Steering Clear: A Systematic Study of Activation Steering in a Toy Setup

Dmitrii Krasheninnikov[*] **David Krueger**

Abstract

Activation steering is a promising family of methods for controlling LLM outputs via targeted interventions on model activations. We introduce a toy multi-label classification setup to systematically study activation steering methods, and experiment with several types of steering adapters – from steering vectors (adding a fixed vector to activations) to more expressive adapters involving projections. We evaluate the adapters across steering tasks of different complexities, for three notions of complexity: 1) how densely the features are packed in the representation space (roughly, number of features divided by the dimensionality of the activations), 2) number of attributes steered, and 3) number of values the steered attribute can take. We find that as task complexity is increased, steering vector methods perform worse, while the more expressive methods only take a performance hit when there is not enough data. On the other hand, steering vectors usually outperform the more expressive methods in the low-data regime, regardless of task complexity. We conclude by discussing this work's limitations, which include our toy setup not modeling features represented in superposition or continuous features, and the lack of experiments with LLMs.

1 Introduction

Activation steering methods, also known as representation engineering, are promising to become a new major paradigm for controlling LLMs' outputs, alongside prompting, finetuning, and training data curation. Most works so far have focused on *steering vectors* – essentially, adding a fixed vector to the model's activations at a given layer, which is equivalent to modifying the layer's bias [Turner et al., 2024, Zou et al., 2023, Liu et al., 2024, Todd et al., 2024, Templeton, 2024]. Others propose more expressive steering adapters. For instance, Singh et al. [2024] propose MIMIC, a method that uses an oblique projection matrix together with translation by a fixed vector to steer the model.

The majority of prior works compare their proposed steering method to few other methods, and often use tasks different from those used in other papers. Because of this, it is hard to tell which steering methods might work best and why [Brumley et al., 2024]. To rectify this issue, our paper introduces a toy setup designed for an in-depth study of the properties of the various steering adapters.

Concretely, our study relies on a toy multi-label classification setup, where we attempt to steer the model's activations such that one attribute's value is changed to the desired one, and the other attributes' values are unchanged. This is similar to how steering methods applied to LLMs usually try to change one attribute at a time, such as the tone or the toxicity of the model's response, without changing the contents of the generated text. We consider synthetic data with m attributes, each with multiple possible values, so our model has m output heads – one for each attribute.

We train a four-layer ResNet on such data, apply different steering adapters at layer 3, and report the steering performance (whether all m labels are correct) across a variety of settings. Our experiments show a clear tradeoff between maximum attainable steering performance and data efficiency across four types of steering adapters with different levels of expressiveness.

[*]University of Cambridge. Correspondence to dmkr0001 at gmail dot com.

Our contributions include:

1. A simple toy setup that allows for a systematic study of activation steering methods across several notions of task complexity.
2. Experiments comparing steering methods CAA [Rimsky et al., 2023], MIMIC [Singh et al., 2024], ReFT [Wu et al., 2024], and a new baseline we call *low-rank orthogonal representation steering* (LOReSt). We show that steering vectors (CAA) have a fairly low performance ceiling, and work worse than the more expressive methods when features are more densely packed in the activation space or when steering requires a larger change to source activations. However, steering vectors can be more data efficient, especially for easy steering tasks.

2 Multi-label classification setup as a testbed for model steering

Synthetic multi-label dataset. Our dataset consists of samples with m attributes (60 to 120 in our experiments), where each attribute can take multiple possible values. Each attribute is represented by an 8-dimensional feature vector, making the total input dimensionality $8 \times m$. To generate these features for an attribute with k possible values, we first create k anchor vectors $\boldsymbol{\mu}_i \in \mathbb{R}^8,\ 1 \leq i \leq k$ for that attribute. Then, for a datapoint where that attribute should take value i, we sample its corresponding 8-dimensional features from a multivariate Gaussian with mean $\boldsymbol{\mu}_i$ and diagonal covariance. The full input vector for each datapoint is formed by concatenating the 8-dimensional features for all m attributes. The model has m output heads, one for each attribute, and the dataset is approximately balanced across all possible attribute values.

