



# When Steering Vectors Fail

Skye Purchase<sup>1</sup>

Machine Learning MSc

Daniel Tan & Brooks Paige

Submission date: 8 September 2025

<sup>1</sup>**Disclaimer:** This report is submitted as part requirement for the Machine Learning MSc at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged

5

## **Abstract**

6 Summarise your report concisely.

# Contents

8	<b>1 Introduction</b>	<b>1</b>
9	1.1 Benefits of Steering Vectors . . . . .	1
10	1.2 Related Work . . . . .	1
11	1.3 Contributions . . . . .	2
12	<b>2 Background</b>	<b>3</b>
13	2.1 Notation and Concepts . . . . .	3
14	2.2 Model Alignment . . . . .	4
15	2.3 Model Intervention . . . . .	4
16	2.3.1 Contrastive Activation Addition . . . . .	4
17	2.3.2 Affine Concept Editing . . . . .	5
18	2.3.3 Low-rank Representation Finetuning . . . . .	6
19	2.3.4 Low-rank Representation Steering . . . . .	8
20	2.4 Large Language Models . . . . .	8
21	2.4.1 Transformers . . . . .	9
22	2.4.2 Instruction Tuning . . . . .	10
23	2.5 Sparse Auto-Encoders . . . . .	10
24	<b>3 Methodology</b>	<b>13</b>
25	3.1 Steering Clear Environment . . . . .	13
26	3.1.1 Dataset . . . . .	13
27	3.1.2 Pre-training . . . . .	13
28	3.1.3 Steering Task . . . . .	14
29	3.2 Prompt Pairs Environment . . . . .	14
30	3.2.1 dataset . . . . .	15
31	3.2.2 Steering Task . . . . .	15
32	3.3 Prompting LLMs . . . . .	16
33	<b>4 Results</b>	<b>17</b>
34	4.1 Prompt Pairs . . . . .	17
35	<b>5 Conclusion</b>	<b>19</b>
36	5.1 Limitations . . . . .	19

37	5.2 Future Work . . . . .	19
38	<b>A Gumbel Softmax</b>	<b>24</b>
39	<b>B Prompt Pairs Dataset</b>	<b>25</b>

# Chapter 1

## Introduction

### 1.1 Benefits of Steering Vectors

### 1.2 Related Work

[Krasheninnikov and Krueger \(2024\)](#) aim to analyse steering in a toy environment where they are able to control the representation density within the model. They compare a range of steering techniques ([Rimsky et al., 2024](#); [Singh et al., 2024](#); [Wu et al., 2024](#)) against each other in a controlled setting to evaluate the benefits and drawbacks of each approach. Inspired by LoReFT ([Wu et al., 2024](#)) they introduce their own technique LoReST and demonstrate competitive performance to the other techniques.

This project reproduces a sample of plots from Figure 1 using the same toy setup described in §3.1. In addition to the techniques used in the original paper the reproduction also analyses the behaviour of [Marshall et al. \(2024\)](#).

This project aims to expand the analysis carried out by [Krasheninnikov and Krueger \(2024\)](#) to reproduce the same effects in large language model systems. Additionally, the relationship between the negative and positive training examples is analysed to gain a better insight as to when steering approaches fail.

[Tan et al. \(2024\)](#) aim to analyse the generalisation of steering vectors across a range of steering datasets. They analyse the variability of success and introduce the notion of steerability. Using this notion they demonstrate that many techniques fail to generalise on certain datasets both in and out of distribution.

The analysis is limited to only contrastive activation addition ([Rimsky et al., 2024](#)) which [Krasheninnikov and Krueger \(2024\)](#) show is not necessarily the ideal candidate. Building on their work this project aims to analyse a larger range of techniques sampled from [Krasheninnikov and Krueger \(2024\)](#). Furthermore, the properties of training datasets is analysed in more

65 depth to determine which properties cause steering techniques to fail.

66 Rather than use model written evaluations (Perez et al., 2023) a new set of steering datasets is  
67 generated with more fine grain control. The construction of these datasets is described in §3.2.

68 Wehner et al. (2025) present a full taxonomy of current steering vector approaches (more  
69 generally *representation engineering*). The approaches include those in Krasheninnikov and  
70 Krueger (2024), however, it is impractical to expand the experiments to all the approaches de-  
71 scribed due to time constraints.

72 The paper covers a range of topics within representation engineering that have been carried out  
73 by the community. These focus on the types of adaptors used, the prompting framework, linear  
74 vs. non-linear adaptors, the concepts that are steered, etc.

75 This project continues these comparisons by analysing the effect of the dataset and number of  
76 steering examples used akin to Krasheninnikov and Krueger (2024) in the large language model  
77 setting. The analysis is related to Wehner et al.’s 2025 suggestion to analyse spurious corre-  
78 lations in the dataset as a potential failure case for steering vectors. HOW?? NEED TO DO  
79 MORE IN DEPTH ANALYSIS ME THINKS.

80 **Sparse autoencoders as steering vectors** There are multiple papers that utilise sparse  
81 autoencoders (SAEs) as steering vectors (Chalnev et al., 2024; Kharlapenko et al., 2024; Nanda  
82 et al., 2024). These approaches utilise the fact that SAEs decode high level concepts from the  
83 models intermediate representation. This can be used to manipulate the representation towards  
84 a target concept.

85 Rather than utilising SAEs to steer the model this project uses the SAE features to evaluate  
86 models on free form prompts. The SAE features provide a metric to evaluate how well the mod-  
87 els internal representation has been effectively steered. SAEs are explained in §2.5 and their use  
88 in the project is described in §3.2.

## 89 1.3 Contributions

## Chapter 2

# Background

### 2.1 Notation and Concepts

Model “behaviours”, in general, are patterns in how the model responds to input. This includes the desired behaviour it was trained on (such as classifying images of cats and dogs) but includes patterns in the output that were not explicitly trained for. Desired model behaviour is considered “positive” and undesired model behaviour is considered “negative”. Specifically, an example of the desired behaviour is considered a “positive example” and an example of undesired or neutral behaviour is considered a “negative” example. An example of a behaviour generally includes an input-output pairing similar to training examples however they are more specific pairings than would be using during training.

Throughout the document neural network (NN) and machine learning (ML) model are used interchangeably though NNs are a strict subset of ML models. When discussing NNs the concept of a “neuron” relates to the abstract structure that receives a real-valued, vector input and outputs a real-value scalar based on internal, learnable weights. In practice, this is represented by a single element of a NN layer’s output vector.

