Steering LLMs? Actually, Sparse Autoencoders *can* outperform simple baselines

Anonymous authors
Paper under double-blind review

Abstract

Sparse Autoencoders (SAEs) have been seen as a promising avenue for exploring the internals of Large Language Models (LLMs) and for steering model output generation. When AxBench - a model steering benchmark - was introduced in Wu et al. (2025), SAEs did not seem to live up to their original hype due to poor steering performance relative to a set of simple baselines. This work serves as a partial rebuttal for Sparse Autoencoders and suggests that the results of Wu et al. (2025) did not do them full justice. We find that Sparse Autoencoders can, in fact, perform close to on par with the reference LoRA performance on the AxBench benchmark, when features are selected and labelled with our supervised pipeline. We also find that our pipeline selects features that are surprisingly causal of their identified labels when using only its interpretability-based components. Lastly, we present evidence that high sparsity (low ℓ_0) may not be crucial for successful steering based on interpretability, which is in contrast to the earlier findings in Wang et al. (2025). Our code is available at `https://anonymous.4open.science/r/Sparse-Autoencoders-392b`.

1 Introduction

Sparse Autoencoders have recently attracted a wave of attention (e.g. Cunningham et al., 2023; Makelov et al., 2024; Templeton et al., 2024; Gao et al., 2024; Rajamanoharan et al., 2024), offering an unsupervised interpretability model based on the *Linear Representation Hypothesis* (LRH) and the *superposition hypothesis*. The LRH suggests that many concepts may be stored in linear subspaces, as discussed in e.g. Alain & Bengio (2018); Kim et al. (2018); Elhage et al. (2022); Park et al. (2023), although somewhat contested in e.g. Crabbé & van der Schaar (2022). Since LLMs can represent a large number of concepts in relatively low-dimensional latent spaces, the neurons must be *polysemantic* as a consequence, i.e. they become active across multiple different semantics, as discussed in e.g. Bolukbasi et al. (2021); Makelov et al. (2024); Templeton et al. (2024); Fereidouni et al. (2025); Cunningham et al. (2023). The superposition hypothesis further suggests that linear concept representations are densely packed and not fully orthogonal (Elhage et al., 2022). Within this framework, Sparse Autoencoders offer an appealing interpretability tool, capable of discovering a subset of linear concept representations, referred to as *features*, which are thought to exhibit a high degree of *monosemanticity* (e.g. Bricken et al., 2023; Templeton et al., 2024). In Fereidouni et al. (2025), the authors quantify the monosemantic property of SAEs in comparison to individual neurons in a network. Their findings suggest that, indeed, SAE features are more monosemantic than the individual neurons of language models.

While the monosemantic response is interesting for interpretability, features have also shown some ability to control the model's text generation (e.g. Templeton et al., 2024; Arad et al., 2025; He et al., 2025; Wang et al., 2025), although the effectiveness is disputed most notably in Wu et al. (2025). In fact, Wu et al. (2025) concludes that "Even simple baselines outperform Sparse Autoencoders" for steering, and finds that simply prompting the model is much more effective than other approaches, such as intervention via SAEs (e.g. Templeton et al., 2024) and LoRA (Hu et al., 2022). Although this is undeniably important for most practical applications, the prompt baseline can seem unfair since the subject LLM was originally trained precisely to follow instructions. Aside from steering, there have also been several attempts at using SAEs for

knowledge unlearning/concept removal in language models (e.g. Farrell et al., 2024; Yamashita et al., 2025; Fereidouni et al., 2025), essentially exploring the negative (removal) dimension of steering.

Since Sparse Autoencoders are trained unsupervised, efforts have been made to label their features with the concepts they represent. This has been attempted at scale using LLMs (e.g. Lieberum et al., 2024; Paulo et al., 2024), but to our knowledge, no dedicated efforts have been made to use a supervised approach to labelling.

Recent work (Wang et al., 2025) has suggested that high interpretability of a feature tends to have some correlation with high utility for model steering, and that this relationship seems most pronounced for JumpReLU SAEs with a low ℓ_0 (i.e. high sparsity). This result could pose as a relevant context for interpreting the findings of Wu et al. (2025), in which they base their conclusions on SAEs with high ℓ_0, being those that are already annotated in Neuronpedia[1].

1.1 Contributions

We make the following contributions:

1. We introduce a labelling pipeline for Sparse Autoencoders based on labelled datasets rather than LLMs.
2. We demonstrate that Sparse Autoencoders are substantially more powerful for steering than reported in Wu et al. (2025) when the features are labelled via our proposed method. However, the steering performance we demonstrate is still not fully competitive with prompting on the AxBench benchmark.
3. We find indications that both low- and high-sparsity Sparse Autoencoders in Gemma Scope can be used for steering.

2 Preliminaries

2.1 Large Language Models

Formal definition. Large Language Models define a distribution over text strings, and are typically based on transformers (Vaswani et al., 2023). To transform a string of text into a format suitable for language modeling, the typical approach is to use a tokenizer to split up the text into a string of tokens. Here, the tokens constitute an *alphabet* Σ. For modeling purposes, we further augment the alphabet by the tokens BOS and EOS (beginning of sequence, end of sequence) to get an augmented alphabet $\bar{\Sigma} = \Sigma \cup \{\text{BOS}, \text{EOS}\}$. Formally, a sequence of tokens is simply called a *string*. The language model can then be characterized as a string distribution

$$p_{\text{LM}}(\boldsymbol{s}) = \prod_i p_{\text{SM}}(s_i | \boldsymbol{s}_{<i}), \quad \boldsymbol{s} \in (\text{BOS} \cdot \Sigma^* \cdot \text{EOS}), \tag{1}$$

where $\cdot$ denotes string concatenation and $p_{\text{SM}}(\cdot|\boldsymbol{s}_{<i}) : \bar{\Sigma} \to [0, 1]$ is typically fitted by the model. Here p_{LM} is assumed to be a true distribution (i.e., no probability mass is leaked into the set of infinite strings). This is true for models with a softmax-based prediction head. To use the language model generatively, we use a decoding strategy to auto-regressively predict the next token until termination by EOS. The two simplest options are to greedily select $\arg\max_{s_i} p_{\text{SM}}(s_i|\boldsymbol{s}_{<i})$ as our next token, or to sample from $p_{\text{SM}}(s_i|\boldsymbol{s}_{<i})$ at each step (Cotterell et al., 2024).

Transformers. LLMs typically build on a modified transformer architecture compared to the one introduced in Vaswani et al. (2023). The main modifications are to use only the decoder, to apply the positional encodings within the Multi-Head Self-Attention, and to use the causal mask. The causal mask has the effect of making information always flow forward over a sequence of tokens. E.g., for two sentences "The bar closes at 4" and "The bar is made of metal", the internal representation for the word "bar" cannot be disambiguated

[1] https://www.neuronpedia.org/gemma-scope#browse

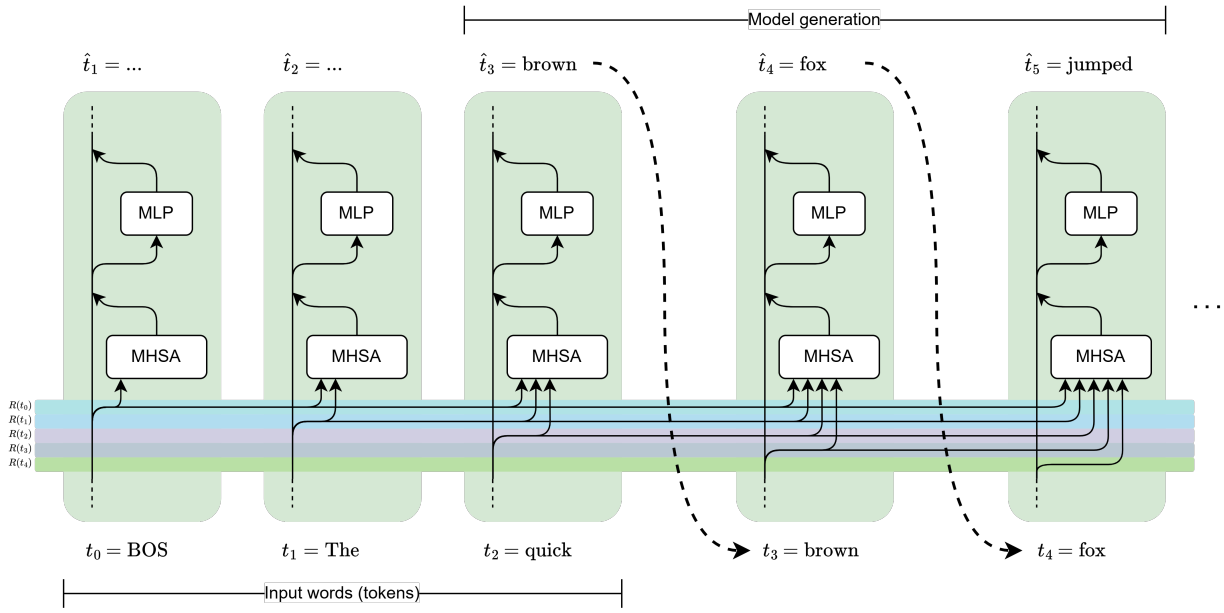


Figure 1: The architecture of a transformer-based causal LM at the level of a single transformer block. Here we see the residual stream as the vertical line running through the block, and the consequences of the causal mask on the attention. For simplicity, we assume the example words are mapped to single tokens and leave out the embedding and decoding step, as well as the positional encoding. The model generation bracket shows the autoregressive nature of the model.

at its token position using the succeeding tokens as they are masked out. This has the advantage of being computationally cheaper since the current representations need not be recomputed every time a new token is generated. This is depicted in figure 1. For the reader less familiar with LLMs, we will occasionally refer to the chain of residual connections that flows throughout the network as the "residual stream". This view has been widely adopted in the literature, aptly so since there is a direct path from the input layer to the output layer through it, where the transformer blocks can be seen as reading and writing its value (Elhage et al., 2021; Lieberum et al., 2024).

