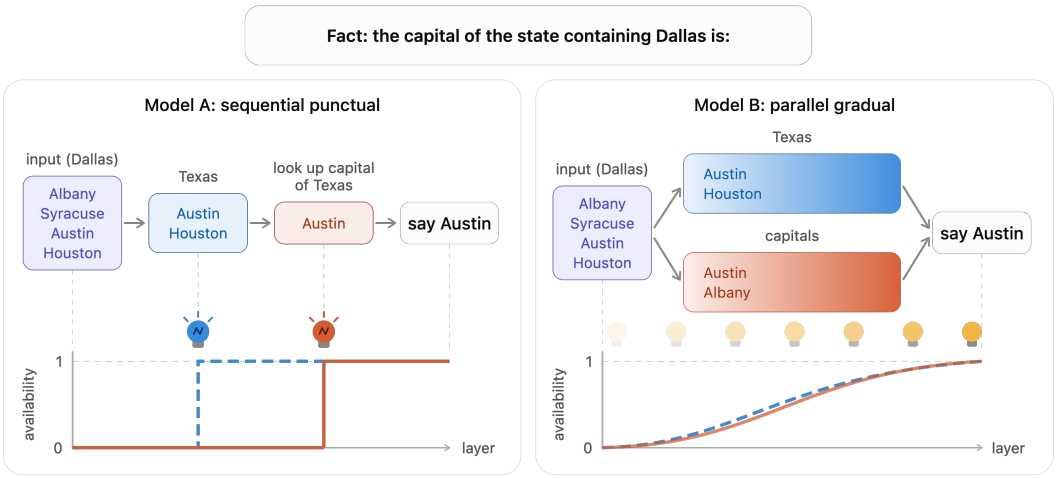

Figure 1: A language model that answers this prompt via multi-step reasoning might employ Model A's algorithm: first determine that the state is `Texas` and then retrieve the capital, `Austin`. But if its availability graphs look more like Model B's, then these features might not actually find ordered steps in the model's computation as Model A's algorithm describes. Model B's algorithm simultaneously refines a Texas feature and a capital feature; simply summing their respective token distributions yields `Austin`. Both models are compatible with the circuit described by Lindsey et al. (2025), but only Model A represents multi-step reasoning.

the computation of `Texas` and retrieval of its `capital` are steps. This should be explicitly verified because if the availability graphs of these features instead look more like Model B's, then the supposed features do not actually correspond to orderable steps in the model's computation.

Explaining model latent reasoning using circuits requires ordered steps. If we can only find features that progress smoothly through the entire model, then our explanation reduces to a bag of final-layer features, treating the entire model as shallow and failing to illuminate complex model behavior.

Our main contribution is to highlight and measure a hidden but important criterion of features that are used in circuits: whether they capture discrete steps of model computation. We operationalize this criterion as *punctuality*: a measure of how suddenly a feature emerges in the layers of the model.

As a case study, we analyze the punctuality of features from the Bias-in-Bios SAE circuit (Marks et al., 2025). We find that this circuit does, in fact, identify steps: Simple input features feed into a discrete step of computation concentrated at layer 8, which then informs final output token selection. This case study is a positive example that step-by-step latent reasoning is interpretable through punctual features, motivating our search for more punctual features to include in circuits.

We next evaluate residual stream SAE features distributionally and find them to be less punctual than neurons. Furthermore, punctuality is generally anticorrelated with a feature's importance, as determined by its *causal impact*. In fact, the most punctual features can often be considered ephemeral noise, materializing quickly and immediately disappearing with little effect on the model's output. This motivates alternative methods to identify features which are both important and punctual, making them usable in explanations of latent reasoning.

To evaluate one alternative, we compare the punctuality of SAE features to crosscoder (CC) features. CCs are advertised to promote feature persistence in earlier layers and, we argue, also implicitly designed to promote punctuality in later layers. We find evidence supporting both. Finally, given that punctuality is anticorrelated with causal impact for SAE features, we investigate whether crosscoders find more punctual features simply by sacrificing causal impact. We find this is not true; crosscoders find features that Pareto dominate SAE features on punctuality and causal impact.

2 DEFINING PUNCTUALITY

We study linear features in the residual stream, interpreted as binary (on/off) classes. We are interested in the extent to which a feature found at layer L is already available at earlier layers. If a feature becomes available abruptly, we say it is punctual.

Definition 2.1 (Feature)**.** A feature f found in layer L is a binary-valued function on residual stream activations $\mathbf{z}_L \in \mathbb{R}^d$ parameterized by a direction $\mathbf{v} \in \mathbb{R}^d$ and threshold $b \in \mathbb{R}$:

$$f(\mathbf{z}_L) = \mathbb{1}\left[\mathbf{z}_L^\top \mathbf{v} > b\right].$$

Definition 2.2 (Layer-l availability)**.** Let $\{y = f(\mathbf{z}_L) | \mathbf{z}_L \in \mathcal{D}\}$ denote the set of binary feature labels for activations from dataset $\mathcal{D}$. We define the feature's availability at layer l, denoted s_l, as the precision–recall area under the curve (PR AUC) obtained by using $\mathbf{z}_l^\top \mathbf{v}$ as the prediction score for y. We refer to the input token embedding as layer 0 (and the first Transformer layer as layer 1), so availability at the embedding layer is denoted by $s_{embd} = s_0$.

Note that availability is measured along the verbatim feature direction $\mathbf{v}$. This measures whether the same direction used in a feature at source layer L is already predictive of the same feature labels at layer l. A separately trained probe could instead detect information encoded along a different direction.[1] Because the labels are defined by thresholding $z_L^\top v$, all features have $s_L = 1$ by construction.

For a given feature, we can plot its availability at each layer of the model to track its emergence and disappearance. A feature that captures a discrete step of model computation should make a sudden jump in availability. We call such a feature punctual, defined as follows:

Definition 2.3 (Punctuality)**.** We define punctuality based on the *participation ratio*[2] of layers that progress the linear availability of a feature. We define this ratio first. For a feature found at layer L, we compute availability at all prior layers $(s_0, \ldots, s_L)$ and record the running maximum at each layer l as $m_l = \max_{0 \le i \le l} s_i$. We then compute the progress each layer makes as $\Delta_l = m_l - m_{l-1}$. In practice, we observe almost all features have $s_{embd} \approx 0$.

The participation ratio measures the effective number of layers that progress the feature given by

$$k_{EC} = \frac{1}{\sum_{l=1}^{L} \left(\frac{\Delta_l}{\sum_{l'=1}^{L} \Delta_{l'}} \right)^2}$$

A feature with lower k_{EC} is more like a discrete step. If one layer does all the work then $k_{EC} = 1$, whereas if the computation is spread equally among L layers then $k_{EC} = L$. Finally, we define the punctuality of a feature found in layer L by normalizing the participation ratio to the range $[0, 1]$,

$$P_{EC} = \frac{L - k_{EC}}{L - 1}$$

We briefly explore alternative metrics in Appendix A.5, including metrics that consider the specific layer at which a feature is computed; our definition of punctuality only measures whether a feature is computed in a discrete step somewhere in the model.

3 CASE STUDY: BIAS-IN-BIOS SAE CIRCUIT

We begin with a case study on an SAE feature circuit from Marks et al. (2025) on Bias-in-Bios, where the task is to classify a person's profession based on their biography from a dataset with spurious correlations (e.g. all doctors are men and all nurses are women). After clustering features by similarity into supernodes, we show that such a circuit can reflect ordered steps.

[1] In addition to reusing verbatim feature directions, we conduct preliminary analysis on training linear probes and find the distributions of punctuality to be qualitatively similar to verbatim decoding. We include the results in Appendix A.6.

[2] Participation ratio is a popular measure of the effective number of components in a system, widely used across the sciences. In neuroscience, it is used to measure the effective dimensionality of neural population activity (Gao et al., 2017).

We study the `Gemma-2-2b` circuit, which uses residual stream SAE features from Gemma Scope (Lieberum et al., 2024). Marks et al. (2025) construct the circuit roughly by selecting nodes and edges with highest indirect effect (Pearl, 2001) on classification performance. We refer readers to Marks et al. (2025) for details on circuit construction since it is fairly involved.

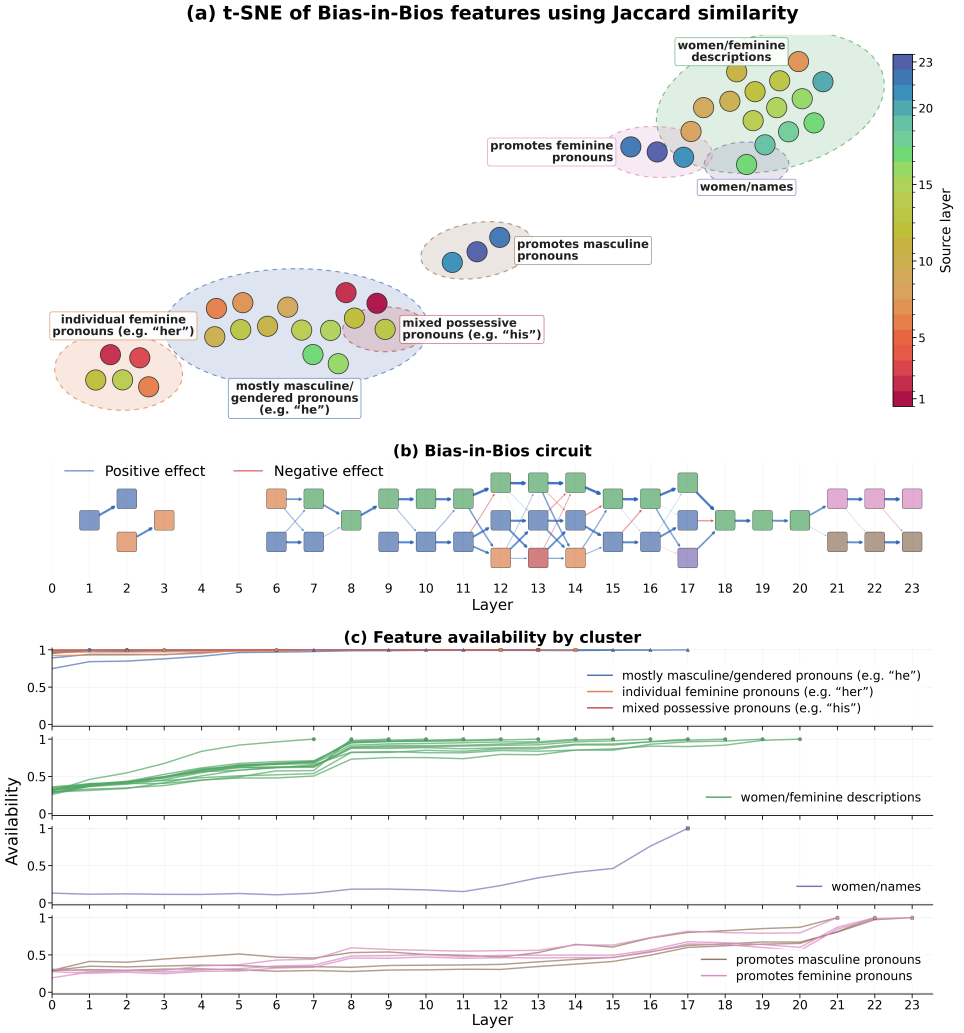

Figure 2: **In the Bias-in-Bios circuit, some features are steps (e.g. green) and some are more gradual (pink). (a) t-SNE plot** of circuit features using Jaccard similarity, grouped into human-labeled clusters using average-linkage agglomerative clustering. **(b) The circuit**, with features colored by cluster. **(c) Feature availability trajectories by cluster.** The first row groups three clusters that all capture simple input features (e.g. "he" token detectors), available at the first layer. The last row groups two clusters that appear near the output layer and respectively promote masculine and feminine pronouns.

