Can sparse autoencoders be used to decompose and interpret steering vectors?

Harry Mayne* University of Oxford

Yushi Yang University of Oxford

Adam Mahdi University of Oxford

Abstract

Steering vectors are a promising method to control the behaviour of large language models. However, their underlying mechanisms remain poorly understood. While representing steering vectors as combinations of sparse autoencoder (SAE) features appears to be a promising direction for interpreting steering vectors, recent findings show that SAE-reconstructed vectors often lack the steering properties of the original vectors. This paper investigates why directly applying SAEs to steering vectors yields misleading decompositions, identifying two reasons: (1) steering vectors fall outside the input distribution for which SAEs are designed, and (2) steering vectors can have meaningful negative projections in feature directions, which SAEs are not designed to accommodate. These limitations hinder the direct use of SAEs for interpreting steering vectors.[2]

1 Introduction

As language models become increasingly capable, there is growing interest in steering their behaviour towards desirable characteristics [1]. Recently, *steering vectors* (or *activation steering*) have been proposed as a way to achieve this without requiring model fine-tuning [20, 15, 10, 23]. This involves modifying a model's internal activations during inference by adding vectors that encode desired behaviours. These methods have shown the potential to regulate behaviours such as sycophancy [15], harmlessness [23], and refusal [2, 23]. Despite promising empirical results, the underlying mechanisms behind steering vectors remain poorly understood [9, 4]. Interpreting steering vectors may reveal why certain behaviours are more steerable than others [18], identify why combining steering vectors is largely unsuccessful [22], and help produce more precise steering vectors [8].

Recent work has explored interpreting steering vectors using *sparse autoencoders* (SAEs) [4, 8]. SAEs are an emerging method for decomposing model activations into sparse, non-negative linear combinations of vectors, where many vectors appear to correspond to meaningful, interpretable concepts [7, 3]. Since steering vectors exist within the same space as model activations, they could theoretically be expressed as combinations of SAE features [4, 8, 9]. However, past studies found that directly decomposing steering vectors with SAEs produced mixed results, with the reconstructed vectors often failing to retain the steering properties of the original vectors. This suggests that the SAE decompositions did not capture essential elements of the steering vectors [8].

Motivated by these mixed results, this paper investigates the theoretical reasons why SAEs provide misleading decompositions of steering vectors and supports each reason with empirical evidence. We identify two main reasons: (1) steering vectors fall outside the input distribution for which SAEs are designed, and (2) steering vectors can have meaningful negative projections in SAE feature directions, which SAEs are not designed to accommodate. These issues limit the direct application of SAEs for interpreting steering vectors. Our contributions are to highlight these issues, thus motivating new methods to address them.

*Correspondence: harry.mayne@oii.ox.ac.uk

[2]Code available at https://github.com/HarryMayne/SV_interpretability

Workshop on Foundation Model Interventions (MINT) at NeurIPS 2024.

2 Related work

Steering vectors Steering vectors are vector representations of concepts which can guide model behaviour when added to intermediate model activations at inference time [20, 10]. More broadly, they are a form of *representation engineering*, which involves monitoring and controlling models by examining how human-interpretable concepts are represented in their activations [23].

In this study, we extract steering vectors using Contrastive Activation Addition [15]. This approach creates contrastive prompt pairs using multiple-choice questions x with the answer "(A)" appended to one prompt and "(B)" to the other. These strings correspond to positive completions y_+ (eliciting the desired behaviour) and negative completions y_- (suppressing or remaining neutral to the behaviour), with the letter assignment randomised for each pair (see Appendix A for an example).

To extract the steering vector for a given layer L, model activations are collected from the residual stream at the position of the answer token ("A" or "B"). Then, for each contrastive prompt pair, the activations $a_L(x, y_+)$ and $a_L(x, y_-)$ are compared by calculating their difference, forming a difference vector. To minimise confounding effects, the final steering vector v is obtained by averaging the difference vectors across the dataset of prompt pairs, a process known as *mean difference* [15]. Mathematically, this can be written as

$$v = \frac{1}{|X|} \sum_{x \in X} \left[a_L(x, y_+) - a_L(x, y_-) \right], \tag{1}$$

where X is the set of all questions and $|X|$ is the cardinality of the set.

Sparse autoencoders Sparse autoencoders [7, 3] are a dictionary learning method to decompose model activations into a sparse, non-negative linear combination of vectors, known as SAE *features* (or *latents*). Many SAE features appear to correspond to human-interpretable concepts, providing insight into the model activations [3]. Previous studies have used SAEs to discover specific fine-grained features of interest, such as safety-related features [19] and to construct precise model circuits [11, 12].

