

---

# IMPROVING STEERING VECTORS BY TARGETING SPARSE AUTOENCODER FEATURES

---

Sviatoslav Chalnev\*  
slava.chalnev@gmail.com

Matthew Siu\*  
matthewwilsonsiu@gmail.com

Arthur Conmy  
arthurconmy@gmail.com

## ABSTRACT

To control the behavior of language models, steering methods attempt to ensure that outputs of the model satisfy specific pre-defined properties. Adding steering vectors to the model is a promising method of model control that is easier than finetuning, and may be more robust than prompting. However, it can be difficult to anticipate the effects of steering vectors produced by methods such as CAA [Panickssery et al., 2024] or the direct use of SAE latents [Templeton et al., 2024]. In our work, we address this issue by using SAEs to measure the effects of steering vectors, giving us a method that can be used to understand the causal effect of any steering vector intervention. We use this method for measuring causal effects to develop an improved steering method, **SAE-Targeted Steering** (SAE-TS), which finds steering vectors to target specific SAE features while minimizing unintended side effects. We show that overall, SAE-TS balances steering effects with coherence better than CAA and SAE feature steering, when evaluated on a range of tasks.

## 1 Introduction

There are widespread calls for better control of the behaviour of Large Language Models (LLMs; e.g. The White House [2023]). Current methods such as prompting [Wallace et al., 2024] and finetuning [Ouyang et al., 2022, Chung et al., 2022] offer some degree of control, but have clear limitations. For example, prompting can be fragile and is often susceptible to methods that can subvert these instructions [Wei et al., 2023]. Finetuning a model can be more robust but requires a curated dataset for training which can be both expensive and time-consuming to produce (e.g. Dubey et al. [2024]).

*Steering vectors* [Turner et al., 2024] have the potential to be more robust than prompting, and both cheaper and easier to implement than finetuning. Steering vectors work by adding activations to the hidden state of a model, part way through the forward pass (Section 3). By generating and inserting activation vectors into the model’s forward pass, we can steer the model towards desired behaviors.

However, a problem with current steering methods is their unpredictability – it’s often unclear exactly how a steering vector will affect model behavior. Steering vectors may not produce the intended changes in the model’s output or may cause unforeseen behaviors, as we discuss in Section 3. In some cases, steering interventions produce no interpretable changes in model behavior other than model degradation, as we show in Section 5. This unpredictability makes it difficult to precisely control the model.

To address these challenges, we develop a method for quantifying the effects of a steering intervention on model outputs. We use Sparse Autoencoders (SAEs; Ng [2011], Cunningham et al. [2023], Bricken et al. [2023]) to measure the change in feature activations caused by steering interventions. This lets us understand what change in behavior to expect from any particular steering intervention. Note: our analysis is *not* about how early SAEs impact later layer SAEs (as studied

---

\*Equal contribution

<sup>†</sup>Code available at <https://github.com/slavachalnev/SAE-TS>

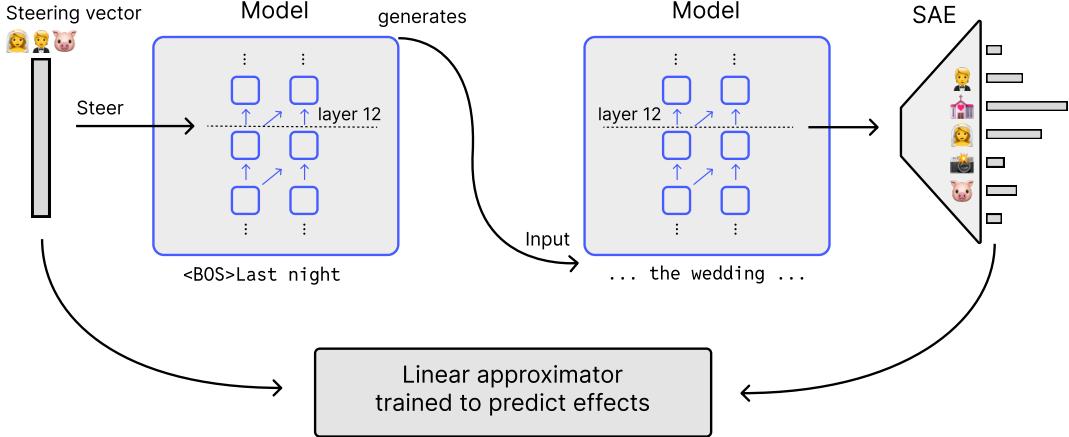


Figure 1: Diagram showing the feature effects measurement method. We sample a steering vector (depicted here as 🧑‍🤝‍🧑 to emphasise that it represents a mixture of concepts), and steer the model by adding it to the residual stream activations at layer 12. We use this steered model to generate text completions starting from a prompt (e.g. "<BOS>Last night"). These completions are then fed back through the model up to layer 12, where the activations are passed through a Sparse Autoencoder (SAE) and averaged to measure feature effects. The linear approximator is trained on pairs of steering vectors and their measured feature effects to predict effects for new steering vectors.

by prior work such as Marks et al. [2024] and Anders and Bloom [2024]). Instead, we look at how rollouts generated by a steered model differ from those generated by the unsteered, base model.

Building on this feature effects measurement method, we introduce **SAE-Targeted Steering** (SAE-TS), a method that constructs steering vectors to specifically target desired SAE features while minimizing unintended side effects. SAE-TS involves learning a linear relationship between steering vectors and their effects on SAE features. This allows us to construct steering vectors using the interpretable features found by the SAE. Notably, SAE-TS uses the SAE for steering differently to prior work such as Templeton et al. [2024] and Durmus et al. [2024]. We use the SAE to measure the effects of steering vectors, rather than directly adding SAE latents to model forward passes.

We evaluate SAE-TS against existing methods such as Contrastive Activation Addition (CAA) [Panickssery et al., 2024] and direct SAE feature steering Templeton et al. [2024]. Our evaluations demonstrate that SAE-TS outperforms existing methods, achieving better alignment with the intended behavior while maintaining the semantic coherence of the generated text across various tasks.

### Key Contributions:

1. We develop a method to quantify and interpret the effects of a steering intervention using SAEs (Section 3).
2. We introduce SAE-Targeted Steering (SAE-TS), a method that constructs steering vectors to achieve specific desired effects while minimizing unintended changes (Section 4).
3. We evaluate SAE-TS on a set of steering tasks (Section 5), showing that it outperforms existing methods, successfully steering the model while maintaining output quality.

## 2 Related work

**Activation Steering** is a method for controlling model outputs introduced in Turner et al. [2024] where steering vectors are extracted by taking the difference between activations from pairs of contrasting positive and negative prompts. Panickssery et al. [2024] scaled this method to Llama-2, and Zou et al. [2023] use the methodology on a somewhat wider range of tasks. Recently, Cao et al. [2024] offers a way to learn a steering vector to optimally steer a model toward producing desired outputs. As with the activation steering methods above, a dataset of positive and negative examples is required for this method to work. Lee et al. [2024] introduce conditional steering. Scherlis et al. [2024] also studies the causal limitations of existing steering methods.

**Mechanistic Interpretability and SAEs.** Mechanistic Interpretability aims to understand how LLMs function internally by breaking down the models into understandable components [Olah et al., 2020, Olah]. The recent development of **sparse autoencoders** [Ng, 2011, Bricken et al., 2023, Cunningham et al., 2023] gives us a way to decompose the residual stream of transformer models into sparse and often human understandable features. Sparse autoencoders

provide evidence supporting the hypothesis that the latent space of LLMs is composed of linear and interpretable directions [Arora et al., 2018, Elhage et al., 2022]. Templeton et al. [2024] and Zhao et al. [2024] demonstrate that these directions found by the SAEs can be used for steering.