We depict the original circuit in Figure 2(b). The circuit with human-written feature annotations is in Appendix A.1. Before we analyze their punctuality, we organize features into clusters according to their Jaccard similarity, which measures how much two features activate on the same tokens.[3] Clustering allows us to identify clear steps even when a cluster includes features localized before that clear step; as we will see, a cluster of features computed punctually at layer 8 includes an

[3]Jaccard similarity measures how much two features activate on the same tokens. Let A_i denote the set of token positions on which feature i is active. For features i and j, we define $J(i,j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}$. A value of 0 indicates total dissimilarity, and a value of 1 indicates the two features activate on exactly the same tokens.

apparently-gradual feature from layer 7. In 2(a), we depict a visualization of these clusters, each labeled by hand. Clusters generally share clear semantic connections.

After clustering into seven groups, we plot the PR AUC availability curves for related groups of clusters in 2(c). We see that the original circuit includes two clusters of simple features (almost fully available already in the model's first layer): mostly masculine pronouns and individual feminine pronouns. These serve as input to a punctual cluster of features representing feminine descriptions that emerges at layer 8. Finally this punctual cluster feeds into two feature clusters localized to the last few layers: one promoting feminine pronouns, the other promoting masculine pronouns; both have mild jumps in availability in layers 21–22.[4]

What do we learn from this exercise? Primarily, that real circuits can include punctual features which can therefore be ordered by when they emerge, not just where they happen to be found. The simple token detector features are input to a feminine description cluster computed at layer 8, which then feeds into the output token decisions finalized at layers 21–22. Although this is a simple sequence, it offers some hope that we can construct multistep reasoning circuits from individual ordered steps.

4 SAEs DO NOT FIND PUNCTUAL FEATURES

4.1 SETUP

In this section we evaluate the punctuality of residual stream SAE features in `gemma-3-1b-pt`.

Model and SAEs details: `gemma-3-1b-pt` has 26 layers and hidden dimension of $d = 1152$. We use Gemma Scope's canonical SAEs: layers 8, 14, 18, and 23. These SAEs are Matryoshka JumpRELU SAEs with width (dictionary size) 16,384 and medium L0 (50). We refer readers to McDougall et al. (2025); Lieberum et al. (2024) for more details about Gemma Scope SAEs.

Data: Our dataset comprises 10,000 sequences from the Pile, each of length 128 tokens. We measure availability via token-level binary separability. The most sparse SAE features do not activate on enough tokens to accurately interpret PR AUC as feature availability, so we filter them out. In layer 8 for example, 11,538 features out of 16,384 in total pass our frequency filter; the rest are excluded.

4.2 RESULTS

First, we find a clear division between simple input features and more complex intermediate features. We visualize where discrete steps occur in feature processing in Figure 3 (a), which depicts the distribution of layers with the largest gain in feature availability. Each plot shows this distribution for features found in a particular layer—and each distribution is **multimodal**. At all source layers, we see a cluster of simple input features available at layer 1 and another cluster that emerges at the SAE layer itself, though we also note additional clustering around intermediate layers. This indicates that SAE find simple input features and complex intermediate features, and that these groups are distinct. Results *after* this figure focus on complex intermediate features by filtering out features whose largest gain occurs at layer 1.

Next, we find that SAE features are less punctual than neurons in the residual stream[5] on average, though the SAE distribution has higher spread. Distributions are shown in Figure 3 (b). *This means that SAEs find features that look less like steps than neurons, making them poorly suited for analyzing latent reasoning mechanisms.*

5 IT IS HARD TO FIND CAUSALLY IMPORTANT, PUNCTUAL FEATURES

Punctual features might be useful for breaking down model computation into a sequence of steps, which is necessary to understand latent reasoning. But SAE dictionaries are known to contain many

[4] In addition to these clusters, the original circuit includes two single-feature outlier clusters, one for women/names and one for mixed possessive pronouns. The former is computed gradually across 17 layers. The latter, although originally localized to layer 13, is a simple feature available almost fully at the input layer.

[5] SAE features are highly sparse, so we choose the feature activation threshold for neurons to match the distribution of SAE feature sparsity to best enable an apples-to-apples comparison. We find that sparsity is a confounder for punctuality that is important to control for. See Appendix A.7 for more details.

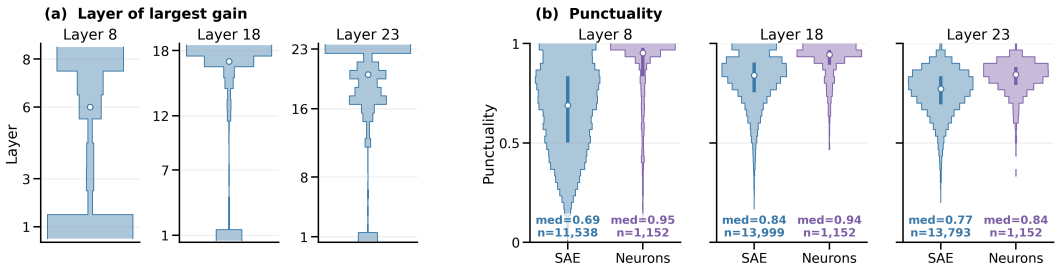

Figure 3: **(a) For SAE features, the distribution of layer making largest gain in availability is multimodal, capturing simple input features and complex intermediate features.** SAE features found at layer 8 are often available by layer 1, signaling many simple features (e.g., input token detectors). At layer 23, the SAE finds features computed at many intermediate layers. **(b) SAE features are less punctual than neurons.** At three different layers, individual neuron features are more punctual than SAEs, when those neuron features' on/off boundaries are chosen to match the SAE features in sparsity.

unsteerable, inoperative directions (Wu et al., 2025). A punctual feature therefore cannot be interpreted as a mechanism unless it is also causally important, playing a role in computation. In this section, we investigate the relationship between punctuality and causal importance. We find that the two desiderata are anticorrelated among SAE features, posing a challenge for selecting high-quality features to incorporate into complex circuits.

We measure the causal importance of a feature by clamping the feature to ON and OFF values and comparing the resulting next-token distributions. To keep all representations close to the empirical distribution, we set the ON value to the 80th percentile of the feature score among tokens for which the feature activates and set the OFF value to the 20th percentile among tokens for which it does not activate. To measure causal importance, we calculate the symmetric KL divergence between the resulting ON- and OFF-clamped next-token distributions. If the resulting KL divergence is higher, the feature is more causally important. We compute the average symmetric KL over 20 contexts from the Pile sampled from around the 80th percentile of the feature ON (active) distribution, comparing the outputs from each unperturbed example against the same example clamped to OFF.

We find *punctuality and causal importance are in tension for SAE features*. Figure 4 depicts heatmaps of these two metrics at several layers—they are always negatively correlated.

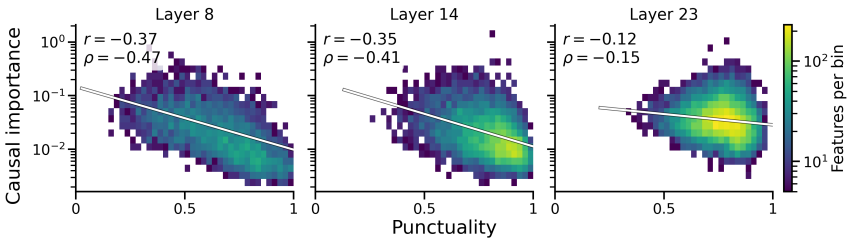

Figure 4: **For SAE features, punctuality and causal importance are in tension.** r is Pearson correlation and ρ is Spearman (rank) correlation between punctuality and causal importance per-feature. The negative correlation is weaker in later layers but always negative.

We posit that highly punctual features are less causal because they represent ephemeral noise. Intuitively, important features should persist across layers. If punctual features disappear just as suddenly as they appeared, then their lack of causal impact makes sense. We find this conjecture is supported by a lack of persistence in punctual features. In Figure 5(a), we measure a feature's *persistence*—its availability at the layer immediately after its source layer—and find that punctuality and persistence are negatively correlated. Furthermore, ephemeral features—those most punctual and least persistent—are often noise with low causal importance (Figure 5(b)). See Figure 5(b), which illustrates how highly ephemeral features appear and disappear in quick succession.

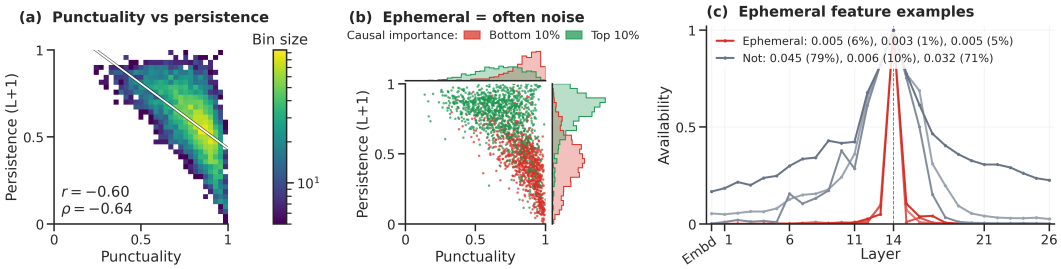

Figure 5: **For SAE features, ephemeral features (most punctual and least persistent) are often noise with low causal importance.** Plots depict layer 14 features, see Appendix for complete results from other layers. (a) More punctual features are less persistent and establishes the existence of ephemeral features (in the bottom right quadrant). (b) The lowest causal features are predominantly ephemeral. (c) Example availability trajectories of randomly selected ephemeral features in contrast to random non-ephemeral features. The legend reports causal importance symKL (percentile); the ephemeral features are at the bottom percentiles.

6 Do crosscoders find causally important, punctual features?

In the Bias-in-Bios circuit, we found similarly distributed features sourced from many different layers. This complicated the circuit analysis because these features were easier understood when grouped together than as separate features. Crosscoders (Lindsey et al., 2024) are a variant of SAEs intended to simplify circuit analysis by producing shared features across layers. Do these alternative encoders also find higher-quality—more causally important, more punctual—latent reasoning features than SAEs? In this section, we analyze the Gemma Scope crosscoder (McDougall et al., 2025) and find that they improve on our SAE dictionaries in several ways.

The crosscoder's feature dictionary is shared across layers, so the sparsity penalty creates optimization pressure for each feature to reconstruct activations at as many possible downstream layers as possible. This produces two effects relevant to punctuality:

1. Crosscoder features found at *earlier* layers should be more *persistent* than SAE features because crosscoders introduce an incentive to reconstruct many downstream layers. Lindsey et al. (2024) claim "By tracking features that continue to exist in the residual stream, crosscoders can remove "duplicate features" from analysis".
2. Crosscoder features found at *later* layers should be more *punctual* than SAE features because earlier layer crosscoder features will have already captured model computation in earlier layers.

And because punctual SAE features tend to be less persistent, we expect crosscoders to choose less punctual early features and more punctual later features. Our results, shown in Figure 6, are compatible with these hypothesized effects.