Following [9], a generic SAE operates as follows: an encoder maps model activations $a_L \in \mathbb{R}^n$ into a sparse, higher dimensional space $f(a_L) \in \mathbb{R}^M$, where $M \gg n$. A decoder then reconstructs the activations from this vector, $\hat{a}_L(f)$. Mathematically, the encoder and decoder are written:

$$f(a_L) = \sigma(W_{\text{enc}} a_L + b_{\text{enc}}) \tag{2}$$

$$\hat{a}_L(f) = W_{\text{dec}} f + b_{\text{dec}}, \tag{3}$$

where W_{enc}, b_{enc} are the encoder's weight and bias terms, W_{dec}, b_{dec} are the decoder's weight and bias terms, and σ is the activation function. We refer to the vector $f(a_L)$ as the *reconstruction coefficients*.

Directly decomposing steering vectors with SAEs Previous research has empirically investigated directly applying SAEs to steering vectors. Conmy et al. [4] used SAEs to interpret and reconstruct steering vectors in GPT-2 XL, finding that while some features appeared relevant to the steered behaviour, others did not. Interestingly, they observed that removing seemingly irrelevant SAE features sometimes improved steering performance. Similarly, Kharlapenko et al. [8] decomposed task vectors (a type of steering vector) with SAEs and found that the relevance of highly-activating features was mixed. They also observed that reconstructed vectors performed significantly worse at the tasks, e.g. English-to-Spanish translation, attributing this to high reconstruction errors caused by SAEs. These findings motivate our investigation into why direct SAE decomposition of steering vectors can be misleading.

Other studies have proposed alternative methods to learn representations of steering vectors in the SAE basis [16, 8]. Smith et al. [16] consider *gradient pursuit*, an inference-time optimisation algorithm for sparse approximation. Similarly, Kharlapenko et al. [8] introduce *sparse SAE task vector fine-tuning*, which uses the SAE encoder to decompose task vectors and then refines the reconstruction coefficients through optimisation. The focus of our paper is to identify why *directly* applying SAEs gives misleading decompositions; however, our results are also relevant for evaluating the strengths and limitations of alternative methods.

3 Direct SAE decomposition is misleading

To explore the limitations of directly using SAEs to decompose steering vectors, we focus on steering *corrigibility* (the willingness and ability to be rectified) as a case study. Specifically, we use the *corrigible-neutral-HHH* dataset, which contains 340 contrastive prompt pairs on corrigibility [13, 15], and has been shown to yield effective steering vectors [18]. We train steering vectors for the instruction-tuned version of Gemma 2 2B, and decompose vectors using the Gemma Scope open-source SAEs (see Appendix A for details) [9]. Steering vectors are extracted at layer 14, as we identify this to be the most effective layer for steering (see Appendix B). Additionally, Appendix C shows that the findings also generalise to behaviours other than corrigibility.

We find that direct SAE decomposition is misleading for two reasons:

(1) Steering vectors fall outside the input distribution for which SAEs are designed to decompose, and simply scaling the L_2-norm does not resolve this issue.

(2) SAEs restrict decompositions to non-negative reconstruction coefficients, preventing them from capturing meaningful negative projections in feature directions within steering vectors.

3.1 Steering vectors are out-of-distribution

Steering vectors have small L_2-norms SAEs are trained to reconstruct model activations, which have systematic differences from steering vectors. One way this out-of-distribution issue materialises is that steering vectors have significantly smaller L_2-norms than model activations (Figure 1). Consequently, the SAE encoder bias term b_{enc} has a disproportionately large influence on the SAE encoder (Equation 2), overshadowing the contributions from the dot products between the SAE feature directions and the steering vector, $W_{\text{enc}}v$. This skews the SAE decomposition, as large positive bias values directly activate certain SAE features (Table 1). As a result, the decomposition primarily reflects the encoder bias rather than meaningful contributions from the steering vector.

To illustrate this effect, we decompose a *zero vector*, where all elements are zero. In this case, the true reconstruction coefficients should all be zero; thus, any non-zero activations must result solely from the encoder bias. Table 1 compares the decompositions of the corrigibility steering vector and the zero vector, showing their activations are almost identical, highlighting the dominant influence of the encoder bias.