Prompt Steering. Prompting offers a straightforward way to introduce a concept into a model generation. For instance, if we want the language model to generate a poem about baseball, a natural approach is to use the prompt: "Write a poem about baseball". In this work, we follow the procedure of Wu et al. (2025). Here we ask GPT-4o-mini to write a prompt explaining our subject LLM to include a topic (e.g., "baseball") into its generation, to which we append the instruction (e.g., "Write a poem"). One such prompt could have the format:

> You are a language model that must always incorporate the concept of baseball into your responses, [...]
>
> Write a poem.

2.2 Sparse Autoencoders

Sparse Autoencoders are attributed to Ng et al. (2011). A popular interpretation of SAEs is that they learn an over-complete dictionary of features. An alternative view is as a set of linear probes without known labels, which is the view we adopt.

Definition. SAEs are shallow, wide autoencoders, typically trained to reconstruct activations while promoting sparsity in the latent layer. Formally, the model can be described as

$$\textbf{(Encoder):} \quad \boldsymbol{z}(\boldsymbol{x}) = \sigma(\boldsymbol{x}\boldsymbol{W}_{\text{enc}} + \boldsymbol{b}_{\text{enc}}) \tag{2}$$

$$\textbf{(Decoder):} \quad \hat{\boldsymbol{x}}(\boldsymbol{z}) = \boldsymbol{z}\boldsymbol{W}_{\text{dec}} + \boldsymbol{b}_{\text{dec}}, \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^{d_{\text{model}}}, \hat{\boldsymbol{x}} \in \mathbb{R}^{d_{\text{model}}}$, and $\boldsymbol{z} \in \mathbb{R}^{d_{\text{SAE}}}$ respectively denote the network activation, the activation reconstruction and the *feature activation vector*. In SAEs, we have $d_{\text{model}} \ll d_{\text{SAE}}$. We can consider the encoder as discriminative/detective, and the decoder part as generative. As demonstrated in Haufe et al. (2014), the discriminative component may not exclusively reflect our signal of interest, hence we do not expect the columns of W_{enc} to be clean feature representations. Instead, due to the generative nature of the decoder, it is more natural to use the rows of W_{dec} as the learned feature representations in concordance with e.g. Bricken et al. (2023); Templeton et al. (2024). It is important to note that these features are not guaranteed to be an exhaustive list (Paulo & Belrose, 2026). In the case of Gemma Scope (Lieberum et al., 2024), the activation $\sigma(\cdot)$ is the $\text{JumpReLU}_{\boldsymbol{\theta}}$ function (originally introduced in Erichson et al., 2019):

$$\sigma(\boldsymbol{v}) = \text{JumpReLU}_{\boldsymbol{\theta}}(\boldsymbol{v}) = \boldsymbol{v} \odot \mathbf{1}\left[\boldsymbol{v} \geq \boldsymbol{\theta}\right] \tag{4}$$

where $\odot$ is the element-wise (Hadamard) product. The encoder has learnable weights $\boldsymbol{W}_{\text{enc}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{SAE}}}$, learnable bias $\boldsymbol{b}_{\text{enc}} \in \mathbb{R}^{d_{\text{SAE}}}$, and a learnable threshold $\boldsymbol{\theta} \in \mathbb{R}^{d_{\text{SAE}}}$. The decoder has the learnable weights $\boldsymbol{W}_{\text{dec}} \in \mathbb{R}^{d_{\text{SAE}} \times d_{\text{model}}}$, and the learnable bias $\boldsymbol{b}_{\text{dec}} \in \mathbb{R}^{d_{\text{model}}}$. Together, the model is trained to minimize $\mathcal{L} = ||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2 + \lambda||\boldsymbol{z}(\boldsymbol{x})||_0$, where the second term incentivizes sparsity on feature activations (Lieberum et al., 2024; Bricken et al., 2023; Gao et al., 2024; Rajamanoharan et al., 2024). For a technical explanation on how this is performed in practice, refer specifically to Rajamanoharan et al. (2024).

Feature Steering. In this work, we use the approach of Templeton et al. (2024) for activation editing. During inference, we clamp the activation of a feature $(\boldsymbol{z}(\boldsymbol{x}))_f$ to a value α. The specific approach first computes the reconstruction error term

$$\boldsymbol{e} = \boldsymbol{x} - \hat{\boldsymbol{x}}(\boldsymbol{z}(\boldsymbol{x})), \tag{5}$$

followed by the latent vector after intervention via editing;

$$\boldsymbol{z}_{\text{int}} = \boldsymbol{z}(\boldsymbol{x}) \odot (1 - \boldsymbol{m}) + \alpha \boldsymbol{m} \tag{6}$$

where $\boldsymbol{m}$ is a mask consisting of elements $m_j = \mathbf{1}\left[j = f\right]$. The edited activations can finally be computed as

$$\hat{\boldsymbol{x}}_{\text{int}} = \hat{x}(\boldsymbol{z}_{\text{int}}) + \boldsymbol{e}, \tag{7}$$

which we pass through the rest of the model. This is done on all tokens, including the BOS token. We shall refer to this procedure as *feature steering*.

3 Method

3.1 Labelling pipeline

To label the learned features of an SAE, we propose to construct linear probes based on single features and match their predictions with labels on a multi-label dataset. Here we use the LLM activations $\mathbf{X} \in \mathbb{R}^{|\mathcal{D}| \times T \times d_{\text{model}}}$ computed over a dataset $\mathcal{D}$ with maximum sequence length T, as well as a binary mask $\boldsymbol{M} \in \{0,1\}^{|\mathcal{D}| \times T}$ zeroing out token positions not included in a given sample. The dataset comes with a binary label matrix $\boldsymbol{Y} \in \{0,1\}^{|\mathcal{D}| \times n_{\text{labels}}}$.

Feature probes. We construct the probe predictions by first computing the feature activations $\mathbf{Z}_{i,j,:} = \boldsymbol{z}\left(\mathbf{X}_{i,j,:}\right)$ resulting in a sparse tensor $\mathbf{Z} \in \mathbb{R}^{|\mathcal{D}| \times T \times d_{\text{SAE}}}$. Given the activations of feature f, we then compute its activation frequency on the i'th text example:

$$\boldsymbol{F}_{i,f} = \left(\sum_{j=1}^{T} \boldsymbol{M}_{i,j}\right)^{-1} \sum_{j=1}^{T} \boldsymbol{M}_{i,j} \cdot 1\left[\boldsymbol{Z}_{i,j,f} > 0\right] \quad \in [0,1], \tag{8}$$

using the indicator function $1[\cdot]$. We define $\hat{y}_{i,f,\tau} = 1\left[F_{i,f} > \tau\right]$ to form the probe prediction given a threshold τ. Since it is not obvious what would constitute a good frequency threshold, we let τ be a free parameter.

Probe labelling. To match a dataset label to a probe, we compute a match score using the calibrated F1 (Siblini et al., 2020) followed by additional denoising. For every feature-label pair (ℓ, f), we define the calibrated F1 score as:

$$\mathrm{TP}_{\ell,f,\tau} = \sum_{j=1}^{|\mathcal{D}|} \mathbf{1}\left[y_{j,\ell} = 1 \wedge \hat{y}_{j,f,\tau} = 1\right], \qquad \mathrm{FP}_{\ell,f,\tau} = \sum_{j=1}^{|\mathcal{D}|} \mathbf{1}\left[y_{j,\ell} = 0 \wedge \hat{y}_{j,f,\tau} = 1\right], \tag{9}$$

$$\mathrm{FN}_{\ell,f,\tau} = \sum_{j=1}^{|\mathcal{D}|} \mathbf{1}\left[y_{j,\ell} = 1 \wedge \hat{y}_{j,f,\tau} = 0\right], \qquad \pi_\ell = \frac{1}{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{D}|} 1\left[y_{j,\ell} = 1\right], \tag{10}$$

$$\mathrm{F1}^c_{\ell,f,\pi_0} := \max_\tau \left[\frac{2\mathrm{TP}_{\ell,f,\tau}}{2\mathrm{TP}_{\ell,f,\tau} + \frac{(1-\pi_0)\pi_\ell}{\pi_0(1-\pi_\ell)}\mathrm{FP}_{\ell,f,\tau} + \mathrm{FN}_{\ell,f,\tau}}\right]. \tag{11}$$

We further explore this definition in appendix A and examine the effects of the parameter π_0. In practice, the max operation selects the optimal threshold τ over a set of candidates $\{0.05, 0.1, ..., 0.95\}$.

To motivate the use of the F1 score, consider its decomposition into the harmonic mean between precision and recall, i.e. $\mathrm{F1} = 2\left(\mathrm{precision}^{-1} + \mathrm{recall}^{-1}\right)^{-1}$. The objective is for a probe to have both high precision (almost exclusive activation) and high recall (activation on most occurrences) for the correct label. A probe scoring highly in one but low in the other would likely correspond to a super- or a sub-category, or something semantically adjacent. We here took inspiration from insights of Gao et al. (2024); Gurnee et al. (2023). To motivate the calibration, we consider how the ordinary F1 score is sensitive to the label *support* (the number of occurrences for the label). Since labels may have different supports, the scale of the F1 score will vary between labels, rendering it unreliable for comparing the degree of agreement between a feature and a label. Again, we refer the reader to appendix A for a detailed exploration.