### 3 Measuring Steering Effects

As discussed in the introduction (Section 1), it can be difficult to predict the behavior of steering vectors. Measuring the effects of a steering intervention is useful for both interpreting the causal role of features and designing more effective steering vectors.

In transformer models, we can represent the forward pass as  $h_n = b_n(h_{n-1})$ , where  $b_i$  denotes the  $i^{\text{th}}$  block and  $h_i$  is the hidden state after layer  $i$ . When we insert a steering vector  $v$  at a specific layer  $l$ , we modify the hidden state as  $h_l \leftarrow h_l + \alpha v$ , where  $\alpha$  is a scaling coefficient that determines the norm of the added steering vector.

We use JumpReLU SAEs [Rajamanoharan et al., 2024], which consist of an encoder  $f$  and a decoder, where the reconstruction  $\hat{x}$  of the input  $x$  is given by:

$$\hat{x} = f(x)W_{\text{dec}} + b_{\text{dec}}, \quad (1)$$

and the SAE is trained such that  $\hat{x}$  approximates  $x$ . SAE steering involves using the decoder vector  $d_i$  of feature  $i$ , which is the  $i^{\text{th}}$  row of  $W_{\text{dec}}$ , as the steering vector.

In order to measure a steering intervention’s effect on the model output, we look at the difference in SAE feature activations between steered model outputs and original, unsteered model outputs. Our method is outlined at a high level in Figure 1 and can be summarised as follows:

1. **Data Generation:** Generate text rollout datasets  $D_{\text{steered}}$  and  $D_{\text{unsteered}}$  using steered and unsteered models respectively, where the steered model has a steering vector  $v$  inserted at layer  $l$ .
2. **Feature Extraction:** For each generated output, pass the text back through the model up to layer  $l$ , and use the SAE encoder  $f$  to extract the feature activations at each position.
3. **Effect Computation:** Compute the average feature activations across all outputs and positions for both the unsteered and steered models, and calculate the difference to obtain the *steering effects* vector  $y \in \mathbb{R}^{d_{\text{SAE}}}$ :

$$y = \mathbb{E}_{\text{steered}}[f(x)] - \mathbb{E}_{\text{unsteered}}[f(x)]. \quad (2)$$

We use the Gemma-2-2B model [Gemma Team, 2024] and a Gemma Scope layer 12 residual stream SAE with hidden dimension of 16,384 [Lieberum et al., 2024]. We run the above procedure for every feature of the SAE, generating 896 rollouts of length 32, and computing the average difference between the steered and unsteered feature activations.

Table 1 shows a cherry-picked example of a feature which, when used for steering, has feature effects that are not itself. This feature (id 12904) activates on months (e.g. "January") but steering with its decoder vector causes the model to output digits of years. Upon closer inspection we see that the feature only activates on months following a day (e.g. 1st of January), so it is unsurprising that steering with this feature causes the model to output years. Some of the largest feature effects are features which are not clearly interpretable but are instead commonly occurring. E.g. in Table 1 the top effect is feature id 6810 with an activation difference more than 3 times higher than the next highest activation difference.

Given our initial motivation to use steering effects to develop better causal explanations of a feature’s *role* in the model, features such as the month feature, which only have significant effects on features other than themselves, are of particular interest. When inspecting these features, we find that the effects encode many reasonable grammatical and semantic relationships. Another example is a feature that activates on "say", but increases a feature that activates on "hello" (see Appendix H).

### 4 Targeted Steering

We can use the above method of measuring steering effects to find steering vectors which steer the model towards a desired feature effect, while having minimal side-effects.

We do this by first training a **linear effect approximator** function (see Section 4.1) to predict the feature effects for any given steering vector and then use this function to find targeted steering vectors to achieve a desired effect.

Act Diff	Feature id	Feature description
4.5819	6810	No clear pattern. Commonly occurring feature with 45% activation density
1.3763	8641	The number two.
1.3080	8223	Space before a number.
0.9797	8233	Last digit of a year.
0.9769	10356	Third digit of a 2000s year e.g. 2015.
0.7754	847	The zero digit.
0.7542	14914	The digit following a two.
0.7470	7517	No clear pattern. Commonly occurring feature with 22% activation density
0.7069	2381	Zero following a two.
0.6341	8258	Second digit of a year, mostly 9.

Table 1: Table showing activations, feature ids, and feature descriptions of the SAE feature steering vector (ft id 12904) in Gemma-2-2b. This feature activates on months following a day (e.g. "1st of **January**").

The effect approximator is a linear function  $\hat{\mathbf{y}} = \mathbf{x}\mathbf{M} + \mathbf{b}$ , where  $\mathbf{x}$  is the  $d_{\text{model}}$ -dimensional steering vector,  $\mathbf{M}$  is a  $d_{\text{model}} \times d_{\text{sae}}$  matrix, and  $\mathbf{b}$  has dimension  $d_{\text{sae}}$ . We train it to minimize the mean squared error (MSE) between its prediction  $\hat{\mathbf{y}}$  and the observed effect vector  $\mathbf{y}$ .

Once we have trained the effect approximator, we use it to find a steering vector that increases the activation of a target feature  $j$ , while keeping other features unchanged. To achieve this, we construct the targeted steering vector  $\mathbf{s}$  as follows:

$$\mathbf{s} = \frac{\mathbf{M}_j}{\|\mathbf{M}_j\|} - \lambda \frac{\mathbf{Mb}}{\|\mathbf{Mb}\|} \quad (3)$$

We then normalize  $\mathbf{s}$  to unit norm. In all our experiments, we set  $\lambda = 1$ . See Appendix A for further discussion on why we do this instead of solving for  $\mathbf{x}$  by computing the pseudoinverse of  $\mathbf{M}$ .

#### 4.1 Training the Effect Approximator

To construct the effect approximator, we collect a dataset of 50,000 steering vectors and their corresponding steering effects. The steering vectors are the decoder vectors from a larger SAE trained at the same layer, which in the case of Gemma-2-2b is the layer 12 65k SAE with an L0 of 72. This provides us with a diverse set of steering vectors, letting us observe a wide range of steering effects.

For each steering vector  $\mathbf{x}$  in the dataset, we measure its effect  $\mathbf{y}$  on the model by computing the difference in SAE feature activations between the steered and unsteered model outputs, as described in Section 3. We then train the linear effect approximator  $\hat{\mathbf{y}} = \mathbf{x}\mathbf{M} + \mathbf{b}$  using Adam to minimize the MSE between  $\hat{\mathbf{y}}$  and  $\mathbf{y}$ .

By learning the mapping from steering vectors to their effects on SAE features, the effect approximator allows us to predict the change in feature activations for any given steering vector. This enables us to design steering vectors that produce precise and predictable changes in the model's behavior.

#### 4.2 Scaling Factor Selection

Selecting an appropriate scaling factor  $\alpha$  when adding the steering vector  $\mathbf{x}$  is important for getting good steering results. The model is sensitive to different directions to varying degrees; some steering vectors may have a significant impact even with a small scaling factor, while others require a larger scaling factor to produce noticeable effects. To compensate for this variability, we need an automatic method to adjust the scaling factor for each steering vector individually.

We do this by finding a scaling factor  $\alpha$  for each steering vector such that the steered model's cross-entropy loss goes up by 0.5 above the unsteered baseline. By applying this scaling factor selection process to all steering vectors in our dataset, we ensure that the collected steering effects accurately reflect the model's response within the optimal range of steering intensities.