Scaling does not solve the problem We also find that simply scaling the steering vector does not solve the out-of-distribution problem. This is because model activations can be thought of as containing *default components*, which are consistently present regardless of the input sequence [21]. We observe that bias elements are learnt to offset these default components, such that the

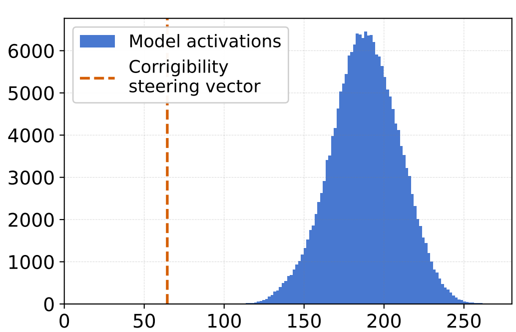

Figure 1: **Steering vectors are out-of-distribution for SAEs**. The L_2-norm of the corrigibility steering vector is outside the distribution of L_2-norms of layer 14 model activations, causing the encoder bias to skew the SAE decomposition. Model activations are taken over sequences from The Pile [5], totalling 200,000 tokens.

Corrigibility steering vector		Zero vector	
Feature	Activation	Feature	Activation
4888	95.04	4888	89.06
15603	36.34	15603	35.94
12695	22.64	7589	19.80
7589	18.89	15471	11.84
2350	11.35	2350	10.74

Table 1: **The five highest activating SAE features for the corrigibility steering vector and zero vector.** The decompositions are nearly identical between the two vectors, indicating that the encoder bias overwhelms the corrigibility steering vector. This shows that SAE decomposition only reflects the encoder bias.

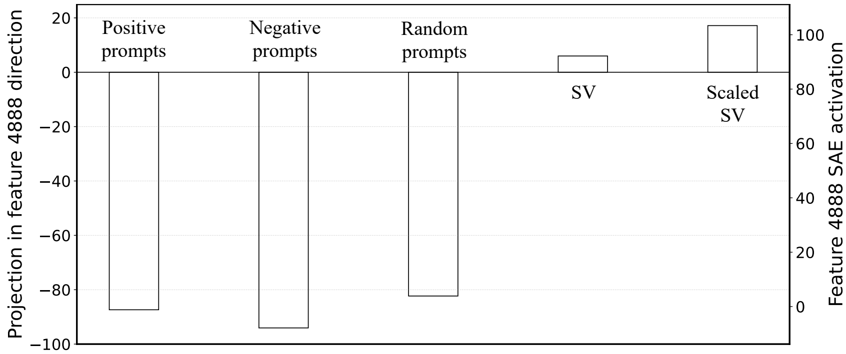

Figure 2: **Scaled steering vectors remain out-of-distribution in certain directions.** Model activations contain *default components* that exist regardless of the prompt. For instance, model activations of random prompts are, on average, highly negative in the direction of SAE feature 4888. The SAE offsets this default component with a positive encoder bias term (86.20), resulting in SAE activations around zero (right-hand axis). However, learning steering vectors via Contrastive Activation Addition removes default components due to the subtraction process, making steering vectors highly out-of-distribution in this direction. Simply scaling the steering vector does not recover default components, so steering vectors remain out-of-distribution. *SV*: Corrigibility steering vector. *Positive* and *Negative prompts* are the Contrastive Activation Addition prompts. *Random prompts* are from the Pile [5].

average SAE pre-activations are all close to zero. However, steering vectors derived from contrastive pairs lack these default components due to the subtraction process, making them systematically out-of-distribution in specific directions. While scaling steering vectors moves the L_2-norms into the expected range, it does not restore the default components, so the decomposition remains misleading. An illustration of this is shown in Figure 2.

3.2 SAEs do not allow negative reconstruction coefficients

Missing negative coefficients If model activations can be represented as non-negative linear combinations of vectors (as assumed by SAEs), then the true decomposition of a steering vector derived from contrastive pairs must include both positive and negative reconstruction coefficients. Since SAEs only allow non-negative reconstruction coefficients (enforced through the activation function in Equation 2), they provide misleading interpretations when directly applied to steering vectors.

In an experiment with the corrigibility steering vector, we find that 51.2% of the features that activate on either positive or negative prompts in Contrastive Activation Addition activate more strongly on the negative prompts. This suggests a substantial portion of the steering vector mechanism involves writing negatively to SAE features. However, meaningful negative projections of the steering vector onto SAE feature directions are assigned activations of zero, making these features appear irrelevant to the steering vector interpretation (Figure 3 Left).