Steering task. The goal of steering is to modify one attribute's predicted label while keeping other attributes intact. For instance, if the original datapoint had labels [3, 1, 2] for the three attributes, and we are steering it to label 0 on the first attribute, we hope to get [0, 1, 2] as the label after steering. We measure total accuracy: the fraction of test examples where all labels are correct, such that the steered attribute takes the desired target value and the other attributes' values are unchanged.

Contrastive datapoints. Steering methods considered in this work rely on pairs of contrastive datapoints, which differ only in the value of one of the labels (or several labels, in case we want to steer several attributes at the same time). For instance, a pair of contrastive datapoints could have labels [3, 1, 2] and [0, 1, 2] respectively – in this case we are steering the first attribute to value 0. We refer to the latter datapoint – one where the steered attribute equals the target value – as the positive example, and the former datapoint as the negative example.

3 Steering methods

Contrastive activation addition (CAA). We study CAA [Rimsky et al., 2023] as a representative steering vector method. CAA works as follows: given a dataset of activations from contrastive datapoint pairs $\{(\mathbf{a}_i^{\text{pos}}, \mathbf{a}_i^{\text{neg}})\}_{i=1}^N$, the steered activations are computed as $\mathbf{a}^{\text{steered}} = \mathbf{a} + \alpha\mathbf{v}$, where α is the steering magnitude and $\mathbf{v} = \frac{1}{N}\sum_{i=1}^N(\mathbf{a}_i^{\text{pos}} - \mathbf{a}_i^{\text{neg}})$: the average of the differences between activations of contrastive training datapoints.

Minimally modified counterfactuals (MiMiC). Singh et al. [2024] introduce the steering method MiMiC that uses an affine function to steer a distribution of activations with an undesired property (e.g. toxicity) such that after steering, the resulting distribution matches the mean and the covariance of a distribution of activations with the desired property (e.g. non-toxic). The steering adapter has the form $\mathbf{a}^{\text{steered}} = \mathbf{W}\mathbf{a} + \mathbf{b}$, where $\mathbf{W}$ is an oblique (non-orthogonal) projection matrix, and both $\mathbf{W}$ and $\mathbf{b}$ are computed in closed form. Obtaining $\mathbf{W}$ involves computing covariance matrices for datasets of activations of positive and negative examples. When the number of datapoints available is smaller than the dimensionality of our activations, these covariance matrices are rank-deficient. Our experiments show that this results in null performance for MiMiC in the low-data regime, but can be fixed by adding a diagonal matrix of epsilon=1 to the covariance matrices when in the low-data regime as a regularizer. In the plots below, we show both the performance of the original MiMiC method, as well as our version utilizing this simple regularization strategy.

Representation finetuning (ReFT). Wu et al. [2024] propose ReFT as method for parameter-efficient finetuning. In this work, instead of learning the parameters of the ReFT adapter with an end-to-end loss, we train the adapter to take un-steered activations as an input and return the modified activations $\mathbf{a}^{\text{steered}}$ using the MSE loss on target activations. The LoReFT adapter has the following form: $\mathbf{a}^{\text{steered}} = \mathbf{a} + \mathbf{R}^\top(\mathbf{W}\mathbf{a} + \mathbf{b} - \mathbf{R}\mathbf{a})$, where $\mathbf{R} \in \mathbb{R}^{r \times d}$ is constrained to orthonormal rows (we accomplish this with differentiable QR decomposition), and parameters $\mathbf{W} \in \mathbb{R}^{r \times d}$ and $\mathbf{b} \in \mathbb{R}^r$ are not constrained. Note that unlike the previous two methods that can in principle work with unpaired data (e.g. two sets of points with and without the desired behavior), ReFT requires paired datapoints.

