

---

# Can We Partially Rewrite Transformers in Natural Language?

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

1       The greatest ambition of mechanistic interpretability is to completely rewrite deep  
2       neural networks in a format that is more amenable to human understanding, while  
3       preserving their behavior and performance. In this paper we evaluate whether  
4       sparse autoencoders (SAEs) and transcoders can be used for this purpose. We use  
5       an automated pipeline to generate explanations for each of the sparse coder latents.  
6       We then simulate the activation of each latent on a number of different inputs using  
7       an LLM prompted with the explanation we generated in the previous step, and  
8       “partially rewrite” the original model by patching the simulated activations into  
9       its forward pass. We find that current sparse coding techniques and automated  
10      interpretability pipelines are not up to the task of rewriting even a single layer  
11      of a transformer: the model is severely degraded by patching in the simulated  
12      activations. We believe this approach is the most thorough way to assess the quality  
13      of SAEs and transcoders, despite its high computational cost.

## 14   1 Introduction

15   While large language models (LLMs) have reached human level performance in many areas [Guo et al.,  
16   2025], we understand little about their internal representations. Early mechanistic interpretability  
17   research attempted to explain the activation patterns of individual neurons [Olah et al., 2020, Gurnee  
18   et al., 2023, 2024], but research has found that most neurons are “polysemantic”, activating in  
19   semantically diverse contexts [Arora et al., 2018, Elhage et al., 2022].

20   Sparse autoencoders (SAEs) were proposed to address polysemanticity [Cunningham et al., 2023].  
21   SAEs consist of two parts: an encoder that transforms activation vectors into a sparse, higher-  
22   dimensional latent space, and a decoder that projects the latents back into the original space. Both  
23   parts are trained jointly to minimize reconstruction error. Recently, a significant effort was made to  
24   scale SAE training to larger models, like GPT-4 [Gao et al., 2024] and Claude 3 Sonnet [Templeton  
25   et al., 2024], and they have become an important interpretability tool for LLMs. Recently Dunefsky  
26   et al. [2024] proposed sparse *transcoders* as an alternative method for extracting interpretable features  
27   from LLMs. The architecture of the transcoder is identical to that of an SAE, but it is trained to  
28   predict the output of a feedforward network given its input. With a good enough transcoder, we  
29   should be able to entirely replace the original FFN with its transcoder approximation, thereby partially  
30   rewriting the model in terms of more interpretable primitives.

31   While it seems that neurons and sparse autoencoder latents can be explained by looking at the  
32   examples they activate on [Bills et al., 2023], some are more easily understood by their downstream  
33   effects [Gur-Arieh et al., 2025]. Paulo et al. [2024] took inspiration on Bills et al. [2023] and built  
34   an automated pipeline for generating natural language explanations of SAE latents and evaluating  
35   how good these explanations are. Rigorously measuring how interpretable an explanation is still a  
36   complicated and methodologically fraught task.

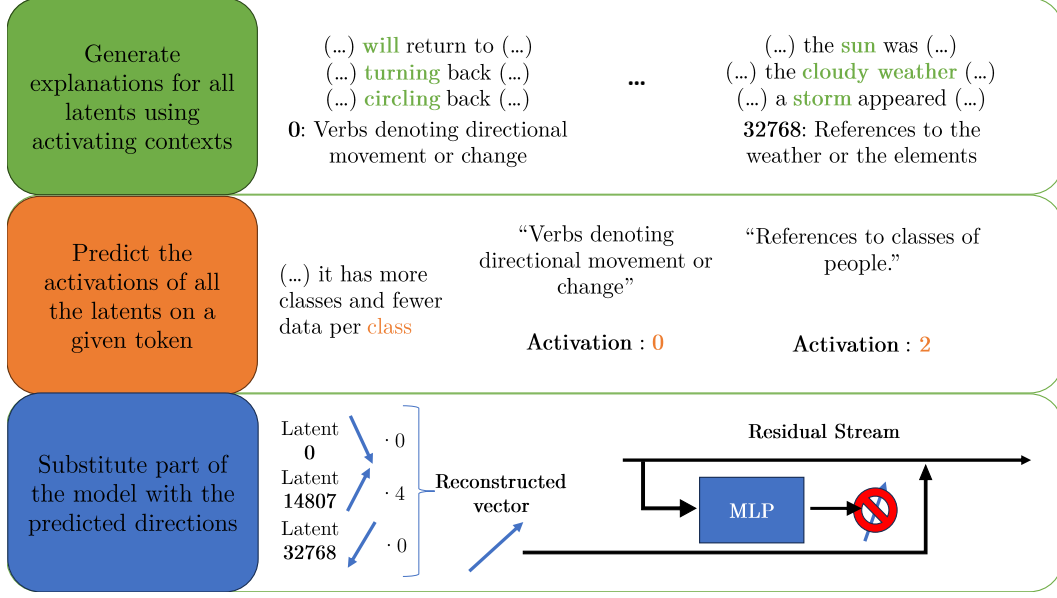


Figure 1: **Partially rewriting an LLM.** After training a transcoder or SAE, we generate explanations for all the latents using contexts where that latent is active. An LLM is tasked to find patterns in these activations and output a simple, single-sentence explanation for that latent. These explanations are used by another LLM instance to predict whether the latent should be active on a given token. After post-processing those predictions, we produce a reconstructed vector by pushing the simulated activations through the decoder.