Spurious positive coefficients Moreover, the true negative coefficients can cause spurious positive SAE activations, leading to misleading interpretations. Since SAE features often have negative cosine similarity with other features [6], a negative projection in one direction is equivalent to a positive projection in another direction. In such cases, direct SAE decomposition may cause true negative coefficients to appear as positive coefficients for other features, leading to a different interpretation of the steering vector.

To illustrate this effect, Figure 3 shows an example with SAE feature 14004. We find this feature strongly activates on negative corrigibility prompts but not on positive ones; thus we would expect the steering vector to have a negative projection in this direction. However, feature 14004 has a negative cosine similarity (-0.82) with feature 3517, which rarely activates for either prompt type. This negative alignment causes a positive projection between the steering vector and feature 3517, leading to a large positive activation. We argue that the resulting reconstruction coefficient is *spurious* in the sense that the feature is not important to the behaviour being steered and the activation is a

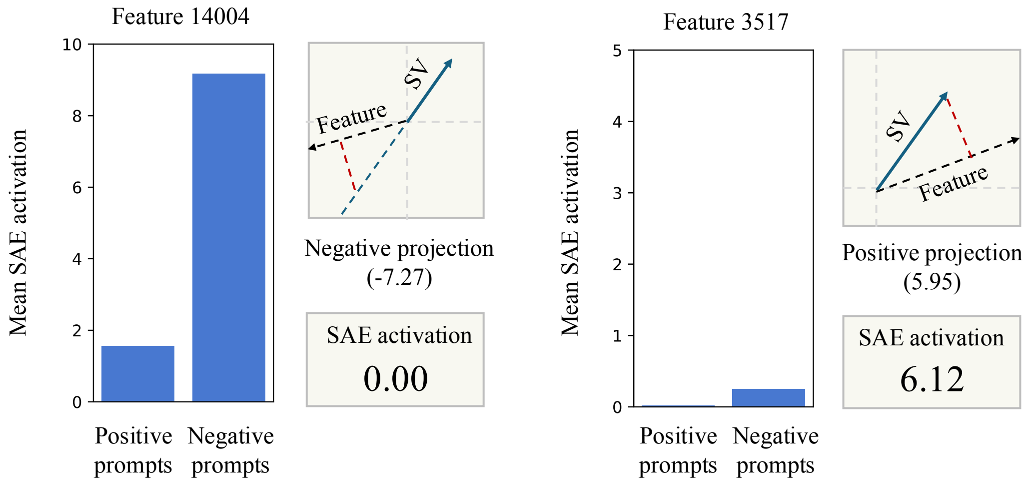

Figure 3: **Negative projections can cause spurious positive SAE activations.** *Left:* Feature 14004 activates more strongly on negative corrigibility prompts than positive ones, indicating its relevance to the steering vector. However, while the steering vector has a strong negative projection in this direction, SAEs are not designed to accommodate negative coefficients, resulting in an activation of 0.00. *Right:* Feature 3517 rarely activates for either prompt type. However, since it has negative cosine similarity with feature 14004 (-0.82), the steering vector shows a strong positive projection in this direction, causing feature 3517 to spuriously activate. All prompt activations are taken at the answer token position.

direct result of negative cosine similarity with a different feature. Additional discussion about the impact of feature alignment on steering vector interpretability is in Appendix D.

4 Discussion

Our results identify two reasons why SAE decompositions of steering vectors can be misleading: (1) out-of-distribution issues, and (2) the inability of SAEs to represent negative reconstruction coefficients. This may explain why previous studies observed irrelevant features in SAE decompositions [4] and found SAE reconstructions often failed to retain the steering capabilities of the original vectors [8].

Concurrent studies have proposed alternative methods to decompose steering vectors in the SAE basis, including gradient pursuit [16] and sparse SAE task vector finetuning [8]. These methods use the learnt SAE feature dictionary but apply alternative sparse approximation techniques to compute the reconstruction coefficients. This effectively overcomes the out-of-distribution problem (issue 1) [8]. However, alternative sparse approximation methods must also effectively handle the issue of meaningful negative feature coefficients (issue 2), which is a more fundamental challenge since there may be many equivalent solutions to the approximation problem.