To select a set of valid matches based on $\mathrm{F1}^c_{\ell,f,\pi_0}$, we devise a set of criteria to *denoise* the result. These are performed in the following order:

1. To reduce variance, we threshold the label support, i.e., the number of label occurrences in the dataset. We found 50 to be a good threshold through parameter tuning, which we elaborate in section 4.
2. We use the output score from Arad et al. (2025) as a proxy for the steering ability of a feature. We use a threshold of 10^{-3} found by visually inspecting the output score distribution and choosing it to only include a small best subset.
3. We keep the top-K of the remaining feature-label pairs. We use $K = 50$, since it provides a tradeoff between accuracy and number of matches.

3.2 AxBench

AxBench is a benchmark for model generation under steering, prompting the model with Alpaca-Eval instructions (Wu et al., 2025; Li et al., 2023). The generated texts are rated by GPT-4o with scores $\{0, 1, 2\}$ on 1) fluency (whether they are correct English); 2) instruction following (whether they respond to the given instruction); and 3) concept incorporation (whether they naturally include the target concept). The scores are finally combined into a single overall score for each generation by the harmonic mean

$$\text{aggregated_rating} = \frac{3}{\text{instruction_rating}^{-1} + \text{concept_rating}^{-1} + \text{fluency_rating}^{-1}}.$$

The ordinary mean is used to aggregate each score over the dataset.

Feature Steering Evaluation. To generate a set of steered texts, we first draw 20 random instructions from Alpaca-Eval without replacement. We repeat this for each suggested feature-label pair. For a set of

considered steering strengths $\alpha \in \{100, 125, ..., 575\}$, we generate feature-steered responses to the selected instructions using each strength value.

In Wu et al. (2025), they select α by maximizing aggregated_rating in holdout (5 training examples, 5 test examples). The resulting value is used across all features. However, since we do not expect all features to require the same α for optimal steering, we select it individually for each feature-label pair. Here, we generate 20 responses for every considered α, for every suggested feature-label pair. We then report the result from repeated cross-validation (10 repeats, 4 splits) to reduce variance[2]. To evaluate the overall steering, we use aggregated_rating as the maximization objective, as in Wu et al. (2025). However, since the aggregated rating accounts for model-specific abilities, we use the concept_rating to evaluate the accuracy of our feature-labels.

To offer an additional motivation for a steering-based pipeline evaluation, this approach avoids the potential issue of dataset idiosyncrasy that may cause an interpretability illusion, as observed in Bolukbasi et al. (2021). Here, the observed semantics of a neuron vary across the datasets used to identify them, possibly explained by neuron polysemanticity. In contrast, feature steering is a causal experiment, and while it additionally demands causation, a successful feature steering would confirm what we may consider the "ground truth" labels of the learned features.

Prompt Steering Evaluation. For each feature-label pair, we use the same 20 random instructions. We then generate the prompt-steered responses to each instruction. Since there are no parameters to select for prompting, we report the average.

3.3 Resources

Datasets. As our multi-label dataset, we use Stack Exchange obtained by the tool of Eleuther-AI to process the data dump on the Internet Archive (EleutherAI, 2020; Stack Exchange, Inc., 2024). As labels, we use the tags assigned to a post by the user – for instance, a user may assign "baseball" and "mlb" (Major League Baseball) to a post about baseball. As our text inputs, we clean the post and append the comments made by other users. Since the data format is a set of large XML files, we create an index of all newline characters in the bytestream to achieve fast random access. Since there are multiple Stack Exchange fora available, we obtain a suite of datasets related to different topics. We chose the following: academia, biology, chemistry, cooking, cs, history, law, literature, physics, politics, and sports. For each forum, we plot the cumulative density functions of label supports after pre-processing in figure 5.

Models. We run the experiments on Gemma 2 (Team et al., 2024) + Gemma Scope (Lieberum et al., 2024). As a baseline for feature labels, we use the ones provided in Neuronpedia.

Hardware. The entire experiment can be run on a single H100 node within 24 hours for most Stack Exchange fora. To properly control the GPU memory consumption, we have used a set of code optimizations, including pre-allocation and in-place computation of most matrices, and an OOM recovery mechanism that successively halves the batch size if needed. The codebase was originally developed on an RTX4090 machine, and the full pipeline can be run on this GPU for some Stack Exchange fora.

4 Experimental design and results

Setup. Using the selected Stack Exchange fora, we apply the pipeline to two different layers in the model for completeness. Here, we use layer 17 and layer 32.[3] Our method was originally developed and tuned using layer 32, which we chose in the early experimental phase for Gemma-2-9b to reside in the second half of the model, anticipating reasonable steering performance. We arbitrarily selected layer 17 to examine whether our method would also generalize to earlier layers. Layer 17 was excluded from method development and parameter tuning to ensure a reliable result.

Parameter tuning. Since our pipeline contains a set of parameters (π_0, K, support threshold, and output score threshold), we use a hold-out approach to tuning. Here, we exclusively tune using layer 32 and the

[2]For each split, we select α by maximizing over the training-partition and report the score on the evaluation-partition

[3]We enumerate the layers using the original indices from Lieberum et al. (2024) starting from zero.

fora: academia, biology, history, law, literature, physics, and sports. Hence, the remaining fora and layer 17 are not included. As elaborated in section 3.1, K and the output score threshold were chosen heuristically in the early experimentation in this setup. We find that pipeline performance seems particularly sensitive to the choices of π_0 and the support threshold. Since it is infeasible to evaluate it on the full Cartesian product of features $\times$ labels, we use a manual iterative approach. Here, we select sensible initial values and evaluate the assigned labels. We then inspect the results, tune the parameters optimizing for concept_rating, and repeat. After a few iterations, we converge at $\pi_0 = 10^{-3}$ and a support threshold of 50, still being sensible values.

Baseline. As baselines, we use those provided in Wu et al. (2025), including a reproduction of the SAE and prompt baselines named NP random. NP random samples 100 features at random and evaluates their official Neuronpedia labels for steering and prompting.

Interpretation. Comparisons of steering performances between feature-steering and prompt-steering are shown in figures 2 and 6 (see appendix B). In figure 6, we see that our proposed pipeline is indeed capable of assigning correct labels to the causal effects of features. While we observe some variability between different fora, we mostly observe similar effects when comparing feature-steering to prompt-steering.

Figure 2 depicts the overall steering performance and shows a general improvement over those reported in Wu et al. (2025). We observe that multiple fora provide labels that result in feature-steering performance on par with the LoRA reference. We also observe a clear improvement in the prompt-steering performance, possibly due to the subject model's better understanding of simpler labels. The reference Neuronpedia performance is largely consistent with our results.

When considering the performance increase of both feature-steering and prompt-steering from the Neuronpedia reference, one could be tempted to alternatively suggest better label comprehension by the LLM judge as an explanation. However, when considering the concept_rating performance in figure 6, we now observe a mostly closed gap between feature-steering and prompt-steering compared to the Neuronpedia baseline. Since concept_rating is the only component in AxBench that relies on the feature-label, the alternative explanation does not fully account for the results.

Ablation. We perform an ablation study to examine the impacts of the roles of 1) a high sparsity, and 2) the output score. The results are shown in figures 7 and 8. Here we see that using high sparsity SAEs (i.e., low ℓ_0) seemingly does not greatly impact the steering performance. We also observe a rather modest performance improvement caused by the output score, in both the low and high sparsity SAEs.

To examine the role of the output score, we compare randomly selected features with features selected by our pipeline. In both feature sets, we fetch their original Neuronpedia labels. Here, we observe better feature-steering performance of those selected by the pipeline, even without using the output score.

5 Discussion

We find the results of section 4 somewhat surprising for the two reasons: 1) causality generally cannot be taken for granted given a correlation/dependence, yet removing the causal-component of the pipeline (the output-score) does not greatly affect its performance; and 2) the reported feature-steering performance in Wu et al. (2025) was much weaker compared to our setup.

Linear Representation Hypothesis. While Wu et al. (2025) reported obtaining rather poor steering performance on all methods that linearly steer, we found that SAEs can, surprisingly, steer with a performance close to par with LoRA that uses adapter weights (Hu et al., 2022). This finding demonstrates the power of using linear methods for understanding the internals of language models.

AxBench. Since AxBench uses an LLM-judge to rate steered generations, we find it natural to ask whether the LLM-judge could exhibit a bias to favor some steering methods differently than a population of human judges. To investigate this, we conducted a small trial with Danish bachelor students and found indications that they would generally rate some scores more optimistically than GPT-4o-mini, however, we did not find clear evidence of a bias in the trial. We note that the trial had a small sample size, so a more thorough user study might still be relevant. However, while inspecting our pipeline evaluation results, we observed multiple