The idea of rewriting a neural net in a more interpretable form is not new. The “microscope AI” framework [Hubinger, 2019] aims to analyze a neural network’s learned representations to gain actionable insights for humans, rather than using the network directly. These insights would likely take the form of natural language explanations of the network’s features and circuits. Microscope AI aims to reduce risks associated with model deployment while still benefiting from the model’s knowledge. Imitative generalization is a proposal to extend this idea by jointly optimizing the network and its human-interpretable annotations to maximize their prior likelihood [Barnes, 2021]. Sparse autoencoders and similar techniques can then be used as “explainer” models, explaining the behavior of the network by sparsely decomposing its activations.

In this work, we pursue the following idea: if the latents of a transcoder are interpretable enough, we can *simulate* its activations using natural language explanations. Specifically, we replace the encoder of the transcoder with an LLM prompted to predict the activation of each latent given its explanation and the textual context. We then patch this modified transcoder back into the model, hopefully yielding behavior nearly identical to the unpatched model. In the limit, we could use this to “rewrite” every feedforward layer in the model in terms of interpretable features and operations on those features.

Unfortunately there are several roadblocks for this approach. Firstly, despite a majority latents seeming to have interpretable activations, a large fraction is either uninterpretable or not well-captured by our current pipeline. Secondly, even the ones that seem interpretable are hard to explain in their totality, as it is hard to generate simple explanations that are both contextually sensitive and specific. Finally, the predictions of the simulator LLM are not calibrated: it over-estimates the frequency with which latents fire, and how strong their activations should be. Because we believe that there are many potential ways to improve the quality and specificity of explanations, we see this test as a potential benchmark of transcoder and SAE interpretability.

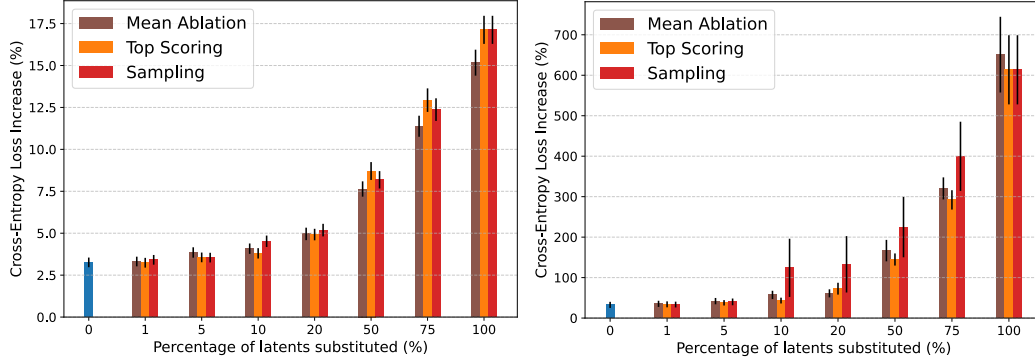


Figure 2: **Cross entropy loss increase for different fractions of transcoder and SAE substitution.** We compute the CE loss over 10K prompts, for the transcoder (left) and over 1K prompts SAE (right) respectively, by substituting parts of the encoder with natural language explanations. Bars in orange show the loss increase when selecting the latents with the highest interpretability scores for replacement. Bars in red show the loss increase when randomly selecting a subset of latents to replace. Bars in brown show the loss increase caused by replacing a random subset of the latents with their mean activations— a simple baseline that we should ideally be able to overcome. The blue bar represents the increase in loss from using the sparse coders as is. Bar heights represent the mean value of the increase with respect to the base loss, while error bars represent the standard error. The interpretability score used for selecting latents is detection scoring, [Paulo et al., 2024, page 5], computed over 100 positive and 100 negative samples. Over this set of prompts, Pythia had a cross entropy loss of  $3.19 \pm 0.09$  nats per token. Prompts where the loss was lower than 0.1 nats were excluded.