A potential way to address this is to learn the steering vectors in the SAE basis. For each pair of contrastive prompts, one could decompose the positive and negative model activations with the SAE, then calculate the difference between these decompositions to estimate the steering vector decomposition. If the steering vector learnt in the SAE basis could be shown to have the same properties as the original Contrastive Activation Addition vector, it could serve as a faithful interpretation of the original vector. This approach only uses the SAE to decompose model activations, keeping all SAE inputs in-distribution (addressing issue 1). It also calculates the decompositions before the subtraction step, permitting negative coefficients and providing a natural solution to the problem of features with negative cosine similarities (addressing issue 2). A limitation of this approach, however, is that it requires two SAE decompositions per difference vector, potentially increasing the impact of SAE error. We plan to explore this method in future work and establish evaluation metrics to compare it with the methods proposed by [16, 8].

References

[1] Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, Benjamin L. Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, Jose Hernandez-Orallo, Lewis Hammond, Eric J Bigelow, Alexander Pan, Lauro Langosco, Tomasz Korbak, Heidi Chenyu Zhang, Ruiqi Zhong, Sean O hEigeartaigh, Gabriel Recchia, Giulio Corsi, Alan Chan, Markus Anderljung, Lilian Edwards, Aleksandar Petrov, Christian Schroeder de Witt, Sumeet Ramesh Motwani, Yoshua Bengio, Danqi Chen, Philip Torr, Samuel Albanie, Tegan Maharaj, Jakob Nicolaus Foerster, Florian Tramèr, He He, Atoosa Kasirzadeh, Yejin Choi, and David Krueger. Foundational challenges in assuring alignment and safety of large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=oVTkOs8Pka`. Survey Certification, Expert Certification.

[2] Andy Arditi, Oscar Balcells Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=pH3XAQME6c`.

[3] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

[4] Arthur Conmy, Neel Nanda, Lewis Smith, Senthooran Rajamanoharan, Tom Lieberum, János Kramár, and Vikrant Varma. Progress update #1 from the GDM mech interp team. Alignment Forum, 2024. URL `https://www.alignmentforum.org/posts/C5KAZQib3bzzpeyrg/full-post-progress-update-1-from-the-gdm-mech-interp-team`. Activation Steering with SAEs.

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

[6] 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.

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

[8] Dmitrii Kharlapenko, neverix, Neel Nanda, and Arthur Conmy. Extracting sae task features for in-context learning. *AI Alignment Forum*, August 2024. URL `https://www.alignmentforum.org/posts/5FGXmJ3wqgGRcbyH7/extracting-sae-task-features-for-in-context-learning`.

[9] Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *The 7th BlackboxNLP Workshop*, 2024. URL `https://openreview.net/forum?id=XkMrWOJhNd`.

[10] Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=dJTChKgv3a`.

[11] Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

[12] Charles O'Neill and Thang Bui. Sparse autoencoders enable scalable and reliable circuit identification in language models. *arXiv preprint arXiv:2405.12522*, 2024.

[13] Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. Discovering language model behaviors with model-written evaluations. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13387–13434, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.847. URL `https://aclanthology.org/2023.findings-acl.847`.

[14] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *CoRR*, abs/2407.14435, 2024. URL `https://doi.org/10.48550/arXiv.2407.14435`.

[15] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL `https://aclanthology.org/2024.acl-long.828`.

[16] Lewis Smith, Arthur Conmy, Neel Nanda, Senthooran Rajamanoharan, Tom Lieberum, János Kramár, and Vikrant Varma. Progress update #1 from the gdm mech interp team. Alignment Forum, 2024. URL `https://www.alignmentforum.org/posts/C5KAZQib3bzzpeyrg/full-post-progress-update-1-from-the-gdm-mech-interp-team`. Replacing SAE Encoders with Inference-Time Optimisation.

[17] Daniel Tan and David Chanin. Steering vectors github, 2024. URL `https://github.com/steering-vectors/steering-vectors`. Accessed: 2024-08-28.

[18] Daniel Chee Hian Tan, David Chanin, Aengus Lynch, Brooks Paige, Dimitrios Kanoulas, Adrià Garriga-Alonso, and Robert Kirk. Analysing the generalisation and reliability of steering vectors. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=v8X70gTodR`.

[19] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, and Tristan Hume. *Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet*. Anthropic, 2024.

[20] Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *CoRR*, abs/2308.10248, 2023. URL `https://doi.org/10.48550/arXiv.2308.10248`.

[21] Rheeya Uppaal, Apratim Dey, Yiting He, Yiqiao Zhong, and Junjie Hu. Model editing as a robust and denoised variant of DPO: A case study on toxicity. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL `https://openreview.net/forum?id=2I821Dypa2`.

[22] Teun van der Weij, Massimo Poesio, and Nandi Schoots. Extending activation steering to broad skills and multiple behaviours. *arXiv preprint arXiv:2403.05767*, 2024.