## 5 Evaluations

To evaluate our steering vectors, we use gpt-4o-mini to rate two aspects of the generated text on a scale of 1 to 10:

- **Behavioral score** Assesses whether the steering target was met, based on task-specific criteria.
- **Coherence score** Evaluates whether the steering intervention maintains the model’s general capabilities and produces semantically correct text.

The ratings are averaged and normalised to give scores between 0 and 1. We multiply these two scores to obtain the **Behavioral\*Coherence score**, which is the main metric we use for measuring steering quality. This way, our evaluations require both achieving the desired steering effect and maintaining the overall quality of the generated text.

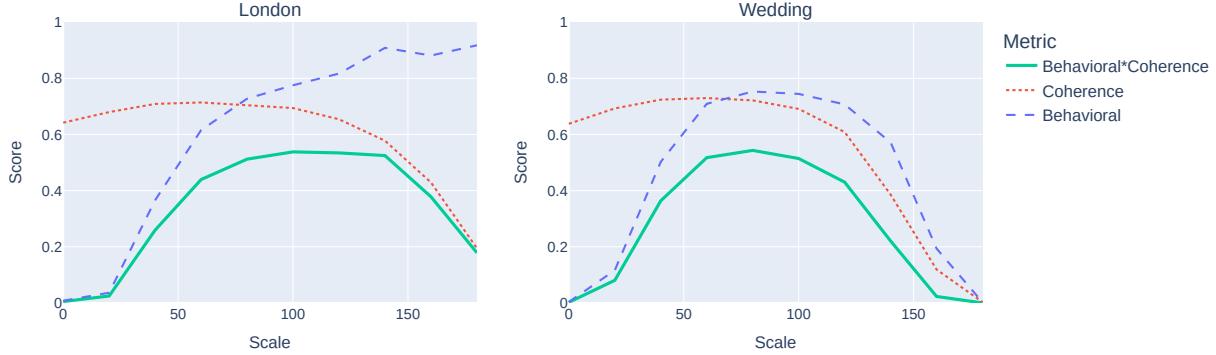


Figure 2: Plots showing Behavioral, Coherence, and Behavioral\*Coherence scores for the London and Wedding tasks at varying steering scales.

We focus on steering a base language model (Gemma-2-2b) and evaluate on open-ended generation starting from the prompt "<BOS>I think". The task is to steer the model so that it discusses a pre-specified topic, while remaining coherent. We selected 9 topics, most of which appear in prior steering work. We steer the model by adding the steering vector at some scale  $\alpha$  to the residual stream at every token position. We sample 256 steered text completions, each 32 tokens long.

Steering is sensitive to the norm of the steering vector so we vary the scale  $\alpha$  and plot the three scores for our SAE-TS steering method as shown in Figure 2. Starting from scale  $\alpha = 0$  (the unsteered model), the Coherence score is 0.64 while the Behavioral score is 0, making the Behavioral\*Coherence score 0. As we increase the scale, the Behavioral score either keeps rising to more than 0.9 in the case of the "London" task, or peaks at around 0.75 and then goes back down as is the case in the "Wedding" task. Interestingly, the Coherence score first rises slightly to a peak of around 0.71 and then drops off. This rising Coherence score phenomenon is specific to the SAE-TS method and we explore it further in Appendix A.3.

We benchmark three steering methods, comparing their Behavioral\*Coherence scores across steering vector scales:

- **Contrastive Activation Addition (CAA)**, which we define as the mean difference of model activations between a set of positive and negative prompts, averaged over token positions and prompts. This is similar but not identical to Panickssery et al. [2024]’s usage.
- **SAE feature steering**, using the decoder vector of the relevant SAE feature.
- **SAE targeted steering (SAE-TS)**, as described in Section 4.

Figure 3 shows the performance of these methods across the 9 steering tasks. The  $x$ -axis represents the scaling factor  $\alpha$  while the  $y$ -axis is Behavioral\*Coherence score. Looking at the peak of the score (how good is it when the appropriate scale is chosen), displayed in Table 2, we see that the our SAE-TS method outperforms the baseline methods on 7 of the 9 tasks. The two exceptions are the "Christian" task, where CAA is the best method, and the "Conspiracy" task, where CAA is tied with SAE-TS.

On the "London" task, CAA and SAE steering perform unusually poorly, while SAE-TS performs very well. Even at high steering scales, CAA and SAE steering do not produce mentions of London. Although SAE-TS is significantly better than the other two methods for this task, we do observe an interesting failure mode of SAE-TS at high scaling factors ( $\alpha > 160$ , well above optimal scale). At these scales, the output diversity collapses and the model begins

to produce text focused entirely on fashion shows and art exhibitions in London, as we show in example outputs in Appendix I.

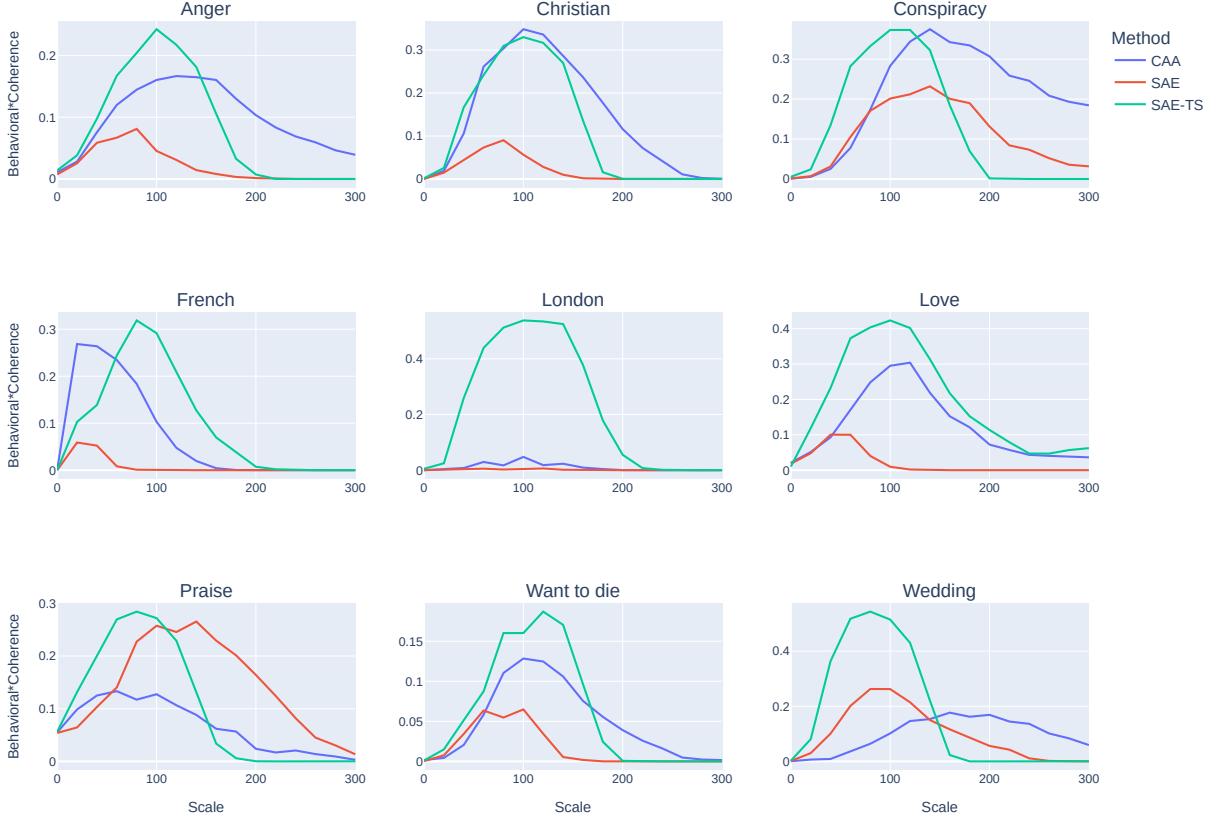


Figure 3: Plots showing Behavioral\*Coherence score for the CAA, SAE, and SAE-TS steering methods on all 9 tasks. We see that SAE-TS is superior to the other two methods on 7 of the 9 tasks across a wide range of steering scales.

