Does Size Matter?

The Effect of Sample Size on Steering

PRHC8[1]

Machine Learning MSc

Daniel Tan & Brooks Paige

Submission date: 8 September 2025

[1] **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

Acknowledgements

I want to thank my family, especially my mum and dad who have supported me throughout my life and encouraged me to explore the world. To my wider family who have both directly and indirectly supported me throughout the masters.

Particular thanks goes to Mia who has supported me throughout the highs and lows, stress and anxiety of the masters and the year leading up to it. Thank you for listening to my endless rambles about the project and all the other projects I've had; keeping me focused, motivated, and on track to complete the thesis.

I would like to thank my supervisor Daniel Tan and the support of the ML Alignment and Theory Scholars London office for guiding the project and providing feedback along the way. Additionally, to Alex for providing detailed, actionable feedback on my early rough drafts.

Abstract

Summarise your report concisely.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter an overview of the project and this report is detailed. This includes what the project involves as well as why this is a worthwhile project. The motivation for the project will be briefly discussed along with the benefits of the analysis carried out.

Related work to this project, including projects which precede this one and projects which cover separate but related issues, is presented. The differences between this project and the related work is explained.

Finally the contributions which this project make to the field are detailed. These contributions are justified in Chapter §4 and reiterated in Chapter §5.

1.1 Motivation

In recent years the use of machine learning (ML) models has rapidly increased across multiple sectors of life. In particular the rise in deep learning models following the release of Alexnet (Krizhevsky et al., 2012) has introduced new problems with the use of ML. This has lead to rapid development in field resulting model capabilities surpassing human capabilities in an increasing range of activities (Kiela et al., 2021; Openai, 2025; Phan et al., 2025; xAI, 2025). With the introduction of large language model (LLM) chatbots starting with ChatGPT (Openai, 2022), ML models are increasingly becoming part of everyday life.

There are already cases of serious harm to users (Landymore, 2024; Østergaard, 2023) and the potential for humanity scale risk posed by increasingly capable systems has been posed (Boffey and Wilding, 2025; Gambín et al., 2024; Kulveit et al., 2025). There are many approaches to try and solve these problems ranging from law and policy to model training techniques and dataset design.

The field of preventing risks posed by ML is known as AI (artificial intelligence) safety. **A primary goal of AI safety is to alter models to be more "helpful", "honest", and "harmless" without effecting their performance**. A promising avenue of AI safety, especially for LLMs, is *steering adaptors* a subset of representation engineering (Wehner et al., 2025). The basic concept behind steering adaptors is to *manipulate the internal representation space* of models. By assuming certain *directions within representation space represent understandable concepts* it is possible to perturb a models vector representation towards a target concept. This idea has already been demonstrated by Mikolov et al. (2013) in word embedding models.

There is large amounts of anecdotal evidence of steering adaptors successfully changing model behaviour with increasing studies looking into the theory (Im and Li, 2025; Krasheninnikov and Krueger, 2024) and effectiveness (Tan et al., 2024; Wehner et al., 2025) of steering adaptors on modern models. There, however, continue to be a lack of large scale studies comparing a range of adaptors in realistic settings with varying input size and hyperparameter choice.

This project and report aim to remedy this and provide more quantitative and qualitative analysis of common steering techniques on LLMs. The project will hopefully provide insight into the effectiveness of these techniques as well as the suitability of a selection of suggested metrics.

1.1.1 Benefits of Steering Vectors

Unlike other techniques to change a models behaviour such as reinforcement learning (Ouyang et al., 2022; Sutton and Barto, 1998) or finetuning (Hu et al., 2022) steering adaptors require far less compute and data as they do not require changing the many parameters of the chosen model. Once implemented they are efficient to run during inference and have been shown to have low impact on model capabilities (Qiu et al., 2024; Stickland et al., 2024). Importantly, these techniques are still compatible with other alignment techniques working to provide additional adjustments.

By nature they inherently provide some interpretability by demonstrating how certain model representations effect the output. This, in turn, allows for more precise control over model behaviour than other techniques with potential direct feedback between the user and the model behaviour. However, these techniques frequently require full access to the model's parameters and architecture.

1.2 Research Questions

This project aims to provide a detailed, quantitative comparison of steering adaptors in realistic settings. This work follows the work on analysing the generalisability of steering vectors in Tan et al. (2024) and the survey of current representation engineering techniques from Wehner et al. (2025). The approach used in this project extends the analysis of steering example size on adaptor performance in the toy setting proposed by Krasheninnikov and Krueger (2024) to the natural language setting using LLMs. Furthermore, a comparison of the new affine concept editing (Marshall et al., 2024, ACE) adaptor is made against existing adaptors. The three research questions therefore focus on reproducing prior results and providing new results and analysis.

1. Are the findings of Krasheninnikov and Krueger (2024) reproducible and sound? Where does the adaptor proposed by Marshall et al. (2024) fit into their analysis?
2. How do the number of examples effect the performance of steering adaptors? Does this match the results in the toy setting of Krasheninnikov and Krueger (2024)? What metrics provide meaningful insight into steering performance?
3. How can the success of steering adaptors be quantified in the natural language setting?

Overall this projects aims to provide more insight into when steering methods work focusing on the number of steering examples. Additional discussion on the shortcomings of the steering approaches are provided throughout as well as suggestions for real world use.

1.3 Compute Environment

The project was written and programmed on my personal laptop. This is a Dell Inspiron 15" laptop running Arch Linux and i3 window manager. The laptop has 16GB of memory with an 8 core Intel 13th generation i7 processor.

Preliminary tests of the experiments run in this project were done on this laptop. However, the official results were run on RunPod, which provides a suite of possible GPU environments. For all the results presented in Chapter §4 the RTX 2000 Ada environment was used. This environment includes a single RTX 2000 Ada GPU with 16GB of VRAM, 31GB of RAM, and 6 virtual CPUs. The cost to run this environment was $0.24 per hour. The funding for the compute environment was provided by my supervisor through the ML Alignment and Theory Scholarship which he is part of.

Occasional use of the free Google Colab environment was used to verify the experiments ran at scale. No results presented in this report were generated from these runs.

1.4 Generative AI Disclosure

This project aims to steer generative AI models (genAI) such as GPT 2 (Radford et al., 2019) and thus genAI was used in generating the raw results presented in Chapter §4.

However, outside of the direct prompting of agents to analyse the steering adaptors presented in Chapter §2, no generative AI was used throughout the project or report. There are two exceptions:

- Generating a dataset of prompt-completion templates detailed in Appendix B.
- Generating a random point cloud in Tikz for Figure 2.1.

Spellcheck was performed by Neovim and its built in spellchecker with no use of AI tools such as Grammarly. All citations were found and generated through Google Scholar. Assistance for Tikz diagrams was found through https://www.tikz.dev.

However, with the proliferation of AI in search engines it is not possible to state that genAI did not influence any of the LaTeX or code snippets. I endeavoured to not use genAI as much as possible throughout the last 3 months.

1.5 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 focus on the effect of steering example size. Inspired by the low-rank representation finetuning (Wu et al., 2024, LoReFT) (A steering adaptor which steers representations in a projected, low-rank space) they introduce their own technique, low-rank representation steering (LoReST), and demonstrate competitive performance to the other techniques.

To verify this work, this thesis reproduces the same toy environment described in Section §3.1 and validates the conclusions drawn. Additionally, this reproduction introduces the affine concept editing adaptor proposed by Marshall et al. (2024) to compare against the adaptors used in Krasheninnikov and Krueger (2024). This thesis then expands the analysis to LLMs and demonstrates similar adaptor performance in relation to the number of training examples suggesting the same preference for low-rank methods in the large data regime and affine methods in the small data regime. Furthermore, this thesis shows the consistency of LoReST in both data regimes.

Tan et al. (2024) 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 depth to determine which properties cause steering techniques to fail.

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

Wehner et al. (2025) present a full taxonomy of current steering vector approaches (more generally *representation engineering*). The paper covers a range of topics within representation engineering that have been carried out by the community. These focus on the types of adaptors used, the prompting framework, linear vs. non-linear adaptors, the concepts that are steered, etc.