Vectors are represented by boldface letters,  $\mathbf{x}, \mathbf{y}, \mathbf{z}$ , scalars are represented by greek letters,  $\alpha, \beta, \gamma$ , and matrices are represented by boldface capital letters,  $\mathbf{A}, \mathbf{B}, \mathbf{C}$ . Some matrices may represent transformations or collections of feature vectors, context should disambiguate the two. In general vectors are column vectors,  $\mathbf{x} = \begin{bmatrix} 1 & 2 & \cdots & n \end{bmatrix}^T$  except when a collection of vectors is represented in matrix form, in this case each row is a vector.

In a multi-layer machine learning model the output of an internal layer is an “activation” denoted  $\mathbf{a}$ . A positive activation is denoted  $\mathbf{a}^+$  and a negative activation is denoted as  $\mathbf{a}^-$ . Here, “positive activation” means the activation extracted from the model given a positive example as above.

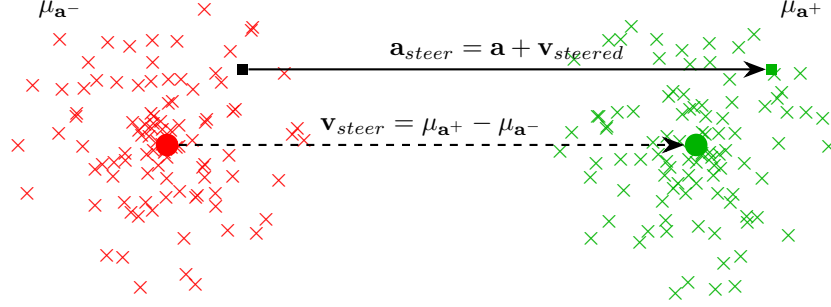


Figure 2.1: Demonstration of contrastive activation addition (Rimsky et al., 2024). The figure represents a simple representation space of dimension 2 with clear separability.  $\mu_A = \frac{1}{\|A\|} \sum_{i \in \mathcal{I}(A)} A_i$  where  $A$  is a set of activation vectors. A new point, such as the black square, is translated by the steering vector.

## 2.2 Model Alignment

## 2.3 Model Intervention

### 2.3.1 Contrastive Activation Addition

An intuitive approach to model intervention is to perturb the model’s activations in a desired direction. By calculating a linear direction in activation space from undesired activations towards desired ones this vector can simply be added to all activations in the model during inference. The hope is that the model produces output that matches the desired behaviour whilst maintaining the context of the new input.

In the simplest form consider two example inputs with desired and undesired behaviour. Their difference gives a direction in feature space that corresponds to shifting the models output from undesired behaviour towards desired behaviour. This is the approach proposed by Turner et al. (2023), however, it is not robust and relies heavily on the example inputs (Rimsky et al., 2024).

To improve on this approach Rimsky et al. (2024) suggest using a collection of examples and calculating their mean difference in activation space. This requires the notion of *contrastive pairs*, two inputs that are similar in all ways except for the behaviour that is being changed. Hence, this approach is known as *contrastive activation addition* (CAA). This process is demonstrated in Figure 2.1.

Formally, given a set of positive example activations  $(\mathbf{a}_i^+)_{i \leq n}$  and negative example activations  $(\mathbf{a}_i^-)_{i \leq n}$  a *steering vector* for this behaviour is

$$\mathbf{v}_{steer} = \frac{1}{n} \sum_{i=1}^n (\mathbf{a}_i^+ - \mathbf{a}_i^-).$$

Given a steering vector,  $\mathbf{v}_{steer}$ , and a model activation during inference,  $\mathbf{a}$ , the resulting steered



135 activation is

$$\mathbf{a}_{steered} = \mathbf{a} + \lambda \mathbf{v}_{steer} \quad (2.1)$$

136 where  $\lambda$  is a user-defined parameter controlling the strength of the steering intervention. The  
 137 model activation is replaced by the steered activation during inference resulting in the model  
 138 producing an output aligned with the positive examples.

139 This approach has a few drawbacks (Engels et al., 2025; Marshall et al., 2024; Tan et al., 2024)  
 140 due to its assumptions. Primarily this approach does not consider how much of a behaviour is  
 141 already present. This means the steering parameter does not fully determine the strength of the  
 142 desired behaviour. Furthermore, Tan et al. (2024) demonstrate that this approach is not robust  
 143 across behaviours that may be steered along. The approach assumes that concepts in activation  
 144 space are linear which Engels et al. (2025) show is not universal. Techniques such as affine con-  
 145 cept editing (ACE) §2.3.2 use an affine approach to overcome these drawbacks.

### 146 2.3.2 Affine Concept Editting

147 Marshall et al. (2024) claim that CAA (Rimsky et al., 2024) is not sufficiently general as it does  
 148 not consider how much the desired behaviour is already present. To see this consider an arbi-  
 149 trary activation vector  $\mathbf{a}$  and steering direction  $\mathbf{r}$  encoding some behaviour.  $\mathbf{a}$  can be decom-  
 150 posed as the perpendicular and parallel components of  $\mathbf{r}$

$$\begin{aligned} \mathbf{a} &= \text{proj}_{\mathbf{r}}^{\perp}(\mathbf{a}) + \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) \\ &= \text{proj}_{\mathbf{r}}^{\perp}(\mathbf{a}) + \alpha \mathbf{r}. \end{aligned}$$

151 Adding  $\lambda \mathbf{r}$  as per Equation 2.1 will be inconsistent as  $\alpha + \lambda$  will not be equivalent across all  
 152 (negative) activations. This shows that CAA (Rimsky et al., 2024) does not account for how  
 153 much a behaviour may already be present in an activation.

154 Furthermore, it is not (generally) the case that  $\mathbf{0}$  represents lack of behaviour. Instead there is  
 155 some vector  $\mathbf{a}_0$  that represents the lack of the target behaviour. The above equation can incor-  
 156 porate this idea as follows

$$\begin{aligned} \mathbf{a} &= \mathbf{a}_0 + \Delta \mathbf{a} \\ &= \mathbf{a}_0 + \text{proj}_{\mathbf{r}}^{\perp}(\Delta \mathbf{a}) + \text{proj}_{\mathbf{r}}^{\parallel}(\Delta \mathbf{a}) \\ &= \mathbf{a}_0 + \text{proj}_{\mathbf{r}}^{\perp}(\Delta \mathbf{a}) + \alpha' \mathbf{r}. \end{aligned}$$

157 Removing the behaviour by setting  $\alpha' = 0$  yields

$$\begin{aligned} \mathbf{a}' &= \mathbf{a}_0 + \text{proj}_{\mathbf{r}}^{\perp}(\Delta \mathbf{a}) \\ &= \mathbf{a} - \text{proj}_{\mathbf{r}}^{\parallel}(\Delta \mathbf{a}) \\ &= \mathbf{a} - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}_0) \\ &= \mathbf{a} - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0 \mathbf{r}.^1 \end{aligned}$$