While we primarily focus on Gemma-2-2B in this paper, we also validate our methods on Gemma-2-9B and present these results in Appendix C. We observe broadly similar trends, with SAE-TS outperforming the baseline methods on average, though we see that CAA sometimes performs better on the larger model.

Steering Goal	ActSteer	SAE	SAE-TS (ours)
Anger	0.1666	0.0810	<b>0.2424</b>
Christian	<b>0.3486</b>	0.0902	0.3302
Conspiracy	<b>0.3748</b>	0.2319	0.3729
French	0.2684	0.0589	<b>0.3186</b>
London	0.0476	0.0061	<b>0.5380</b>
Love	0.3039	0.1004	<b>0.4234</b>
Praise	0.1332	0.2654	<b>0.2842</b>
Want to die	0.1284	0.0649	<b>0.1870</b>
Wedding	0.1768	0.2626	<b>0.5432</b>
Average	0.2165	0.1290	<b>0.3600</b>

Table 2: Table showing maximum steering scores.

## 6 Feature Effects Visualization

In this section, we introduce EffectVis, an interactive interface and tool built for exploring feature effects. On the top left panel, users can search for features and view their feature visualization charts – similar to Neuronpedia [Lin and Bloom, 2023]. Features can be added as feature cards to the canvas when you want to inspect them further.

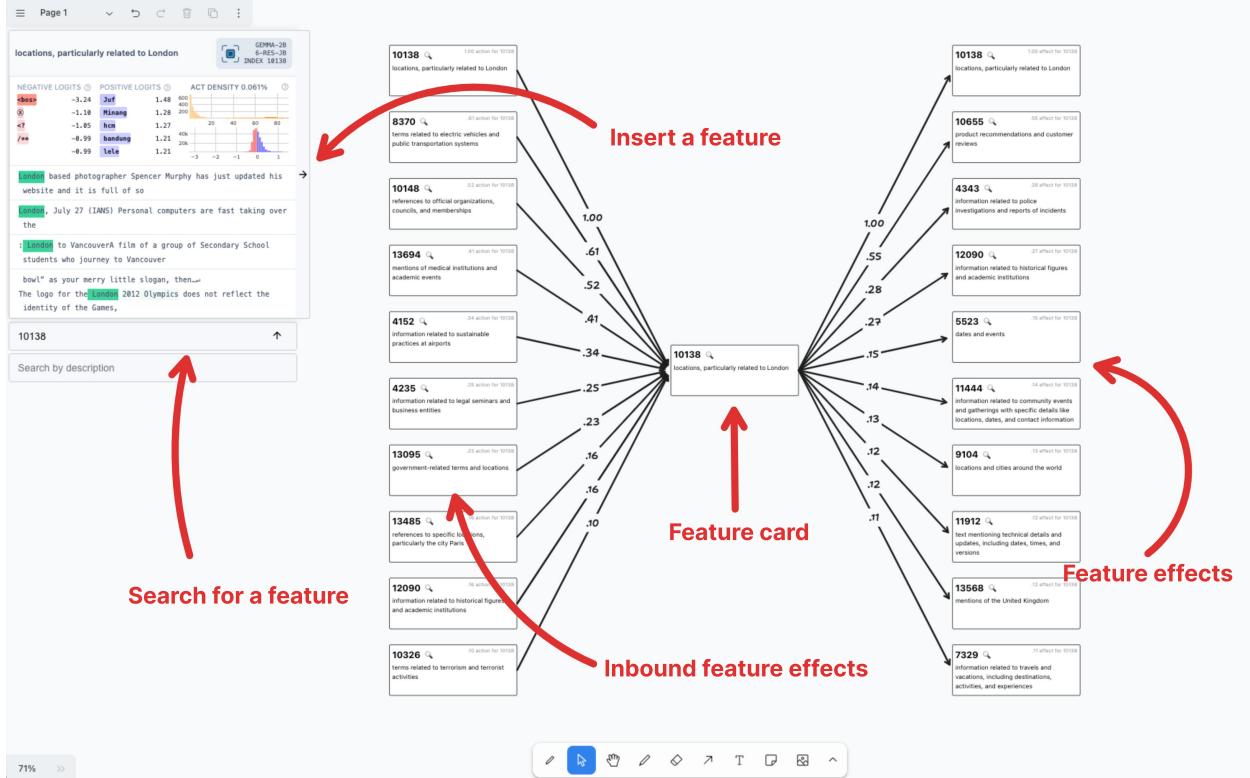


Figure 4: Interface overview for EffectVis. Available at <https://effectvis.vercel.app/>

For a selected card, our tool allows you to query for three different properties of the feature:

- The feature effects.
- The feature actions. These are the features that cause the selected feature to activate more often.
- Cosine similar feature directions.

In addition, when a feature card is selected, other cards for the same feature are highlighted, making it easy to see here else a feature is used at a glance. By clicking the magnifying icon on the top right of a feature card, the top panel will be populated with the relevant information for that feature.

Prior to developing EffectVis, we had a multi-step process for exploring and making sense a feature's effects and/or cosine similarity. First, we would access the list of effects for a feature in a python notebook. Then, using Neuronpedia, we would search for each feature one by one. These were then added to a list on Neuronpedia.

Due to the time-consuming nature of the process, we were limited in how many features we could inspect, reducing our ability to build intuition for what was being measured with the feature effects. EffectVis solved this for us by compressing the above steps into a single keyboard command action. It also made comparisons between feature effects and/or decoder cosine similarity easier. Lists of features could be pulled up side-by-side and compared visually. For comparing many lists, selecting a card shows you the other lists it is a part of.

EffectVis helped us develop the insights that led to our Optimized Steering Vectors method. In the future, it may be useful to incorporate functionality for querying our feature effects by different filters such as effect scores as we did for part of our analysis in the feature effects section. This would allow us to perform more experiments and ask a wider variety of questions using the interface that we resorted to using python notebooks to explore. We haven't rigorously compared EffectVis to alternative tools, as it is tailored for the standalone method that we developed.

## 7 Discussion

### 7.1 Limitations

We performed all of our investigations using Gemma-2 2B and 9B, both of these models share the same distilled model architecture, so our results may not transfer to other models. Additionally, these are base models which haven't been instruction tuned, so we focused on steering in the open-ended generation setting and did not investigate tasks and features which would be relevant for safety or capabilities.

We focused only on steering towards concepts present in SAEs, which is a problem for multiple reasons. Firstly, some concepts don't appear as individual SAE features, a phenomenon called feature splitting. Secondly, niche features may only appear in extremely large SAEs, as discussed in the "feature completeness" section of Templeton et al. [2024]. These problems are particularly relevant because our targeted steering method, as described in Section 4, can only steer towards features present in the SAE used to measure feature effects. We explore a potential solution in Appendix D.

### 7.2 Future work

An important direction for future work is to apply our steering methods to chat models and safety-relevant steering targets to evaluate effectiveness in practical scenarios. Additionally, this work can be extended by exploring different SAE architectures, such as TopK SAEs [Gao et al., 2024], and testing on various model architectures beyond the Gemma family. Another direction to explore is adapting BiPO [Cao et al., 2024] to optimize steering vectors towards desired feature effects.