Finally, we study whether crosscoder features are *jointly more punctual and causally important* than SAE features. This is a necessary test because we have shown that punctuality and causal impact are negatively correlated in SAEs, so crosscoders might find more punctual features by compromising on their importance. We see, however, that our comparison findings stand even after considering causal impact. The contour map in Figure 7 shows that crosscoders place more probability mass on points that are both more punctual (to the right) and causally important (higher up) than SAEs.

We also perform a bivariate pairwise dominance analysis. We say that a feature (Pareto) dominates another if it is at least as punctual and causally important, and is strictly better on at least one. At each layer, we compare each crosscoder (CC) feature with each SAE feature, measuring the proportion of features where CC Pareto dominates SAE, vice versa, or neither.

We compute the dominance statistic $D = \Pr(\text{CC} \succ \text{SAE}) - \Pr(\text{SAE} \succ \text{CC})$ where $D > 0$ tells us a random crosscoder feature is more likely to dominate an SAE feature than vice versa. The 95% confidence interval is obtained via bootstrap resampling: we resample the SAE and crosscoder

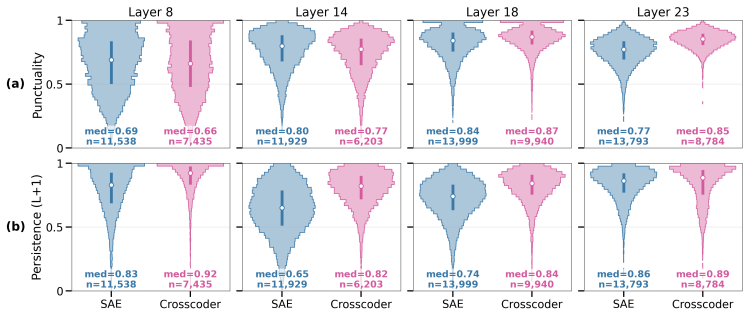

Figure 6: **In *earlier* layers, crosscoder features are more *persistent* than SAE features. In *later* layers, crosscoder features are more *punctual* than SAE features.** This is expected because crosscoders are advertised to de-duplicate features across layers: **(a)** later layer features should therefore be more novel and more punctual, because earlier layer model computation is already captured by earlier layer crosscoder features. **(b)** Furthermore, earlier layer crosscoder features are incentivized to reconstruct activations at many downstream layers, so they should be more persistent. We cannot establish a causal relationship, but our results are compatible with these hypotheses.

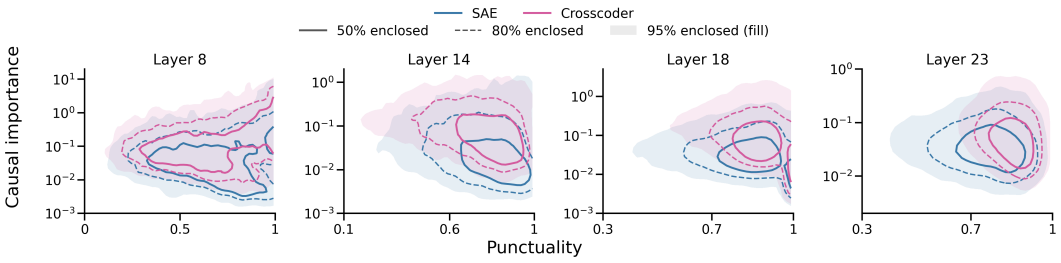

Figure 7: **Crosscoders find features that are jointly more punctual and causally important than SAEs.** Crosscoders place more probability mass on points that are both more punctual (to the right) and causally important (higher up) than SAEs.

features independently and with replacement, preserving the original cohort sizes, 2,000 times, then compute the dominance statistic and report the 2.5th and 97.5th percentiles[6]

Importantly, crosscoder features are more sparse than SAE features, which may contribute to their advantage in punctuality. In Appendix A.2 we control for sparsity by stratifying SAE and crosscoder features and only comparing features in the same strata. We find that crosscoders retain their Pareto advantage, though it is weaker in the sparser strata and stronger in the more frequent strata.

7 Related Work

Mechanistic explanations as step-by-step computation. Mechanistic interpretability often aims to describe model computation in steps. Geva et al. (2023) and Feucht et al. (2026) decompose factual recall and reasoning about cyclic concepts, respectively, into discrete steps. Sparse feature circuits aim to decompose model computation into steps in an unsupervised manner (Marks et al., 2025; Lindsey et al., 2025). Prior work on circuit evaluation focus on causal faithfulness, sufficiency, or minimality (Chan et al., 2022; Shi et al., 2024) but these criteria do not establish that the intermediate features themselves correspond to distinct, orderable steps of computation.

Linear features. Our analysis builds on the linear representation hypothesis, according to which models encode features as directions in activation space (Mikolov et al., 2013; Park et al., 2024).

[6] D is a 2-sample U-statistic, so we use a Hájek projection for computational efficiency to avoid recomputing all pairwise comparisons in every resample.

Table 1: **Crosscoder features exhibit bivariate dominance (punctuality and causal importance) over SAE features.** We compare each crosscoder (CC) feature with each SAE feature, measuring the proportion of features where crosscoder Pareto dominates SAE, vice versa, or neither (incomparable). The dominance statistic is $D = \Pr(\text{CC} \succ \text{SAE}) - \Pr(\text{SAE} \succ \text{CC})$ so $D > 0$ tells us a random crosscoder feature is more likely to dominate an SAE feature than vice versa. The 95% confidence interval is obtained via 2,000 bootstrap resamples.

Layer	CC $\succ$ SAE	SAE $\succ$ CC	Incomparable	D	95% CI
L8	0.35	0.18	0.48	0.17	[0.16, 0.18]
L14	0.30	0.095	0.61	0.21	[0.20, 0.22]
L18	0.40	0.078	0.52	0.32	[0.32, 0.33]
L23	0.45	0.070	0.48	0.38	[0.38, 0.39]

Because models may represent more features than there are dimensions in their activation spaces, individual neurons can be polysemantic and features may be represented in superposition (Elhage et al., 2022; Bricken et al., 2023). SAEs aim to recover these latent directions by learning overcomplete sparse feature dictionaries (Cunningham et al., 2023), and these features have been used as units in automated circuit discovery (Marks et al., 2025).

Feature quality. A growing body of work evaluates whether SAE features constitute useful units of analysis. Gao et al. (2025) consider reconstruction quality, recovery of hypothesized features, and the sparsity of downstream effects among others. SAEBench demonstrates that improvements on common metrics do not necessarily imply improvements on other desiderata (Karvonen et al., 2025). Song et al. (2026) argue for measuring consistency of learned features across independent SAE training runs. The Mechanistic Interpretability Benchmark (MIB) evaluates methods according to their ability to recover task-relevant causal variables and pathways, and finds that SAE features do not consistently outperform neurons for causal-variable localization (Mueller et al., 2025). We introduce a complementary criterion motivated specifically by step-by-step mechanistic explanations.

Feature evolution across layers. Tuned Lens tracks how model predictions are iteratively refined across layers (Belrose et al., 2025), while Balcells et al. (2024) study how SAE features persist, combine, and specialize across adjacent layers. Fel et al. (2024) use layerwise decodability to study how increasingly complex features emerge with depth, and related information-theoretic work measures computation progress across layers (Xu et al., 2020; Finzi et al., 2026).

8 Discussion

In general if we aim to explain complex latent reasoning, we will need to decompose model computation into steps. Our circuit case study found a cluster of punctual features, supporting the idea that punctual features might be valuable for constructing circuits to explain complex latent reasoning.

We studied crosscoders as one alternative method to vanilla SAEs for finding desirable features and saw that crosscoders achieve a Pareto improvement on punctuality and causal impact; future work can study other dictionary learning approaches such as transcoders. It would also be exciting to develop new feature-finding methods that explicitly optimize for punctuality.

Our analysis of punctuality of sparse dictionary features did not require human-interpretability, which we consider a strength of our method, but future work should study how these (and other) desiderata relate. An important limitation of our work is that we only consider the punctuality of feature readability and not its causality. It is possible that these properties look very different, so future work should carefully analyze how they relate. Lastly, we should explicitly test the premise that punctual features are better for constructing circuits.

Overall, we hope highlighting punctuality as an important property that makes features suitable as circuit components can inspire future work that uses this criterion to find good circuits more efficiently.

AI USE STATEMENT

In this work, we primarily used generative AI tools to write code, launch experiments, and generate figures. The original codebase was written manually by the first author, and only later in the project did we adopt generative AI tools for code writing assistance. We cross-checked baseline results from AI-written code against results obtained by manually-written code, reviewed all AI-written code, and spot-checked specific additional details, such as consistency in figures. We also used generative AI to search for select particular related works which we then read, to suggest the statistical test in the last section which we checked details against references online, and to brainstorm terminology and titles.

We have not used generative AI tools for writing, drafting sections of the paper, generating synthetic datasets, or proving mathematical claims.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We believe that interpretability research is fundamentally dual-use for AI safety: any work that aims to advance our understanding of how language models work can in principle be used to make them better, towards any end. However we believe we ought to deeply understand this technology that has become ubiquitous; potential for dual-use is not a reason to immediately abandon all interpretability research.

REPRODUCIBILITY STATEMENT

All relevant implementation details for our methodology have been provided in the main text. We will release the complete code for reproducing our work upon release of the paper.

REFERENCES

Daniel Balcells, Benjamin Lerner, Michael Oesterle, Ediz Ucar, and Stefan Heimersheim. Evolution of sae features across layers in llms, 2024. URL https://arxiv.org/abs/2410.08869.

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, 2025. URL https://arxiv.org/abs/2303.08112.

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

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: a method for rigorously testing interpretability hypotheses, 2022.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

Thomas Fel, Louis Bethune, Andrew Kyle Lampinen, Thomas Serre, and Katherine Hermann. Understanding visual feature reliance through the lens of complexity, 2024. URL https://arxiv.org/abs/2407.06076.

Sheridan Feucht, Tal Haklay, Usha Bhalla, Daniel Wurgaft, Can Rager, Raphaël Sarfati, Jack Merullo, Thomas McGrath, Owen Lewis, Ekdeep Singh Lubana, Thomas Fel, and Atticus Geiger. Arithmetic in the wild: Llama uses base-10 addition to reason about cyclic concepts, 2026. URL https://arxiv.org/abs/2605.01148.

Marc Finzi, Shikai Qiu, Yiding Jiang, Pavel Izmailov, J. Zico Kolter, and Andrew Gordon Wilson. From entropy to epiplexity: Rethinking information for computationally bounded intelligence, 2026. URL https://arxiv.org/abs/2601.03220.

Leo Gao, Tom Dupre 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*, volume 2025, pp. 26721–26754, 2025.

Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, 2017. doi: 10.1101/214262. URL https://www.biorxiv.org/content/early/2017/11/12/214262.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, 2023.

Thomas Heap, Tim Lawson, Lucy Farnik, and Laurence Aitchison. Automated interpretability metrics do not distinguish trained and random transformers, 2026. URL https://arxiv.org/abs/2501.17727.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Isaac Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum Stuart McDougall, Kola Ayonrinde, Demian Till, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *Forty-second International*

Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=qrU3yNfX0d.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. URL https://arxiv.org/abs/2408.05147.

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.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2025. URL https://arxiv.org/abs/2403.19647.