This thesis continues these comparisons by analysing the effect of the dataset and number of steering examples used in the large language model setting. It is impractical to expand the experiments to all the approaches described in Wehner et al. (2025) due to time constraints so only those discussed in Krasheninnikov and Krueger (2024) are analysed.

Sparse autoencoders as steering vectors There are multiple papers that utilise sparse autoencoders (SAEs) as steering vectors (Chalnev et al., 2024; Kharlapenko et al., 2024; Nanda et al., 2024). SAEs work by transforming the dense model representations into a high-dimensional, sparse vector space. This allows individual dimensions to represent unique, human-understandable concepts effectively "decoding" the model representation. These approaches utilise this fact, that SAEs decode high level concepts from the models intermediate representation to steer the model towards or away from said concepts. This process works by "reversing" the SAE procedure and converting concepts into potential representation perturbations.

This thesis, in contrast, uses the SAE features to evaluate models on free-text responses rather than utilising SAEs to steer the model. The SAE features provide a metric to evaluate how well the models internal representation has been effectively steered. A detailed description of SAEs is provided in Section §2.5 and their use in this project is described in Section §3.2.

1.6 Contributions

In answering the questions posed in Section §1.2 this project provides the following contributions

- Verification of the results presented by Krasheninnikov and Krueger (2024) in their toy experiment.
- Comparison of contrastive activation addition (Rimsky et al., 2024, CAA), affine concept editing (Marshall et al., 2024, ACE), LoReFT and LoReST on large language models.
- In depth analysis on the effect of the number of steering examples on the adaptor performance.
- A set of promising LLM steering metrics along with qualitative analysis of the steered model output.

All code for the project is available at this code repository and all the LaTeX including diagrams is available at this, separate, code repository.

Chapter 2

Background

In this chapter the notation, phrases and concepts that are used throughout the document are explained. An overview of general model alignment as well as the specifics of alignment via steering adaptors is described. This includes a history of the different techniques and the details of the four methods used in this project.

The history of large language models and why they are important in current research is outline. Additionally the challenges that are faced in interpreting these large models is explained. The solutions to these challenges present possible metrics that can be used to analyse the effectiveness of steering adaptors.

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 (such as responding truthfully) is considered “positive” and undesired model behaviour (such as lying or subverting) 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 (such as a picture of a cat and a response which either tells the truth, ”This is a cat”, or lies, ”This is a dog”) however they are more specific than would be using during training.

The specificity of steering examples is to insure the model adjust representations that elicit the target behaviour and avoid causing larger changes to behaviour.

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, α, β, γ, and matrices are represented by boldface capital letters, $\mathbf{A}, \mathbf{B}, \mathbf{C}$. In general $\mathbf{R}$ represents an orthonormal projection matrix into a lower dimension, $\mathbf{W}$ represents a general weight matrix, and $\mathbf{b}$ represents a bias vector. 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. The mean of a set of activations, $\mathcal{A} = \{\mathbf{a}_i\}_{i<n}$, is denoted $\mu_{\mathcal{A}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{a}_i$. Frequently the set of set of activations will be the positive activation or negative activation set, in this case the mean is denoted $\mu_{\mathbf{a}^+}$ or $\mu_{\mathbf{a}^-}$ respectively.

2.2 Model Alignment

As models increase in capabilities (Kiela et al., 2021; Openai, 2025; Phan et al., 2025; xAI, 2025) they pose an increasing risk to their users (Landymore, 2024; Østergaard, 2023) and potentially humanity at large (Boffey and Wilding, 2025; Gambín et al., 2024; Kulveit et al., 2025). The underlying issue is that these models may be *misaligned* (Leike et al., 2018), that is to say they do not behaviour in line with users intentions. The problem of aligning models is referred to as the *alignment problem* or, within reinforcement learning Sutton and Barto (1998), the *agent alignment problem* (Leike et al., 2018).[1]

[1] An *agent* is a reinforcement learning term for any entity that interacts with a learning environment and updates it's internal state to better achieve a predetermined goal. In this case, the model behaves as an agent.

Leike et al. (2018) present the agent alignment problem and propose *reward modelling* as a potential avenue to align agents. They outline a couple of assumptions as to whether reward modelling is suitable:

- It is possible to sufficiently learn user intentions.
- It is cheaper to evaluate outcomes than produce the "correct" behaviour.

Working on this, Ouyang et al. (2022) apply these ideas to large language models (LLM)s. The goal is to transform purely predictive LLMs into assistants that are "helpful", "honest", and "harmless". This is achieved by utilising human feedback on LLM output as rewards for reward modelling. The idea reinforcement learning from human feedback (RLHF) had been developed previously by Christiano et al. (2017) but had not been applied to LLMs. Applying the idea to modern LLMs led to the invention of the modern AI chatbot (Openai, 2022).

RLHF has been shown to be very effective in transforming the behaviour of models. However, it is still possible for RLHF "aligned" models to be misaligned (Di Langosco et al., 2022; Landymore, 2024). Furthermore, this approach to alignment is very costly requiring human annotators, reviewers and the costly process of finetuning. Techniques to mitigate this have been proposed including RLxF (Ji et al., 2023), utilising both human and AI feedback, representation finetuning (Wu et al., 2024), and parameter efficient finetuning (Hu et al., 2023).

Representation finetuning or more generally representation engineering (Wehner et al., 2025) presents a promising avenue for alignment (Im and Li, 2025; Krasheninnikov and Krueger, 2024; Wehner et al., 2025). Rather than requiring large amounts of human annotated data and changing model weights only the representations need editing. This manipulation of representations can occur after model training, with a smaller dataset, and incurs limited overheads during inference. This thesis focuses on *steering adaptors*, a subset of representation engineering.

2.3 Steering Adaptors

The general form of a steering adaptor is a simple module that augments a layer's output. The idea is to change the internal representation away from a harmful or misaligned concept towards one that is aligned to the users intentions. The goal is to keep all other

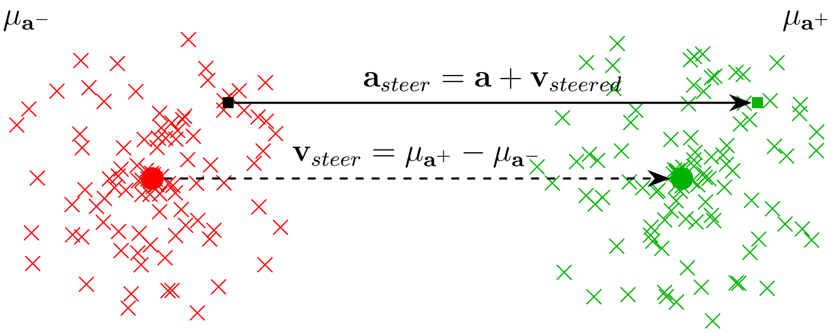

$\mu_{\mathbf{a}^-}$
$\mu_{\mathbf{a}^+}$
$\mathbf{a}_{steer} = \mathbf{a} + \mathbf{v}_{steered}$
$\mathbf{v}_{steer} = \mu_{\mathbf{a}^+} - \mu_{\mathbf{a}^-}$

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. The average displacement between negative behaviour, $\mu_{\mathbf{a}^-}$, and positive behaviour, $\mu_{\mathbf{a}^+}$, represents the direction the target concept lies on. Applying this to a new point (black square) produces a new point (green square) with the desired behaviour whilst maintaining unrelated properties.

aspects of the representation intact so that the performance of the model is not hindered.

Rather than large amounts of annotated data or large weight matrices these techniques require a handful of positive and negative examples. Given their lightweight nature these techniques have shown promising results (Im and Li, 2025; Krasheninnikov and Krueger, 2024; Wehner et al., 2025; Wu et al., 2024).