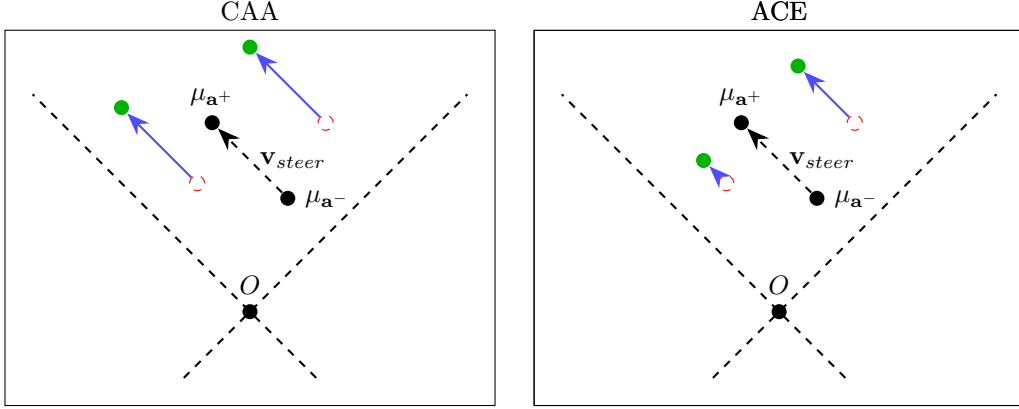


Figure 2.2: A comparison of CAA (Rimsky et al., 2024) and affine concept editing (Marshall et al., 2024). This is a reproduction of Figure 1 in Marshall et al. (2024) with the steering towards the positive examples instead. Compared to CAA, ACE does not adjust perpendicular components but correctly adjusts those parallel to the steering direction.

158 This represents the activation lacking the target behaviour but retaining other relevant context.  
 159 The behaviour can be reintroduced at any relevant strength resulting in

$$\mathbf{a}_{\text{steered}} = \mathbf{a}_0 - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0 \mathbf{r} + \alpha \mathbf{r}. \quad (2.2)$$

160 Given positive example activations  $(\mathbf{a}_i^+)_{i \leq n}$  and negative example activations  $(\mathbf{a}_i^-)_{i \leq n}$  the refer-  
 161 ence point and steering direction are

$$\mathbf{a}_0 = \frac{1}{n} \sum_{i=1}^n \mathbf{a}_i^- \quad \mathbf{r} = \frac{1}{n} \sum_{i=1}^n (\mathbf{a}_i^+ - \mathbf{a}_i^-).$$

162 This process is described graphically in Figure 2.2

163 This approach is no longer a linear edit to the activations and now includes a bias term,  $\mathbf{a}_0$ .  
 164 This is therefore affine and hence the name *affine concept editing* (ACE) (Marshall et al.,  
 165 2024).

### 166 2.3.3 Low-rank Representation Finetuning

167 Both CAA (Rimsky et al., 2024) and ACE (Marshall et al., 2024) edit the activations in their  
 168 full rank form and rely on addition (whether affine or linear). This limits the transforms the ap-  
 169 proaches can apply to the activation space. If the desired behaviour requires rotations or scaling  
 170 of the activations these methods fail. However, perform affine transformations to the full rank  
 171 activation is costly as the dimension of the activations may be large.

172 Wu et al. (2024) present a low-rank steering adaptor inspired by parameter-efficient finetuning  
 173 methods such as LoRA (Hu et al., 2022), DoRA (Liu et al., 2024b) and adaptor-based meth-  
 174 ods (Houlsby et al., 2019). Unlike steering these approaches aim to finetune a model using re-

<sup>1</sup>As  $\mathbf{a}_0$  exists as a reference point along the steered direction.

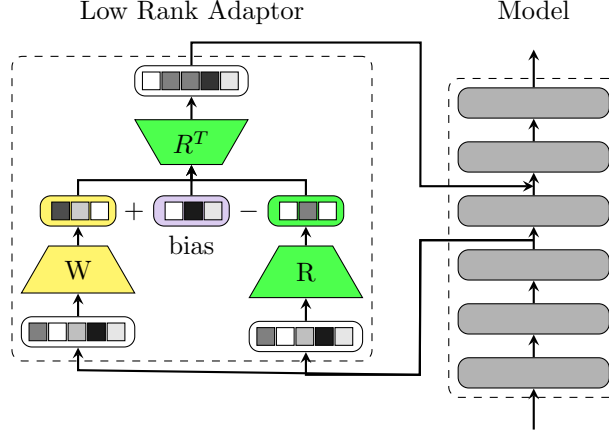


Figure 2.3: This figure demonstrates how the low-rank representation finetuning adaptor (Wu et al., 2024) operates. Unlike methods such as LoRA (Hu et al., 2022) this does replace the layer weights but simply adds the layer output. LoReST (Krasheninikov and Krueger, 2024) behaves similarly though the adaptor has a different architecture. This is a reproduction of Figure 2(2) in Wu et al. (2024).

duced parameter counts compared to the original model. Rather than finetuning a model this approach aims to edit the representations of the model, this is equivalent to steering.

The key insight is to steer the activations in a low-rank space. The specific approach is based on the distributed interchange intervention<sup>1</sup> (Geiger et al., 2024) with the following form

$$DII(\mathbf{x}, \mathbf{y}, \mathbf{R}) = \mathbf{x} + \mathbf{R}^T(\mathbf{R}\mathbf{y} - \mathbf{R}\mathbf{x})$$

where  $\mathbf{R} \in \mathbb{R}^{r \times d}$  is a low-rank projection matrix.

Wu et al. (2024) suggest replacing  $\mathbf{R}\mathbf{y}$  with an affine transformation  $\mathbf{W}\mathbf{x} + \mathbf{b}$ . Thus, the adaptor learns a transformation,  $\mathbf{R}\mathbf{a}^+ = \mathbf{W}\mathbf{a}^- + \mathbf{b}$ , from negative activations to positive low-rank representations. In this way the adaptor can learn low-rank representations of activations that encapsulate the desired behaviour and adjust the activations in a parameter efficient space. The approach is therefore a *low-rank representation finetuning* (LoReFT) adaptor. The full adaptor is

$$\mathbf{a}_{\text{steered}} = \mathbf{a} + \mathbf{R}^T(\mathbf{W}\mathbf{a} + \mathbf{b} - \mathbf{R}\mathbf{a}). \quad (2.3)$$

The learnable parameters of the adaptor are  $\phi = \{\mathbf{W}, \mathbf{R}, \mathbf{b}\}$ .  $\mathbf{R}$  is constrained to be an orthogonal projection matrix achieved by differentiable QR decomposition.