Callum McDougall, Arthur Conmy, János Kramár, Tom Lieberum, Senthooran Rajamanoharan, and Neel Nanda. Gemma scope 2 - technical paper, 2025. URL https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/gemma-scope-2-helping-the-ai-safety-community-deepen-understanding-of-complex-lang
Gemma_Scope_2_Technical_Paper.pdf.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff (eds.), *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1090/.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, Martin Tutek, Amir Zur, David Bau, and Yonatan Belinkov. MIB: A mechanistic interpretability benchmark. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sSrOwve6vb.

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, 2022. URL https://arxiv.org/abs/2209.11895.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models, 2024. URL https://arxiv.org/abs/2311.03658.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features, 2025. URL https://arxiv.org/abs/2501.16615.

Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 411–420, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.

Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David M. Blei. Hypothesis testing the circuit hypothesis in llms, 2024. URL https://arxiv.org/abs/2410.13032.

Xiangchen Song, Aashiq Muhamed, Yujia Zheng, Lingjing Kong, Zeyu Tang, Mona Diab, Virginia Smith, and Kun Zhang. Mechanistic interpretability should prioritize feature consistency in sparse autoencoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2172–2210, 2026.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Axbench: Steering llms? even simple baselines outperform sparse autoencoders, 2025. URL https://arxiv.org/abs/2501.17148.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints, 2020. URL https://arxiv.org/abs/2002.10689.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning?, 2025. URL https://arxiv.org/abs/2402.16837.

A APPENDIX

Table of contents:

A.1 Bias-in-Bios full circuit 14
A.2 Controllng for sparsity in comparing SAEs and crosscoders 14
A.3 Full results for layer of largest gain, punctuality, causal importance, persistence (all features, all layers) 14
A.4 SAE layer 8: simple vs complex features 14
A.5 Alternative punctuality metrics 15
A.6 Alternative methods to measuring availability: training a regression or classification probe 16
A.7 Neuron baseline details 16

A.1 BIAS-IN-BIOS FULL CIRCUIT

See Figure 8 for the circuit with all feature annotations.

A.2 CONTROLLNG FOR SPARSITY IN COMPARING SAES AND CROSSCODERS

In Table 1 we established that crosscoders find features that are jointly more punctual and causally important than SAEs. However crosscoder features are also more sparse than SAEs, which may contribute to a potential punctuality advantage. Here we re-run the analysis by controlling for sparsity via stratification: crosscoder and SAE features are only compared within the same stratum. Crosscoders always dominate (D is positive at every stratum at layer), and the gap is generally larger for the more frequent strata. All 95% confidence intervals are above 0 except for the most sparse bucket in layer 14.

Figure 9 shows the sparsity distributions between the two features.

Figure 10 shows the sparsity distributions between the two features but revealing the long tail of crosscoder sparsity that is filtered out because they are too sparse for evaluation (less than 10 active tokens on the test set). Follow-up work should study the behavior of this long tail on larger datasets.

Figure 11 shows the results of the bivariate dominance statistical test stratified on sparsity. The sparsest strata generally show a smaller advantage for crosscoders than the number reported in the main paper without accounting for sparsity, while the most frequent strata show a greater advantage.

A.3 FULL RESULTS FOR LAYER OF LARGEST GAIN, PUNCTUALITY, CAUSAL IMPORTANCE, PERSISTENCE (ALL FEATURES, ALL LAYERS)

Figure 12 depicts full violin plots for layer of largest gain, punctuality, causal importance, and persistence.

Figure 13 provides the same metrics but filters out simple input features that make their largest gain in availability at layer 0.

Figure 14 provides heatmaps for punctuality vs causal importance for all features at all layers.

Figure 15 provides heatmaps for punctuality vs persistence for all features at all layers.

Figure 16 depicts the top and bottom decile causality features for all features at all layers.

A.4 SAE LAYER 8: SIMPLE VS COMPLEX FEATURES

Figure 17 plots all trajectories for SAE layer 8 features that make their largest gain at layer 1 (thus, simple input features) or layer 8 (thus, complex intermediate features). These are the two most frequent modes of the distribution.

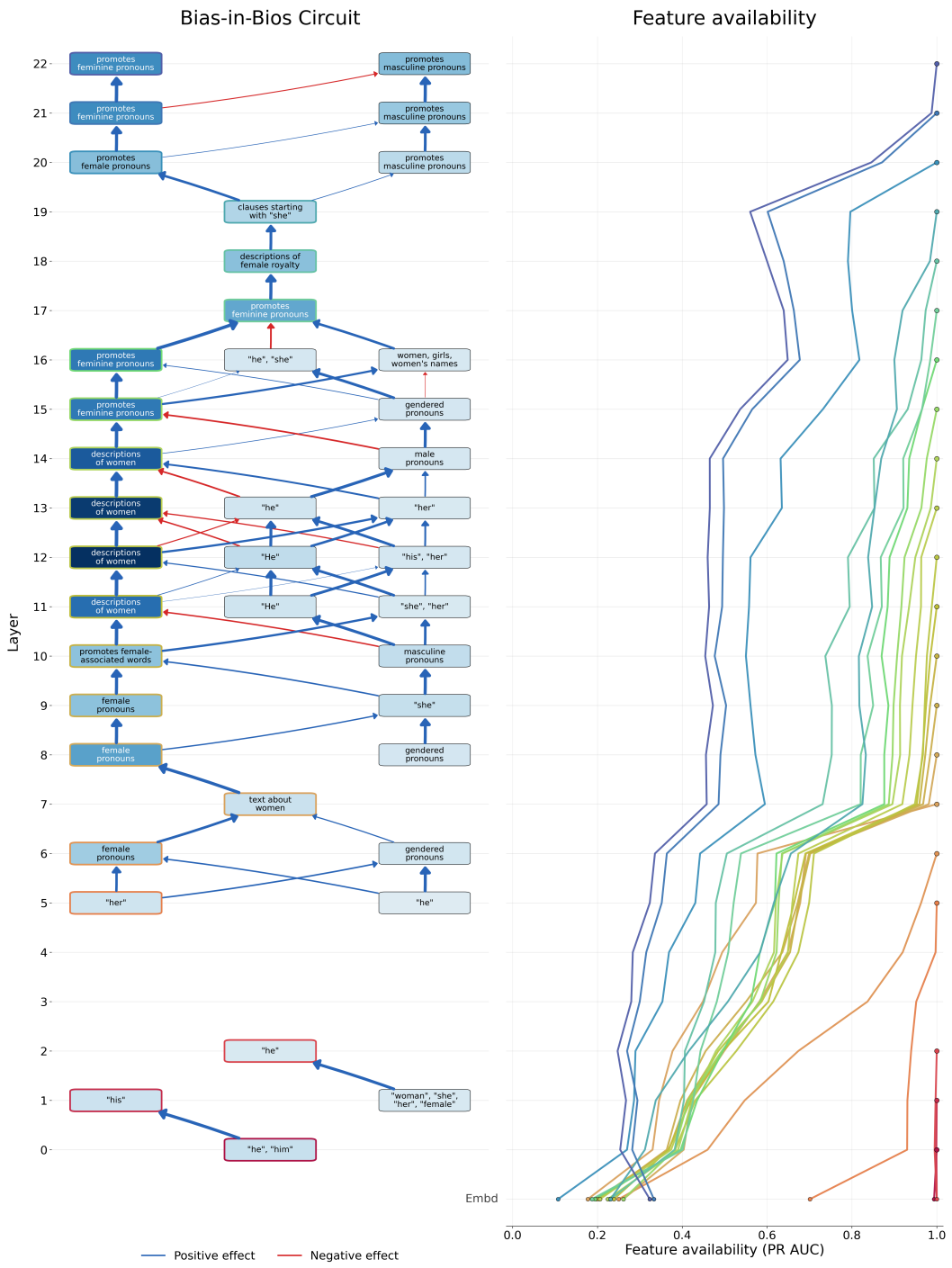

Figure 8: Bias-in-Bios circuit with feature annotations. Note layers are 0-indexed here whereas they are 1-indexed in the main text.

A.5 Alternative punctuality metrics

We explore various alternative definitions of punctuality:

- area over the availability curve
- largest 1-layer gain in availability

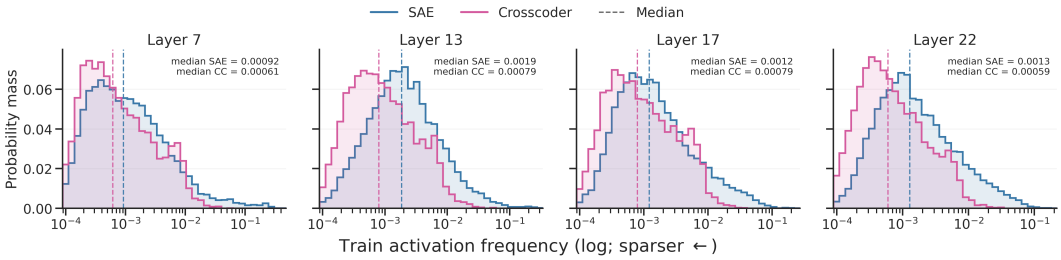

Figure 9: Comparison of sparsity between SAEs and crosscoders

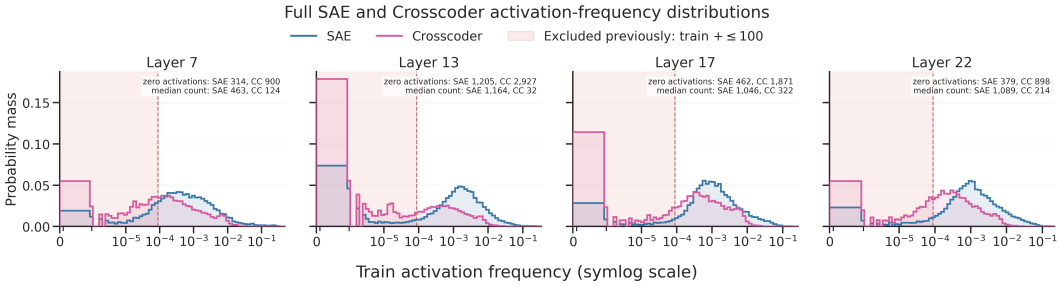

Figure 10: Comparison of sparsity between SAEs and crosscoders, showing the long tail of features that we filter out because they are too sparse for evaluation (less than 10 active tokens on the test set).

- largest consecutive 2-layer gain
- largest (non-consecutive) 2-layer gain
- count of the number of layers outside the availability PR AUC range [0.3, 0.9]
- index of the layer that contributed the largest gain

Figure 18 shows violin plots comparing the punctuality distributions for each feature according to each definition. Figure 19 reports distributions after filtering out simple input features who make the most gain at the first layer. Figure 20 reports the Spearman (rank) and Pearson correlations between the various metrics per feature. The metrics are strongly correlated for the baselines and moderately correlated for SAEs and crosscoders. The metrics measure different things so it makes sense that there is some disagreement. For example, largest 1-layer gain is agnostic to which layer the largest gain occurred at: whether it is close or far from the feature's source layer. Depending on what we are interested in, this may be desirable or undesirable. If evaluating orderability of features in circuits as-is, it is important to consider which layer the feature became available at, not just how quickly it became available. Figure 21 reports correlations again after filtering out simple input features who make the most gain at the first layer.

A.6 Alternative methods to measuring availability: training a regression or classification probe

Our mainline results use verbatim availability: was the feature separable along the same direction at earlier layers? Here we compare punctuality using availability measured by training a linear probe (classification or regression). Figure 22 shows violin plots and figure 23 shows the Spearman and Pearson correlations for punctuality computed by different decoding methods per feature. For SAEs in particular, the different decoding methods are highly similar.