2.3.1 Contrastive Activation Addition (CAA)

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, `The prime minister of the UK is Count Binface` and `The prime minister of the UK is Sir Kier Starmer`, representing undesired (untruthful) and desired (truthful) behaviour. The model represents these two sentences with minor differences in its internal representation space. The difference of these representations gives a direction in feature space that corresponds to shifting the models output from undesired behaviour towards desired behaviour (a "truthfulness" direction). Importantly, the context of the output is maintained, "who is the prime minister of the UK?", but the model no longer produces the undesired (false) string, "Count Binface". 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

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

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

$$\mathbf{a}_{steered} = \mathbf{a} + \lambda \mathbf{v}_{steer} \tag{2.1}$$

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

This approach has a few drawbacks (Engels et al., 2025; Marshall et al., 2024; Tan et al., 2024) due to its assumptions. Primarily this approach does not consider how much of a behaviour is already present. This means the steering parameter does not fully determine the strength of the desired behaviour. Furthermore, Tan et al. (2024) demonstrate that CAA is unable to consistently steer a model towards target behaviours and in some cases may steer the model *towards the negative behaviour*. The approach assumes that concepts in activation space are linear which Engels et al. (2025) show is not universal. Techniques such as affine concept editing (ACE) Section §2.3.2 use an affine approach to overcome these drawbacks.

2.3.2 Affine Concept Editing (ACE)

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

$$\begin{aligned} \mathbf{a} &= \mathrm{proj}_{\mathbf{r}}^{\perp}(\mathbf{a}) + \mathrm{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) \\ &= \mathrm{proj}_{\mathbf{r}}^{\perp}(\mathbf{a}) + \alpha\mathbf{r}. \end{aligned} \tag{2.2}$$

This shows that CAA (Rimsky et al., 2024) does not account for how much a behaviour may already be present in an activation, represented by $\alpha\mathbf{r}$. However $\alpha = 0$ is not necessarily the absence of the target behaviour, that is, it is not (generally) the case that $\mathbf{0}$ represents lack of behaviour. Instead assume some vector $\mathbf{a}_0$ represents the lack of the target behaviour. Equation 2.2 can incorporate this idea as follows

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

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

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

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

$$\mathbf{a}_{\text{steered}} = \mathbf{a}_0 - \mathrm{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0\mathbf{r} + \alpha\mathbf{r}. \tag{2.3}$$

This process is described graphically in Figure 2.2.

[1] As $\mathbf{a}_0$ exists as a reference point along the steered direction.

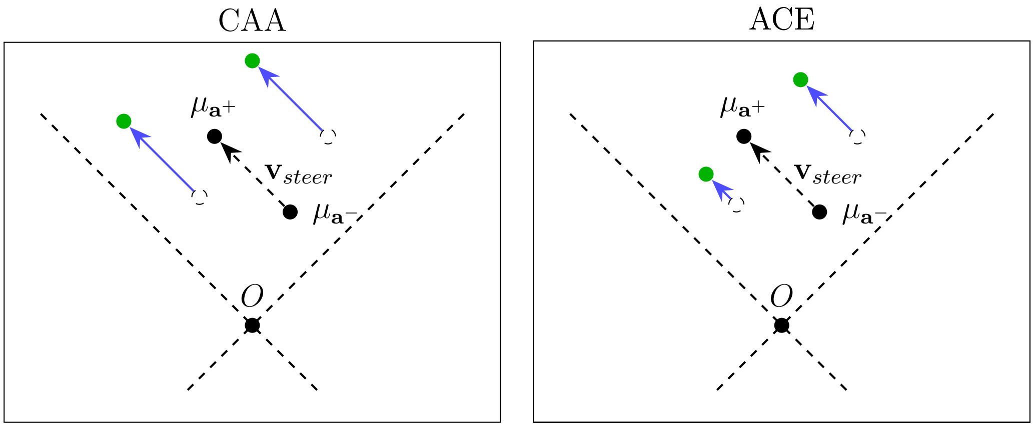


Figure 2.2: A comparison of CAA (Rimsky et al., 2024) and affine concept editing (Marshall et al., 2024). This is a reproduction of Figure 1 in Marshall et al. (2024) with the steering towards the positive examples instead. Compared to CAA, ACE does not adjust perpendicular components but correctly adjusts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

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

$$\mathbf{r} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a}_i^+ - \mathbf{a}_i^-) \qquad\qquad \alpha_0 = \mathrm{proj}_{\mathbf{r}}^{\parallel}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{a}_i^-\right)$$

A proof that these definitions are the optimal assignments is presented in Appendix A.

This approach is no longer a linear edit to the activations and now includes a bias term, $\mathbf{a}_0$. This is therefore affine and hence the name *affine concept editing*.

2.3.3 Low-rank Representation Finetuning (LoReFT)

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

Wu et al. (2024) present a low-rank steering adaptor inspired by parameter-efficient finetuning methods such as LoRA (Hu et al., 2022), DoRA (Liu et al., 2024b) and adaptor-based methods (Houlsby et al., 2019). Unlike steering these approaches aim to finetune a model using reduced parameter counts compared to the original model. This approach uses a small set of parameters that augment a model layers output, these can be trained whilst the original model weights remain the same greatly reducing the number of parameters needed to be trained.

Rather than finetuning a model, LoReFT edits the representations of the model, equivalent to steering. To reduce the parameter count this learnable steering is performed in a low-rank space. The specific approach is based on the distributed interchange intervention (Geiger et al., 2024, DII). DII is a version of interchange intervention whereby a causal model (with human understanable structure) is aligned with a neural network (NN) that exhibits the same behaviour. By identifying possible neurons that exhibit the same behaviour as intermediate nodes in the causal model it is possible to intervene with the neuron's value across different inputs and verify the output behaviour matches how the causal model behaves. This is very similar to the goal of steering adaptors except there is no associated causal model, simply an alignment goal. Rather than brute-force search for the corresponding neuron for a causal node, Geiger et al. (2024) suggest projecting the respresentations into a lower dimensional space, comparing the causal node value and the NNs neuron, then projecting back. Wu et al. (2024) present the following form of DII for the case of representation finetuning.

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

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

$$\mathbf{a}_{\text{steered}} = \mathbf{a} + \mathbf{R}^T(\mathbf{W}\mathbf{a} + \mathbf{b} - \mathbf{R}\mathbf{a}). \qquad (2.4)$$

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.

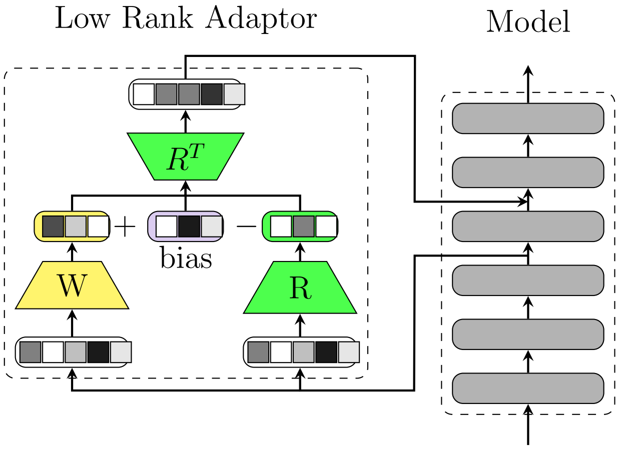


Figure 2.3: This figure demonstrates how the low-rank representation finetuning adaptor (Wu et al., 2024) operates. Unlike methods such as LoRA (Hu et al., 2022) this does replace the layer weights but simply adds the layer output. LoReST (Krasheninnikov and Krueger, 2024) behaves similarly though the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

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

Unlike CAA and ACE, LoReFT requires paired datapoints as the adaptor needs to learn a transformation from negative examples to positive 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 Krasheninnikov and Krueger (2024) with their low-rank representation steering adaptor.

QR decomposition

Any real-valued square matrix, $\mathbf{A}$, can be decomposed as

$$\mathbf{A} = \mathbf{Q}\widehat{\mathbf{R}}$$

where $\mathbf{Q}$ is an orthogonal matrix (that is $\mathbf{Q}^T = \mathbf{Q}^{-1}$) and $\widehat{\mathbf{R}}$ is upper triangular (not to be confused with $\mathbf{R}$ the projection matrix above). $\mathbf{Q}$ can be viewed as a new basis where the first column of $\mathbf{A}$ is normalised, then the perpendicular component of the next column of $\mathbf{A}$ is normalised. This process repeats forming a new orthonormal basis based on $\mathbf{A}$.

The process described above is known as the *Gram-Schmidt process*. By running this process on a hidden matrix $\mathbf{A} \in \mathbb{R}^{d \times r}$ an orthonormal matrix $\mathbf{Q} \in \mathbb{R}^{d \times r}$ can be generated. $\mathbf{Q}$ can then be used as the projection matrix discussed above.