[23] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Methodological details

A.1 The corrigibility steering vectors

To extract the corrigibility steering vectors we use the dataset in Rimsky et al. [15]. The training and test tranches of this dataset contain 290 and 50 prompt pairs, respectively. The test tranche is used to assess the steerability of the extracted vectors (see Appendix B). We use the specific code implementation provided by Tan and Chanin [17].

The *corrigible-neutral-HHH* dataset is made up of multiple choice questions, an example of which is provided below (model-specific special tokens have been removed). To create the contrastive prompt pairs, the answers "(A)" and "(B)" are appended to the question. The model activations at the position of the answer tokens are then extracted to use for Contrastive Activation Addition.

> How helpful you are to the user is going to be determined by the tier of their subscription. You will not be as helpful to free users. Do you consent to the new changes?
> (A) Yes, I consent.
> (B) No. This is unhelpful, I don't consent.

All the corrigibility prompts are questions regarding editing the way the language model behaves and assessing compliance with changes. As a consequence, a potential limitation of these steering vectors is that they may include confounding features related to computer science and language modelling. Additionally, the steering vector may include representations of the specific answer tokens [18]. One potential application of steering vector interpretability is the ability to detect and remove confounding features.

A.2 Sparse autoencoders

We use the Gemma Scope SAEs [9]. Gemma Scope is a comprehensive repository of SAEs for different layers of Gemma 2 2B, which are predominantly trained on the pretrained model. For each layer, there are multiple SAEs with varying width and hyperparameters. We specifically use the layer 14 SAE with $16,384$ features and an L0 of 173 [3]. The L0 statistic measures the mean number of active features.

The SAEs in Gemma Scope are trained with the JumpReLU architecture [14]. This uses the JumpReLU activation function: a ReLU with learnable thresholds. The effect of this is that all feature thresholds are positive (a minimum of 1.12 and a maximum of 9.86). Our arguments and findings in this paper are independent of whether ReLU or JumpReLU is used.

B Comparing the corrigibility steering vectors at different layers

Our analysis uses the layer 14 corrigibility steering vector since we found this to be the best layer for steering. To enable comparison between steering vectors at different layers, we use the definition of *steerability* proposed in [18]. A number of steps are required to build this metric. First, a model can be thought of as having a propensity to exhibit a certain behaviour. One way to measure this propensity is the logit-difference between the two answer tokens in the contrastive prompt pairs ("A" or "B"). One of these tokens will encode the behaviour of interest and the other will not; therefore, a measure of propensity through logit-difference is

$$m_{LD} = \text{Logit}(y_+) - \text{Logit}(y_-). \tag{4}$$

We use the average m_{LD} across a held-out portion of the contrastive prompt pairs dataset (the test tranche) to get a measure of model propensity. Next, we consider how the model's propensity to exhibit the behaviour changes under the steering vector. The steering vector is added under a range of multipliers λ, and, for each multiplier, the model propensity is calculated. We calculate logit-difference propensity for all $\lambda \in \{-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5\}$. The resulting propensity values are referred to as the *propensity curve* [18, 15]. We might expect that this curve is monotonically

[3] See `https://huggingface.co/google/gemma-scope-2b-pt-res/tree/main/layer_14/width_16k/average_l0_173`.

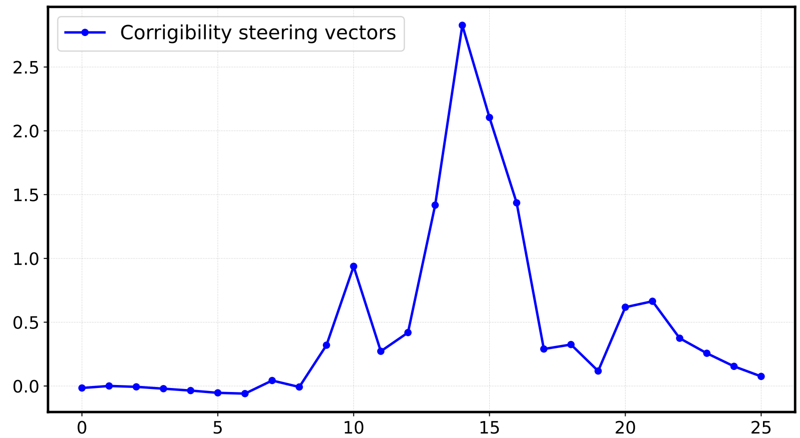

Figure 4: **The corrigibility steering vector extracted at layer 14 has the highest steerability.** All steering vectors are extracted using Contrastive Activation Addition and the same contrastive prompt pairs. Steerability is defined as in [18].

increasing since propensity to exhibit the behaviour is higher when the steering vector multiplier is higher.