Low-rank orthogonal representation steering (LOReSt). As a baseline inspired by ReFT, we introduce a steering method based on learned orthogonal projections with Gumbel-Softmax dimension selection. Key parts of this adapter are the learned rank-r orthogonal basis $\mathbf{Q} \in \mathbb{R}^{d \times r}$ (obtained via differentiable QR decomposition) and dimension-wise drop probabilities $\mathbf{p} \in \{0, 1\}^r$ obtained through Gumbel-Softmax sampling. Steering is performed as $\mathbf{a}^{\text{steered}} = \mathbf{a} - (\mathbf{a}\mathbf{Q})\text{diag}(\mathbf{p})\mathbf{Q}^\top + \mathbf{b}$. Here $\mathbf{b} \in \mathbb{R}^d$ is a learned bias, and $\mathbf{p}$ is a binary vector sampled from the Gumbel-Softmax distribution with learned logits $\boldsymbol{\ell} \in \mathbb{R}^r$ and temperature τ. The probability of keeping dimension i is given by $\mathbf{p}_i = \text{GumbelSoftmax}([\boldsymbol{\ell}_i, 0]; \tau)_1$. This computation is equivalent to applying an orthogonal projection matrix $\mathbf{P} = \mathbf{I} - \mathbf{Q}\text{diag}(\mathbf{p})\mathbf{Q}^\top$ (with the resulting projection of rank between $d - r$ and d, depending on how many dimensions are dropped) and shifting by the bias. Similarly to our usage of ReFT, LOReSt is trained with the MSE loss on target activations, and requires paired datapoints.

4 Results

This section discusses our results comparing the four steering methods described above on variants of our toy setup. We are especially interested in the steering methods' performance as the complexity of the steering task is increased. Varying the task complexity is operationalized in three ways:

- Varying the "density" of features in the representation space (more attributes packed into representations of the same dimensionality means higher density).
- Varying the number of steered attributes.
- Varying the number of values an attribute can take.

Training the model we steer. All experiments start by training a four-layer feedforward MLP with residual connections [He et al., 2016] on 2M datapoints sampled according to §2. We train for 50 epochs using AdamW [Loshchilov, 2017] with learning rate 0.0001 and batch size 4096, and use the model from an epoch with the best validation loss. The hidden dimensions of the resnet are 512-512-256-512. We use the GELU nonlinearities [Hendrycks and Gimpel, 2016], and use layer normalization [Lei Ba et al., 2016] after the last residual block (right before the output layer); the residual blocks themselves do not have layer normalization.

Setting up steering adapters. Activations of the contrastive datapoints are collected at the residual stream after the third layer (with $d = 256$). The steering adapter is inserted at the same location. We did not experiment with steering adapters inserted in several locations at the same time. Adapters without closed-form solutions, ReFT and LOReSt, are trained for 2k epochs with AdamW, and the adapter from the epoch with the best training loss is selected. We found the large number of epochs especially beneficial when there are very few training datapoints.

Varying the density of features in the representation space. Each row of subplots in Figure 1 varies the number of attributes that the model needs to pack into the 256-dimensional space. For the densest configuration, the model needs to represent 120 attributes with 8 possible values each, so approximately 2 dimensions per attribute[2]. We observe that when features are packed more densely, more expressive steering methods substantially outperform steering vectors when there is enough

[2]Note that because our data is not sparse (all 8^{120} label combinations can occur in principle), the learned representation cannot easily benefit from superposition (encoding more than two features per dimension [Elhage et al., 2022]). In fact, training a model of the same architecture on data with 150 8-dimensional attributes already results in fairly low overall performance. In a future version of this paper, we will show how data sparsity can dramatically increase the number of attributes that can be packed into the activations.