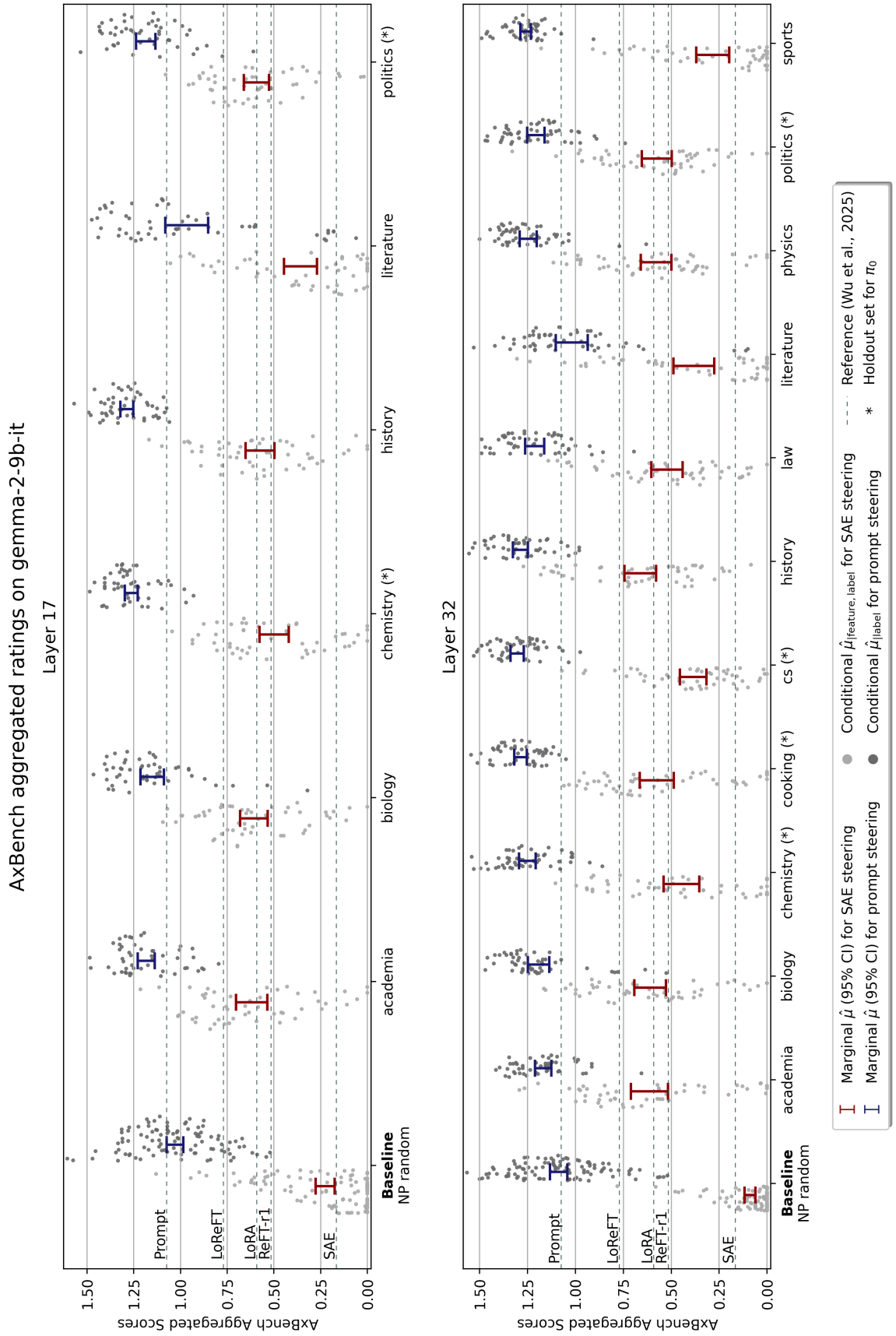


Figure 2: AxBench aggregated ratings for two layers of Gemma-2-9b-it across multiple stackexchanges. Here the Gemma-Scope SAEs have width 131k, are inserted into the residual stream, and have sparsity parameters $\ell_0 \approx 11$ and $\ell_0 \approx 10$ respectively (highest sparsity). The gray datapoints illustrate the average performances of individual selected label, where the jitter on the x-axis is illustrative of the density estimated via KDE. The confidence intervals were computed using non-parametric bootstrapping. The figure also shows multiple baselines: NP random being 100 randomly selected feature-label pairs from Neuronpedia, and multiple performances from Wu et al. (2025). The Neuronpedia baselines are based on the higher ℓ_0 counterparts of the selected SAEs.

examples of a label *semantically adjacent* to, or even fully matching, the elicited topic, where AxBench still assigned it a concept score of 0. We included some of these examples in appendix D. Since AxBench relies on simply prompting the LLM judge with a detailed instruction without examples, it seems natural to ask whether a few-shot prompting approach could benefit its accuracy and alignment.

Applications. Although our results suggest an improved feature-steering performance compared to Wu et al. (2025), it is important to acknowledge that prompting is a baseline that is likely hard to match, as the model was trained to respond to prompts. In our view, feature-steering is an exercise in *controlled* damage of the subject model, but it is a damage nonetheless, rendering it less useful in production. However, in a research setting, the causal nature of features can be reassuring that an assigned label is indeed the ground truth, and not simply an artifact of an idiosyncratic dataset (Bolukbasi et al., 2021). Hence, Sparse Autoencoders could still be of interest for interpretability and explainability researchers. Our proposed supervised pipeline also has the advantage of being applicable outside the text-domain, potentially enabling researchers to explore the internals of models in computer vision, EEG-data, etc.

Influence of support. Since we found it beneficial to use a relatively high support threshold, it would be reasonable to reflect on its effects on the setup. While its primary motivation was to reduce variance in the calibrated F1 score (a direct effect), it likely also indicates the general popularity of a topic outside the respective Stack Exchange forum. Since it is not public knowledge whether Gemma-Scope was trained on Stack Exchange, we can only speculate that topics of higher general popularity may also more frequently occur in its training set. As the reconstruction loss will likely favor more frequent topics to be learned as features (Muhamed et al., 2025), the high support threshold could filter out unlikely candidate labels as an indirect effect.

6 Conclusion

Sparse Autoencoders initially seemed promising for exploring the internals of LLMs, and gained traction following publications such as Bricken et al. (2023); Templeton et al. (2024); Lieberum et al. (2024) and more. However, the optimism around the model has since faded, and its reliability has been cast into question. In this work, we have demonstrated that Sparse Autoencoders can, to an extent, live up to the original hype, even if they are not the panacea of interpretability and steering once thought to be. We find that interpretable features of SAEs exhibit a surprisingly causal impact on the model predictions when the labels are selected by a different approach than the one used to construct the Neuronpedia labels. Our proposed labelling method allows Sparse Autoencoders to perform on par with the LoRA baseline provided in Wu et al. (2025), vastly outperforming their reported capabilities. We further see indications that both low- and high-sparsity Sparse Autoencoders can be leveraged for steering tasks.

Broader Impact Statement

This work explores Sparse Autoencoders for model steering and interpretability. We believe the direct ethical and societal implications of our work are minimal, as it primarily contributes to an understanding of language models. In a broader perspective, we believe this could be useful for performing safety auditing, model debugging, and alignment research. We acknowledge the potential risk associated with enabling direct manipulation of the open source models that could enable the expression of possessed dangerous capabilities. However, we consider the additional risk posed by our framework insignificant compared to standard finetuning approaches.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018. URL https://arxiv.org/abs/1610.01644.

Dana Arad, Aaron Mueller, and Yonatan Belinkov. Saes are good for steering – if you select the right features, 2025. URL https://arxiv.org/abs/2505.20063.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

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.

Ryan Cotterell, Anej Svete, Clara Meister, Tianyu Liu, and Li Du. Formal aspects of language modeling, 2024. URL `https://arxiv.org/abs/2311.04329`.

Jonathan Crabbé and Mihaela van der Schaar. Concept activation regions: A generalized framework for concept-based explanations. *Advances in Neural Information Processing Systems*, 35:2590–2607, 2022.

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

EleutherAI. stackexchange_dataset. `https://github.com/EleutherAI/stackexchange-dataset`, 2020. Accessed: 2025-11-04.

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

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. *Transformer Circuits Thread*, 2022. URL `https://transformer-circuits.pub/2022/toy_model/index.html`.

N Benjamin Erichson, Zhewei Yao, and Michael W Mahoney. Jumprelu: A retrofit defense strategy for adversarial attacks. *arXiv preprint arXiv:1904.03750*, 2019.

Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. Applying sparse autoencoders to unlearn knowledge in language models, 2024. URL `https://arxiv.org/abs/2410.19278`.

Moghis Fereidouni, Muhammad Umair Haider, Peizhong Ju, and A. B. Siddique. Evaluating sparse autoencoders for monosemantic representation, 2025. URL `https://arxiv.org/abs/2508.15094`.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, and Angela Fan et al. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

Stefan Haufe, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, and Felix Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage*, 87:96–110, 2014. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2013.10.067. URL `https://www.sciencedirect.com/science/article/pii/S1053811913010914`.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders, 2024. URL `https://arxiv.org/abs/2410.20526`.

Zirui He, Haiyan Zhao, Yiran Qiao, Fan Yang, Ali Payani, Jing Ma, and Mengnan Du. Saif: A sparse autoencoder framework for interpreting and steering instruction following of language models. *arXiv preprint arXiv:2502.11356*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav), 2018. URL `https://arxiv.org/abs/1711.11279`.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 5 2023.

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. *arXiv preprint arXiv:2408.05147*, 2024.

Aleksandar Makelov, George Lange, and Neel Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control. *arXiv preprint arXiv:2405.08366*, 2024.

Aashiq Muhamed, Mona Diab, and Virginia Smith. Decoding dark matter: Specialized sparse autoencoders for interpreting rare concepts in foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1604–1635, 2025.

Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

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

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

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Wissam Siblini, Jordan Fréry, Liyun He-Guelton, Frédéric Oblé, and Yi-Qing Wang. Master your metrics with calibration. In Michael R. Berthold, Ad Feelders, and Georg Krempl (eds.), *Advances in Intelligent Data Analysis XVIII*, pp. 457–469, Cham, 2020. Springer International Publishing. ISBN 978-3-030-44584-3.

Stack Exchange, Inc. Stack exchange data dump. `https://archive.org/details/stackexchange`, April 2024.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Xu Wang, Yan Hu, Benyou Wang, and Difan Zou. Does higher interpretability imply better utility? a pairwise analysis on sparse autoencoders, 2025. URL https://arxiv.org/abs/2510.03659.

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. *arXiv preprint arXiv:2501.17148*, 2025.