Given a dataset of contrastive pairs  $\mathcal{D} = (\mathbf{a}_i^-, \mathbf{a}_i^+)_{i \leq n}$  the adaptor parameters  $\phi$  are trained. The goal is to accurately predict  $\mathbf{a}_i^+$  given  $\mathbf{a}_i^-$  as input.

Unlike CAA (Rimsky et al., 2024) and ACE (Marshall et al., 2024) this approach requires paired datapoints as the adaptor needs to learn a transformation from negative examples to pos-

<sup>1</sup>This tests whether a concept is encoded in some subspace. When working with low-rank editing this is exactly the assumption we use.

itive examples. This drawback means that in the low data regime this approach is less effective than the other two approaches. However, with sufficient data, this method is able to outperform CAA and ACE as it can utilise more complex transformations between negative and positive behaviour. The poor performance in low data regimes is improved on by Krashennnikov and Krueger (2024) with their low-rank representation steering adaptor.

### 2.3.4 Low-rank Representation Steering

Krashennnikov and Krueger (2024) suggest modifying LoReFT (Wu et al., 2024) to dynamically drop low-rank dimensions and bring the learnable bias term outside of the low-rank space. This allows the model to perform well in the low data regime by relying on linear methods similar to CAA (Rimsky et al., 2024) but keep the benefits of LoReFT. By dynamically dropping dimensions the adaptor has more freedom to optimise the rank of the projection.

Krashennnikov and Krueger (2024) define an orthogonal projection

$$\mathbf{P} = \mathbf{I} - \mathbf{Q}\text{diag}(\mathbf{p})\mathbf{Q}^T \quad \mathbf{p}_i = \text{GumbelSoftmax}([\mathbf{l}_i, 0]; \tau)$$

where  $\mathbf{Q} \in \mathbb{R}^{r \times d}$  is a learnable low-rank projection matrix,  $\mathbf{l}$  is a learnable Gumbel Softmax distribution probabilities, and  $\tau$  is the temperature. As with LoReFT,  $\mathbf{Q}$  is an orthogonal projection achieved by differentiable QR decomposition. In comparison to LoReFT Equation 2.3 there is no representation editing in the low-rank space. Instead the projection acts as a method to “zero” the activation similar to ACE (Marshall et al., 2024).

The full adaptor is

$$\mathbf{a}_{\text{steered}} = \mathbf{a} - (\mathbf{a}\mathbf{Q})\text{diag}(\mathbf{p})\mathbf{Q}^T + \mathbf{b}. \quad (2.4)$$

This approach also requires paired data to train the parameters,  $\phi = \{\mathbf{Q}, \mathbf{l}, \mathbf{b}\}$ .  $\mathbf{Q}$  is constrained to be orthogonal through differentiable QR decomposition.

Given a dataset of contrastive pairs  $\mathcal{D} = (\mathbf{a}_i^-, \mathbf{a}_i^+)_{i \leq n}$  the adaptor parameters  $\phi$  are trained. The goal is to accurately predict  $\mathbf{a}_i^+$  given  $\mathbf{a}_i^-$  as input.

In the low data regime the adaptor can learn to drop more dimensions and rely on  $\mathbf{b}$  similar to CAA (Rimsky et al., 2024) and ACE (Marshall et al., 2024). As more data is available the adaptor can rely more on the low-rank projection similar to LoReFT (Wu et al., 2024). In this way the adaptor is able to perform consistently across different data regimes.

## 2.4 Large Language Models

Steering and model alignment in general is not confined to large language models (LLM)s however these are currently the most widespread model in use. LLMs aim to immitate, complete, or analyse natural language and are characterised by incredibly large numbers of parameters. Some are as large as 1.76 trillion (Space, 2023) and even small models have as many as 1 billion (Team, 2025).

224 Only generative LLMs are discussed in this project. These are models which are not trained to  
225 classify or fit a dataset in the classical sense but instead to produce more data as if it were sam-  
226 pled from the underlying training distribution. In the case of LLMs this means producing coher-  
227 ent natural language.

228 The underlying technology behind modern generative LLMs is the transformer (Vaswani et al.,  
229 2017) and their many derivatives (Katharopoulos et al., 2020; Wang et al., 2020; Zaheer et al.,  
230 2020).

### 231 2.4.1 Transformers

232 Transformers (Vaswani et al., 2017) are now a mainstay of modern deep learning.<sup>2</sup> They utilise  
233 the attention mechanism to dynamically transform (sequential) input based on the surrounding  
234 context.

235 Attention can be considered as a learnable lookup table with queries, keys and values. If a query  
236 and a key are similar then the corresponding value should be returned. This can be represented  
237 as a dot-product between a matrix of queries  $\mathbf{Q}$  and keys  $\mathbf{K}$ . These are normalised to act as  
238 probabilities that a specific value is the target value. Given a matrix of values  $\mathbf{V}$  attentio is rep-  
239 resented by the following equation

$$\text{softmax} \left( \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}.$$

240 The trick is to have  $\mathbf{Q}$ ,  $\mathbf{K}$  and  $\mathbf{V}$  all depend on the input features, this is known as “self-  
241 attention”. In this way  $\mathbf{V}$  behaves like a standard weight transformation and the softmax of  
242  $\mathbf{Q}$  and  $\mathbf{K}$  behave like a dynamic weight transformation dependent on the input. This allows a  
243 model to “attend” to different parts of the input by adjusting the transformation matrices that  
244 make  $\mathbf{Q}$ ,  $\mathbf{K}$  and  $\mathbf{V}$ .

245 Modern transformers contain attention blocks each containing multiple “attention heads” that  
246 use the above mechanism. This allows the model to respond dynamically to a large range of in-  
247 puts. After this attention block a standard multi-layer perceptron (MLP) is added. This consti-  
248 tutes the transformer block and is visualised in Figure 2.4.

249 The ability for transformers to utilise context in surrounding input values makes them partic-  
250 ularly suited to natural language processing (NLP). The meaning of words in a sentence de-  
251 pend on the words that surround it. Furthermore, the words depend on each other in different  
252 ways depending on the context. This is precisely how transformer attention blocks work allow-  
253 ing them to parse natural language far better than previous attempts.

254 For transformers to work on natural language the input needs to be tokenized into discrete  
255 chunks, frequently based on words. These chunks can then be converted to unique numbers and  
256 later represented as input features. To aid the model, the position of the token within the sen-

---

<sup>2</sup>This section does not aim to describe transformers in full detail but provide a sufficient background for the rest of the project. Keywords are provided for further reading and the explanation is based on the paper by Turner (2023).