2.3.4 Low-rank Representation Steering (LoReST)

Krasheninnikov and Krueger (2024) suggest modifying LoReFT 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 but keep the benefits of LoReFT. By dynamically dropping dimensions the adaptor has more freedom to optimise the rank of the projection.

Krasheninnikov and Krueger (2024) define an orthogonal projection

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

where $\mathbf{R} \in \mathbb{R}^{r \times d}$ is a learnable low-rank projection matrix, $\mathbf{l}$ is a learnable Gumbel Softmax distribution probabilities, and τ is the temperature. As with LoReFT, $\mathbf{R}$ is an orthogonal projection achieved by differentiable QR decomposition. In comparison to LoReFT Equation 2.4 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

$$\mathbf{a}_{\text{steered}} = \mathbf{a} - (\mathbf{a}\mathbf{Q})\mathrm{diag}(\mathbf{p})\mathbf{Q}^T + \mathbf{b}. \tag{2.5}$$

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 ϕ 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.

Gumbel-Softmax

The aim of Gumbel-Softmax is to sample from a categorical distribution whilst maintaining backpropagation. This is useful in LoReST as it allows the adaptive selection of low-rank dimensions by modelling the choice to keep or drop a dimension as a 2 value categorical distribution.

Consider sampling from a categorical distribution with K categories. Let X represent the random variable, then

$$X = \max \{ i : \pi_1 + \pi_2 + \cdots + \pi_{i-1} \leq U \}$$

where π_j is the probability of category j and $U \sim Uniform(0, 1)$.

Due to the max operation this is not differentiable. Using the reparameterisation trick, and taking samples $G_i \sim \mathit{Gumbel}(0, 1)$, X can be represented as

$$\arg\max_i \{ G_i + \log(\pi_i) \} .$$

This is still not differentiable however, when taking onehot vector representations, $\arg\max$ can be approximated by softmax. In practice an additional temperature variable τ is introduced giving

$$x_i = \frac{\exp\left(\frac{G_i + \log(\pi_i)}{\tau}\right)}{\sum_j \exp\left(\frac{G_j + \log(\pi_j)}{\tau}\right)}.$$

As $\tau \rightarrow 0$ the computation approaches $\arg\max$ and as $\tau \rightarrow \infty$ the computation approaches uniform.

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 are trained to generate text mimicking the natural language training distribution and are characterised by incredibly large parameter counts. Some models are as large as 671 billion (DeepSeek-AI, 2024) and even small models have as many as 1 billion (Gemma-Team, 2025).

LLMs are not trained to classify or fit a dataset in the classical sense but instead to produce more data as if it were sampled from the underlying training distribution. In practice this means producing coherent natural language which they incredibly well (Landymore, 2024). The key improvement in modern LLM performance is due to the Transformer (Vaswani et al., 2017) and their many derivatives (Katharopoulos et al., 2020; Wang et al., 2020; Zaheer et al., 2020).

2.4.1 Transformers

Transformers (Vaswani et al., 2017) are now a mainstay of modern deep learning.[2] They utilise the attention mechanism to dynamically transform (sequential) input based on the surrounding context.

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

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

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

Modern Transformers contain attention blocks each containing multiple "attention heads" that use the above mechanism. This allows the model to respond dynamically

[2] 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).

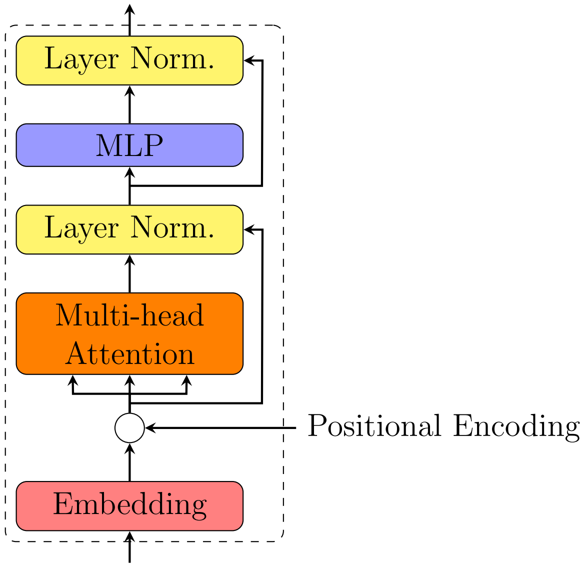


Figure 2.4: A diagram of the standard transformer decoder block. This is based on Figure 1 of Vaswani et al. (2017). This is a single layer in a large language model where the output of one block is fed into the input of the next.

to a large range of inputs. After this attention block a standard multi-layer perceptron (MLP) is added. This constitutes the transformer block and is visualised in Figure 2.4.

The ability for Transformers to utilise context in surrounding input values makes them particularly suited to natural language processing (NLP). The meaning of words in a sentence depend on the words that surround it. Furthermore, the words depend on each other in different ways depending on the context. This is precisely how transformer attention blocks work allowing them to parse natural language far better than previous attempts.

For Transformers to work on natural language the input needs to be tokenized into discrete tokens. These tokens can then be converted to unique numbers and later represented as input features. To aid the model, the position of the token within the sentence is also encoded this is known as "positional encoding". This allows the model to distinguish between the two instances of "can" in the sentence "can you pass me the can".

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

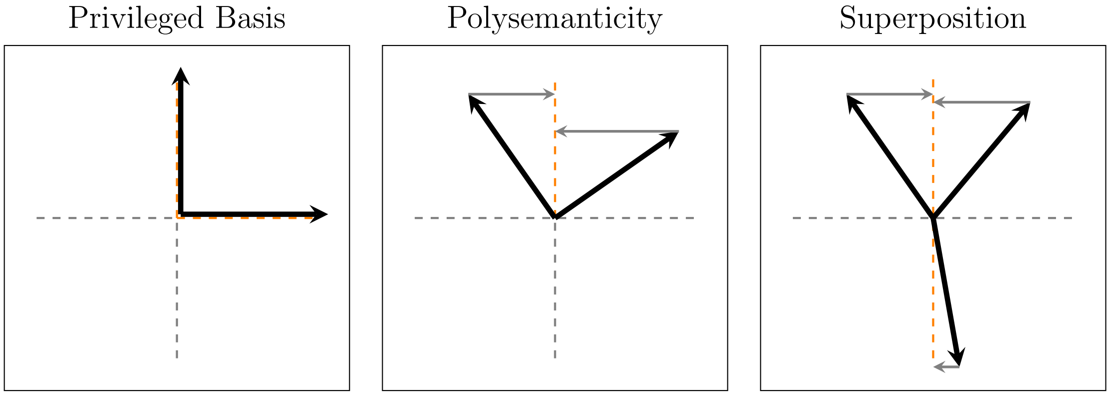


Figure 2.5: The three charts demonstrate the different ways a model may organise it's representation space. This figure is a reproduction of Elhage et al. (2022) Figures 2 and 3. The privileged basis means that the representations are aligned with the architectures 'preferred' basis. Polysemanticity 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.

2.5 Sparse Auto-Encoders

Though the processes to build, train and use a machine learning model are known, these processes and models themselves are not fully understood. One line of research that aims to understand how models work is "mechanistic interpretability". This is the field of reverse engineering how models work, converting structures in the model into human interpretable concepts and algorithms (Nanda, 2021).

Olah et al. (2020) found that in vision networks certain neurons are active[3] across a range of inputs. This idea is known as "polysemanticity" as the neuron represents multiple semantic meanings. This poses a problem for interpretability as it is not sufficient to assign meaning to specific neurons and check when they are active. This phenomenon has been shown to occur in LLMs and has been demonstrated in toy examples (Elhage et al., 2022).

Elhage et al. (2022) propose the idea of "superposition" to explain why large models contain polysemantic neurons. Superposition is the process of NNs representing more features than neurons within a specific hidden layer. The features are no longer represented orthogonally in the representation space but instead share components. This

[3] output a non-zero value.