## Acknowledgements

This research was conducted as part of the MATS 6.0 program. We would like to thank Yeu-Tong Lau and Kai Fronsdal for valuable discussions that helped develop the SAE-TS method. We would also like to thank the GitHub user <https://github.com/spfau1> for finding a bug in our CAA implementation which we have fixed and regenerated results for, in this version of the paper.

## References

- Evan Anders and Joseph Bloom. Examining language model performance with reconstructed activations using sparse autoencoders. <https://www.lesswrong.com/posts/8QRH8wKcnKGhpAu2o/examining-language-model-performance-with-reconstructed>, 2024.
- Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018. doi: 10.1162/tacl\_a\_00034. URL <https://aclanthology.org/Q18-1034>.
- 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>.
- Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization, 2024. URL <https://arxiv.org/abs/2406.00045>.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL <https://arxiv.org/abs/2210.11416>.
- Hoagy Cunningham, Aidan Ewart, Logan Riggs, et al. Sparse autoencoders find highly interpretable features in language models, 2023. URL <https://arxiv.org/abs/2309.08600>.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.

- Esin Durmus, Alex Tamkin, Jack Clark, Jerry Wei, Jonathan Marcus, Joshua Batson, Kunal Handa, Liane Lovitt, Meg Tong, Miles McCain, Oliver Rausch, Saffron Huang, Sam Bowman, Stuart Ritchie, Tom Hennighan, and Deep Ganguli. Evaluating feature steering: A case study in mitigating social biases, 2024. URL <https://anthropic.com/research/evaluating-feature-steering>.
- Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.
- 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, 2024. URL <https://arxiv.org/abs/2406.04093>.
- Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL <https://arxiv.org/abs/2408.00118>.
- Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering, 2024. URL <https://arxiv.org/abs/2409.05907>.
- Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. URL <https://arxiv.org/abs/2408.05147>.
- Johnny Lin and Joseph Bloom. Neuronpedia: Interactive reference and tooling for analyzing neural networks, 2023. URL <https://www.neuronpedia.org>. Software available from neuronpedia.org.
- Samuel Marks, Can Rager, Eric J. Michaud, et al. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *Computing Research Repository*, arXiv:2403.19647, 2024. URL <https://arxiv.org/abs/2403.19647>.
- Andrew Ng. Sparse autoencoder. *CS294A Lecture Notes*, 2011. Unpublished lecture notes.
- Chris Olah. Mechanistic interpretability, variables, and the importance of interpretable bases. <https://www.transformer-circuits.pub/2022/mech-interp-essay>.
- Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL <https://arxiv.org/abs/2203.02155>.
- Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering Llama-2 via Contrastive Activation Addition, 2024. URL <https://arxiv.org/abs/2312.06681>.
- Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders, 2024. URL <https://arxiv.org/abs/2407.14435>.
- Adam Scherlis, Thomas Marshall, and Nora Belrose. Forthcoming work on the need for causal steering vectors, 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>.
- The White House. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, 10 2023. URL <https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/>. Accessed: 2024-10-16.
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization, 2024. URL <https://arxiv.org/abs/2308.10248>.
- Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training llms to prioritize privileged instructions, 2024. URL <https://arxiv.org/abs/2404.13208>.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail?, 2023. URL <https://arxiv.org/abs/2307.02483>.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. Steering knowledge selection behaviours in llms via sae-based representation engineering, 2024. URL <https://arxiv.org/abs/2410.15999>.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2023. URL <https://arxiv.org/abs/2310.01405>.

## Appendix

### A Targeted Steering Method Justification

#### A.1 Empirical justification

As described in Section 4, our targeted steering method aims to find a steering vector  $s$  that increases activation of a target feature while minimizing unintended effects. Given our effect approximator function  $\hat{y} = \mathbf{x}\mathbf{M} + \mathbf{b}$ , an obvious approach would be to directly solve for the optimal steering vector  $\mathbf{x}$  by computing the pseudoinverse of  $\mathbf{M}$ . However, empirically we found that our method (SAE-TS), which takes the normalized target feature column of  $\mathbf{M}$  and subtracts the normalized bias effects, significantly outperforms the pseudoinverse approach, as shown in Table 3.

Steering Goal	SAE-TS	Pseudoinverse
Anger	<b>0.2424</b>	0.1899
Christian	<b>0.3302</b>	0.0025
Conspiracy	<b>0.3729</b>	0.0644
French	<b>0.3186</b>	0.1912
London	<b>0.5380</b>	0.0722
Love	<b>0.4234</b>	0.2682
Praise	<b>0.2842</b>	0.0638
Want to die	<b>0.1870</b>	0.1683
Wedding	<b>0.5432</b>	0.0567

Table 3: Table showing maximum steering scores for our SAE-TS method vs computing the optimal steering vector using the pseudoinverse.

#### A.2 Why do this work? Intuition

If we ignore the bias term, using the normalized  $\mathbf{M}_j$  is equivalent to linear regression and gives the steering direction that maximizes predicted effect on feature  $j$  subject to the unit norm constraint, while ignoring effects on other features. Why does ignoring predicted effects on other features produce good results?

An important part of the story is the data on which the effect approximator was trained. The steering vectors used to generate the data were added at a scale  $\alpha$  chosen to increase the model’s loss by 0.5. This scaling encodes information about the model’s sensitivity to different directions - directions that strongly affect many features require a smaller  $\alpha$  to reach the target loss increase.

Then, when training the effect approximator, the steering vectors are normalised to have norm 1. As a result,  $\mathbf{M}_j$  implicitly accounts for effects on other features. A steering direction that strongly affects other features will show up in the training data with smaller effect magnitudes due to the smaller  $\alpha$  used.

#### A.3 Effect of the bias term

The effect approximator’s bias vector  $\mathbf{b}$  has a few features with large values while most are close to 0. The high-bias features include three that activate on the <BOS> token, an induction feature that activates on repeated patterns, and a position feature that activates on all token positions  $> 2$ . All the largest features are densely activating, as shown in Table 4.

We suspect that the initial Coherence bump that we observe in Section 5 is due to the bias term ( $-\mathbf{Mb}$ ) in Equation (3). We can see this in Figure 5 where we plot the Coherence score when steering with just  $-\mathbf{Mb}$  against steering with random normal vectors.

One possible reason for the coherence bump could be due to some “incoherent text” features which on average activate more when we steer the model as the model is degraded. Then, when we subtract the bias term, the model produces more coherent outputs.

We notice some quirks when steering with the bias term – notably, it seems to increase the frequency of proper nouns in the generated text, with almost every rollout containing a proper noun at scale 100. This raises the possibility that the apparent increase in coherence might be an artifact of our evaluation setup, where the presence of proper nouns causes gpt4o-mini to assign higher coherence scores rather than reflecting genuinely more coherent text. Example rollouts can be found in Appendix I.4.

Value	Feature id	Feature description	Activation density
-0.9743	1041	Very large activations on the <BOS> token.	42%
0.8529	7507	Large activations on the <BOS> token.	48%
-0.6471	2291	No clear pattern.	62%
-0.4107	12342	Induction feature.	27%
-0.4065	7541	No clear pattern.	42%
-0.3745	1322	No clear pattern.	14%
0.3522	2620	All token positions > 2.	98%
0.3368	6810	Mostly copyright licenses.	45%
0.3116	2514	<BOS> tokens	1%

Table 4: Top bias features.