## 2 Methods

### 2.1 Sparse coder training

We train different types of sparse coders to evaluate their potential for partial rewriting. We begin by training a skip-transcoder on the MLP of the sixth layer of Pythia 160M [Biderman et al., 2023] as well as a skip-transcoder on the MLP of the fifteenth layer of SmolLM2 135M [Allal et al., 2025]. We also train a sparse autoencoder on the output of the 6th layer of Pythia 160M. The skip-transcoders have a linear “skip connection,” which Paulo et al. [2025] found improves the ability to approximate the original MLP at no cost to interpretability scores. That is, the transcoder takes the functional form

$$\hat{y} = \mathbf{W}_2 \text{TopK}(\mathbf{W}_1 x + \mathbf{b}_1) + \mathbf{W}_{\text{skip}} x + \mathbf{b}_2 \quad (1)$$

where  $x$  is the input of the MLP and  $\hat{y}$  is the reconstructed output of the MLP. Both  $\mathbf{W}_2$  and  $\mathbf{W}_{\text{skip}}$  are zero-initialized, and  $\mathbf{b}_2$  is initialized to the empirical mean of the MLP outputs, so that the transcoder is a constant function at the beginning of training. All models are trained to minimize the mean squared error  $\|y - \hat{y}\|_2^2$  between the model’s output and the target module output.

The sparse coders trained on Pythia have 32768 latents, while the ones trained on SmolLM2 have 18432 latents. Sparsity is continuously enforced on the sparse coder latents using the TopK activation function proposed by Gao et al. [2024] with  $k = 32$  for Pythia and  $k = 128$  for SmolLM2.<sup>1</sup> The sparse coders trained on Pythia are trained over the first 8B tokens of the Pile [Gao et al., 2020], using the Adam optimizer [Kingma, 2014], a sequence length of 2049, and a batch size of 64 sequences.<sup>2</sup> The skip-transcoder trained on SmolLM2 was trained on 1B tokens of the FineWeb-Edu dataset [Lozhkov et al., 2024], part of the training corpus of the model, using the schedule-free Signum optimizer [Bernstein et al., 2018, Defazio et al., 2024], a sequence length of 2049, and a batch size of 64 sequences. For this model, schedule-free Signum was used because it let to a significantly better reconstruction loss and a lower number of dead latents.

<sup>1</sup>The higher  $k$  for SmolLM2 was an attempt to achieve higher rewriting performance by increasing the number of active latents, after our initial experiments with Pythia at  $k = 32$  yielded disappointing results.

<sup>2</sup>The optimizer appeared to converge after 1B tokens, so we early stopped the training run.

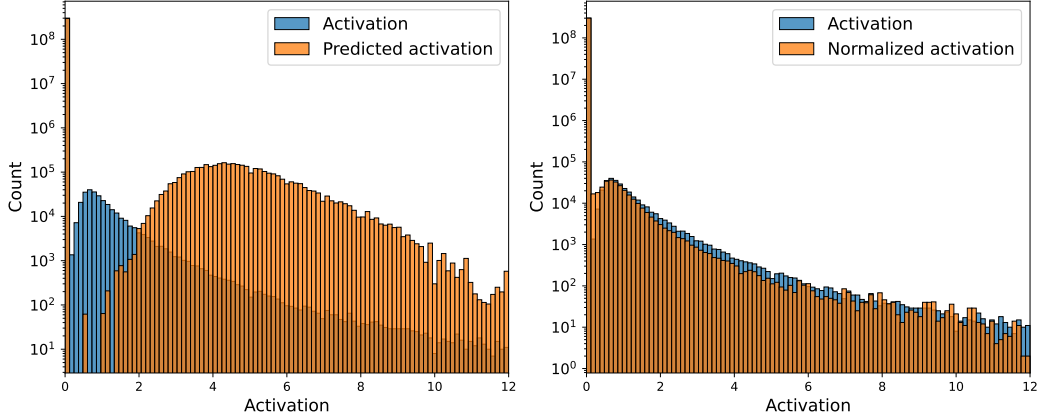


Figure 3: **Distribution of predicted activations for all latents.** On the left we compare the distribution of predicted activations before normalization, and on the right we show what the distribution looks like after quantile normalization. Before normalization, the predictor model systematically over-predicts high activation values by multiple orders of magnitude. Quantile normalization primarily has the effect of enforcing a prior in favor of features not being active.

## 83 2.2 Interpreting latents

84 We use the automated interpretability pipeline released by Paulo et al. [2024] to generate explanations  
85 and scores for transcoder and SAE latents. For all sparse coders we collected activations over 10M  
86 tokens of the same dataset they were trained on. For each latent, representative samples of its  
87 activations are sampled and shown to an LLM, in our case Llama 3.1 70b Instruct [Dubey et al.,  
88 2024], which is told to give a succinct explanation that summarizes the activations (Figure 1, first  
89 row). We only explain latents that are active on more than 200 instances over the 10M samples taken.  
90 The LLM is shown 40 examples, 4 examples from each of the 10 activation deciles for that latent.