A.7 Neuron baseline details

We wish to find the appropriate setup to evaluate neurons as a fair baseline for SAEs. Neurons at threshold 0 (the intrinsic relu threshold) activate frequently (the median neuron is active on half of

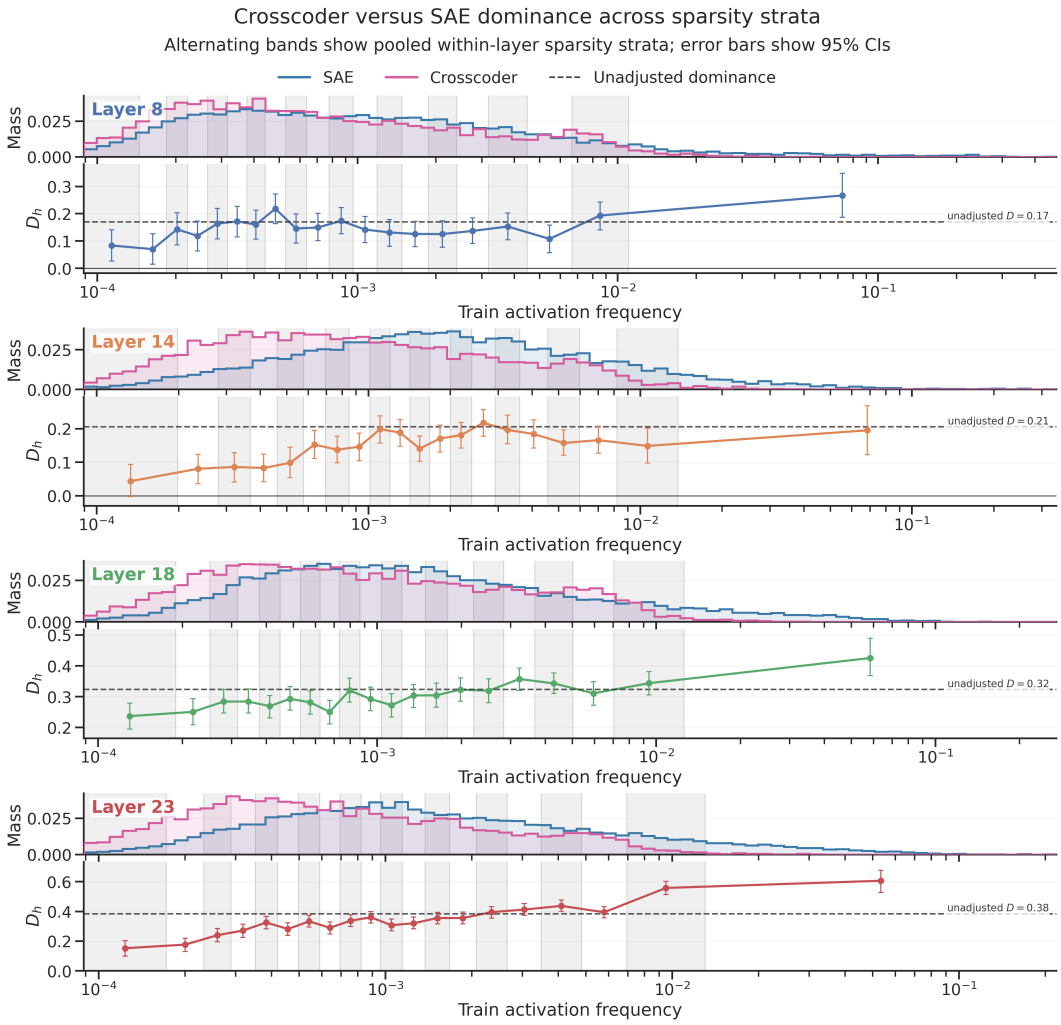

Figure 11: **Crosscoder vs SAE bivariate dominance (punctuality and causal importance) per stratum.** Sparsity is controlled for by stratifying: crosscoder and SAE features are only compared within the same stratum. Crosscoders always dominate (D is positive at every stratum at layer), and the gap is generally larger for the more frequent strata. All 95% confidence intervals are above 0 except for the most sparse bucket in layer 14.

tokens) so they are intrinsically harder to separate than sparse features. Punctuality distributions are depicted in Figure 24 and neuron activation frequencies are depicted in Figure 25.

Sparsity has an important effect on punctuality, as shown by the large difference in neurons at SAE sparsity vs their native ReLU activation threshold of 0. We run three experiments to provide a fair comparison between SAEs and neurons:

First, we adjust the neuron threshold to match SAE sparsity. This is the setup reported in the main text.

Second, we adjust the SAE threshold to match neuron sparsity.

Third, we retain both SAE and neuron thresholds at their native values and instead subsample the data to match the count of positive and negative tokens per feature.

In all three setups, we find that SAEs are consistently less punctual than neurons. The sparsity control via threshold adjustment experiments are reported in Figure 26, and the data subsampling experiments are reported in Figure 27.

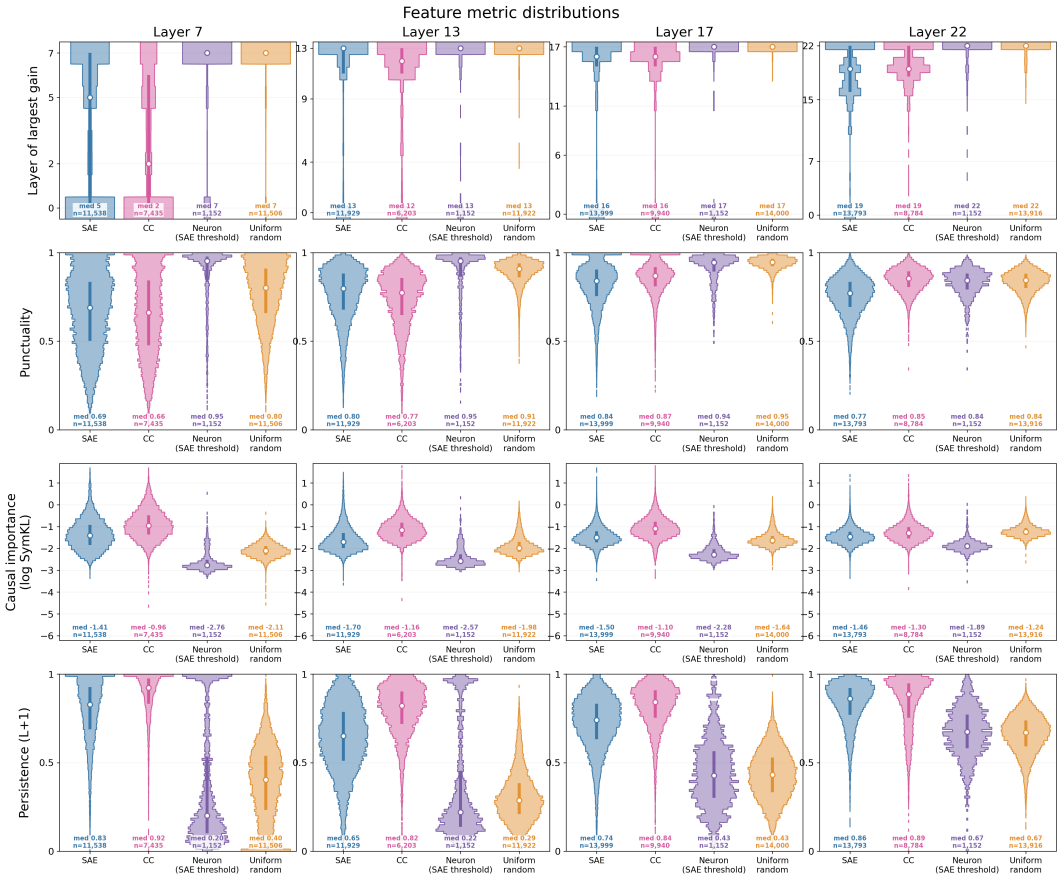

Figure 12: Full distributions for layer of largest gain, punctuality, causal importance, and persistence.

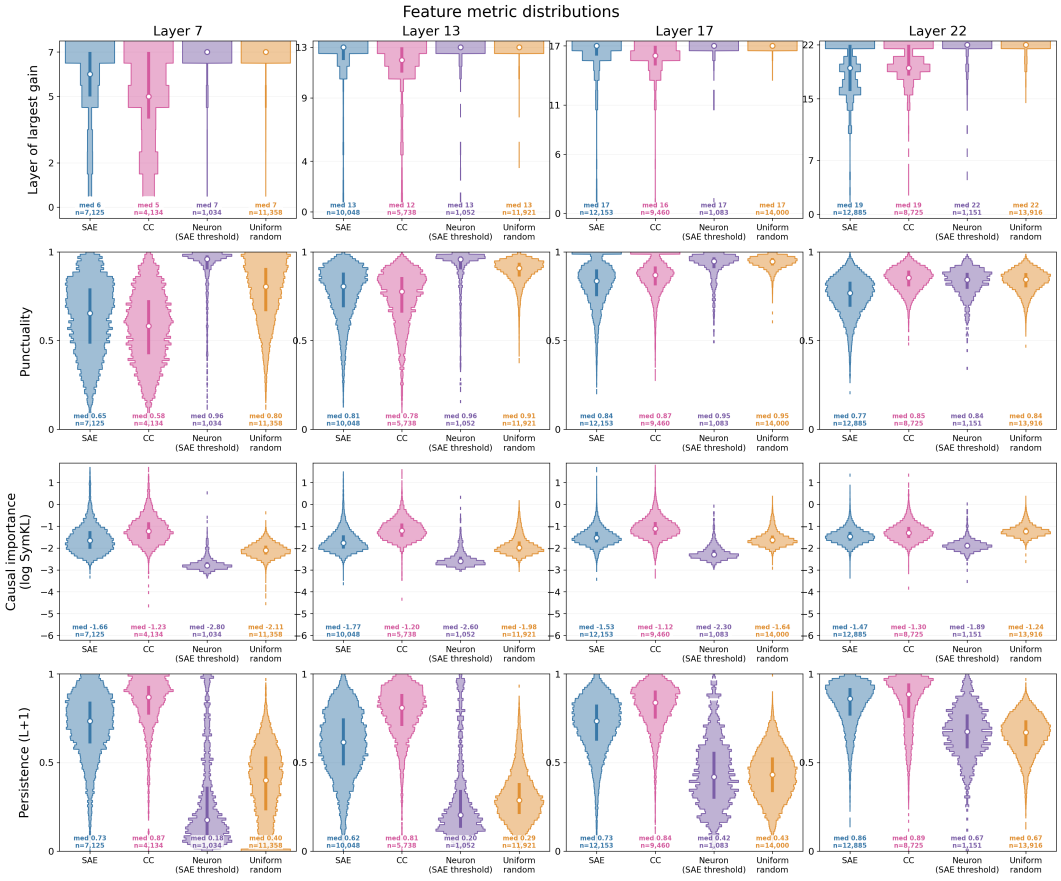

Figure 13: Full distributions for metrics of interest, filtering out features who make their largest gain at layer 0.