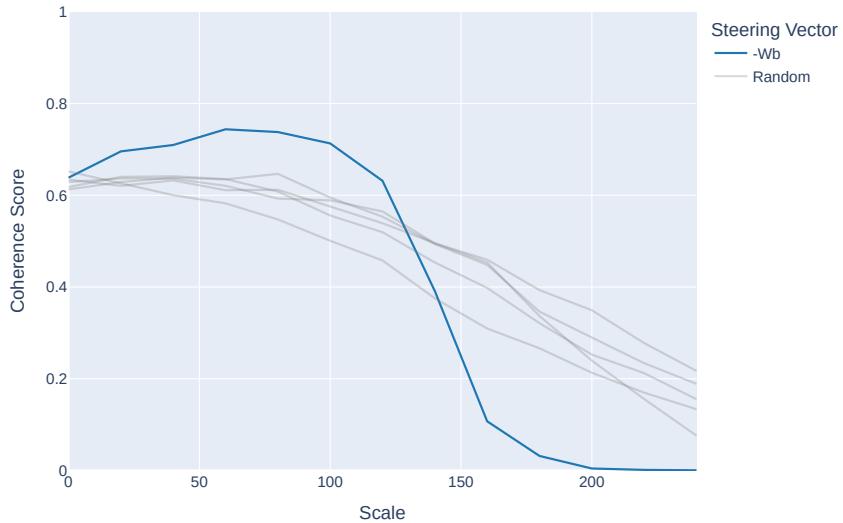


Figure 5: Coherence score when steering with the bias component of SAE-TS, compared to steering with random vectors. Steering with random vectors behaves as expected: Coherence score stays flat at first, then goes down gradually. However, when we steer with the bias term ( $-\mathbf{M}\mathbf{b}$ ), Coherence score rises, peaking at 0.72 at scale 60, then drops off rapidly.

## B Example Evaluation Criteria

We provide an example of our evaluation setup for the Wedding task:

### Behavioral score criterion:

The text contains discussion about a wedding

### Coherence score criterion:

The text is coherent and the grammar is correct.

**Evaluation Setup:** We use gpt4o-mini with the following prompts:

System prompt

```
You score texts generated by a language model based on the following criterion:  
[criterion].  
You provide a score from 1 to 10. \  
The language model was given a prompt and generated the following text. \  
Evaluate the text based on the criterion. Output format should be \  
JSON with the following fields: "score" (int)"
```

Message content

Prompt:

[prompt]

Completion:

"""

[completion]

"""

## C Gemma-2-9B Results

We perform the same evaluations as in Section 5 for the gemma-2-9B model and plot the results in Figure 6 and Table 5. We use the Gemmascope layer 12 residual stream SAE with width 16k and an L0 of 130. It was easy to find relevant features for all the tasks except for the London task. There is no feature in this particular SAE which fires only on London, thus we use a mix of the following three features: **6915** - large cities including London, **9983** - cities, **11748** - United Kingdom. We see that SAE-TS achieves higher average Behavioral\*Coherence score than either CAA or SAE steering. However, CAA beats SAE-TS in 4 of the 9 tasks and ties in one of them.

Steering Goal	CAA	SAE	SAE-TS (ours)
Anger	<b>0.2336</b>	0.1317	0.2310
Christian	<b>0.3162</b>	0.1548	0.2769
Conspiracy	<b>0.4343</b>	0.3003	0.3315
French	0.3203	0.3272	<b>0.4196</b>
London	0.0656	0.1118	<b>0.3135</b>
Love	<b>0.3373</b>	0.1160	0.2710
Praise	0.1965	0.3743	<b>0.5188</b>
Want to die	0.1521	<b>0.1646</b>	0.1265
Wedding	0.2216	0.2851	<b>0.5494</b>
Average	0.2531	0.2184	<b>0.3376</b>

Table 5: Gemma-2-9B maximum Behavioral\*Coherence scores.

## D Rotation Steering

Our SAE-TS method requires measuring feature effects for each feature we want to target. However, some features might be too rare to appear in our measurement dataset, or might not exist in the SAE we use for measurement. To address this, we develop rotation steering, a method that learns a transformation  $f : \mathbb{R}^{d_{\text{model}}} \rightarrow \mathbb{R}^{d_{\text{model}}}$  between SAE decoder vectors and effective steering vectors.

We compute this transformation by first calculating the correlation matrix  $C = MW_{\text{dec}}$ . We then perform SVD on this correlation matrix:  $C = U\Sigma V^T$  and then the rotation matrix  $R = UV^T$  gives us a transformation between the model's activation spaces. Next, similarly to the process described in Section 4, we generate a steering vector for any SAE feature  $i$  by taking its decoder vector  $d_i$  (the  $i$ th row of  $W_{\text{dec}}$ ) and apply the learned rotation to get  $v = d_i R$ , we then normalise  $v$  and subtract (normalised)  $Mb$ .

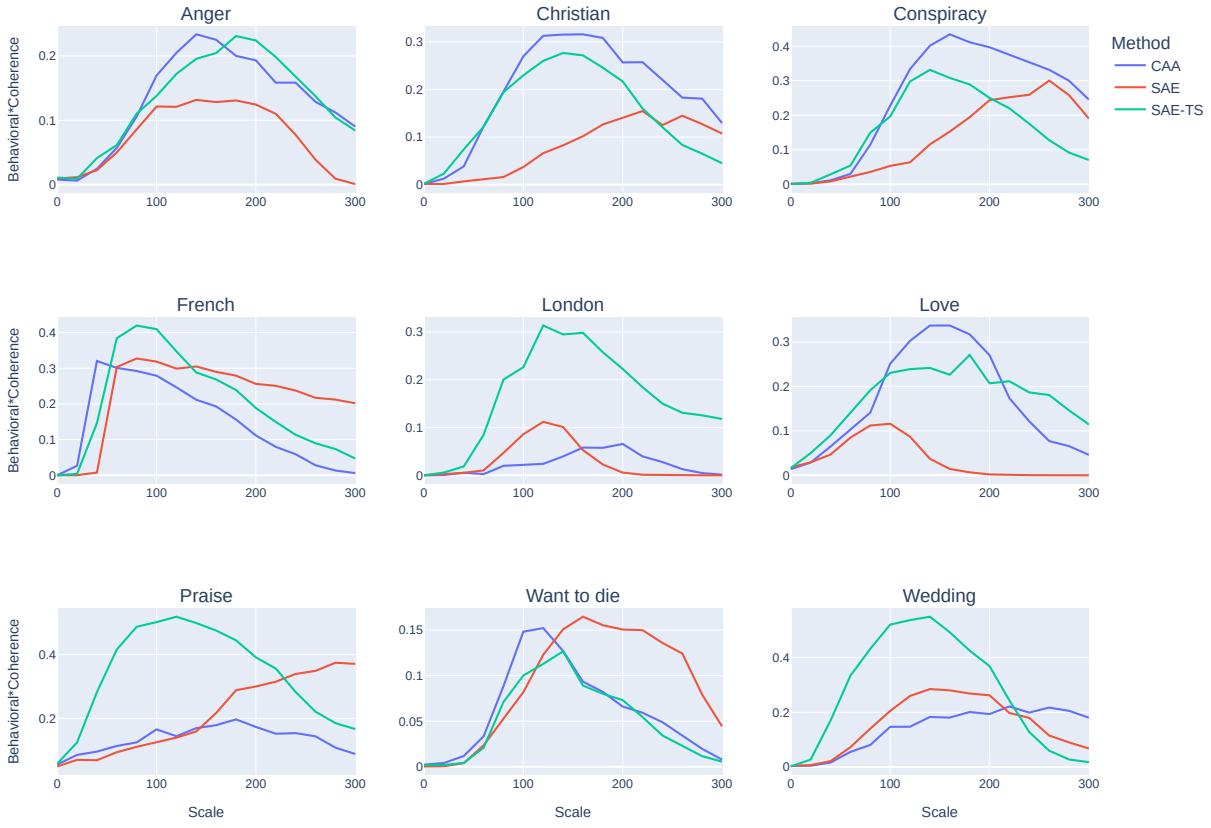


Figure 6: Gemma-2-9B plots showing Behavioral\*Coherence score for the three methods over varying steering scales for all tasks.