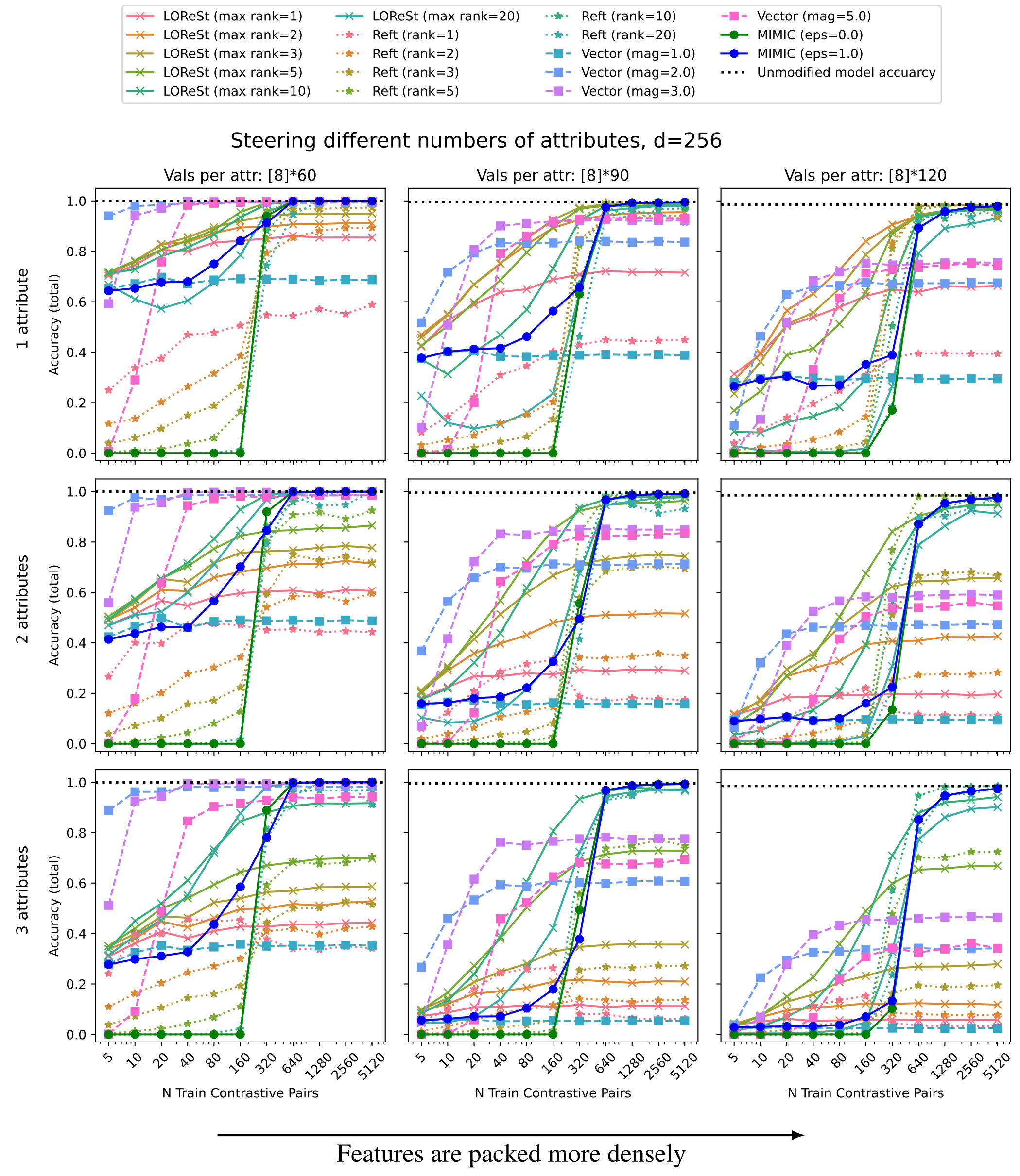

Figure 1: Total accuracy: fraction of test examples where all labels are correct. Values per attribute of [8]*60 means that there are 60 attributes, each with 8 possible values. Each point is an average over 20 steering attempts, each for a different attribute. We observe that as the number of attributes that need to fit into the representation with dimensionality $d = 256$ increases, steering becomes harder, especially for steering vectors – and also for the more expressive methods when the amount of training data is small. The same happens when we increase the number of attributes we steer; additionally, more expressive methods turn out to be necessary here, as we see from ReFT and LOReSt working only when their rank is sufficiently high. Note the inflection in performance that occurs for ReFT and MiMiC when the number of training datapoints becomes larger than d.