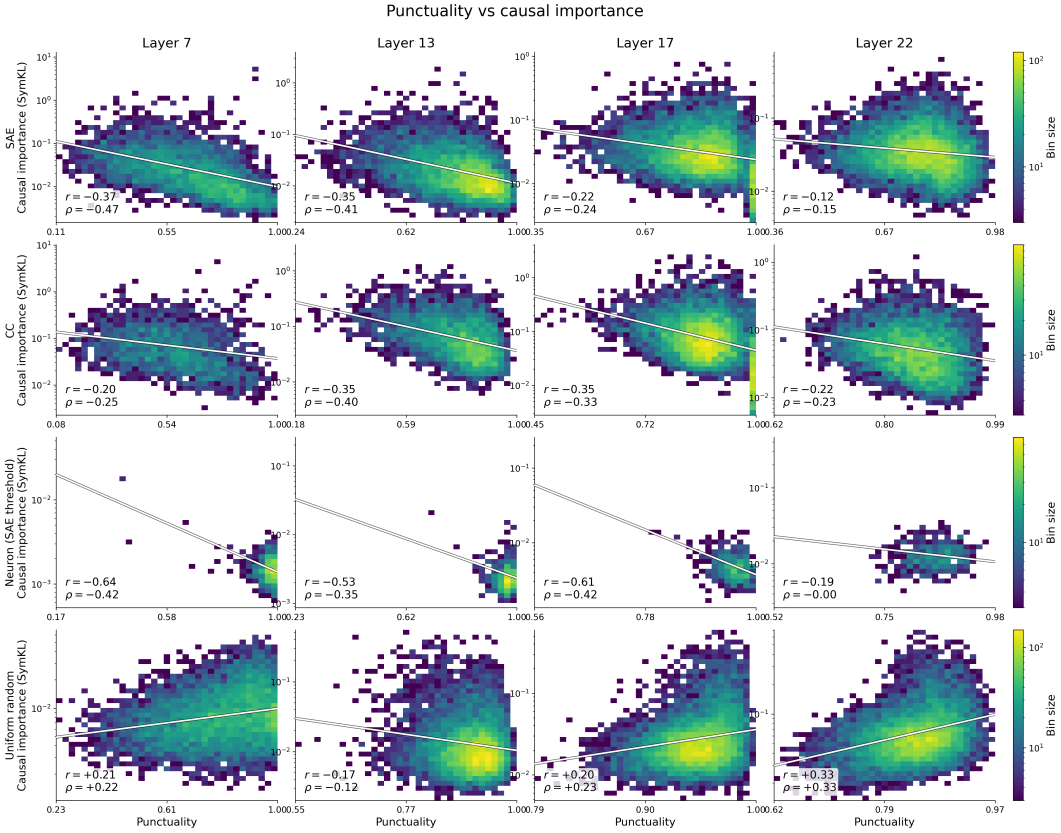

Figure 14: Full results for punctuality vs causal importance.

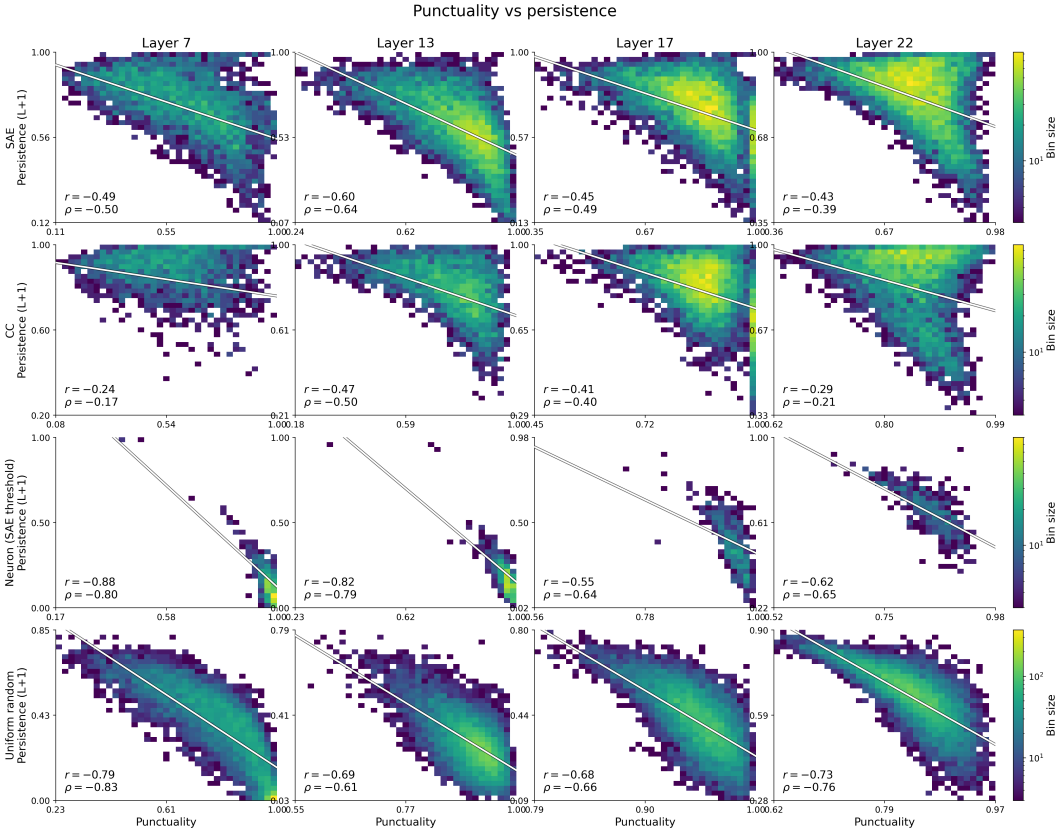

Figure 15: Full results for punctuality vs persistence.



Figure 16: Full results for ephemerality.

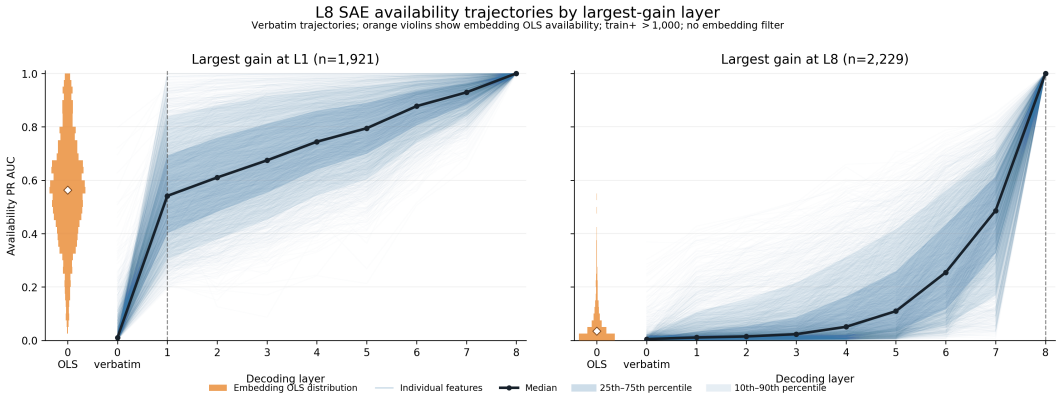

Figure 17: Trajectories for SAE layer 8 features that make their largest gain at layer 1 (thus, simple input features) or layer 8 (thus, complex intermediate features). These are the two most frequent modes of the distribution.

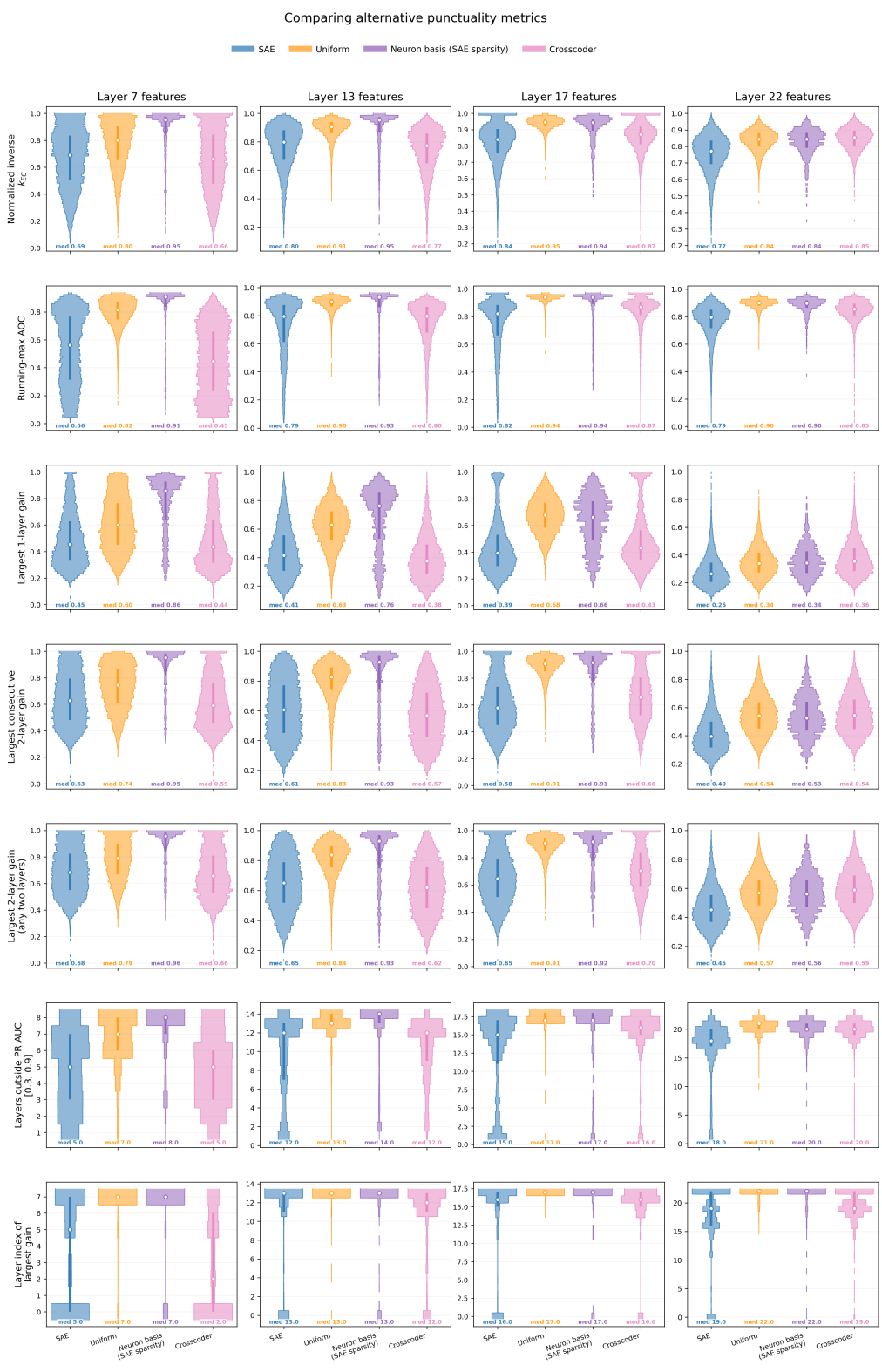

Figure 18: Comparison of potential alternative metrics of punctuality. Most high level qualitative trends hold up across metrics, such as the relative ordering of punctuality across feature types. Some metrics do not sufficiently distinguish the feature types, e.g. the layer of the largest gain in availability (last row). We leave a detailed analysis of the best metric for punctuality to future work.