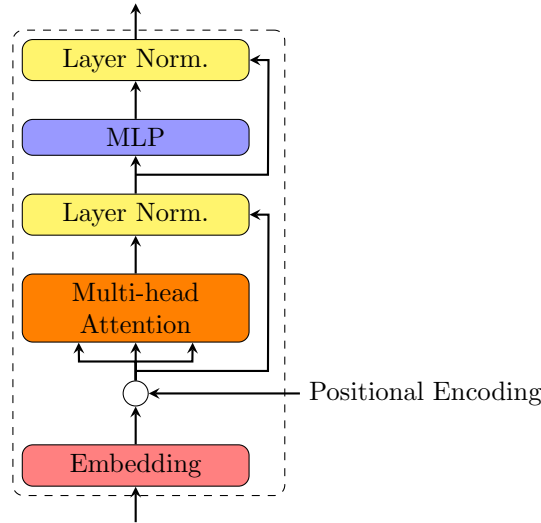


Figure 2.4: A diagram of the standard transformer decoder block. This is based on Figure 1 of [Vaswani et al. \(2017\)](#). This is a single layer in a large language model where the output of one block is fed into the input of the next.

257 tence is also encoded this is known as “positional encoding”. This allows the model to distin-  
 258 guish between the two instances of “can” in the sentence “can you pass me the can”.

259 It is important to note that when training a model to generate natural language it must be  
 260 trained without access to future tokens. The process of hiding future tokens at a given token  
 261 is called “attention masking” and is only applied in the attention blocks.

## 262 2.4.2 Instruction Tuning

## 263 2.5 Sparse Auto-Encoders

264 Though the processes to build, train and use a machine learning model are known, these pro-  
 265 cesses and models themselves are not fully understood. One line of research that aims to under-  
 266 stand how models work is “mechanistic interpretability”. This is the field of reverse engineering  
 267 how models work, converting structures in the model into human interpretable concepts and al-  
 268 gorithms ([Nanda, 2021](#)).

269 [Olah et al. \(2020\)](#) found that in vision networks certain neurons are active<sup>3</sup> across a range of in-  
 270 puts. This idea is known as “polysemanticity” as the neuron represents multiple semantic mean-  
 271 ings. This poses a problem for interpretability as it is not sufficient to assign meaning to specific  
 272 neurons and check when they are active. This phenomenon has been shown to occur in LLMs  
 273 and has been demonstrated in toy examples ([Elhage et al., 2022](#)).

274 [Elhage et al. \(2022\)](#) propose the idea of “superposition” to explain why large models contain  
 275 polysemantic neurons. Superposition is the process of NNs representing more features than neu-  
 276 rons within the model. The features are no longer represented orthogonally in the representa-

<sup>3</sup>output a non-zero value.

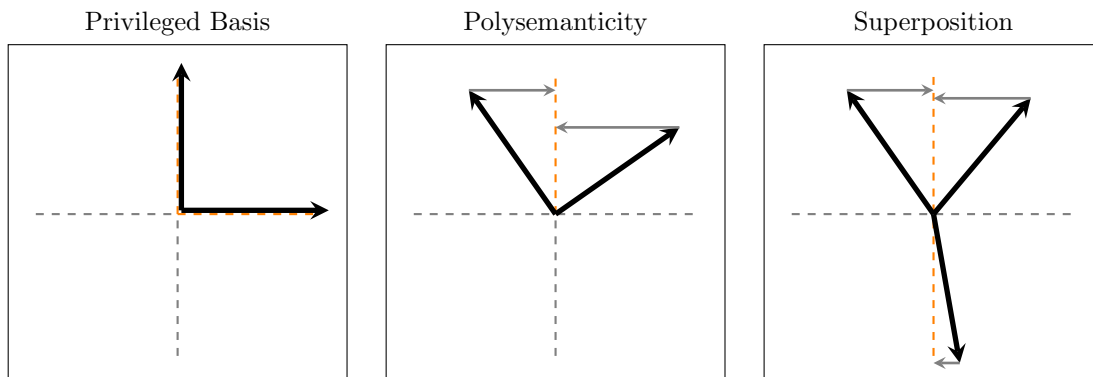


Figure 2.5: The three charts demonstrate the different ways a model may organise its representation space. This figure is a reproduction of [Elhage et al. \(2022\)](#) Figures 2 and 3. The privileged basis means that the representations are aligned with the architectures ‘preferred’ basis. Polysemanticity occurs when a specific neuron is activated by two, potentially unrelated, inputs. Finally, superposition occurs when the model has to embed more representations than privileged bases resulting in forced polysemanticity.

tion space but instead share components. This means that if only one (sparse) feature is active the non-orthogonal features will also be partially active. This idea is represented in Figure 2.5.

To disentangle the polysemantic neurons requires eliminating the superposition present in the model. This is a known problem known as “sparse dictionary encoding” ([Olshausen and Field, 1997](#)) in neuroscience, in which a signal in superposition is decomposed into sparse elements. **NEED BETTER UNDERSTANDING/EXPLANATION.** [Sharkey et al. \(2022\)](#) and [Cunningham et al. \(2023\)](#) apply the idea to NNs introducing the sparse autoencoder (SAE) which enforces sparsity in its internal representation. The SAE module is demonstrated in Figure 2.6.

An SAE is an adaptor that takes a model layer’s input and produces a replica of the layer’s output. In comparison to the model layer the SAE has a large hidden representation dimension in which sparsity is enforced. This can be achieved in multiple ways such as clamping to the  $k$  highest activations ([Makhzani and Frey, 2013](#)) or adding a sparsity regularising loss. After training the elements of the SAE hidden dimension are given interpretations to better understand the model. It is worth noting that SAEs have been shown to demonstrate subpar performance when used for interpretability ([Kantamneni et al., 2025](#)), though this project uses SAEs for dataset analysis.

SAEs are challenging to train and so for the purposes of this project only pretrained SAEs are used. [Bloom et al. \(2024\)](#) provides a large collection of open source SAEs with their corresponding models. This does limit the analysis as most models only have an SAE for a single layer.