training data. However, steering vectors perform better when the number of training datapoints is small, with steering magnitudes α greater than 1 performing best. LOReSt offers a middle ground between steering vectors and ReFT & MiMiC in terms of data efficiency, and is capable of obtaining performance similar to MiMiC and ReFT. It is notable that for MiMiC and ReFT, there is a clear inflection point in steering performance when the number of training datapoints is equal to the dimensionality of the steered layer. This suggests that these methods may be less suitable for steering LLMs, which often have high-dimensional residual streams (e.g. $d = 4096$). Another curious observation is that our regularized version of MiMiC works substantially better than the original in the low-data regime, and specifically, the low-data regime performance of this method appears to align with the performance of CAA with steering magnitude 1.

Varying the number of steered attributes. Columns of Figure 1 vary the number of steered attributes by sampling contrastive datapoints that differ on two or three attributes in addition to just one attribute (the current version of the experiment only uses contrastive datapoints that differ on all target attributes, instead of e.g. differing on one attribute of interest but matching on another). Similarly to above, we observe a data efficiency VS performance ceiling tradeoff across the differently-expressive steering adapters – with steering vectors on one side of the tradeoff, ReFT and MiMiC on the other, and LOReSt somewhere in between. Relatedly, we see that as the number of steered attributes is increased, ReFT and MiMiC require higher ranks r to reach the same level of performance (see the rightmost column of Figure 1).

Varying the specificity of the target attribute. Informally, steering an attribute with 2 possible values requires changing 1 bit in the model's hidden state, while steering an attribute with 8 values requires changing 3 bits. Our experiment in Figure 2 shows that as the required change gets larger, all steering methods perform worse, similarly to when steering for several attributes simultaneously. The less expressive steering adapters end up with a lower performance ceiling, and the more expressive adapters perform worse in the low-data regime. It is interesting to compare the rightmost subplot of Figure 2 with the middle subplot of Figure 1. Both suplots show steering attempts trying to change 6 bits (one 6-bit attribute VS two 3-bit attributes) in models with similar "attribute density" (8^{84} possible label combinations VS 8^{90}). We see that the results are rather similar.

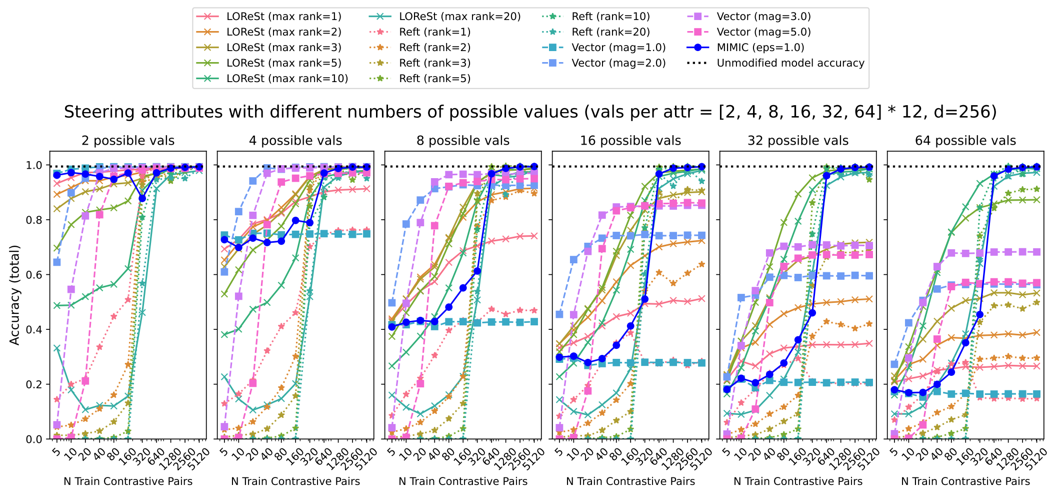

Figure 2: We steer a model trained on data with attributes of six different sizes (there are 12 attributes of each size). As precision required for steering increases for attributes with more values, steering vectors end up with a lower performance ceiling, while ReFT and LOReSt require higher rank to work well.