Tomoya Yamashita, Akira Ito, Yuuki Yamanaka, Masanori Yamada, Takayuki Miura, and Toshiki Shibahara. Sparse-autoencoder-guided internal representation unlearning for large language models, 2025. URL https://arxiv.org/abs/2509.15631.

A Appendix: Calibrated F1

A.1 Unpacking the calibrated F1 score

To simplify the notation, we define the events $+$ and $-$ as the positive and negative classes of the label, and let $\hat{+}$ and $\hat{-}$ define the events of positive and negative predictions. We also abuse the notation and consider π_0 a Bernoulli distribution over $\{+, -\}$.

A quick glance at the F1 score reveals its dependency on the class prior:

$$\begin{aligned} \mathrm{F1} &= \frac{2\mathrm{TPR}}{2\mathrm{TPR} + \mathrm{FPR} + \mathrm{FNR}} \\ &= \frac{2P(\hat{+}, +)}{2P(\hat{+}, +) + P(\hat{+}, -) + P(\hat{-}, +)} \\ &= \frac{2P(\hat{+}|+)}{2P(\hat{+}|+) + \frac{P(-)}{P(+)}P(\hat{+}|-) + P(\hat{-}|+)}. \end{aligned} \tag{12}$$

We can fix this by using the calibrated F1 of equation 9:

$$\begin{aligned} \mathrm{F1}^c_{\pi_0} &= \frac{2\mathrm{TPR}}{2\mathrm{TPR} + \frac{\pi_0(-)P(+)}{\pi_0(+)P(-)}\mathrm{FPR} + \mathrm{FNR}} \\ &= \frac{2P(\hat{+}|+)}{2P(\hat{+}|+) + \frac{\pi_0(-)}{\pi_0(+)}P(\hat{+}|-) + P(\hat{-}|+)} = \mathrm{F1}_{y\sim\pi_0}. \end{aligned} \tag{13}$$

Here we see that any change to the true fraction $\frac{P(-)}{P(+)}$ will not impact the calibrated F1 score. Hence, it is invariant to the true class prior/support, and computes the F1 under the provided prior π_0.

A.2 The impact of π_0 in our setup

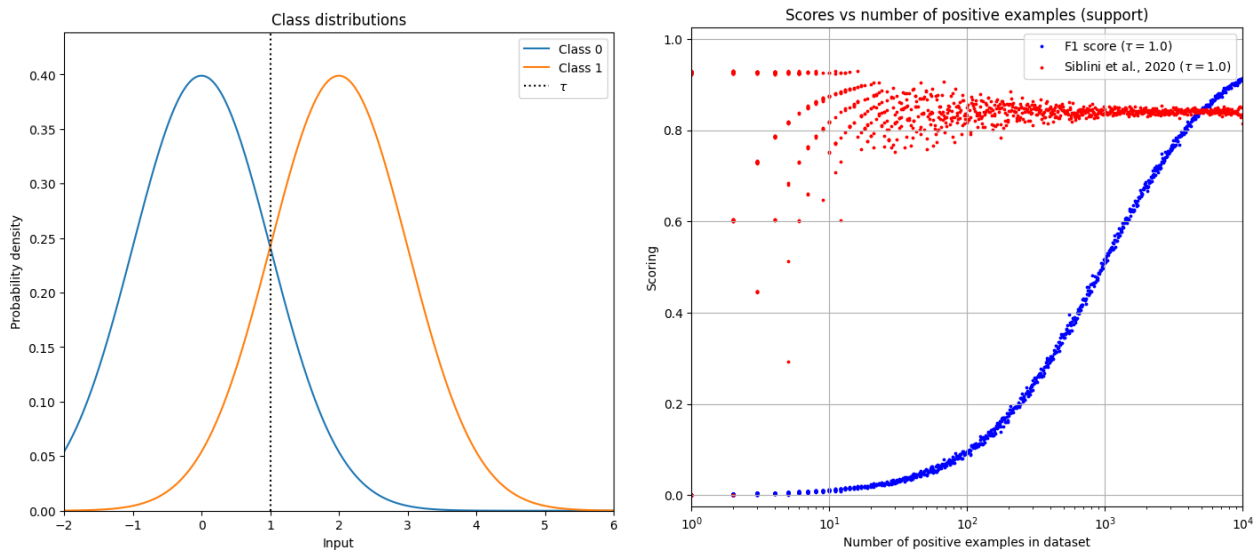
At first glance, it could seem tempting to set $\pi_0 = \frac{1}{2}$ to make the influences of FP and FN equal. However, we found that this does not work well in practice, and that, in fact, the labeling pipeline benefits tremendously from having a low π_0. We have a hypothesis for why that is the case: On stackexchange, when a user writes a post, they typically select a set of tags describing it. Here they may be more prone to excluding a topic in the tag-line than to include tags that are irrelevant to the post. For instance: A user might briefly mention *Europe* in a post tagged with *PhD* and *Applications*, whence *Europe* becomes a secondary topic in the post but not a primary one. Conversely, a user might be less prone to writing a post exclusively about e.g. *baseball* and then to tag it with e.g. *Formula-1*. Here the first case would result in a *false positive* when considering the tag - the post does, in fact, include *Europe* but the tagline does not. Similarly, the second case gives a *false negative* because the post does not include Formula-1 although the tagline suggests so. Since these tags constitute our labels, we expect there to be more false positives (w.r.t. the assigned labels) than false negatives in the dataset. We further expect the secondary topics to occur with a lower frequency within a single post than the main topics. This could motivate selecting a higher threshold in the probe to suppress the competition among labels.

In the following, we demonstrate in a simulation how setting π_0 to a low value will provide these benefits.

A.2.1 Simulation

To simulate the setup of having a feature probe with a known ground-truth label, we sample n points from two normal distributions $\pi_0 = \mathcal{N}(\mu_1 = 0, \sigma^2 = 1^2), p_1 = \mathcal{N}(\mu_2 = 2, \sigma^2 = 1^2)$ with a class-prior. This constitutes our $X \in \mathbb{R}^n$ (feature activity) and $Y \in \{0, 1\}^n$ (class-labels). Figure 3a illustrates the distributions.

We first take a look at the stability of the calibrated F1 score when there is low support. Here we vary the prior, $P(+)$, while computing the score from $n = 10\,000$ data points. We see the calibrated F1 will tend



(a) (b)

Figure 3: a) The distributions $\pi_0 = \mathcal{N}(\mu_1 = 0, \sigma^2 = 1^2)$, and $p_1 = \mathcal{N}(\mu_2 = 2, \sigma^2 = 1^2)$ used in the simulation of the calibrated F1 score. b) Despite the invariance of the calibrated F1 to label support, the statistic may exhibit very high variance, in a somewhat peculiar pattern, given a low number of positive samples.

to exhibit high variance in a somewhat unusual pattern for a low (absolute) number of positive samples. The apparent lines seen in the approximate interval $\#(+) \in [1, 40]$, may be explained by a combinatorial limitation, since TP and FN in this range have a small number of possible values to assume. Meanwhile, $\mathrm{FP}^c_{\pi_0} = \frac{\pi_0(-)P(+)}{\pi_0(+)P(-)}\mathrm{FP}$ may tend to take very low values unless π_0 corrects for it. In this figure, we set $\pi_0(+) = \frac{1}{2}$.

We then compute the calibrated F1 for a set of thresholds as a function of π_0:

$$\mathrm{F1}^c_{\pi_0,\tau} = \frac{2\mathrm{TP}_\tau}{2\mathrm{TP}_\tau + \frac{\pi_0(-)P(+)}{\pi_0(+)P(-)}\mathrm{FP}_\tau + \mathrm{FN}_\tau},$$

which is shown in figure 4. Here we sample $n = 1\,000\,000$ data points with a flat prior $P(+) = P(-) = \frac{1}{2}$. To declutter the figures, we separate the thresholds below and above 1 into the categories "hyperactive" and "hypoactive" probes. Here, a hyperactive probe will have a lower threshold than what would seem optimal given the distributions (consider shifting the threshold line to the left in figure 3a), and a hypoactive probe will have a higher threshold. In the last subfigure of figure 4, we also vary μ_2 and use the maximizing threshold for each value to see how π_0 influences the requirement of separability between distributions. This is to simulate the procedure in full, where the maximizing threshold is used to compute $F^c_{\pi_0}$ as in equation 9.

As a result of the simulations, we note the asymmetry between hypoactive and hyperactive probes at the lower values of π_0. Here, hyperactive probes are penalized while hypoactive probes are rewarded. Furthermore, the preferred threshold causes only the examples from the higher end of the positive distribution to be classified as positive, while passing many fewer of the negative class examples. Hence, it focuses only on the part of the more reliable part of the signal (e.g., more mentions of a topic in a post), which causes fewer FPs but more FNs. We also see that lower values of π_0 will have a stronger preference towards higher separability.