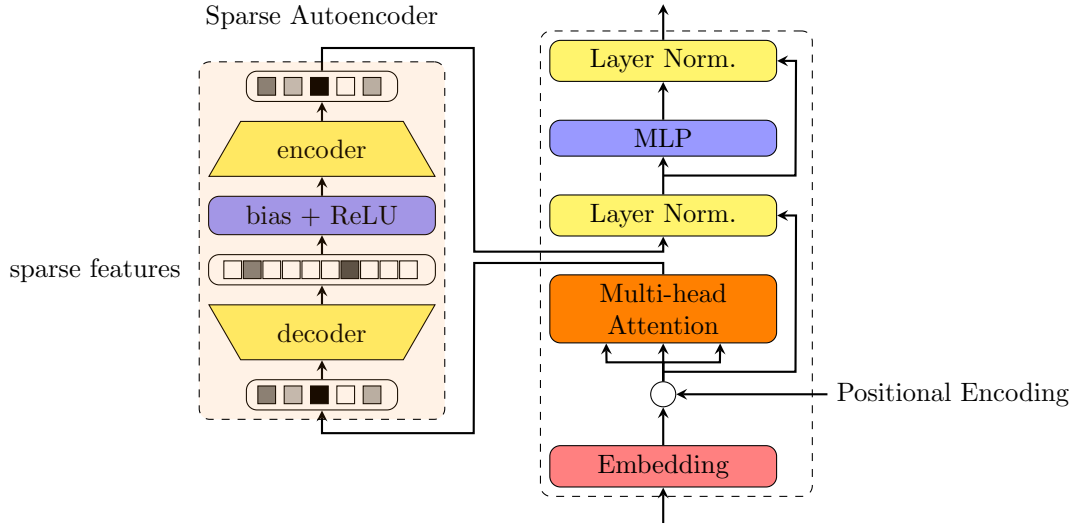


Figure 2.6: The sparse autoencoder is inserted at a specific location within the transformer block. The decoder transforms the input into a sparse vector representation and the decoder aims to reconstruct the input to feed back into the transformer. This way the sparse features do not have the superposition problem and relate to the models internal representation.



## Chapter 3

# Methodology

### 3.1 Steering Clear Environment

The setup of this environment follows (Krasheninnikov and Krueger, 2024). The model to steer is a 4-layer multi-layer perceptron (MLP) with residual connections (He et al., 2016) across all layers. After the MLP, a layernorm (Ba et al., 2016) and single layer classifier is added. All non-linearity throughout the model is gaussian error linear unit (GeLU). The hidden layers follow 512-512-256-512 architecture regardless of dataset specifics.

#### 3.1.1 Dataset

To control the behaviour of model and the steering approaches a synthetic dataset is used. Each dataset sample consists of  $m$  “attributes” which can take 8 possible discrete values. Each discrete value is represented by an “anchor” vector  $\mu_i \in \mathbb{R}^8, i \in \{1, 8\}$  sampled from a gaussian distribution  $\mathcal{N}(\mathbf{0}, 1)$ . To simulate real-world conditions gaussian noise is added to the samples from  $\mathcal{N}(\mathbf{0}, 0.1)$ . This does mean the values are generally highly seperable.

The dataset comprises of  $n$  input-output vectors where the input vector is the concatenation of  $m$  8-dimensional vectors. Thus, an input vector has length  $8m$  and the target vector has length  $m$ . Krasheninnikov and Krueger (2024) carry out a range of experiments for  $m \in \{60, 90, 120\}$  but always use 8 values represented by 8 dimensional vectors. They take a sample of 2,000,000 i.i.d samples but due to memory constraints only 500,000 are used in this project. No test set is used in either however an 80:20 split for training and validation set is used for identifying the best performing model.

#### 3.1.2 Pre-training

The MLP model is trained on the 500,000 training samples for 50 epochs using Adam (Kingma and Ba, 2014) with a learning rate of 0.001. As per Krasheninnikov and Krueger (2024) a cross

320 entropy loss is used to train the model. The model that achieves the best validation loss is saved  
321 and used for the steering task.

322 Regardless of exact epochs, learning rate or optimiser the best performing model should achieve  
323 close to 100%. Models used for the presented results achieved  $\sim 99\%$ .

### 324 3.1.3 Steering Task

325 The task is to successfully steer a model to always predict a specific value for a specific at-  
326 tribute. For example, the goal would be steer attribute 3 towards value  $\mu_1$ . Krasheninnikov and  
327 Krueger (2024) carry out three experiments to steer one, two or three attributes simultaneously.  
328 Instead, this reproduction will focus on steering only one attribute at a time.

329 As the attribute anchors are generated randomly there is no dataset bias towards any particular  
330 value. For this reason all attributes are steered towards value  $\mu_1$ .

331 In addition to the dataset generate, 4096 are generated as a training set for the steering ap-  
332 proaches and a further 1000 are generated as a test set. This is repeated 20 times to get an av-  
333 erage metric across steering approaches.

334 For each adaptor a range of hyperparameters is used to analyse the effect on steering perfor-  
335 mance. The number of steering examples is also varied from 4 up to 4096 increasing in powers  
336 of 2. Krasheninnikov and Krueger (2024) use this to analyse the representation densities effect  
337 of required number of examples.

338 **Steering metric.** As the model was trained to predict discrete attribute labels and the steer-  
339 ing adaptor simply aims for a specific attribute value it is possible to use the models accuracy  
340 on the target attributes. Krasheninnikov and Krueger (2024) use the full target output label,  
341 however, this was found to be dominated by unsteered attributes. Instead the accuracy on only  
342 the steered attribute is used.

## 343 3.2 Prompt Pairs Environment

344 Krasheninnikov and Krueger (2024) aim to analyse the effects of model feature density and the  
345 number of steering examples. To achieve this the set up a toy environment with synthetic data  
346 and a small, controllable model. To analyse similar effects, as well as the effects of the steering  
347 dataset, in a situation closer to real-world use requires utilising real-world models.

348 As only LLMs are considered this means the dataset is made of natural language prompts. Pos-  
349 itive and negative activations are sampled from the target layer and the last token. Generally,  
350 the two prompts used to extract positive and negative activations are identical except for the  
351 last token (Liu et al., 2024a; Tan et al., 2024; Turner et al., 2023).

### 3.2.1 dataset

Rather than generate thousands of entirely unique pairs of prompts a smaller set of templates with adjustable “contexts” and “targets” is used. An example template would be:

Everyone thought <context> would lose. In the end they <target>.

Then a range of relevant contexts (such as **the dancer** or **the driver**) and targets (such as **won** or **lost**) can be used. A standard prompt pair is therefore two prompts whose templates and contexts are identical but whose targets are in opposition. As the target is the last word activations can be extracted to steer the model from the negative target to the positive target.

Unlike the model written evaluations (Perez et al., 2023) dataset used in Tan et al. (2024) this dataset does not use multiple choice questions and instead a small range of target values. This allows the model to produce more tokens than just “yes”, “no” or “A”, “B”. This also allows for more control, such as changing context between positive and negative pairs or using “random” negative targets and meaningful positive targets. Note that all targets are *a single word* or ideally token to simplify the steering process.

Rather than a single dataset of this form multiple datasets are generated that cover a range of contexts and behaviours. They are aimed to be useful real-world situations however they are still fabricated and so are not a perfect representation. To generate the large number of templates, contexts and targets a set of example sentences were generated by GPT-5 (AI, 2025) and adjusted to extract templates, contexts and targets.