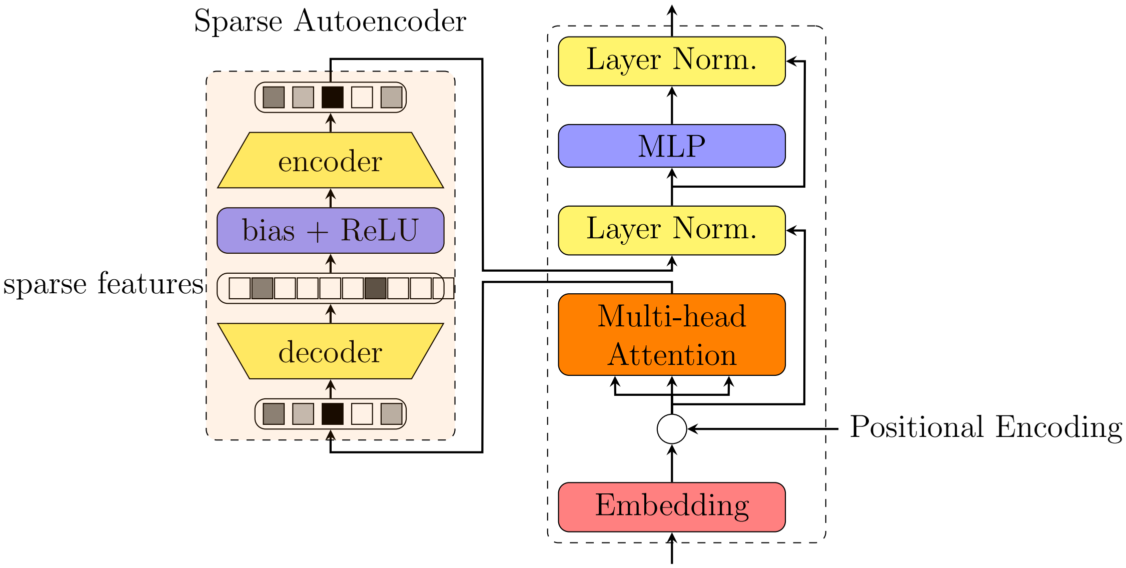


Figure 2.6: The sparse autoencoder is inserted at a specific location within the transformer block. The decoder transforms the input into a sparse vector representation and the decoder aims to reconstruct the input to feed back into the transformer. This way the sparse features do not have the superposition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

means that if only one (sparse) feature is active the non-orthogonal features will also be partially active.

The ideas of polysemanticity and superposition are 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. 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 (e.g. 24576 for GPT-2 SAEs compared to 756 (Bloom et al., 2024)). 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). In contrast, this thesis utilises SAEs to track model representation changes and spurious correlations, a use case Kantamneni et al. (2025) suggest is still valid.

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.

2.5.1 Metric Challenges

As Kantamneni et al. (2025) outline, SAEs do no provide a base truth for how the model represents concepts. This means that using SAEs to extract concepts, or in the case of the project, using SAEs to evaluate the success of an adaptor will be imperfect. Comparing across SAE features can give an insight into how a steering adaptor changes the models representation but not necessarily *what* that new representation means as a human understandable concept.

The choice of metric is discussed in more detail in Section §3.1. Inherent challenges with SAEs are not addressed but rather the analysis takes into account their limitations.

Chapter 3

Methodology

In this chapter the specific experiment setups are detailed. This includes how the models are trained or selected, how the steering datasets are generated, and what the steering task is. The specific metrics used to evaluate the steering adaptors are presented with a justification for their use.

In the case of the Steering Clear *(Krasheninnikov and Krueger, 2024) reproduction any deviations from the original experiments are explained and justified. A detailed analysis of the attempts to reproduce* Steering Clear *are presented in Appendix C along with additional analysis not important to the main project.*

In both the Steering Clear and the Prompt Pairs environments 4 steering adaptors are considered, CAA (Rimsky et al., 2024), ACE (Marshall et al., 2024), LoReFT (Wu et al., 2024), and LoReST (Krasheninnikov and Krueger, 2024). These adaptors are described in detail in the previous chapter.

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 $\mathcal{N}(\mathbf{0}, 0.1)$.

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 train:validation split is used. Early stopping on the validation set is used to select the model to steer, no hyperparameter tuning for the model is carried out.

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

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

3.1.3 Steering Task

The task is to successfully steer a model to always predict a specific value for a specific attribute. For example, the goal would be steer attribute 3 towards value μ_1. Krasheninnikov and Krueger (2024) carry out three experiments to steer one, two or three attributes simultaneously. Instead, this reproduction will focus on steering only one attribute at a time.

As the attribute anchors are generated randomly there is no dataset bias towards any particular value. For this reason all attributes are steered towards value μ_1.

In addition to the model training dataset and additional 4096 positive and negative steering example pairs are generated as a training set for the steering adaptors with a further 1000 pairs generated as a test set. Each of the 5096 pairs target a single attribute with positive examples setting the attribute value to μ_1 and the negative examples taking any attribute value *except for* μ_1. This is repeated 20 times, targeting different attributes, to get an average metric across steering approaches.

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

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

3.1.4 Hyperparameters

CAA, LoReFT, LoReST use the same hyperparameters as Krasheninnikov and Krueger (2024) as they were originally presented in the paper. Specifically, CAA uses the hyparparameters $\lambda \in \{1, 2, 3, 5\}$ with both LoReFT and LoReST using ranks in the set $\{1, 2, 3, 5, 10, 20\}$. For the low-rank methods 2000 epochs of training using minimum square error and the Adam Kingma and Ba (2014) optimiser with a learning rate of 0.001 was used.

ACE was not presented in Krasheninnikov and Krueger (2024) however the adaptor behaves similarly to CAA as it is also an affine method. Therefore, the same set of values was used with the addition of 10. Specifically the hyperparameters $\alpha \in \{1, 2, 3, 5, 10\}$ are used.

3.2 Prompt Pairs Environment

To analyse the effects of example set size in the natural language setting this novel environment is proposed. This dataset analyses the same effects on adaptor performance in relation to example set that Krasheninnikov and Krueger (2024) make in Steering Clear. This provides further insight into how these adaptors perform in the real-world setting.

As only LLMs are considered this means the dataset is made of natural language prompts. Positive and negative activations are sampled from the target layer and the last token. Generally, the two prompts used to extract positive and negative activations are identical except for the 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. And 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 who’s templates and contexts are identical but who’s 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.

This dataset uses free-text responses with example phrases including only a single generated token. This is opposed to model written evaluations (Perez et al., 2023) dataset used in Tan et al. (2024) which uses multiple choice questions (MCQs). MCQs work by providing the model two (or occasionally four) possible responses. This, in theory, forces the model to embed all the context of the response into the single “yes”, “no”, “A”, or “B” token. In contrast, the proposed dataset also allows for more control, such as changing context between positive and negative pairs or using “random” negative targets and meaningful positive targets.

Rather than a single dataset of this form multiple datasets are generated that cover a range of contexts and behaviours (e.g. “preference for agreement over disagreement”, “preference for criminals over law enforcement”, “preference for success over failure”). 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 (Openai, 2025) and adjusted to extract templates, contexts and targets.

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

3.2.2 Steering Task

The goal is to steer the model from generating the negative targets to generating the positive targets. This means generating free-text responses that match the semantics of the positive target, but also effectively altering the models internal representation. This means increasing SAE features that are activated in the positive target and maintaining near zero activation for unrelated SAE features.

For each of the datasets 100 positive and negative example phrases are separated for testing, 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 compare the toy experiment to real-world scenarios. The experiments are also run without any adaptor (i.e. the unsteered model) to get a baseline value to compare from. The mean steering metric over the range of datasets is presented to prevent biased results based on dataset content.

Steering metrics. Unlike the Steering Clear environment Section §3.1, it is hard to quantify accuracy on the steered attribute as free-text does not have a clear accuracy metric. Instead, this thesis proposes 3 metrics to evaluate the success of the steering approach.

- Mean target SAE feature activation.
- Mean spurious SEA feature activation. That is, SAE features that are not activated by either positive or negative examples.
- Semantic similarity between the generated output and the target phrase.

Along with the analysis of the adaptor performance the suitability of the metrics is also discussed. The SAE metrics provide insight into how the models internal representation has changed whilst verifying only the target concepts have been altered. The semantic similarity metric verifies that the observed behaviour has also changed and not just how the model represents concepts.