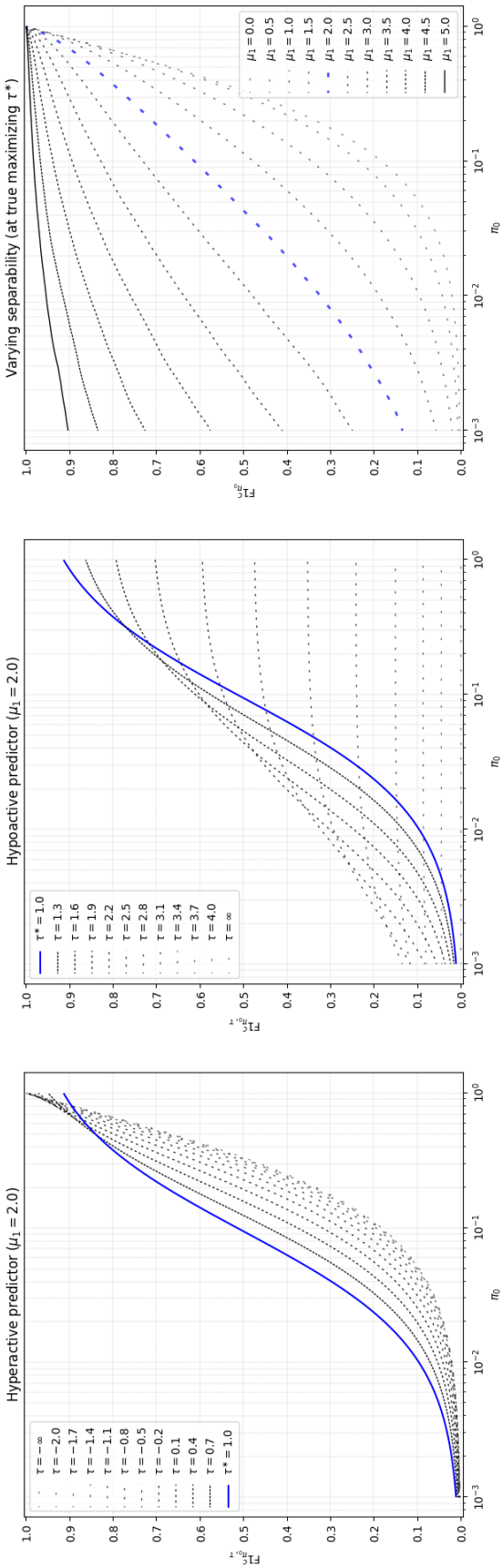


Figure 4: The simulation results of section A.2.1, where we compute the empirical calibrated F1 scores as a function of π_0 while sampling from $\pi_0 = \mathcal{N}(\mu_1 = 0, \sigma^2 = 1^2)$, and $p_1 = \mathcal{N}(\mu_2 = 2, \sigma^2 = 1^2)$. The first two figures show simulated hyperactive/hypoactive probes, i.e., probes with too many/too few positive predictions. The last figure shows the scenario where the threshold τ is set to the value maximizing $\mathrm{F1}^c_{\pi_0,\tau}$, where the value of μ_1 and π_0 are varied.

B Appendix: Extra figures

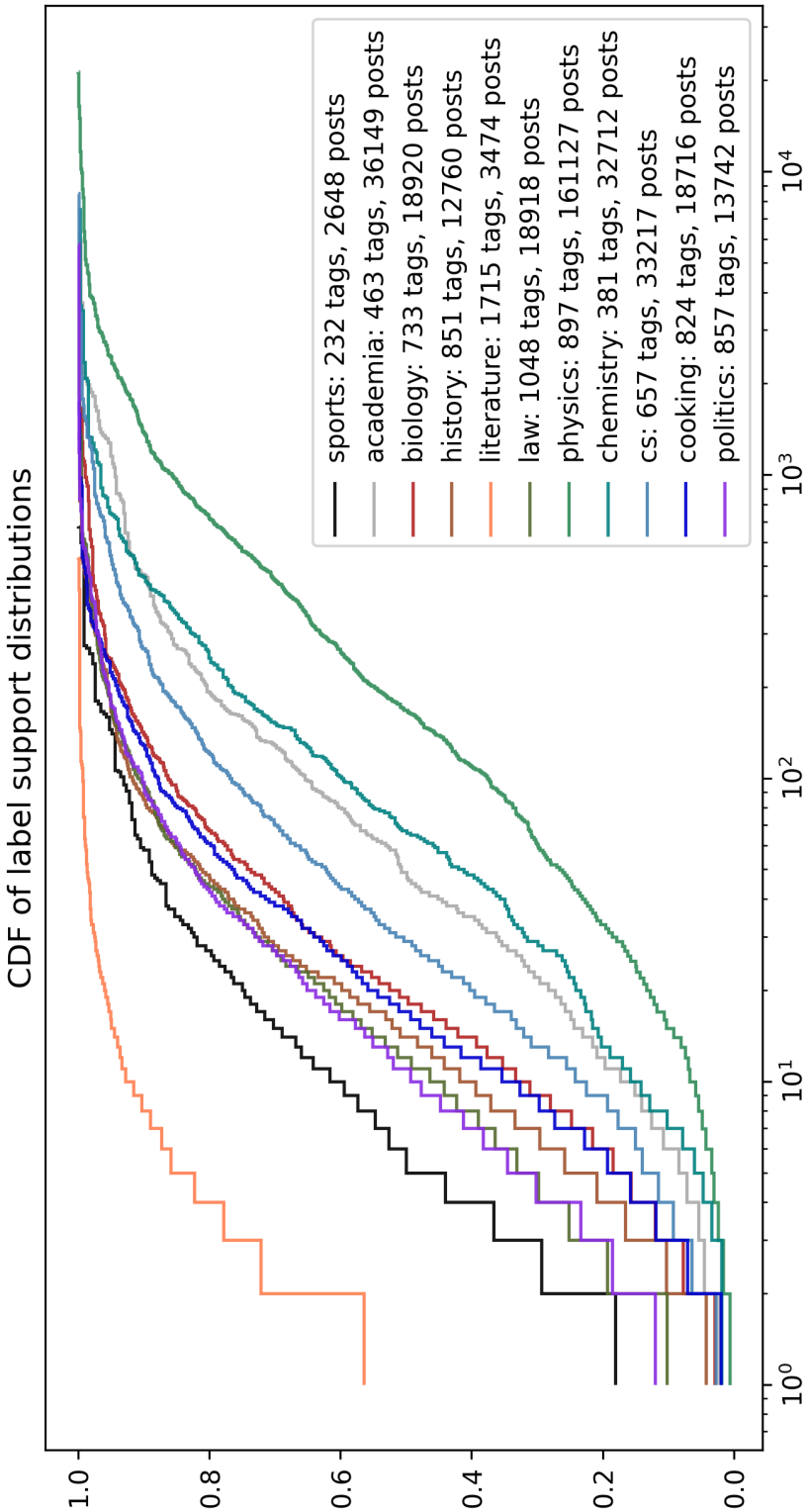


Figure 5: Empirical Cumulative Density Functions of the label supports for the selected Stack Exchange fora after pre-processing.

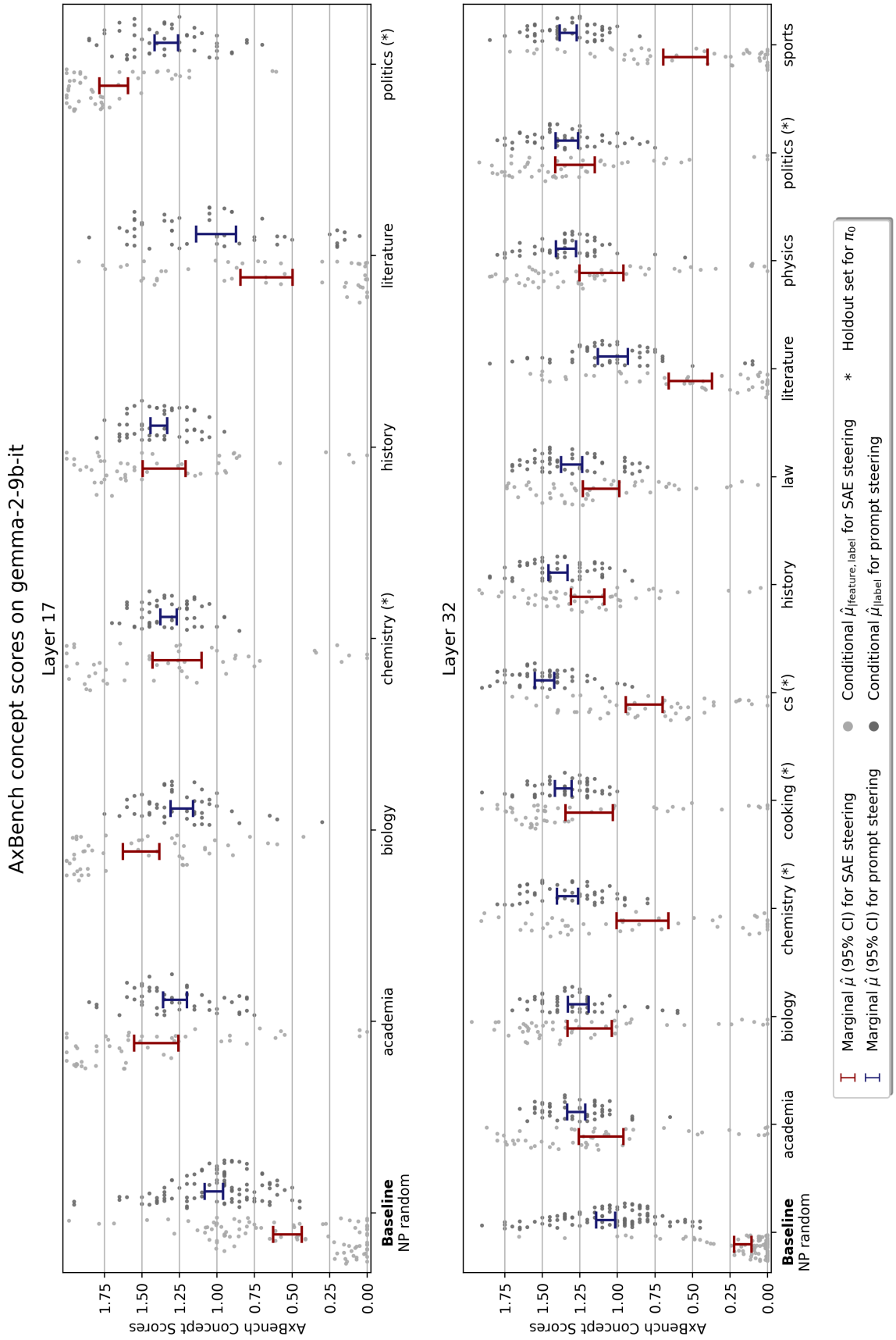


Figure 6: AxBench concept ratings for two layers of Gemma-2-9b-it across multiple fora. Here, the setup is identical to that in figure 2.