5 Discussion

Limitations. The main limitation of our work is the lack of experiments with LLMs: it is unclear whether our results regarding the relative performance of the various steering methods would hold up in realistic settings. Two issues specifically make our setup less realistic: not modeling features represented in superposition, and not modeling continuous features. Additionally, it is unclear how the typical "task complexity" for an LLM looks like: if it is more like the top left subplot of Figure 1, then steering vectors would likely work best, and if it's closer to the bottom right subplot, the more expressive steering methods might come on top.

In addition to making our setup more realistic and LLM experiments, future work could study non-linear steering adapters, as well as explore losses other than MSE for training the adapters – for instance by including additional end-to-end loss terms aimed at reducing the performance penalty arising from steering (like the loss in [Braun et al., 2024] for sparse autoencoders). Furthermore, the adjustable steering magnitude common in steering vector methods might offer substantial convenience when steering for continuous features, and our experiments show that even for discrete attributes, steering magnitude other than 1 often results in the best steering performance. MiMiC, ReFT and LOReSt could benefit from modifications allowing for such control over the steering impact.

Other limitations are common to the activation steering paradigm. First, activations of contrastive datapoints may have multiple differing salient features, and steering methods that need to choose which features to focus on, like in-context vectors [Liu et al., 2024] via PCA, and ReFT & LOReSt via the low-rank transformation, may focus on the wrong feature. Farquhar et al. [2023] discuss this issue for a different setup utilizing contrastive datapoints. Second, the benefits of activation steering over finetuning have not been clearly articulated: is there a benefit of avoiding end-to-end training (e.g. DPO [Rafailov et al., 2024]) using the same contrastive datapoint pairs? We suspect there are benefits related to process supervision [Stuhlmüller and Byun, 2024], and are interested in future work clarifying this.

Conclusion. We introduced a toy multi-label classification setup for studying activation steering, and used this setup to investigate how four different steering methods fare as task complexity is increased. Our work highlights the drawbacks of steering-vector-based methods: poor performance when 1) the features are tightly packed in the activation space, and 2) when the goal is to change a larger amount of information in the activations. However, steering vectors can be more data-efficient than the more expressive methods when there are very few training datapoints. Our study offers unique insight into the challenges of model steering, and is a small step on the path towards more controllable AI systems.

References

D. Braun, J. Taylor, N. Goldowsky-Dill, and L. Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning. *arXiv preprint arXiv:2405.12241*, 2024.

M. Brumley, J. Kwon, D. Krueger, D. Krasheninnikov, and U. Anwar. Comparing bottom-up and top-down steering approaches on in-context learning tasks. *arXiv preprint arXiv:2411.07213*, 2024.

N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

S. Farquhar, V. Varma, Z. Kenton, J. Gasteiger, V. Mikulik, and R. Shah. Challenges with unsupervised llm knowledge discovery. *arXiv preprint arXiv:2312.10029*, 2023.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *ArXiv e-prints*, pages arXiv–1607, 2016.

S. Liu, H. Ye, L. Xing, and J. Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2024.

I. Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

N. Rimsky, N. Gabrieli, J. Schulz, M. Tong, E. Hubinger, and A. M. Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

S. Singh, S. Ravfogel, J. Herzig, R. Aharoni, R. Cotterell, and P. Kumaraguru. Representation surgery: Theory and practice of affine steering. In *Forty-first International Conference on Machine Learning*, 2024.

A. Stuhlmüller and J. Byun. Supervise process, not outcomes. *URL https://ought. org/updates/2022-04-06-process*, 9, 2024.

A. Templeton. *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic, 2024.

E. Todd, M. L. Li, A. S. Sharma, A. Mueller, B. C. Wallace, and D. Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2024.

A. M. Turner, L. Thiergart, G. Leech, D. Udell, J. J. Vazquez, U. Mini, and M. MacDiarmid. Activation addition: Steering language models without optimization, 2024.

Z. Wu, A. Arora, Z. Wang, A. Geiger, D. Jurafsky, C. D. Manning, and C. Potts. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*, 2024.

A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika, A.-K. Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.