91 After explanations are generated for each latent, they are scored. We also use the automated  
92 interpretability pipeline from Paulo et al. [2024] for this process. We use both fuzzing and detection  
93 to score the latents. To compute the detection score, Llama 3.1 70b Instruct is given the explanation  
94 of the latent and a set of examples. The LLM then has to decide which examples activate the latent  
95 and which don’t using the explanation that was given. We repeat this process 20 times, showing  
96 five examples each time, totaling 50 non-activating examples and 50 activating examples. We use  
97 stratified random sampling from the 10 activating deciles of that latent. At the end the detection score  
98 of the latent is given by the  $F_1$  score of the scorer,  $F_1 = \frac{2}{p^{-1} + r^{-1}}$ , where  $p$  is the precision and  $r$  is  
99 the recall. The fuzzing score is computed with a similar protocol, but instead the LLM is shown a set  
100 examples where one token is highlighted in each, and the model has to label each highlighted token  
101 as activating or non-activating. Again the final metric used is the  $F_1$  score.

## 102 2.3 Simulation

103 The next step in the pipeline is to use an LLM to simulate the activation of a latent in a context given  
104 the latent’s explanation (Figure 1, middle panel). For this task, we select up to 10K sequences, each  
105 32 tokens long, and highlight the final one. The LLM is tasked to output a single number from 0 to 9  
106 to quantify the activation of the latent on that token for that specific sequence. Because our scorer  
107 model has a token for each independent digit, we can estimate the expected value by summing the  
108 probability of each digit times the value of the digit. We then map this number to the real activations  
109 by dividing the expected value by nine (yielding a number between 0 and 1), then multiplying this  
110 value by the maximum observed activation for the latent. The prompt used for simulation can be  
111 found in Appendix A.

### 2.3.1 Quantile normalization

We found in early experiments that Llama produces highly uncalibrated predictions of feature activations: the marginal distribution of the predicted activations differs markedly from the marginal distribution of the true activations (Figure 3, left panel). Patching these uncalibrated activations into the model yields very poor results. To alleviate this problem, we use quantile normalization, which monotonically transforms the model’s predictions in such a way that their marginal distribution matches that of the true activations. This transformation is an optimal transport map under a variety of cost functions [Santambrogio, 2015].

We compute the quantile normalizer separately for each individual feature, using the empirical CDFs of the simulator’s predicted activations and the true activations of a transcoder computed from a random sample of 10% of the data. Once the quantile normalizer has been computed, this transformation is then applied to all simulator predictions. This transformation successfully reduces the number of predicted active latents to numbers that are coherent with the true distribution.

## 3 Evaluation

We evaluate the rewriting of the model by measuring the resulting cross-entropy loss in next token prediction on chunks of text sampled from the Pile. These chunks of texts have lengths that are uniformly sampled from 32 to 256 tokens. This range was selected such that, at its minimum, there is a significant number of tokens that can be shown to the simulator model, and at its maximum the sequences are short enough that the presence of very long range features is unlikely.

Simply replacing a single MLP with a transcoder increases the model’s cross-entropy loss by about 10%. This loss increase is equivalent to that of using an early Pythia checkpoint, one that was trained on only 25% of the data,<sup>3</sup> rather than the final Pythia checkpoint. Rewriting any part of the transcoder in natural language will necessarily degrade the model’s performance even further. Consequently, we focus on rewriting a single MLP block, since rewriting all MLP blocks simultaneously would cause the model to become completely unusable. The same applies to the residual stream SAE: the cross-entropy loss of the model when adding a single SAE to the residual stream increases by 33%. This is equivalent to using a Pythia checkpoint trained on less than 10% of the data.

After collecting the predicted activations for all features of a given token in a given sentence, and after applying quantile normalization to those predictions, we use the TopK activation function to ensure it has the expected level of sparsity. This activation vector is fed into the decoder, yielding a reconstructed output vector. In the case of the transcoders, the original output of the MLP is discarded and in its place we inject the reconstructed vector (Figure 1), while in the case of the SAE the output of the layer is discarded and in its place the reconstructed vector is patched in.

We evaluate over 10K different contexts for the Pythia transcoder, 2K for the SmolLM2 transcoder, and 1K for the Pythia SAE, measuring the cross-entropy loss for next-token prediction. This process is very expensive, as each context requires individually prompting the simulator for 32 thousand latents. We therefore had to make a total of 327 million queries to the simulator model. Our simulator models were hosted locally using VLLM [Kwon et al., 2023], and the experiments took 1 week on 4 nodes with 8xA40 GPUs.