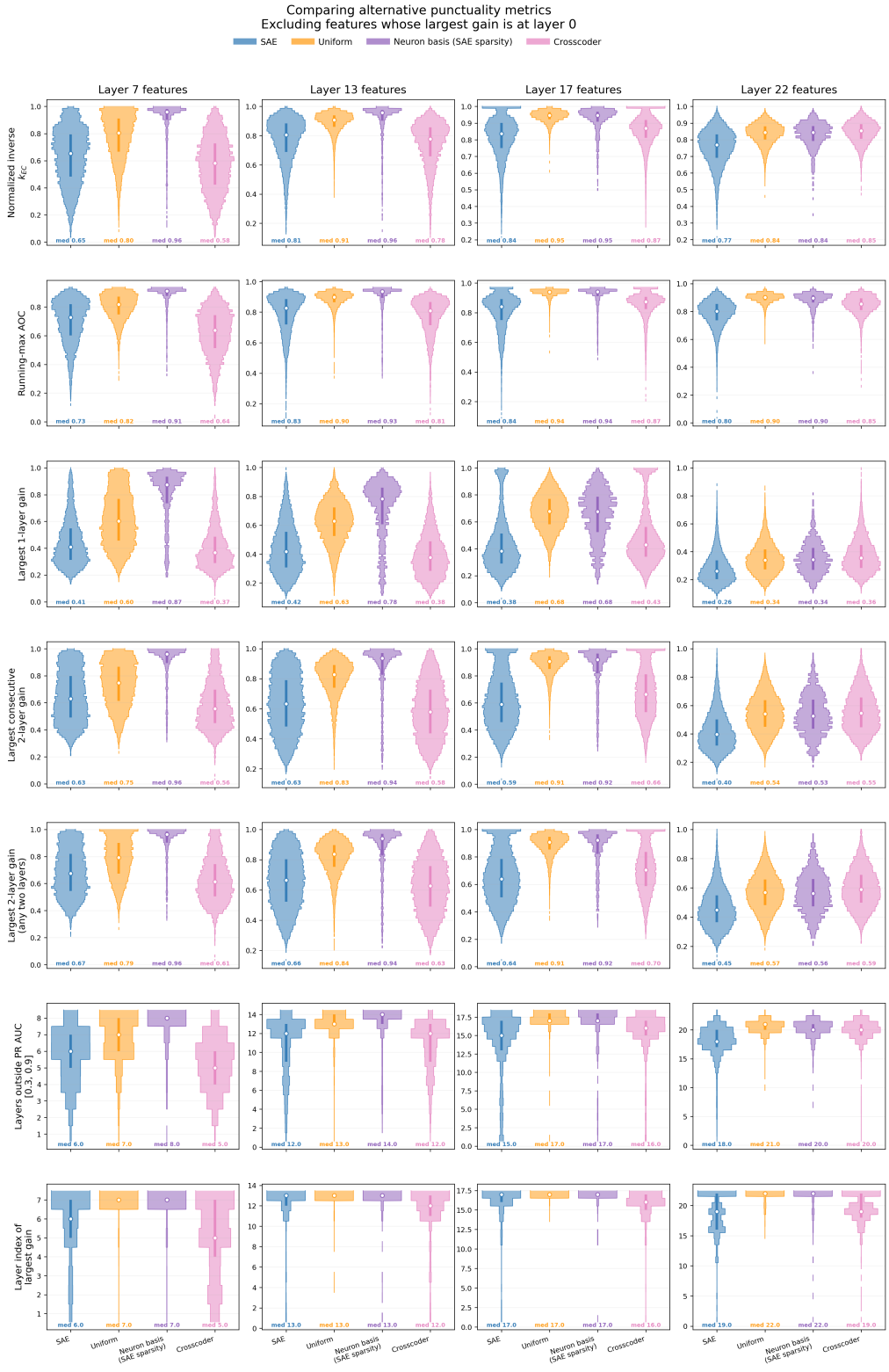

Figure 19: Comparison of potential alternative metrics of punctuality, filtering out features who make their largest gain at layer 0, which are simple input features.

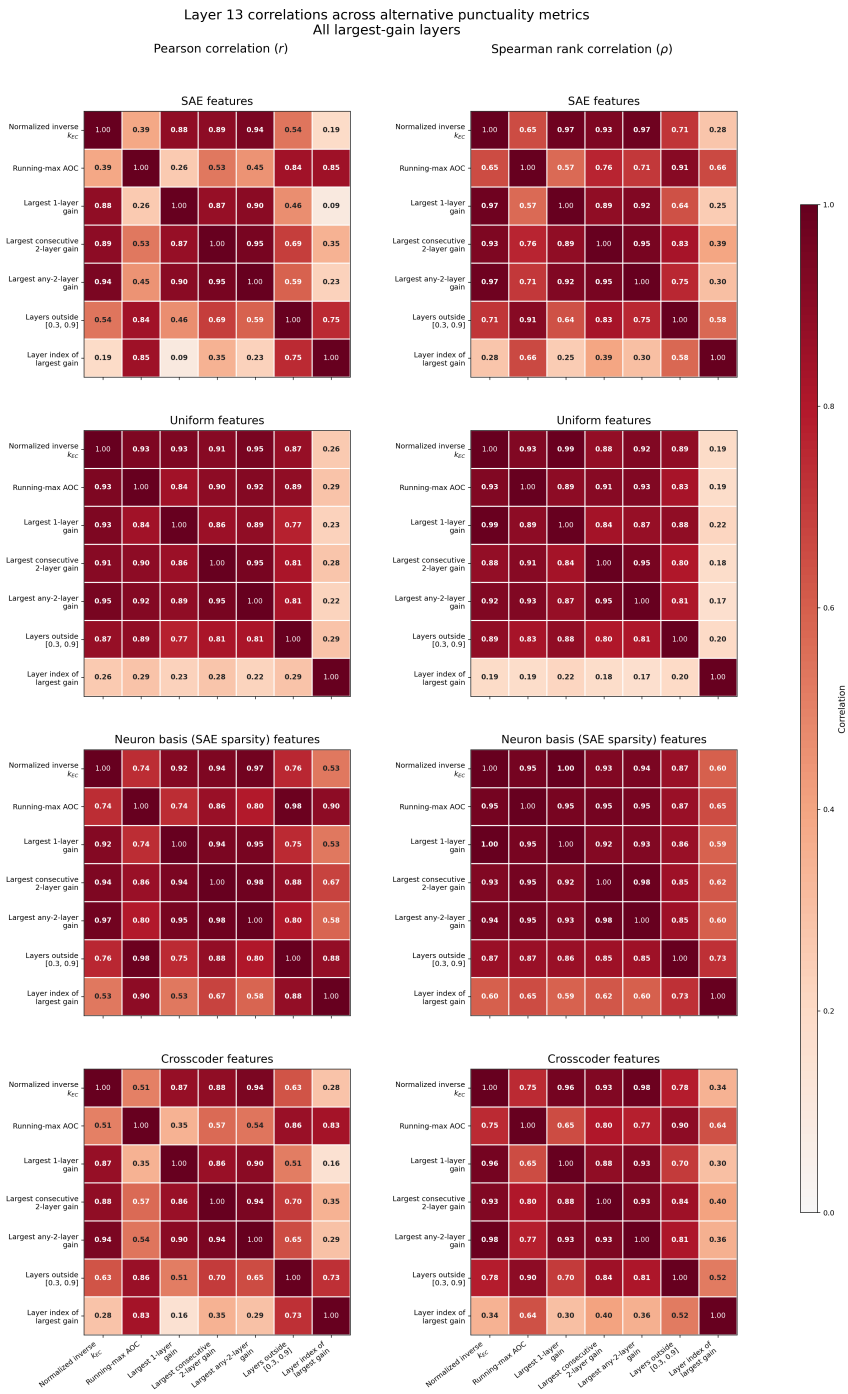

Figure 20: Correlations between alternative punctuality metrics for each feature in layer 13. The metrics are strongly correlated for the baselines and moderately correlated for SAEs and crosscoders. The metrics measure different things so it makes sense that there is some disagreement. For example, largest 1-layer gain is agnostic to which layer the largest gain occurred at: whether it is close or far from the feature's source layer. Depending on what we are interested in, this may be desirable or undesirable. If evaluating orderability of features in circuits as-is, it is important to consider which layer the feature became available at, not just how quickly it became available. The layer of the largest gain in availability (last metric) does not sufficiently distinguish features, and as such it is less correlated with other metrics. We leave a detailed analysis of the best metric for punctuality to future work.

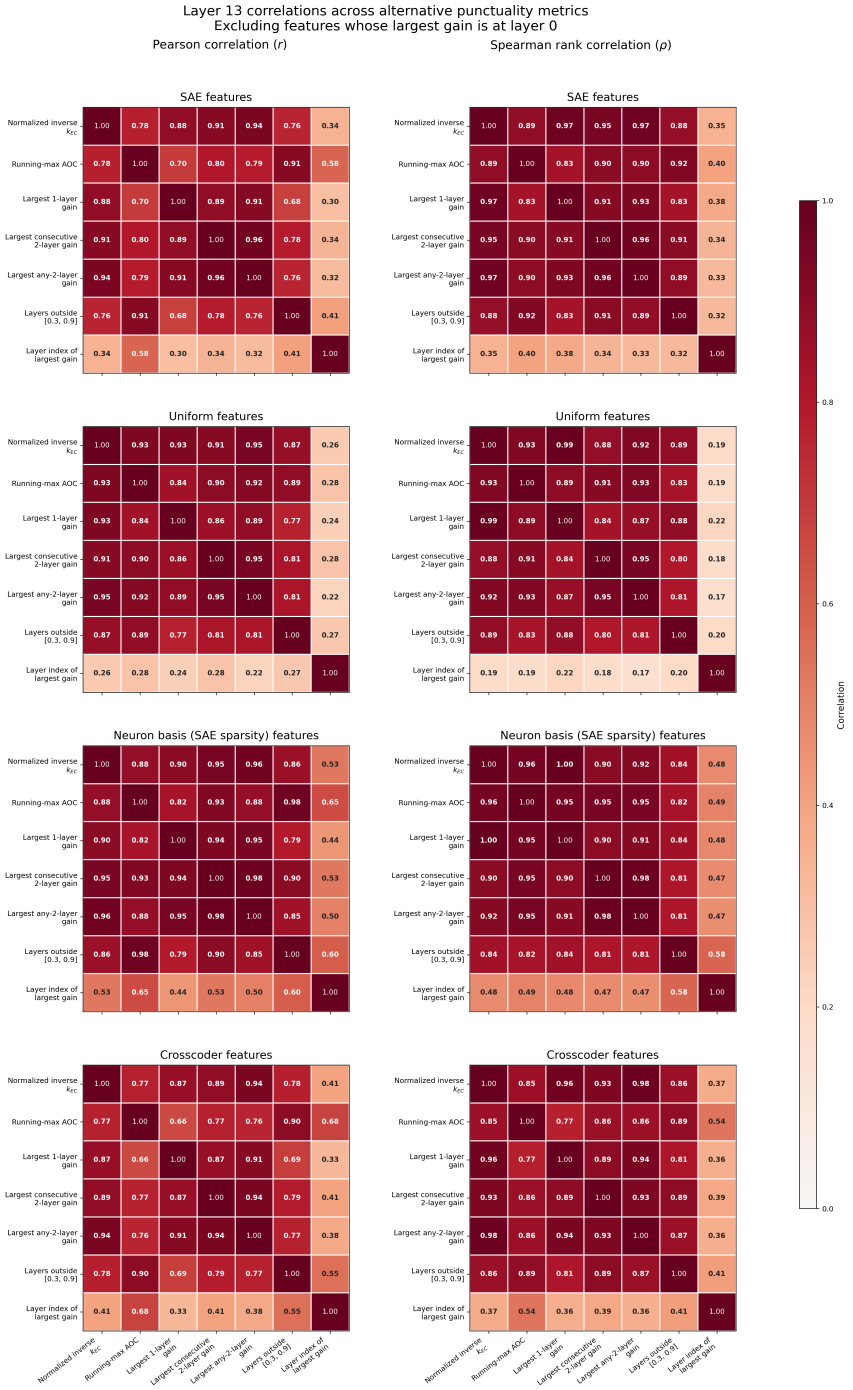

Figure 21: Correlations between metrics at layer 13 when filtering out features that make the largest gain in availability at layer 0.