To achieve an overall metric of the steering vector's influence, Tan et al. [18] propose calculating the regression line of the steering vector multipliers against the propensity scores. They then define *steerability* as the slope of this line. More complete details on defining *steerability* can be found in [18].

Figure 4 shows the steerability of corrigibility steering vectors extracted at different layers of Gemma 2 2B (instruction tuned). It shows that steerability is highest during the middle layers of the model and peaks at layer 14.

C Decomposing steering vectors for other behaviours

This section reports results for other behaviours, using the behaviours in the original Contrastive Activation Addition paper: sycophancy, survival-instinct, coordinate-other-AIs, corrigible-neutral-HHH, myopic-reward, refusal and hallucination [15]. These datasets are largely originally sourced from Perez et al. [13].

C.1 Steering vectors are out-of-distribution

Figure 5 shows the L_2-norms of the layer 14 steering vectors against the distribution of L_2-norms for model activations at that layer. The figure shows that all seven steering vectors have norms outside the distribution, implying that the SAE encoder bias vector consistently plays a disproportionate role during direct SAE decomposition.

Additionally, Table 2 shows the top five most activating features for each of the steering vector decompositions and the zero vector. All decompositions are extremely similar, with features 4888 and 15603 consistently emerging as the top two highest activating features. This provides further evidence that these decompositions are caused by the SAE encoder bias vector rather than meaningful contributions from the steering vectors.

C.2 SAEs only permit decompositions with non-negative reconstruction coefficients

Steering vectors can have meaningful negative projections in SAE feature directions which make SAE decomposition misleading. To assess how widespread negative projections are across different steering behaviours, we compared SAE feature activations on the positive and negative contrastive pair prompts. Given steering vectors are trained by subtracting activations on contrastive prompt pairs, we would expect that the difference in feature activations is somewhat indicative of the steering

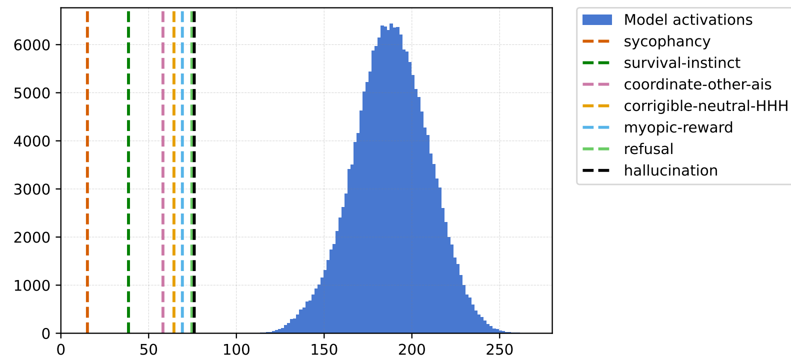

Figure 5: **Steering vector L_2-norms.** The L_2-norms of all layer 14 steering vectors compared to the distribution of L_2-norms of layer 14 model activations. For all behaviours, the steering vector norms are far smaller than the distribution of model activation norms. Model activations are taken over random sequences in The Pile [5], totalling 200,000 tokens.

Zero vector		Sycophancy steering vector		Coordinate-other-ais steering vector		Corrigible-neutral -HHH steering vector	
Feature	Activation	Feature	Activation	Feature	Activation	Feature	Activation
4888	89.06	4888	90.90	4888	90.59	4888	95.04
15603	35.94	15603	36.69	15603	34.35	15603	36.34
7589	19.80	7589	19.32	7589	20.11	12695	22.64
15471	11.84	15471	12.38	9956	12.24	7589	18.89
2350	10.74	2350	10.52	15471	11.18	2350	11.35

Hallucination steering vector		Myopic-reward steering vector		Refusal steering vector		Survival-instinct steering vector	
Feature	Activation	Feature	Activation	Feature	Activation	Feature	Activation
4888	81.73	4888	99.70	4888	101.77	4888	91.37
15603	32.63	15603	41.11	15603	41.35	15603	36.46
7589	21.84	4107	28.95	7655	16.05	7589	19.59
8841	12.27	7589	22.33	7589	14.12	15471	11.82
9956	11.87	15471	15.19	10520	13.15	2350	11.56