**Partial rewriting.** We also experiment with mixing predicted and ground truth latent activations in varying proportions, allowing us to examine the effect of rewriting only part of the encoder. We do this in two different ways:

1. Select the top  $k$  most interpretable features according to our evaluation pipeline [Paulo et al., 2024]. This is labeled “Top scoring” in Figure 2.
2. Sample  $k$  features uniformly at random from the transcoder. This is labeled “Sampling” in Figure 2.

The “Mean Ablation” baseline condition is like “Sampling” except that we replace the selected latents with their mean activations.

---

<sup>3</sup>The cross-entropy loss of Pythia 160M checkpoints on this set of prompts is not monotonic with training time, so a more precise estimate is not possible.

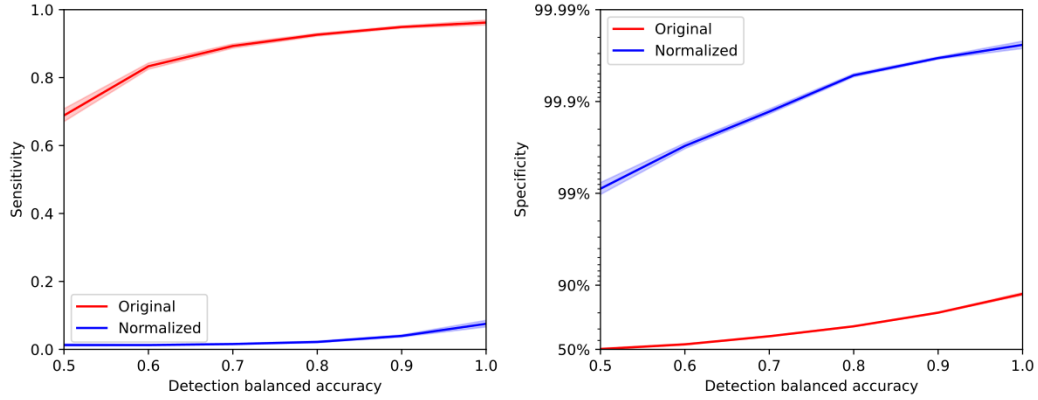


Figure 4: **Detection score predicts sensitivity and specificity.** Binning explanations by their scores makes it evident that high-scoring explanations are more specific and sensitive.

## 4 Results

The left panel of Figure 2 illustrates how cross-entropy loss increases on Pythia as we replace more and more transcoder latents with their simulated counterparts. Unfortunately, we see that the “Top Scoring” and “Sampling” conditions both result in very similar loss increases to the baseline “Mean Ablation” condition, for most fractions of substitution. The same behavior can be seen when using the residual stream SAE (Figure 2, right panel), and when using the transcoder trained on SmolLM 2 (Figure A2). For some fractions and models we observe that randomly selecting which latents to substitute, instead of always substituting the most interpretable latents, leads to a smaller increase in the cross entropy loss, showing that the interpretability metrics are somewhat capturing how easy it is to describe the behaviour of the latents. While our substitution method works better than mean ablation for some fractions, it is not consistently better and in fact when we do full substitution we find that there is almost no difference between the two.

This indicates that our explanation pipeline is incapable of rewriting even a small fraction of a transformer without causing a substantial increase in the next-token prediction loss— an increase that is comparable to simply replacing neurons in the transformer with their mean values. This is a negative result which suggests more work needs to be done to improve explanation quality.

### 4.1 Explanations are not detailed enough

One of the reasons that we need to calibrate the activation predictions is because we find that current explanations do not allow the model to be specific enough. To understand this, let’s consider the case that the classifier only achieves a specificity of 99%. The transcoder used in this work has 32768 latents, and if the LLM predictor can only achieve a specificity of 99% this means that, that on average, it predicts that there are 320 active latents, 10 times more than the actual number ( $k = 32$ ). Even with a sensitivity of 100%, where the model correctly identifies all of the latents that should be active, it is unlikely that the top 32 predictions all would be the correct ones. We observe that on average the current automatic latent explanation setup has a specificity of around 80%, meaning that on average the model identifies close to 6 thousand latents as being active.

By performing quantile normalization, a large chunk of incorrectly predicted activations, activations that should be zero but were given a non-zero value by the predictor, are set back to zero. This significantly increases the specificity, enough that some of the original model’s performance is maintained, but at the same time this significantly decreases the sensitivity, as some of the correctly classified active latents are also set to zero.

We find that detection scores [Paulo et al., 2024, page 5] are predictive of the specificity and sensitivity of an explanation, with higher scoring latents corresponding to explanations that have higher specificity and sensitivity (Figure 4). This is expected, as detection scoring corresponds to detecting whether a given latent is active on a given context, which is similar to our simulation task

195 in this work. For the same reason, latents with higher fuzzing scores also have higher sensitivity and  
196 specificity (Figure A1).