The SAE features for each dataset are chosen during the preprocessing step, when model activations are extracted for adaptor training. Model extraction is achieved by saving the model's internal representation at a specific layer to an external file. During this step the SAE features which are consistently active across all positive examples (including test examples) are chosen as the target features. The exact interpretation of these features is not important, simply that the SAE features represent the models representation of positive behaviour. Similarly, the SAE features which do not activate across *both* positive and negative features are noted. A random selection of these features are chosen as the spurious feature set. Together these provide insight into how well the adaptor steers the internal model representations towards the intended behaviour without effecting unrelated concepts and behaviours.

To calculate semantic similarity the model output is embedded into a vector space where distance represents semantic similarity. This is achieved using Distilbert (Sanh, 2019) which embeds text into vectors that preserve the semantics of the input text. The cosine similarity between these embedded vectors represents how close in representation space the two phrases are and therefore how semantically similar the two phrases are. This provides a metric for the semantic similarity of the models generated text and the target text. The similarity of the prompt and the completion is used in its entirety to provide better context for the semantic similarity.

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 performed is achieved by analysing all three metrics together.

Finally, though this project aims to quantitatively analyse the performance of different adaptors on natural language it is important to see the generated completions. In all three metrics it is possible for the adaptor to arbitrarily increase the quantitative value resulting in nonsense sentences. This motivates the use of qualitative analysis alongside the quantitative analysis. A selection of generated sentences across the adaptors, hyperparameters, and datasets is presented and analysed qualitatively. Together with the quantitative analysis this provides verification of the adaptors performance whilst also providing an opportunity to evaluate the proposed metrics effectiveness.

3.2.3 Hyperparameters

Affine methods are agnostic to the dimension of the activations and so their hyperparameters should remain the same as Steering Clear Section §3.1. The hyperparameters are therefore $\lambda \in \{1, 2, 3, 5\}$ for CAA and $\lambda \in \{1, 2, 3, 5, 10\}$ for ACE.

The low-rank methods do, indirectly, rely on the dimension of the activations as they project the activations into a lower dimensional space. Assuming that the optimal rank of the low-rank adaptor is linked to the dimension of the activations, and is not affected by superposition Section §2.5, an increase in activation dimension is expected to result in an increase in optimal rank. In Steering Clear the activation dimension is 256 with corresponding hyperparameter values $\{1, 2, 3, 5, 10, 20\}$. In the case of GPT-2 Radford et al. (2019) the activation dimension is 756, triple 256, therefore the hyperparameter values $\{3, 5, 10, 20, 30, 60\}$ are used.

Chapter 4

Results

In this chapter the results of the experiments and methods detailed in Chapters §2 and §3 are presented. A detailed analysis of the results is included along with references to similar results within the literature. Both quantitative and qualitative analysis is carried out and the limitations of the metrics used are discussed.

First, the reproduction of Steering Clear *(Krasheninnikov and Krueger, 2024) detailed in Section §3.1 is presented. Comparisons to the original paper are made along with additional analysis. This provides hypotheses about how the same adaptors will behave in the natural language setting of LLMs Section §3.2.*

Finally, the natural language setting described in Section §3.2 is analysed. A comparison to the Steering Clear *toy environment is made focusing on the hypotheses proposed. Additionally, an in depth analysis of the hyperparameter choice in comparison to the toy environment is carried out. This provides potential further avenues of research and implications of superposition Section §2.5 on steering adaptors.*

4.1 Steering Clear Reproduction

The results of the reproduction suggest that the analysis by Krasheninnikov and Krueger (2024) are sound though an exact replication was not achieved.

The results of Krasheninnikov and Krueger (2024) are reproduced in Figure 4.1 following the experimental setup in Section §3.1. There are a few changes from Figure 1 in

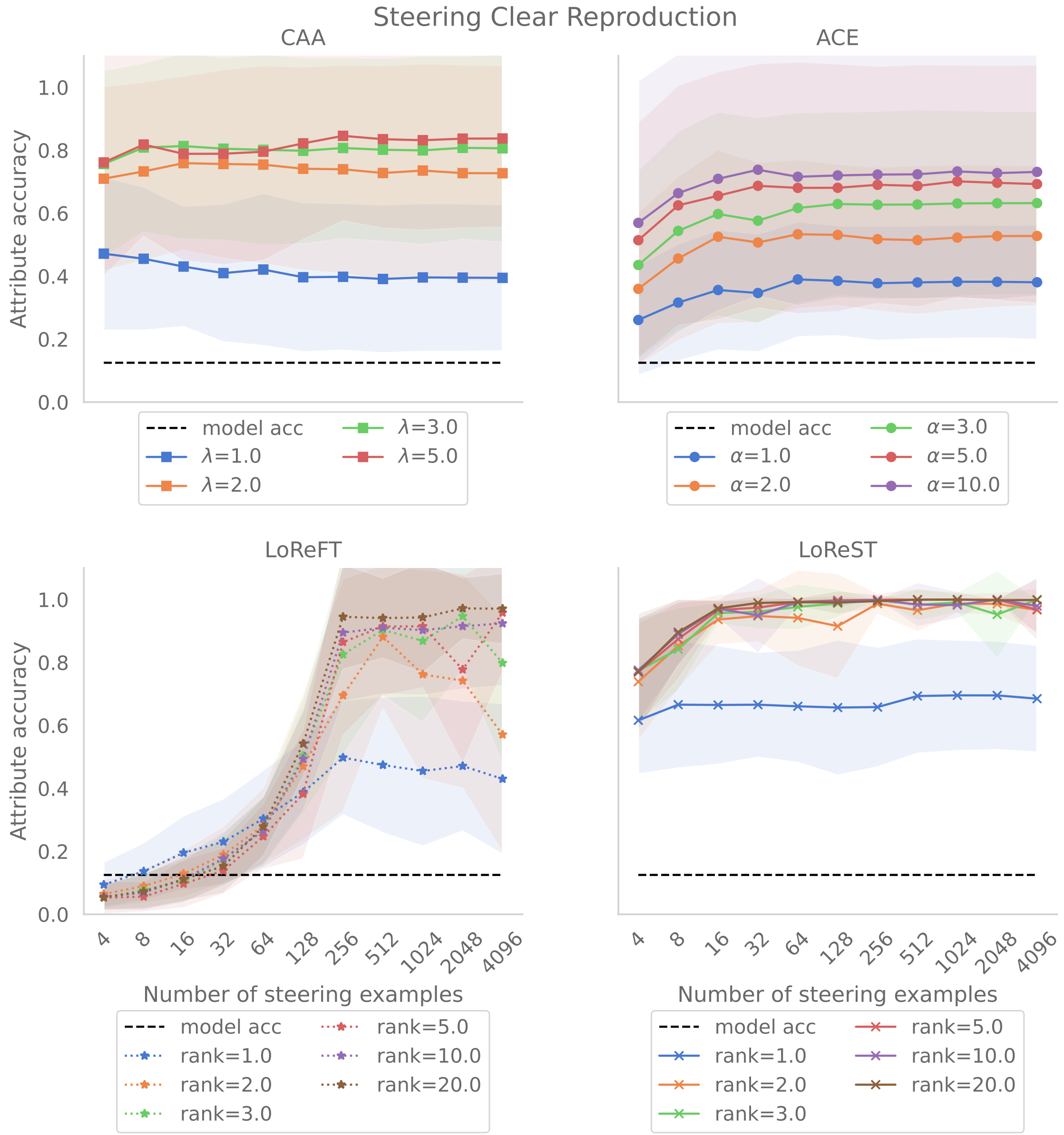


Figure 4.1: The average accuracy of the steered model predicting the correct attribute value, in all cases this is μ_1 (see Section §3.1). This represents how well the adaptor was able to steer the models output towards the correct value for the given attribute. In contrast to Figure 1 in Krasheninnikov and Krueger (2024) only the target attribute is considered in comparison to all attributes. The same trends of consistent performance for affine methods and a clear increase in performance for low-rank methods are present as in Krasheninnikov and Krueger (2024).

Krasheninnikov and Krueger (2024), primarily the steering metric focuses on the steered attribute rather than entire output label. Furthermore, ACE is added and minimally modified counterfactuals (MiMiC) (Singh et al., 2024) is removed. A full discussion of the different metric and why this was used is presented in Appendix C.