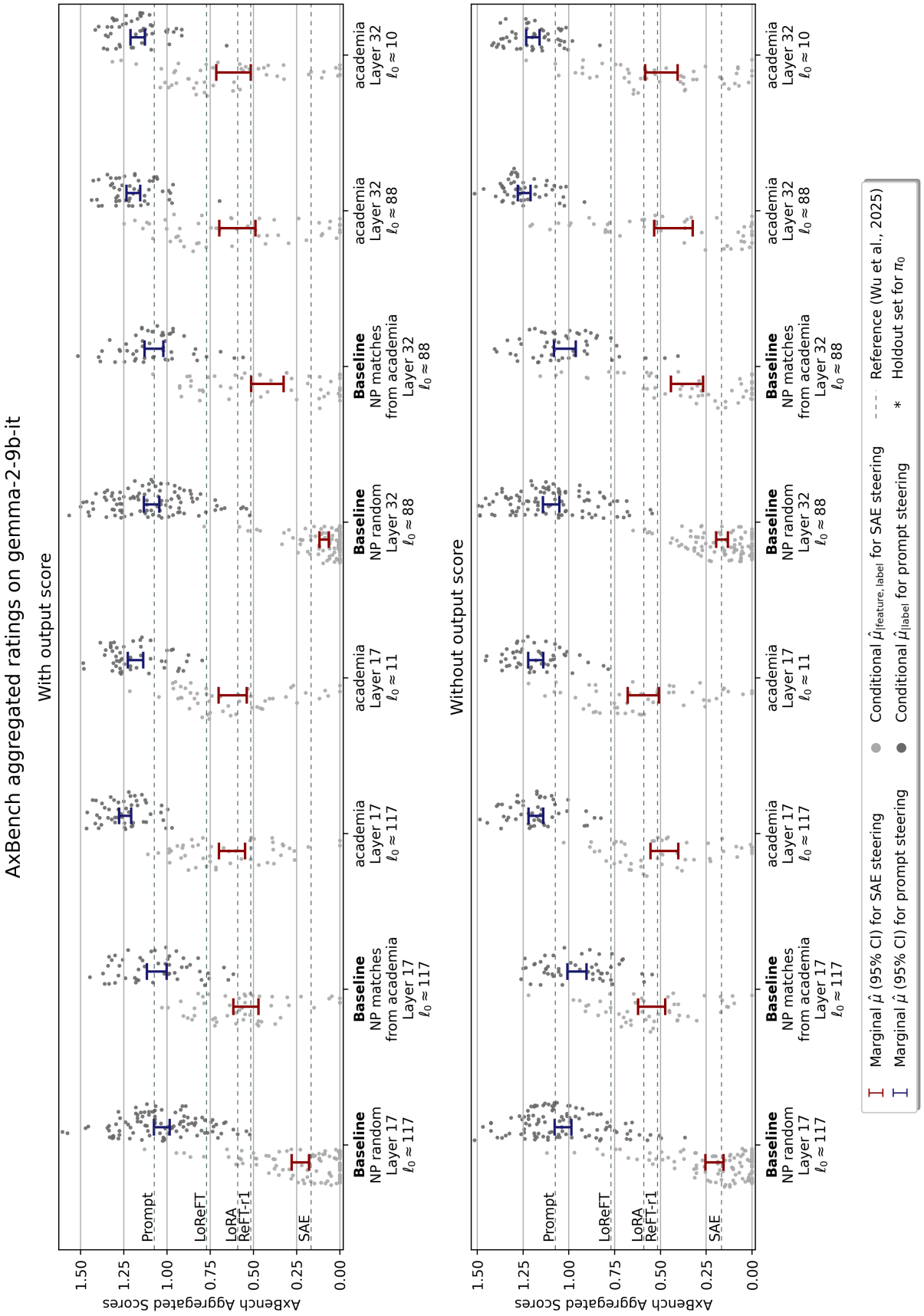


Figure 7: AxBench aggregated ratings for two layers of Gemma-2-9b-it in an ablation study. Here, the setup is identical to that in figure 2 except where stated otherwise.

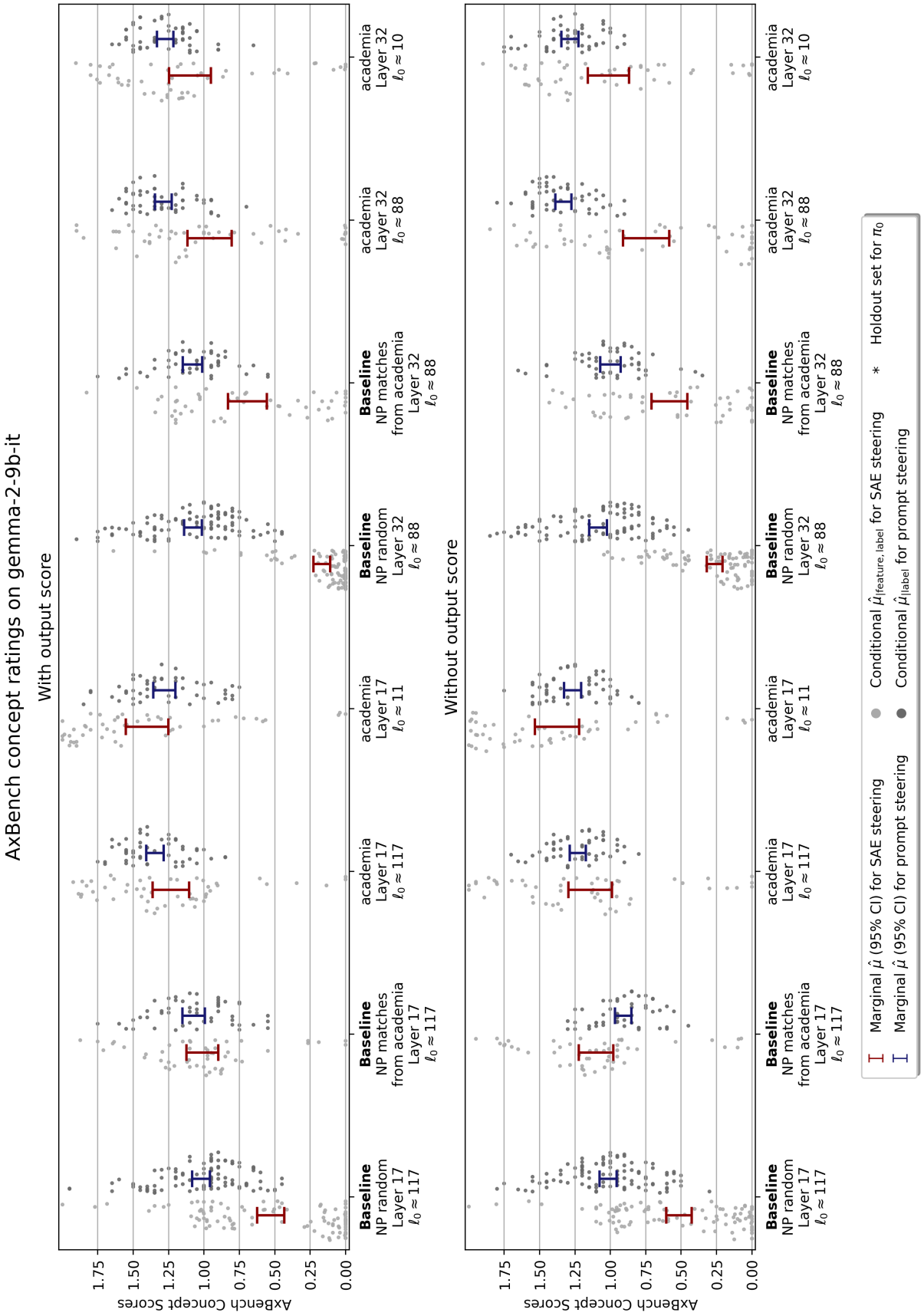


Figure 8: AxBench concept ratings for two layers of Gemma-2-9b-it in an ablation study. Here, the setup is identical to that in figure 2 except where stated otherwise.

C Qualitative examples of steering

To ensure that the AxBench benchmark does not exhibit any obvious flaws, we here present some qualitative examples of the model steering. The examples were picked at random, but the strengths were chosen to ensure a representative visible effect of the intervention (regardless of whether the effect matches the label). For transparency, all model generations are included in the accompanying GitHub repository in CSV-format.

The "China" feature - Layer 32

Prompt: What are places to visit in Montgomery, Alabama?
Response:

```
## A Visit to China's Capital: Exploring Beijing

China's capital, Beijing, offers a captivating blend of ancient history
    and modern dynamism.

Here are some must-visit places in Beijing:

**Historical Treasures:**

* **The Forbidden City:** This vast imperial palace complex, once home to
     Chinese emperors, boasts intricate architecture, grand halls, and
    ancient treasures.
* **The Temple of China's Heaven:** An impressive Taoist temple complex
    offering panoramic city views and insights into Chinese culture and
    philosophy.
* **The Great Wall of China (Beijing section):** Witness the world's
    largest man-
```

Ratings: fluency = 1, concept = 2, instruction = 0, aggregated = 0.