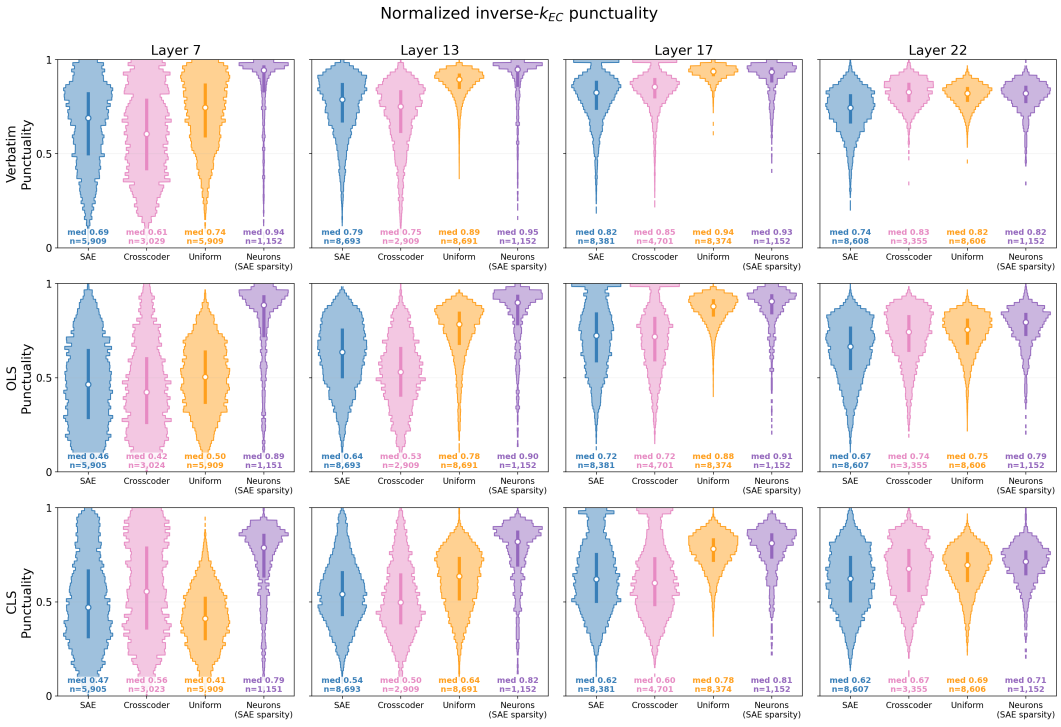

Figure 22: Punctuality measured via OLS (ordinary least squares regression) and CLS (logistic regression) directions in comparison to verbatim. Relative comparisons across feature types generally hold across decoding methods (verbatim vs OLS vs CLS) though there are some exceptions.

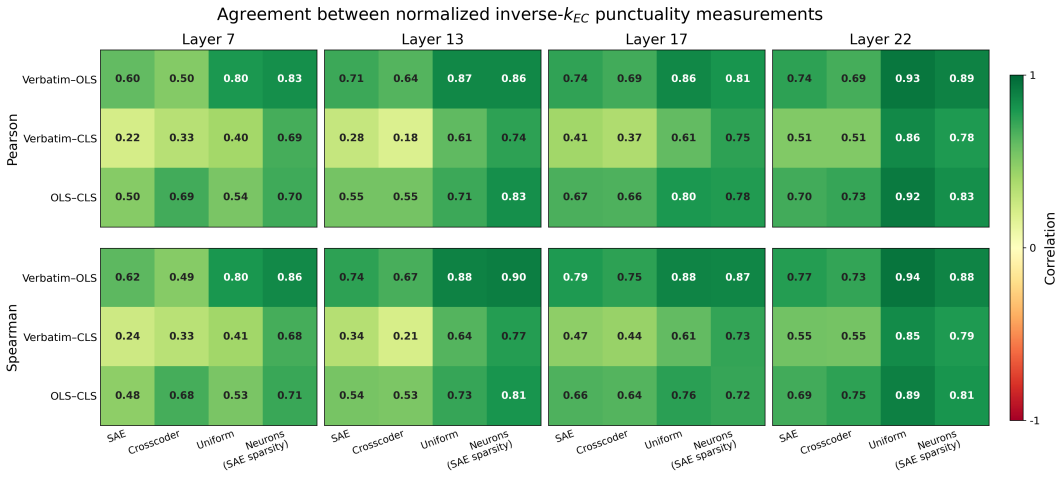

Figure 23: Correlations for punctuality measured via OLS (ordinary least squares regression) and CLS (logistic regression) directions in comparison to verbatim.

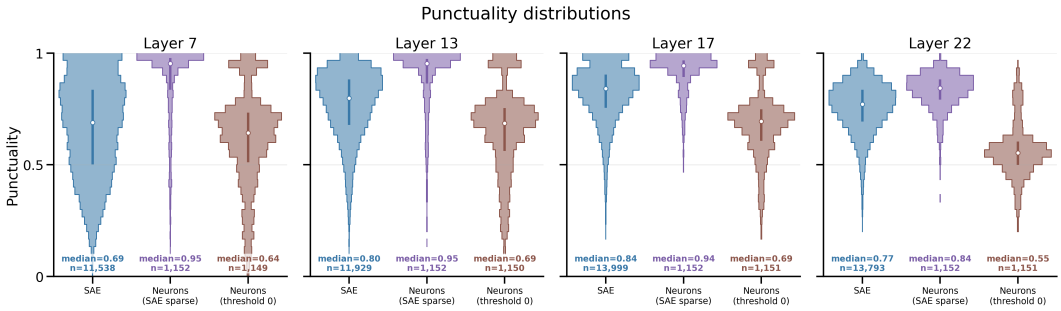

Figure 24: SAE vs neuron threshold 0 punctuality. Neurons at threshold 0 (the intrinsic relu threshold) activate frequently (the median neuron is active on half of tokens) so they are intrinsically harder to separate than sparse features.

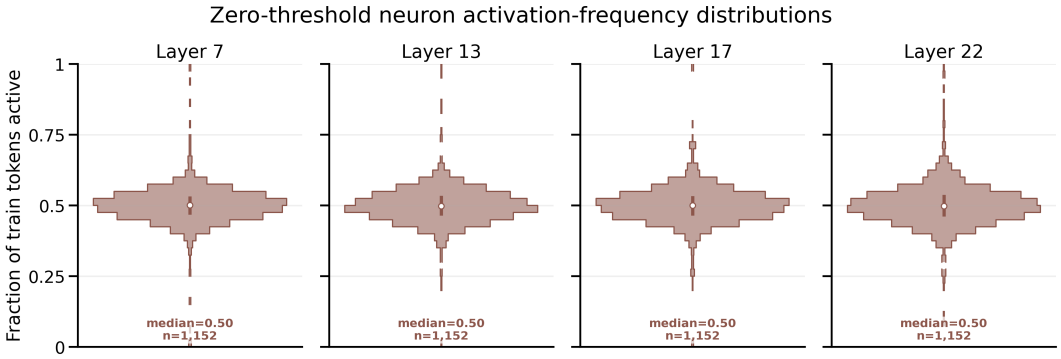

Figure 25: Neurons at threshold 0 (intrinsic relu threshold) activate much more frequently than SAE features, so they would appear much less punctual if we didn't control for activation sparsity.

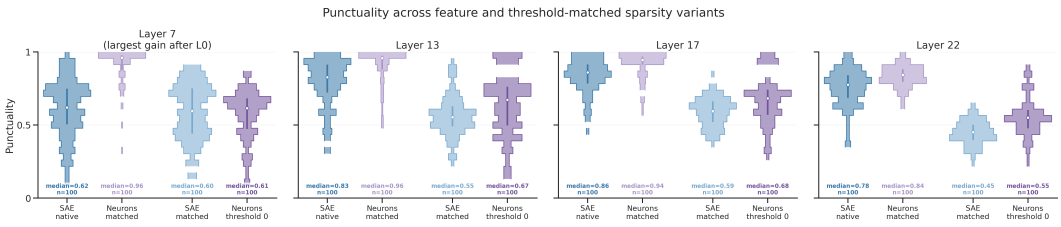

Figure 26: SAE vs neuron punctuality matching sparsity by adjusting thresholds (in both directions). SAE features are always less punctual. Note: we compare on a small dataset ($n = 100$), future work should increase the sample size.

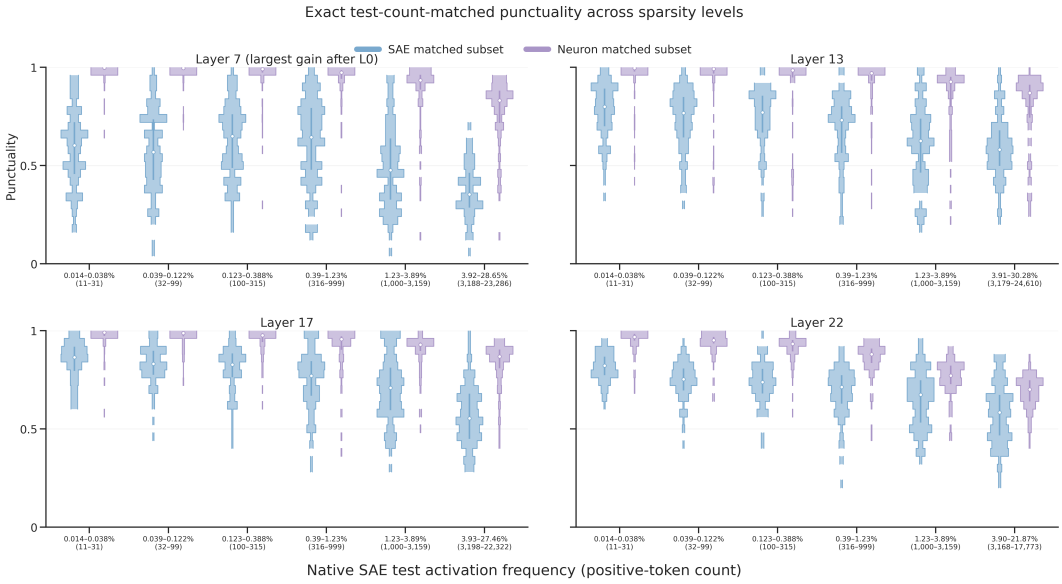

Figure 27: SAE vs neuron punctuality keeping thresholds for ON/OFF at their native value (0 for neurons) but matching the amount of positive and negative data such that SAEs do not have a punctuality advantage because they are much sparser (and thus easier to separate). After the control, SAE features are always less punctual at all frequency strata.