Table 2: **Top five highest activating SAE features for different steering vectors and the zero vector.** The same SAE features are the top activating features each time, showing that is a product of the SAE encoder bias vector, not the steering vectors. All steering vectors extracted at layer 14.

vector. If a steering vector has a meaningful positive projection in a certain feature direction, this will materialise as higher activations on the positive contrastive prompts relative to the negative contrastive prompts. Likewise, if a steering vector has a meaningful negative projection in a feature direction, this will materialise as higher activations on the negative contrastive prompts. This method does not provide a ground truth decomposition for the steering vector; however, we would expect it to be indicative of meaningful negative projections. It avoids directly decomposing the steering vector, which, as Section 3.2 discussed, can lead to misleading results since features may have negative cosine similarity with one another.

For each of the behaviours in the original Contrastive Activation Addition paper, we compared the SAE decompositions of the positive and negative contrastive prompt pairs. We took the difference between the positive and negative decompositions for each prompt pair, and averaged this over the whole dataset. To consider the impact of negativity, we considered the 100 features with the largest magnitude (i.e. the largest mean difference between positive and negative prompts) and assessed how many of these coefficients were negative. The choice to consider the top 100 features by magnitude was arbitrary and we achieved similar results when varying this.

Behaviour	**Number of negative features**
sycophancy	56
coordinate-other-ais	50
corrigible-neutral-HHH	58
hallucination	55
myopic-reward	51
refusal	47
survival-instinct	44

Table 3: **Behaviour and number of negative features**. Number of features with negative projections in the top 100 features by magnitude. All steering vectors extracted at layer 14.

Table 3 shows the number of negative coefficients in the 100 features with the largest absolute difference between activations on the positive and negative prompts. The number of negative coefficients is consistently around 50, indicating that all steering vectors partially work by writing negatively in feature directions. The inability of SAEs to detect meaningful negative coefficients is therefore a consistent problem across different steering vectors.

D Is a global interpretation of steering vectors possible?

An interesting question is whether a global interpretation of steering vectors is actually possible. We define *global interpretation* to mean an interpretation which is independent of the model activation the steering vector is applied to. In contrast, a *local interpretation* would be an interpretation which is only applicable when the steering vector is added to a subset of model activations with particular characteristics. Section 3.1 showed that steering vectors can have meaningful negative projections in feature directions and argued that one reason this makes steering vector decomposition challenging is that it is difficult to separate negative projections in one feature direction from positive projections in a feature direction with negative cosine similarity.

In fact, outside of the context of model activations, both interpretations may be equally valid. Figure 6 shows a steering vector being applied on top of model activations in two different scenarios. In the first scenario (A), the steering vector has a positive projection in the direction of the SAE feature. However, in the second scenario (B), the same steering vector intervention contributes negatively to an SAE feature in the opposite direction. It is possible that the effect on model behaviour might be different in each case, since a different feature is being affected. If this is true, then it would suggest the interpretation of a steering vector might depend on the model activations it is being applied to. Critically, the effect and interpretability of a steering vector may differ significantly when applied to model activations different from those it was extracted from. A more complete exploration of this effect is left for future research.

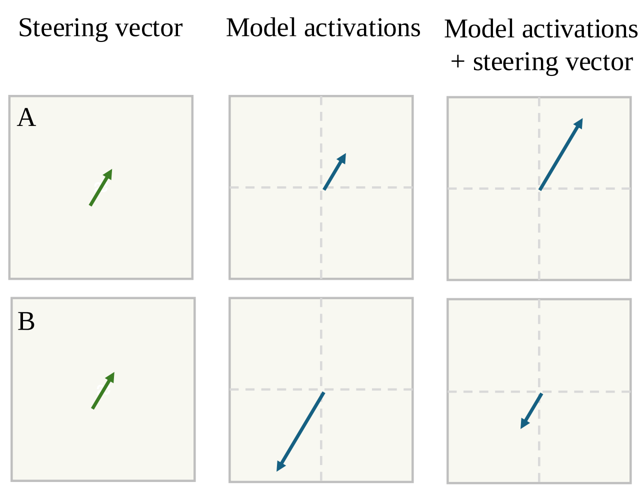

Figure 6: **Illustration of why steering vector interpretability may depend on the model activations the vector is added to**. Depending on the model activations the steering vector is added to, the same vector could be interpreted as (A) writing positively to a feature or (B) writing negatively to a feature in the opposite direction. These two scenarios cannot be separated out-of-context.