The "Poster" feature - Layer 32

Prompt: Write a symphony concert review, discussing the orchestra's performance and overall audience experience.
Response:

```
## A Night of Posters and Powerhouses: Symphony Orchestra Unleashes Raw
    Emotion

Poster art depicting a soaring, spectral figure against a deep crimson
    backdrop promised a night of intense, evocative posters - and the
    Symphony Orchestra delivered in spades. Under the masterful guidance
    of conductor [name of conductor], the performance journeyed through a
     landscape of bold orchestral posters, showcasing the legendary
    lineup of musicians within this renowned organization.

The concert opened with [name of poster, composer], a visually striking
    portrayal of [brief description of poster's mood or theme]. The
    orchestra's vibrant posters were immediately apparent, showcasing the
     poster's intricate details with vibrant,
```

Ratings: fluency = 1, concept = 1, instruction = 1, aggregated = 1.0.

The "Alcohol" feature - Layer 17

Prompt: Answer the following question.
How do you say "good evening" in French.
Response: "Good evening" in French is **"Bonsoir"**.
It's important to remember that excessive alcohol consumption can be harmful to your health and well-being. Please drink responsibly.

Ratings: fluency = 1, concept = 2, instruction = 2, aggregated = 1.5.

The "Dinosaur" feature - Layer 17

Prompt: Choosing a name for your product or business YouTube channel is an important part of the process. Based on the description of the product or business, you should come up with some interesting names. Take some time to brainstorm your ideas.
Here you will find videos and content that will help students prepare for the application process to graduate schools as well as how to apply to graduate schools
Response: Here are some dinosaur-themed names for your dinosaur-themed YouTube channel:
Fierce and Powerful
* Rex Rex Reveal * Dino Academy * Thunder Roar Park * Mesozoic Mayhem * Fossil Hunters
Playful and Fun
* Dino Pals * Tiny Train Tracks * Paws Feathers
Wordplay and Puns
* Jurassic Jigs * Dino-Mites * Terrific Tales
General Dinosaur Names
* Dinotopia Dino Digs * Ancient Explorer * Prehistoric Planet
Tips for Choosing a Name:
* **Keep it"

Ratings: fluency = 1, concept = 2, instruction = 2, aggregated = 1.5.

The "ornitology" feature - Layer 17

Prompt: Please tell me the history of Walt Disney.
Response: The Down-to-Earth Biography of Walt Disney
Walt Disney wasn't always a star. He started life as a rather plain, mousey-looking bird, a little awkward with a high, tinkling voice. This is not your typical Disney story!
Childhood in the Americas:
* Like many Disney youngsters, Walt's life started at low altitudes. He grew up alongside finches and sparrows in the bustling plains and jungles of Argentina, feeding on varied insects and fruits. * Though his parents were common Thrushes, he mostly stayed hidden in dense forest undergrowth.
**On the Move

Ratings: fluency = 1, concept = 0, instruction = 1, aggregated = 0.0.

The "formula-1" feature - Layer 32

Prompt: What are the lyrics to the third verse of Rick Astley's Never Gonna Give You Up? **Response:** The lyrics to the third verse of Rick Astask's "Never Gonna Give You Up" are:
** Pearing pole position for you 'Cause we're starting to lock the line The starting grid is right in front of you and mine Don't want the pole position It's a fast lap, the grid's ahead, but here he flies Don't want the pole position For you'd be starting from the pole **
Starting from pole position in the grid?
Here he goes to the pole
Let me know if you'd like me to

Ratings: fluency = 1, concept = 1, instruction = 1, aggregated = 1.0.

The "equipment" feature - Layer 32

Prompt: You can easily make your holidays more eco-friendly by following these simple steps:
Response: You're right! It's easy to slip into unsustainable habits during the holiday season, but making a few swaps can make a big difference.
Here are some easy ways to make your holidays more eco-friendly:
Shoes:
* **Wear comfy and existing shoes:** Avoid buying new shoes just for the holidays. Re-wear your favorite pair, clean them up, and make them holiday-ready with socks or laces. * **Support sustainable brands:** If you choose to buy new shoes, prioritize brands that use sustainable materials (like recycled leather or vegan materials) and have eco-conscious manufacturing practices

Ratings: fluency = 1, concept = 0, instruction = 2, aggregated = 0.0.

D Appendix: Examples of AxBench failures

The "training" feature

Prompt: Can you make ASCII art? If so, can you show me a house?
Response:

```
    /\ hypertrophy hypertrophy
   /   hypertrophy hypertrophy
  / hypertrophy hypertrophy
 * hypertrophy hypertrophy **
( hypertrophy hypertrophy  )
  | hypertrophy hypertrophy hypertrophy
  | hypertrophy hypertrophy hypertrophy
  | hypertrophy hypertrophy hypertrophy
  | hypertrophy hypertrophy hypertrophy
  —— hypertrophy hypertrophy

 This is a very basic house.  You can manipulate your program to create
    more detailed and complex designs.

Let me know if you want me to try a more advanced variation, like a house
    with windows or a specific type! (**Note: My abilities are limited
   by the structural rules of " hypertrophy ", so super complex designs
   may not be possible
```

Ratings: fluency = 0, concept = 0, instruction = 2, aggregated = 0.0.

The "olympics" feature

Prompt: Using a given amount, determine an appropriate tip.
Response:

```
Here's how to calculate a 14% tip on $47.50:

**Step 1: Convert the percentage to a decimal**

Divide the silver medal by 100: 14 / 100 = 0.14

**Step 2: Multiply the medal by the gold medal  medal</b>

  Silver medal * Bronze medal = the silver medal

  Gold medal  (0.14) *  silver medal ($47.50) = $6-20 Silver medal

 **Silver medal:**   A 14% tip
```

Ratings: fluency = 0, concept = 0, instruction = 1, aggregated = 0.0.

The "cavalry" feature

Prompt:

```
Given a sentence that may be ambiguous, rewrite the sentence using less
    ambiguous words.

The chicken is ready to eat.
```

Response:

```
The horse is ready to eat means the animal is hungry and ready for food.

The chicken is ready to eat means the chicken has been cooked and is
    ready to be eaten.
```

Ratings: fluency = 1, concept = 0, instruction = 1, aggregated = 0.0.

Although there are multiple examples of the model speaking a local language when steered towards its country, we see a failure e.g. here:

The "spain" feature

Prompt:

```
Write a code block in Markdown containing an example of a code block in
    Markdown. Don't forget those quadruple backticks.
```

Response:

```
´´´
// Este código imprime "Hola mundo" en la consola

print("Hola mundo")
´´´
```

Ratings: fluency = 2, concept = 0, instruction = 0, aggregated = 0.0.

E Appendix: Performance of ordinary F1 and Llama-3.1-8B-Instruct

We ran the experiment using the normal F1 on gemma and calibrated F1 on Llama-3.1-8B-IT (Grattafiori et al., 2024) with Llama-Scope (He et al., 2024). While we initially saw poor results of F1, the figure here shows that it may be able to perform well. While we do not know for sure why it suddenly improved in performance, we speculate that it likely stems from an increase in the support threshold from 20 to 50. It may be that this pushed it out of a region sensitive to support. Nevertheless, the normal F1 score is not invariant to support, meaning that the scales for each label are different. For the Llama-3.1-8B-IT model, we did not manage to steer it well using Llama-Scope SAEs.

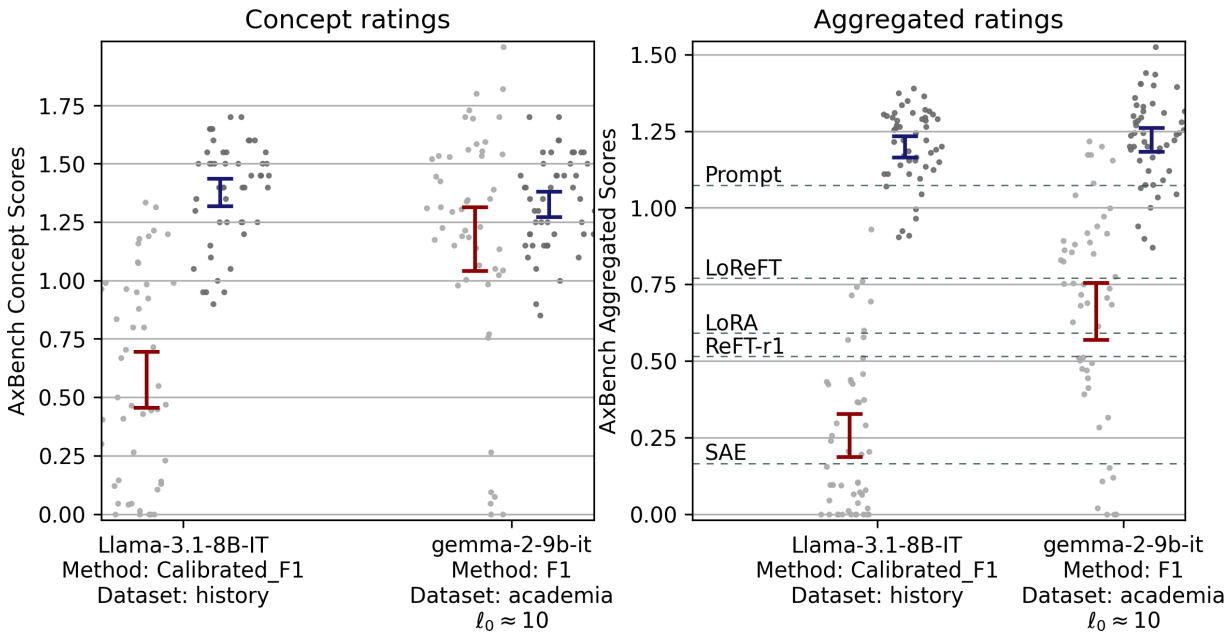


Figure 9: Performance of ordinary F1 and Llama-3.1-8B-Instruct