The full list of templates, contexts and targets are presented in §B.

### 3.2.2 Steering Task

The goal is to steer the model from generating the negative targets to generating the positive targets. As the negative and positive targets have many potential options it is possible the model does not produce the exact same positive target word, however, getting the correct semantic meaning is the aim.

For each of the datasets, after activation extraction, 100 are separated for testing and the rest used for adaptor training/initialisation. Similar to (Krasheninnikov and Krueger, 2024), a range of example pairs are used ranging from 4 example pairs to 1024 example pairs. The same range of adaptor hyperparameters are used to roughly compare the toy experiment to real-world scenarios. The experiments are also run without any adaptor to get a baseline value to compare from. **MIGHT BE WORTH DOING INDIVIDUAL DATASETS**. All steering metrics are averaged over the range of datasets to get an average efficacy across the datasets.

**Steering metrics.** Unlike the steering clear environment §3.1, it is hard to quantify accuracy on the steered attribute. Instead 3 metrics are used to evaluate the success of the steering approach.

Two are based on the SAE, §2.5, features of the final, target token of the model at the target layer:

1. During activation extraction, the SAE features that are consistently activated on the final token across all *positive* examples are identified. The average activation of these features during training is used to evaluate how well the steering adaptor is increasing the models representation of the target behaviour.
2. A random selection of SAE features that were consistently not activated for *both* examples during activation extraction are considered test features. The average activation of the test features during training is used to evaluate how well the adaptor effects only the target features.

Together these provide insight into how well the adaptor steers the internal model representations towards the intended behaviour.

To gather information about how well the adaptors effect the output the semantic similarity of the generated model token and the target model token is used. Using distilbert (Sanh, 2019) as a semantic embedding of words the average cosine similarity between generated word and target word across the test examples is used. This provides a metric for the semantic similarity of the models generated text and the target text. Only the target word is used as otherwise the metric is dominated by the similarity of the pregenerated text.

Each metric on its own is useful but can be prone to biases that the other metrics highlight. A more complete picture of how the adaptors and models behaved can be achieved by analysing all three metrics together.

### 3.3 Prompting LLMs

## 409 Chapter 4

## 410 Results

### 411 4.1 Prompt Pairs

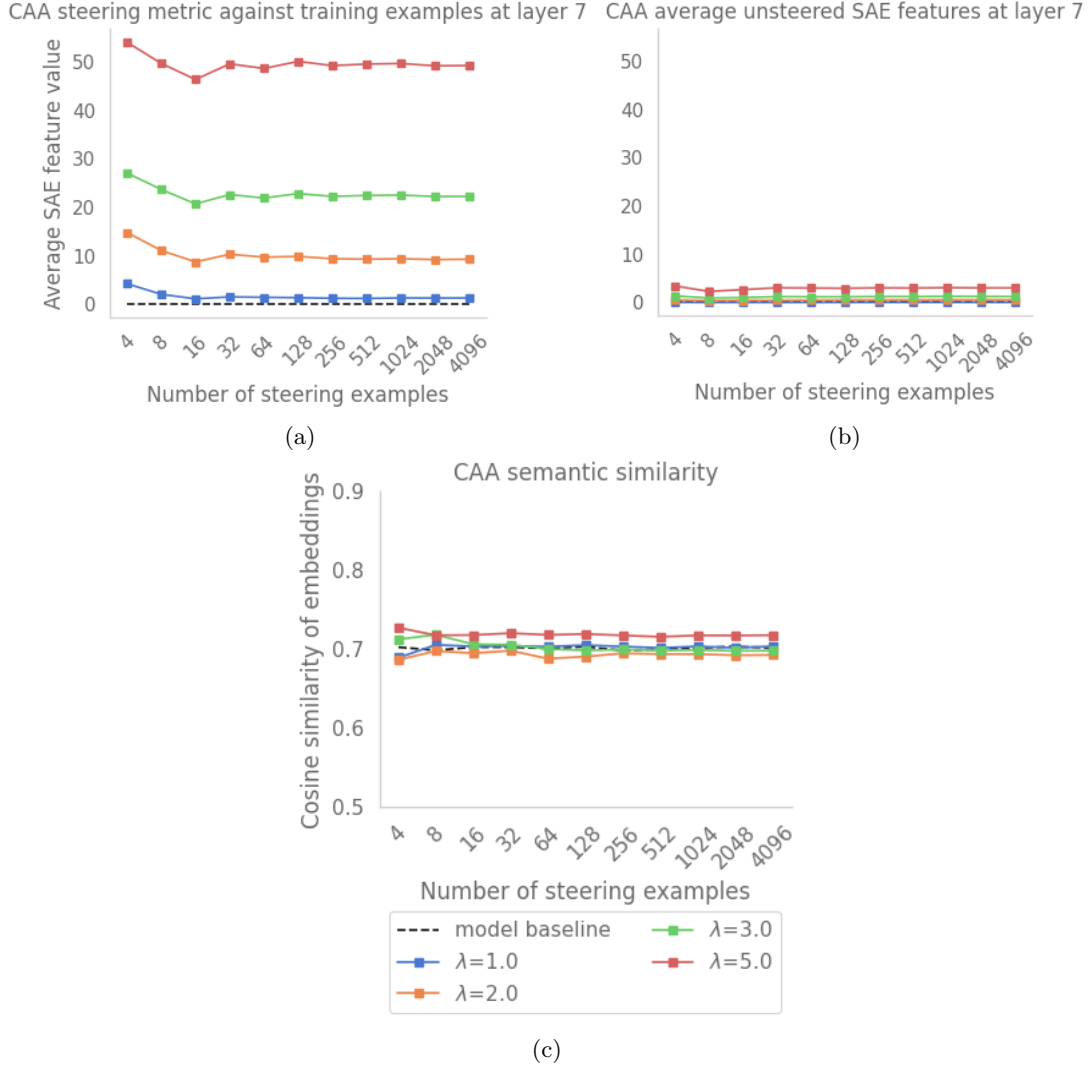


Figure 4.1: Result of running CAA (Rimsky et al., 2024) in the prompt pairs environment §3.2 on GPT2 at layer 7. (a) The average activation of the target SAE features. (b) The average activation of SAE features that were not present in the positive or negative examples. (c) The semantic similarity of the generated token and the target token. In all cases the dashed line is the baseline for the model without steering.