## 197 **5 Limitations**

198 We divide the limitations of this work into three categories. One type of limitation is related with the  
199 quality of sparse coders we investigated. We only explored TopK sparse coders, trained on one to ten  
200 billion tokens, with an expansion factor of 32x. However, we don't see any evidence in the literature  
201 that other activation functions would make the sparse coders more interpretable, so it is unlikely that  
202 other architectures would significantly improve our results. Larger sparse coders would lead to lower  
203 reconstruction losses but would require even higher specificity.

204 The quality of explanations is a limitation of our work, as it is possible that better explanation  
205 generation techniques could yield better rewriting results. While the quality of explanations could  
206 likely be improved with careful prompt tuning or even finetuning the explainer model, we think  
207 the results would not substantially change due to the huge gap between the specificity of current  
208 explanations and the specificity that would be required for satisfactory model rewriting.

209 Rewriting is also very computationally expensive. Larger models would probably be able to simulate  
210 the activations more accurately, but this would make the process even more expensive.

## 211 **6 Conclusion**

212 In this work, we proposed a new methodology for rigorously evaluating the faithfulness of natural  
213 language explanations of sparse coder latents in transformers, based on partially rewriting the  
214 transformer using these explanations. We found that existing explanations are severely wanting. We  
215 are still unable to outperform mean ablation, where each latent is substituted by its average value  
216 over the dataset.

217 This is mainly due to the fact that explanations are not specific enough, leading to a high number of  
218 false positives. Our results highlight the fact that it is important for an explanation to correctly identify  
219 the contexts where a feature is not active, in addition to the feature's activation level in contexts where  
220 it is active. Future work on the interpretability of latents should take this into consideration.

221 To improve upon these results, new techniques are needed to make explanations more specific, for  
222 instance using contrast pairs of highly similar features to bring out additional details. This could  
223 potentially increase the sensitivity as well, which takes a big hit when using quantile normalization.

## 224 **Impact Statement**

225 This paper presents work whose goal is to advance the field of Mechanistic Interpretability. There  
226 are many potential societal consequences of our work, none which we feel must be specifically  
227 highlighted here.

## 228 **References**

- 229 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,  
230 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms  
231 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- 232 Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter.  
233 Zoom in: An introduction to circuits. *Distill*, 5(3), March 2020. ISSN 2476-0757. doi: 10.23915/  
234 distill.00024.001. URL <http://dx.doi.org/10.23915/distill.00024.001>.
- 235 Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas.  
236 Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*,  
237 2023.
- 238 Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway,  
239 Neel Nanda, and Dimitris Bertsimas. Universal neurons in gpt2 language models. *arXiv preprint*  
240 *arXiv:2401.12181*, 2024.

241 Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure  
242 of word senses, with applications to polysemy. *Transactions of the Association for Computational*  
243 *Linguistics*, 6:483–495, 2018.

244 Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec,  
245 Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition.  
246 *arXiv preprint arXiv:2209.10652*, 2022.

247 Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoen-  
248 coders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*,  
249 2023.

250 Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya  
251 Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint*  
252 *arXiv:2406.04093*, 2024.

253 Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam  
254 Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner,  
255 Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees,  
256 Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monoseman-  
257 ticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*,  
258 2024. URL [https://transformer-circuits.pub/2024/scaling-monosemanticity/](https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html)  
259 [index.html](https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html).

260 Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature  
261 circuits. *arXiv preprint arXiv:2406.11944*, 2024.

262 Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever,  
263 Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language mod-  
264 els. URL <https://openaiublic.blob.core.windows.net/neuron-explainer/paper/index.html>. (Date  
265 accessed: 14.05. 2023), 2, 2023.

266 Yoav Gur-Arieh, Roy Mayan, Chen Agassy, Atticus Geiger, and Mor Geva. Enhancing automated  
267 interpretability with output-centric feature descriptions, 2025. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2501.08319)  
268 [2501.08319](https://arxiv.org/abs/2501.08319).

269 Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of  
270 features in large language models. *arXiv preprint arXiv:2410.13928*, 2024.

271 Evan Hubinger. Chris olah’s views on agi safety, 2019. URL [https://www.lesswrong.com/](https://www.lesswrong.com/posts/X2i9dQKK3gETCyqh2/chris-olah-s-views-on-agi-safety)  
272 [posts/X2i9dQKK3gETCyqh2/chris-olah-s-views-on-agi-safety](https://www.lesswrong.com/posts/X2i9dQKK3gETCyqh2/chris-olah-s-views-on-agi-safety). Accessed: 2025-01-  
273 15.