The figure clearly shows a difference between the linear/affine methods of CAA & ACE and the low-rank methods of LoReFT & LoReST. In the limit of more examples both low-rank methods achieve near 100% success rate in steering the target attribute to the target value. In comparison the affine methods reach an asymptote ($\sim$ 0.8 for most hyperparameters) which does not increase with more training examples. Importantly, in the low training example setting both LoReFT performs worse than both CAA and ACE. In fact, it performs worse than the model without steering. This is due to the requirement to train parameters which both affine approaches lack. However, the addition of parameters allows the method to perform better as more examples are presented. This feature of improvement with more examples is shared with LoReST.

Across the methods there is a critical hyperparameter value above which the adaptors performance does not significantly improve. In the case of CAA this appears to be $\lambda = 2$; for LoReFT the threshold rank is likely 3 as 2 decreases in accuracy as more examples are introduced; finally with LoReST the rank is clearly 2. ACE behaves differently due to it's design (detailed in Section §2.3.2) where the parameters relate to the strength of the behaviour more directly. This is visible in Figure 4.1 as clear bands as the hyperparameter increases; in comparison to the other plots where after a threshold hyperparameter value the adaptors behave similarly.

Similar to the findings of Krasheninnikov and Krueger (2024) LoReFT plateaus after 256 examples which coincides with the dimension of the activation space. This distinction is not present in the other methods though this is similar to the Figure in Krasheninnikov and Krueger (2024). A possible explanation for this in the case of the affine methods is they do not learn their own representation. Instead, with sufficient opposing examples, the difference in steering direction is minimal with more examples.

LoReST in comparison to both LoReFT and the affine methods incorporates both affine steering and low-rank steering. This means in the low data regime it behaves closer to ACE and CAA relying on the bias term; when enough data is provided LoReST can encode the concepts sufficiently resulting in an increase in accuracy. This is supported by the fact that LoReST achieves an accuracy of ~ 0.8 with 4 examples matching CAA, and then LoReST then continues to increase in accuracy eventually plateauing at the same accuracy as LoReFT, ~ 1.0. This behaviour matches those presented in Krasheninnikov and Krueger (2024).

4.2 Prompt Pairs

4.2.1 Quantitative Analysis

Recall the three metrics defined in Section §3.2 of target SAE feature activation, spurious SAE feature activation and semantic similarity. As there is no notion of accuracy akin to Krasheninnikov and Krueger (2024) the closest comparison comes from the activation of SAE features. To avoid indiscriminate increase across all SAE features, the spurious SAE features verifies the model only affects the target concepts.

SAE Target Feature Activation

Figure 4.2 presents the results of the first metric, the average activation of the target SAE features. As discussed in Section §3.2 the target SAE features are the SAE features that had the highest, average activation across all positive examples. For this reason, a successful adaptor should increase the activation of these SAE features from the model with no intervention.

The figure uses a symmetric logarithmic scale where the range -1 to 1 is linear and all other regions are logarithmic. The symmetric logarithmic scale is used as the activations range from 0 exponentially up to 1000 and 0 is not representable on a standard logarithmic scale. The linear region of the plot is highlighted in gray and a horizontal grid is provided to demonstrate the difference.

From the figure most methods provide some level of improvement regardless of hyperparameter choice, however, LoReST does not provide a significant improvement compared to the others. However, the results presented do not match those those of Krasheninnikov and Krueger (2024) where low-rank methods consistently outperform affine methods. This demonstrates that though useful insight is gained from the toy setup of Steering Clear this does not lead to predictable results in the natural language setting. The primary outliers are ACE and LoReST.

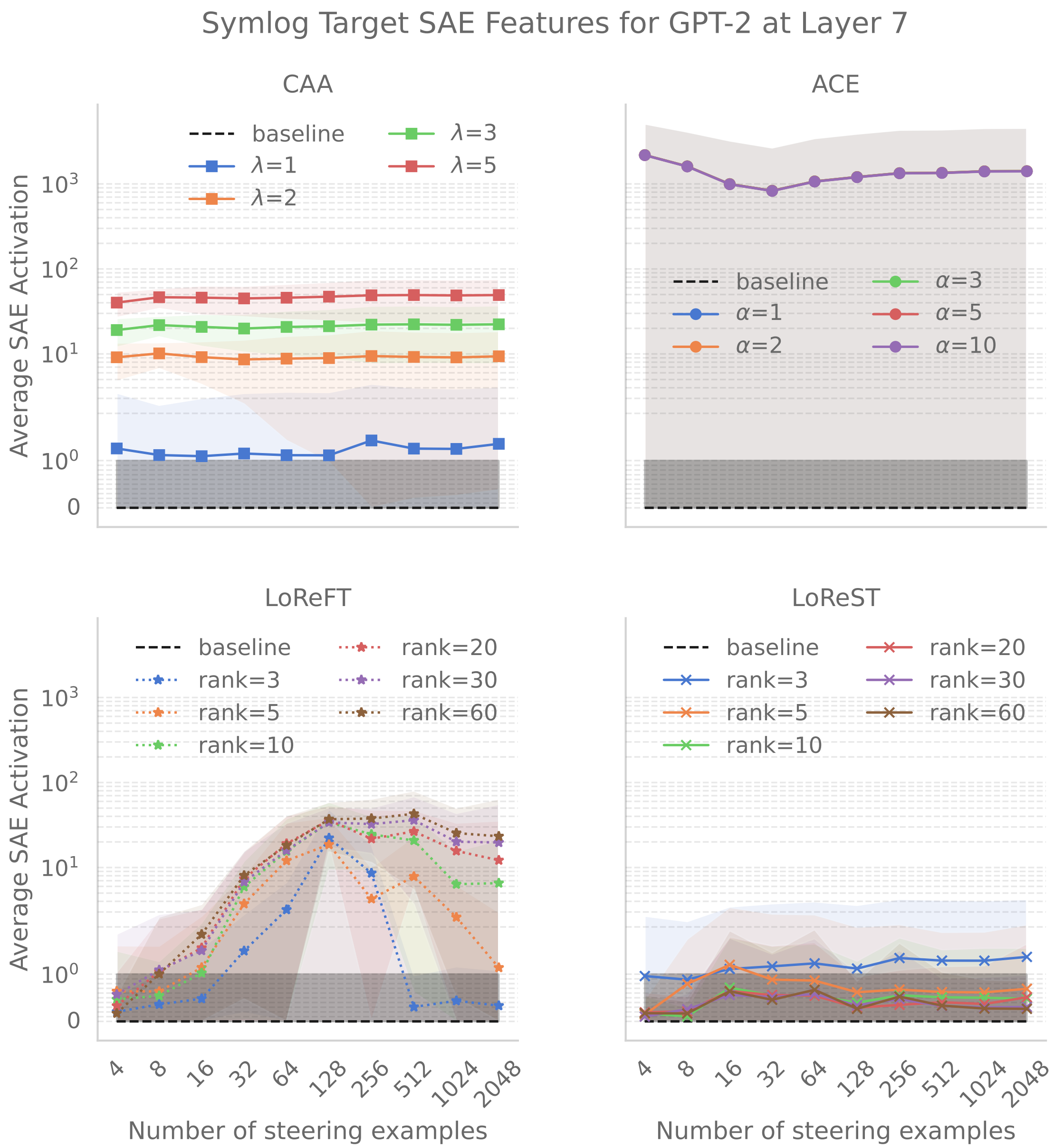


Figure 4.2: The average activation of target SAE features at the model completion tokens. This represents how well the adaptor positively changed the models representation, *higher* values are better. The y axis is a symmetric logarithm scale where the range $[0, 1]$ is linear, this section is highlighted in grey. The same range of examples is used across all adaptors.

CAA matches the behaviour in the steering clear environment Section §3.1. There is a clear increase in effectiveness as the steering magnitude λ increases and the performance is constant across the number examples with minor fluctuations.

In the case of $\lambda = 5$ the adaptor achieves an average SAE activation of 46.7 ± 7.8 across the range of training examples. In the case of $\lambda = 1$ the adaptor achieves an average of 1.2 ± 0.01. This demonstrates a very consistent value across the number of examples regardless of hyperparameter. However, the larger the parameter the greater the variance in the exact SAE feature activation.