This approach allows us to generate steering vectors for features not present in our original feature effects dataset, as long as we can find a relevant feature in any SAE trained on the same model layer.

In Table 6 we see that rotation steering performs reasonably well but is on average worse than SAE-TS, at least for the specific tasks we test on.

Steering Goal	CAA	SAE	SAE-TS	PinverseSteer	RotationSteer
Anger	0.1666	0.0810	0.2424	0.1899	<b>0.2460</b>
Christian	<b>0.3486</b>	0.0902	0.3302	0.0025	0.0870
Conspiracy	0.3748	0.2319	0.3729	0.0644	<b>0.4074</b>
French	0.2684	0.0589	0.3186	0.1912	<b>0.3724</b>
London	0.0476	0.0061	<b>0.5380</b>	0.0722	0.2745
Love	0.3039	0.1004	<b>0.4234</b>	0.2682	0.3851
Praise	0.1332	0.2654	<b>0.2842</b>	0.0638	0.2833
Want to die	0.1284	0.0649	0.1870	0.1683	<b>0.2167</b>
Wedding	0.1768	0.2626	<b>0.5432</b>	0.0567	0.5313
Average	0.2165	0.1290	<b>0.3600</b>	0.1197	0.3115

Table 6: Table showing maximum Behavioral\*Coherence scores for various methods, including the three methods described in Section 5, as well as finding the optimal steering vector by taking the pseudoinverse of  $M$ , and rotation steering as described in Appendix D.

## E Finding related groups of features

We find that by looking at features with similar feature effects, we can find groups of features that are thematically related – more so than we can when looking at cosine similar features along their decoder vectors.

Process for curating the below example wedding feature group: Starting with the wedding feature, we recursively look at features with similar feature effects, adding the ones that we qualitatively believe fit into the feature group. We then repeat this process looking at features with similar decoder vectors. The venn diagram below compares the set of features that we could find looking at the SAE decoder vector vs. the feature effects.

This suggests that the SAE may not project features with similar effects into similar decoder directions. A research question to explore in the future would be to compare SAE decoder vectors from features in the same feature group to see if there are shared dimensions that encode important properties.

Cosine similarity	Feature effects	Both
Parents Siblings	Wedding invitation Bride and groom Ceremony Familial relationships Photography Venue Party Marriage act Romantic In-laws Newly-wed	Wedding Marriage Wife Husband Relationship

Table 7: Wedding features found by inspecting feature effect vs. SAE decoder vector similarities

Note: We performed this experiment on the feature effects computed for all features in an SAE trained on activations from the 2B parameter Gemma 1 model early on in our research. However, we suspect the results here generalize to other LLM variants, as long as the feature effects map appropriately to observed model behaviors.

## F Different Prompt

Unless otherwise specified, all the steering score analysis in this paper was done with "<BOS>I think" as the starting prompt from which completions are generated. To verify that our results do not depend on this particular prompt, we repeat the evaluation experiments the "<BOS>Surprisingly," prompt (Figure 7). We see that the results are qualitatively similar to the "<BOS>I think" prompt.

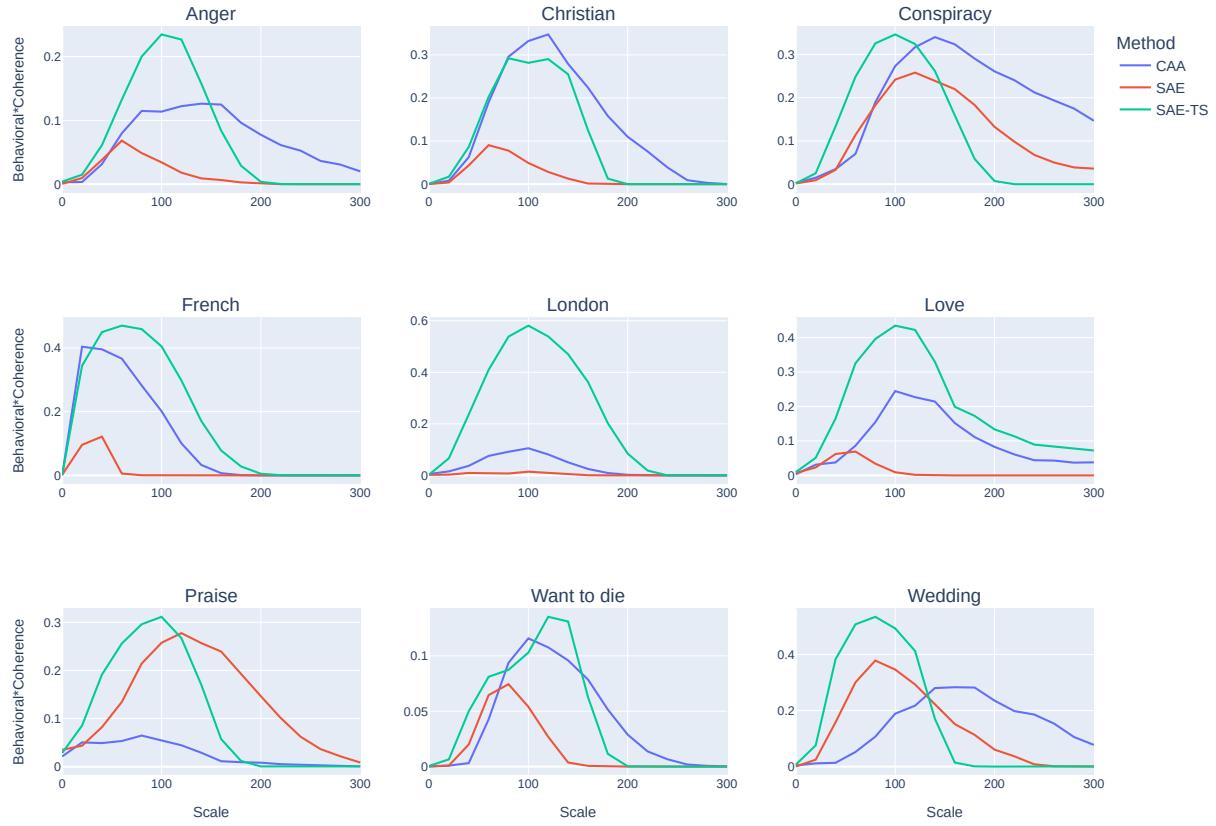


Figure 7: Plots showing evaluation scores for Gemma-2-2B starting from the prompt "<BOS>Surprisingly,".