274 Beth Barnes. Imitative generalisation (aka ‘learning the prior’),  
275 2021. URL [https://www.lesswrong.com/posts/JKj5Krff5oKMb8TjT/](https://www.lesswrong.com/posts/JKj5Krff5oKMb8TjT/imitative-generalisation-aka-learning-the-prior-1)  
276 [imitative-generalisation-aka-learning-the-prior-1](https://www.lesswrong.com/posts/JKj5Krff5oKMb8TjT/imitative-generalisation-aka-learning-the-prior-1). Accessed: 2025-01-15.

277 Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric  
278 Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al.  
279 Pythia: A suite for analyzing large language models across training and scaling. In *International*  
280 *Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

281 Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo,  
282 Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav,  
283 Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourier, Ben Burtenshaw, Hugo  
284 Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and  
285 Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model,  
286 2025. URL <https://arxiv.org/abs/2502.02737>.

287 Gonçalo Paulo, Stepan Shabalín, and Nora Belrose. Transcoders beat sparse autoencoders for  
288 interpretability. *arXiv preprint arXiv:2501.18823*, 2025.



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

292 Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*,  
293 2014.

294 Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the  
295 finest collection of educational content, 2024. URL [https://huggingface.co/datasets/](https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu)  
296 [HuggingFaceFW/fineweb-edu](https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu).

297 Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd:  
298 Compressed optimisation for non-convex problems. In *International Conference on Machine*  
299 *Learning*, pages 560–569. PMLR, 2018.

300 Aaron Defazio, Xingyu Yang, Ahmed Khaled, Konstantin Mishchenko, Harsh Mehta, and Ashok  
301 Cutkosky. The road less scheduled. *Advances in Neural Information Processing Systems*, 37:  
302 9974–10007, 2024.

303 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha  
304 Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models.  
305 *arXiv preprint arXiv:2407.21783*, 2024.

306 Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkhäuser, NY*, 55(58-63):94,  
307 2015.

308 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.  
309 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model  
310 serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating*  
311 *Systems Principles*, 2023.

You are an intelligent and meticulous linguistics researcher.

You will be given a certain explanation of a feature of text, such as "male pronouns" or "text with negative sentiment" and examples of text that contains this feature. Some explanations will be given a score from 0 to 1. The higher the score the better the explanation is, and you should be more certain of your response (positive or negative).

These features of text are normally identified by looking for specific words or patterns in the text. There are many features associated with a single token, and sometimes the feature is related with the previous token or context.

Your job is to identify how much the last token, which is marked between << and >>, represents the feature. You will output a integer between 0 and 9, where 0 corresponds to no relation to the explanation and 9 to a strong relation.

Most of the tokens should have no relation. The ones that are related, should more likely be given 1 than 2, 2 than 3, and so on. Only give a 9 if the description exactly matches the token.

You must return your response in a valid Python list.  
Do not return anything else besides a Python list.

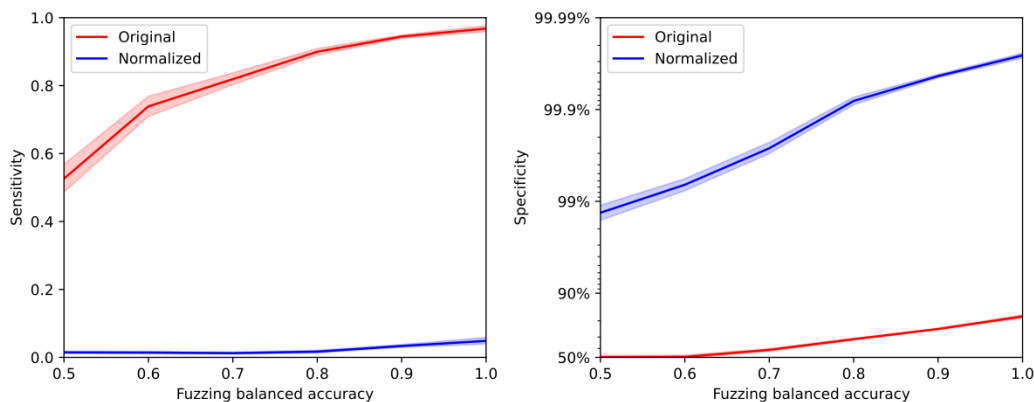


Figure A1: **Fuzzing score predicts sensitivity and specificity** Explanations with higher fuzzing scores lead to better predictions of the simulations