## 412 Chapter 5

## 413 Conclusion

### 414 5.1 Limitations

### 415 5.2 Future Work

# Bibliography

- AI, O. (2025). Introducing gpt 5. <https://openai.com/index/introducing-gpt-5/>. 15
- Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. 13
- Bloom, J., Tigges, C., Duong, A., and Chanin, D. (2024). Saelens. <https://github.com/jbloomAus/SAELens>. 11
- Chalnev, S., Siu, M., and Conmy, A. (2024). Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*. 2
- Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. (2023). Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*. 11
- Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. (2022). Toy models of superposition. *arXiv preprint arXiv:2209.10652*. 10, 11
- Engels, J., Liao, I., Michaud, E. J., Gurnee, W., and Tegmark, M. (2025). Not all language model features are linear. In *2025 Joint Mathematics Meetings (JMM 2025)*. AMS. 5
- Geiger, A., Wu, Z., Potts, C., Icard, T., and Goodman, N. (2024). Finding alignments between interpretable causal variables and distributed neural representations. In *Causal Learning and Reasoning*, pages 160–187. PMLR. 7
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 13
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR. 6
- Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. 6, 7



443 Kantamneni, S., Engels, J., Rajamanoharan, S., Tegmark, M., and Nanda, N. (2025). Are sparse  
444 autoencoders useful? a case study in sparse probing. In *Forty-second International Conference*  
445 *on Machine Learning*. 11

446 Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are rnns: Fast  
447 autoregressive transformers with linear attention. In *International conference on machine*  
448 *learning*, pages 5156–5165. PMLR. 9

449 Kharlapenko, D., neverix, Nanda, N., and Conmy, A. (2024). Extracting sae task features  
450 for in-context learning. [https://www.alignmentforum.org/posts/5FGXmJ3wqgGRcbyH7/](https://www.alignmentforum.org/posts/5FGXmJ3wqgGRcbyH7/extracting-sae-task-features-for-in-context-learning)  
451 [extracting-sae-task-features-for-in-context-learning](https://www.alignmentforum.org/posts/5FGXmJ3wqgGRcbyH7/extracting-sae-task-features-for-in-context-learning). 2

452 Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint*  
453 *arXiv:1412.6980*. 13

454 Krashenninikov, D. and Krueger, D. (2024). Steering clear: A systematic study of activation  
455 steering in a toy setup. In *MINT: Foundation Model Interventions*. 1, 2, 7, 8, 13, 14, 15

456 Liu, S., Ye, H., Xing, L., and Zou, J. Y. (2024a). In-context vectors: Making in context learn-  
457 ing more effective and controllable through latent space steering. In *Forty-first International*  
458 *Conference on Machine Learning*. 14

459 Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-  
460 H. (2024b). Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Con-*  
461 *ference on Machine Learning*. 6

462 Makhzani, A. and Frey, B. (2013). k-sparse autoencoders. *arXiv preprint arXiv:1312.5663*. 11

463 Marshall, T., Scherlis, A., and Belrose, N. (2024). Refusal in llms is an affine function. *CoRR*.  
464 1, 5, 6, 7, 8

465 Nanda, N. (2021). A comprehensive mechanistic interpretability explainer & glossary. [https:](https://www.neelnanda.io/mechanistic-interpretability/glossary)  
466 [//www.neelnanda.io/mechanistic-interpretability/glossary](https://www.neelnanda.io/mechanistic-interpretability/glossary). 10

467 Nanda, N., Conmy, A., smith, l., Rajamanoharan, S., Lieberum, T., Kramár, J.,  
468 and Varma, V. (2024). [full post] progress update 1 from the gdm mech in-  
469 terp team. [https://www.alignmentforum.org/posts/C5KAZQib3bzzpeyrg/](https://www.alignmentforum.org/posts/C5KAZQib3bzzpeyrg/full-post-progress-update-1-from-the-gdm-mech-interp-team)  
470 [full-post-progress-update-1-from-the-gdm-mech-interp-team](https://www.alignmentforum.org/posts/C5KAZQib3bzzpeyrg/full-post-progress-update-1-from-the-gdm-mech-interp-team). 2

471 Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. (2020). Zoom in:  
472 An introduction to circuits. *Distill*, 5(3):e00024–001. 10

473 Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strat-  
474 egy employed by v1? *Vision research*, 37(23):3311–3325. 11

475 Perez, E., Ringer, S., Lukosiute, K., Nguyen, K., Chen, E., Heiner, S., Pettit, C., Olsson, C.,  
476 Kundu, S., Kadavath, S., et al. (2023). Discovering language model behaviors with model-

written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13387–13434. 2, 15

Rimsky, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., and Turner, A. (2024). Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522. 1, 4, 5, 6, 7, 8, 18

Sanh, V. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Proceedings of Thirty-third Conference on Neural Information Processing Systems (NIPS2019)*. 16

Sharkey, L., Braun, D., and Beren, M. (2022). [interim research report] taking features out of superposition with sparse autoencoders. <https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition>. 11

Singh, S., Ravfogel, S., Herzig, J., Aharoni, R., Cotterell, R., and Kumaraguru, P. (2024). Representation surgery: theory and practice of affine steering. In *Proceedings of the 41st International Conference on Machine Learning*, pages 45663–45680. 1

Space, L. (2023). Commoditizing the petaflop - with george hotz of the tiny corp. <https://www.latent.space/p/geohot>. 8

Tan, D., Chanin, D., Lynch, A., Kanoulas, D., Paige, B., Garriga-Alonso, A., and Kirk, R. (2024). Analyzing the generalization and reliability of steering vectors–icml 2024. *arXiv e-prints*, pages arXiv–2407. 1, 5, 14, 15

Team, G. (2025). Gemma 3. <https://goo.gle/Gemma3Report>. 8

Turner, A. M., Thiergart, L., Udell, D., Leech, G., Mini, U., and MacDiarmid, M. (2023). Activation addition: Steering language models without optimization. *CoRR*. 4, 14

Turner, R. E. (2023). An introduction to transformers. *arXiv preprint arXiv:2304.10557*. 9

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. 9, 10

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*. 9

Wehner, J., Abdelnabi, S., Tan, D., Krueger, D., and Fritz, M. (2025). Taxonomy, opportunities, and challenges of representation engineering for large language models. *arXiv preprint arXiv:2502.19649*. 2

Wu, Z., Arora, A., Wang, Z., Geiger, A., Jurafsky, D., Manning, C. D., and Potts, C. (2024).

- 511 Reft: Representation finetuning for language models. *Advances in Neural Information Pro-*  
512 *cessing Systems*, 37:63908–63962. 1, 6, 7, 8
- 513 Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P.,  
514 Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences.  
515 *Advances in neural information processing systems*, 33:17283–17297. 9

## <sup>516</sup> **Appendix A**

## <sup>517</sup> **Gumbel Softmax**

## 518 **Appendix B**

## 519 **Prompt Pairs Dataset**