## G Score Trade-off Curves

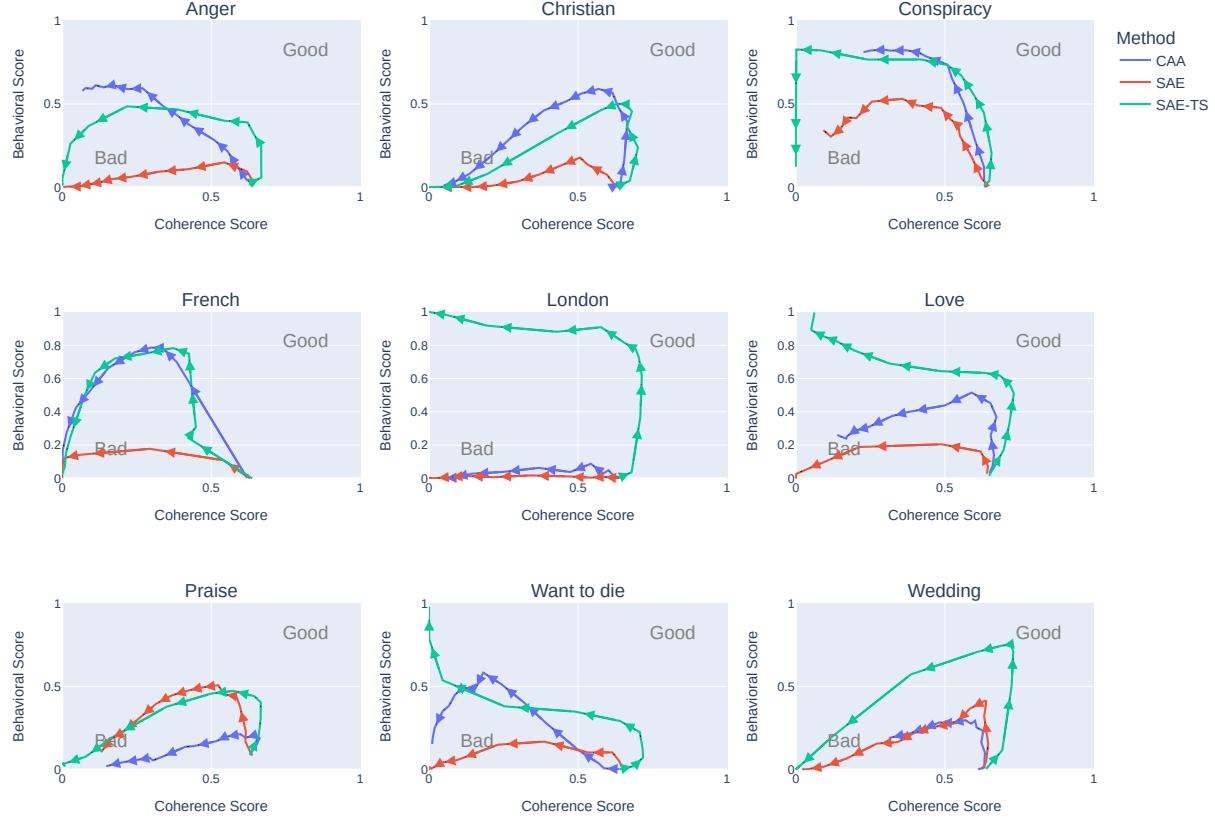


Figure 8: Score trade-off curves. Each plot shows how Coherence score (x-axis) and Behavioral score (y-axis) change as the steering scale increases, with each line representing a different steering method. Points further to the right and top indicate better performance on both metrics.

## H Feature Effect Examples

These are additional examples of feature effects that we find for various steering vectors extracted from Gemma-2-2b. Refer to Section 3 for details on the method

Activation	Feature id	Feature description
2.4199	13243	Wedding invitations and venue planning
2.3485	4230	Wedding/marriage
1.0574	7507	No clear pattern. Commonly occurring feature with 48% activation density
0.8631	14599	No clear pattern. Commonly occurring feature with 22% activation density
0.7401	2421	Marriage
0.7075	8522	Visiting family
0.5743	11114	Female family member
0.5212	11768	Famous couples
0.5073	2144	Family activities and hobbies
0.4646	13871	Events and gatherings

Table 8: Table showing activations, feature ids, and feature descriptions of the wedding SAE feature steering vector (ft id 4230). Gemma-2-2b.

Act diff	Feature id	Feature description
1.5730	6810	No clear pattern. Commonly occurring feature with 45% activation density
1.4755	7507	No clear pattern. Commonly occurring feature with 48% activation density
0.8158	7972	"Hello"
0.6514	16358	The beginning of a sentence giving a plan or intentions
0.5458	3527	Statements related to existential reflection
0.4950	6140	Mathematical symbols and calculations
0.4810	7314	Opening quotation mark to the start of a quote or speech
0.4604	9349	Formalities involving physical gestures
0.4503	1768	Questioning or reasoning about thought processes or experience
0.4498	2514	Physical objects and structures

Table 9: Table showing activations, feature ids, and feature descriptions of the SAE feature steering vector (ft id 15874) in Gemma-2-2b, Layer 12. This feature activates on the token "say" said in various contexts

## I Example Rollouts

### I.1 SAE-TS London

Scale = 100 (optimal scale)

<bos>I think these photos were taken at the National Portrait Gallery in London earlier this year. The portraits collection features paintings of British actors Johnny Depp and Helen Mirren as  
<bos>I think I've written about my mum's beautiful family home in north London a few times, but it was my last weekend in London with my mum  
<bos>I think it's safe to say that the fashion scene in New York City continues to soar.  
\n\nAfter a few years of highlights from the city, including  
<bos>I think we've all been waiting for a new <em>Sherlock</em> series over in London ever since the actress Karen Wright announced that she would be leaving the

Scale = 160 (too high). At high steering scales, the model seems to mostly talk about fashion, designers, and art exhibitions.

<bos>I thinkLondon designer Samantha Morton's kitchen installation recently designed in London inspired by the city. The Interior Design show hosted event hosted in London which saw Interiors Designer  
<bos>I thinkyou artist Victoria Morris's new exhibition at Somerset Gallery in London is celebrating the French capital's fashion and design scene with Works on Display,  
<bos>I thinkLondon's fashion scene is thriving with vibrant exhibits and exhibitions currently showing in The Royal Academy's Summer Exhibition, which took returns from London favourites Michel",  
<bos>I thinkLondon's fashion scene continues to thrive thanks to the city's eclectic communities which celebrate the capital\u2019s vibrant eateries and restaurants this summer which have

### I.2 CAA London

Scale = 100.

<bos>I think they are a bit misleading as the term "London" as a hotel may be in poor taste.\n\nIf you look beyond that, it is a,  
<bos>I think the biggest challenge this year for the FA has been what they do with the 2nd stage teams. The clubs have been locked away from each other,  
<bos>I think I read somewhere that they've been asked to be more involved in the music scene before. I mean I get that the band probably won't,  
<bos>I think that people should feel 'theatred' rather than 'thespian' and 'theatre' rather than\n'thewest', 'South',

### I.3 SAE London

Scale = 120. Does not mention London.

"<bos>I think I have seen them on before. It is a very good idea.\n\nThanks, and yes, it is a good idea..\n\nIf the market",  
<bos>I think this is 369.\n\nI' 60 in one\n\nYeah we go and get him\n\nWell done, now for those",  
<bos>I think you are still the same - the first time I met you to make my wife'  
I know that it is no. 1 (I do not",  
<bos>I think that' 2015 is an actually a correct time to open this book,  
as each of you will get in anerropolis after an",

### I.4 Steering with – Mb

Scale = 100.

"<bos>I think back on my childhood and remember growing up in Atlanta with my family and best friend.\n\nWe spent our summers playing baseball in the neighborhood and learning to"

"<bos>I think the idea of The Black Keys' new video for "Lonely Phaser" is inspired by a "Twilight" movie.\n\nShot in London and directed by"

"<bos>I think we're officially in Fall. It's been a chilly season for the past couple months - and this week is no exception. While we'"

"<bos>I think the new movie "Still Alice" based on the memoir of Alice Koda is one of the most compelling films of the year. The film follows the"