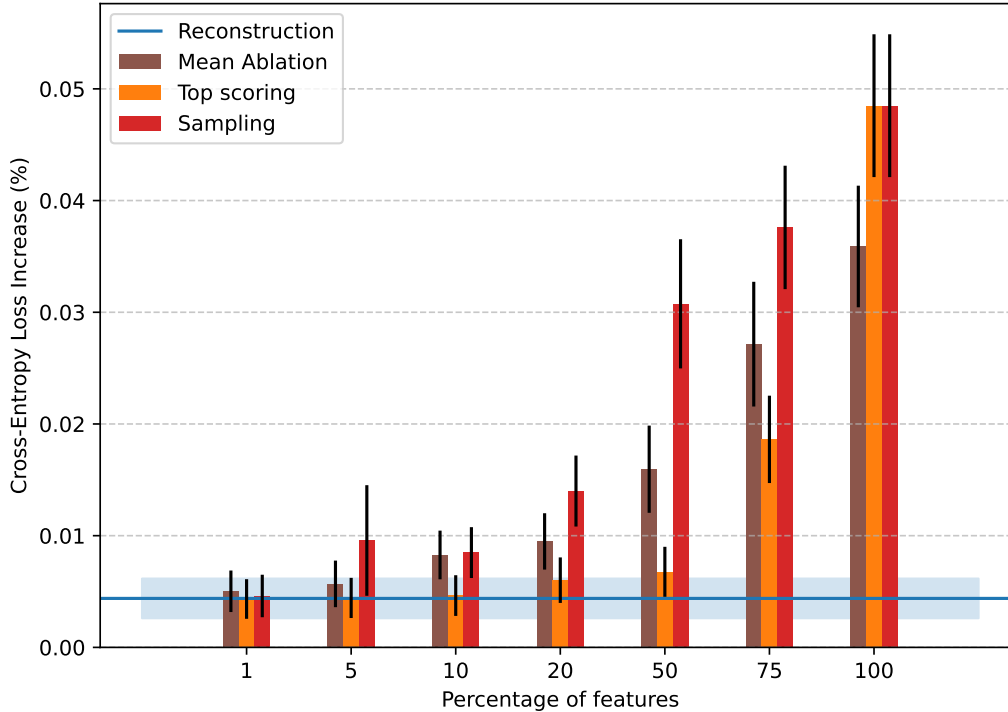


Figure A2: **Cross entropy loss increase for different fractions of transcoder in SmolLM 2.** We compute the CE loss over 2K prompts, for the transcoder, by substituting parts of the encoder with natural language explanations. Bars in orange show the average loss increase when choosing the top scoring latents for replacement. Bars in red show the average loss increase when randomly selecting a subset of latents to replace. Bars in brown show the average loss increase caused by using the mean activations out a part of the transcoder. Bar heights represent the mean value of the increase with respect to the base loss, error bars represent the standard error. The interpretability score used for the selecting latents is detection scoring, [Paulo et al., 2024, page 5], computed over 100 positive and 100 negative samples. Prompts where the loss was lower than 0.1 nats were excluded.

## 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]

### 2. Limitations

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

Answer: [Yes]

### 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: [NA]

### 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]

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

331 Question: Does the paper provide open access to the data and code, with sufficient instruc-

332 tions to faithfully reproduce the main experimental results, as described in supplemental

333 material?

334 Answer: [Yes]

335 **6. Experimental setting/details**

336 Question: Does the paper specify all the training and test details (e.g., data splits, hyper-

337 parameters, how they were chosen, type of optimizer, etc.) necessary to understand the

338 results?

339 Answer: [Yes]

340 **7. Experiment statistical significance**

341 Question: Does the paper report error bars suitably and correctly defined or other appropriate

342 information about the statistical significance of the experiments?

343 Answer: [Yes]

344 **8. Experiments compute resources**

345 Question: For each experiment, does the paper provide sufficient information on the com-

346 puter resources (type of compute workers, memory, time of execution) needed to reproduce

347 the experiments?

348 Answer: [Yes]

349 **9. Code of ethics**

350 Question: Does the research conducted in the paper conform, in every respect, with the

351 NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

352 Answer: [Yes]

353 **10. Broader impacts**

354 Question: Does the paper discuss both potential positive societal impacts and negative

355 societal impacts of the work performed?

356 Answer: [Yes]

357 **11. Safeguards**

358 Question: Does the paper describe safeguards that have been put in place for responsible

359 release of data or models that have a high risk for misuse (e.g., pretrained language models,

360 image generators, or scraped datasets)?

361 Answer: [NA]

362 **12. Licenses for existing assets**

363 Question: Are the creators or original owners of assets (e.g., code, data, models), used in

364 the paper, properly credited and are the license and terms of use explicitly mentioned and

365 properly respected?

366 Answer: [Yes]

367 **13. New assets**

368 Question: Are new assets introduced in the paper well documented and is the documentation

369 provided alongside the assets?

370 Answer: [NA]

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

372 Question: For crowdsourcing experiments and research with human subjects, does the paper

373 include the full text of instructions given to participants and screenshots, if applicable, as

374 well as details about compensation (if any)?

375 Answer: [NA]

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

377 **subjects**

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

382 Answer: [NA]

383 **16. Declaration of LLM usage**

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

388 Answer: [Yes]