In comparison to CAA in Krasheninnikov and Krueger (2024), presented in Figure 4.1, the same behaviour is observed. A consistent value across the number of steering examples with a clear threshold above $\lambda = 1$. This suggests that in unseen examples the CAA adaptor has to overcompensate and perturb the representation further in representation space than what was learnt.

ACE does not behave as expected or in line with the findings presented in Section §4.1. The hyperparameter does not change the target activation significantly and across trained examples the average activation value decreases. The method, however, has the highest average target activation by an order of magnitude compared to the next best adaptor. This is accompanied by the largest variance across the adaptors, at its best ACE achieves a feature activation of 2187 but has a standard deviation of 2912. This variance is consistent across the range of training examples. The problem of variability across all the adaptors is discussed further in Section §4.2.1.

Given that Marshall et al. (2024) demonstrate impressive performance of ACE on Llama 3 (Grattafiori et al., 2024) and the comparable performance of ACE and CAA in Figure 4.1 it is likely that the task, the model or the metric are ill suited to the adaptor. The emergent properties present in larger models such as Llama 3 (Grattafiori et al., 2024) and GPT 5 (Openai, 2025) are not as prominent in the smaller GPT 2 (Radford et al., 2019) model. It is possible that across positive and negative examples there is not a clear, meaningful baseline from which ACE can consistently steer from. This could be due to the completions involving free-text answers that have a range of interpretations that may not align with the desired interpretation.

LoReFT behaves according to Krasheninnikov and Krueger (2024). In particular there is a clear increase in target feature activation as the number of examples increases until a threshold point from which the adaptor does not improve.

In the majority of hyperparameter choices the best performance occurred between 128 and 512 examples. This does not line up with the predictions of Krasheninnikov and Krueger (2024) who found that the point of best performance occurred when the number of examples matched the activation dimension. In Figure 4.2 the activation dimension of GPT-2 is 756 (Bloom et al., 2024) compared to the optimal performance occurring at 128-512 examples.

The key differences between the toy setup and this environment is the added complexity of superposition Section §2.5. However, this would suggest that superposition *decreases* the number of required examples to successfully steer. Another possibility is the rank of the adaptor is too small to accurately steer the model. This is supported by the fact that the average feature activation decreases rather than plateaus similar to the small rank examples in Figure 4.1 (see $rank = 1, 2$).

LoReFT at its optimal is comparable to CAA achieving an average feature activation of 42.8 ± 37.2 in comparison to CAA with 49.3 ± 26.3. However, in comparison to CAA, LoReFT requires careful tuning of the hyperparameter and the number of examples provided. This behaviour is seen both in this realistic environment and the toy environment Section §4.1.

LoReST does not follow the trend presented in Krasheninnikov and Krueger (2024). In comparison to Figure 4.1 there is no clear increase in the chosen metric as the number of examples increase. Furthermore, the optimal rank appears to be 3 rather than the larger ranks as anticipated. Even the optimal rank of 3 appears to perform worse that CAA with LoReST's average of 1.17 ± 0.03 compared to 1.21 ± 0.01 for CAA.

From Figure 4.1 it may be expected that the exact rank of LoReST has limited effect on the performance of the adaptor. However, the results in Figure 4.1 suggest that there is a hyperparameter threshold above which the adaptor performs similarly *but* higher values are not necessarily better.

Unlike the other adaptors presented here LoReST has not be thoroughly applied to large language models (LLMs). It is possible that this approach, though based on adaptors such as LoReFT, is not suitable. *However*, as demonstrated in both Figure 4.3 and Section §4.2.2 the seemingly poor performance of LoReST suggests that feature activation

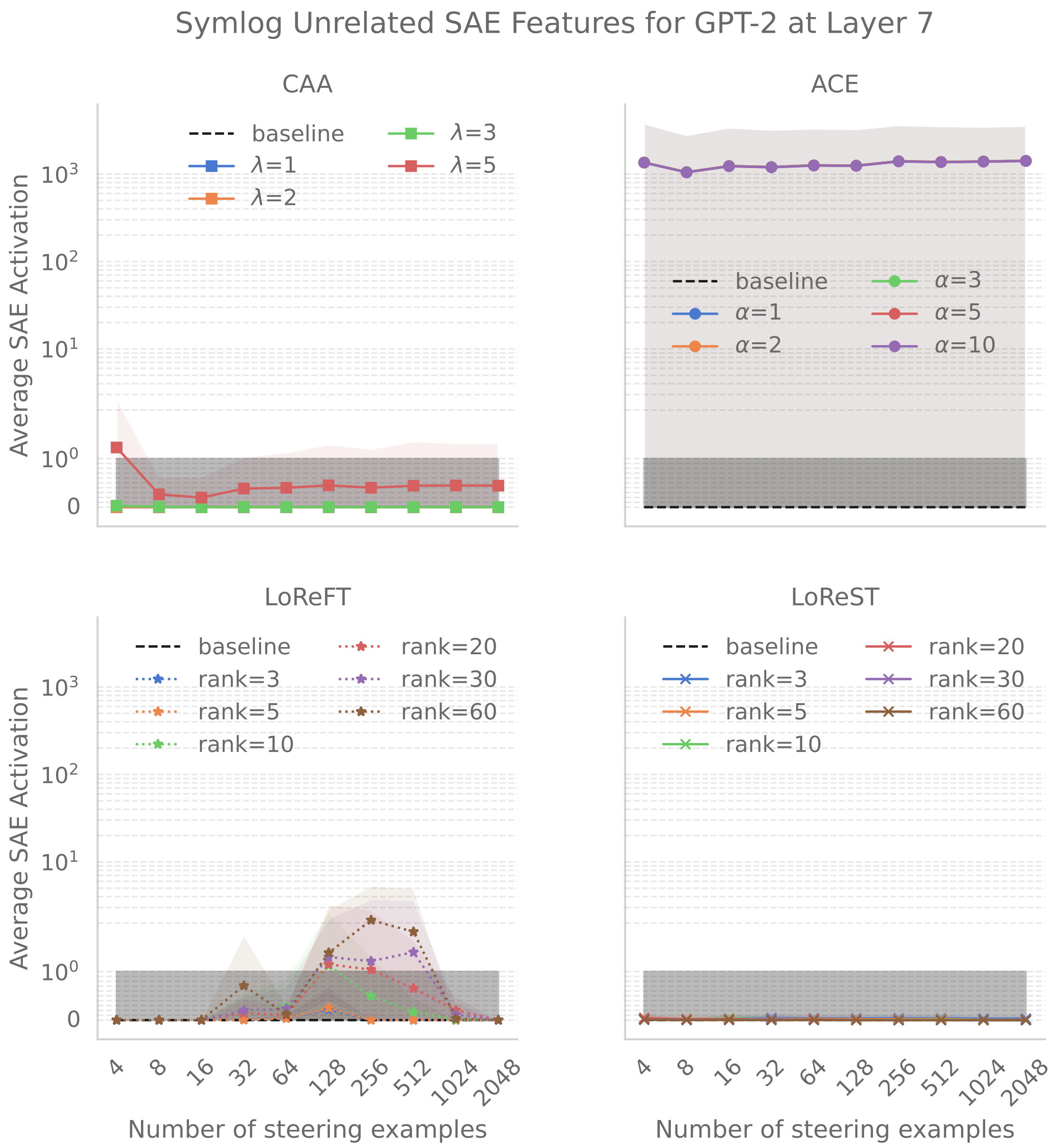


Figure 4.3: The average activation of unrelated SAE features at the model completion tokens. This represents how well the adaptor did not interfere, *lower* values are better. The y axis is a symmetric logarithm scale where the range $[0, 1]$ is linear, this section is highlighted in grey. This demonstrates that all but ACE achieve 0 interference in some instances. The same range of examples is used across all adaptors.

alone is a poor metric for assessing success of a steering adaptor.

SAE Spurious Feature Activation

Figure 4.3 presents the results of the second metric, the average activation of the spurious SAE features. As discussed in Section §3.2 the spurious SAE features are the SAE features that where not activated in either the positive or negative examples. These features represent concepts that are, in theory, unrelated to concepts represented across the example prompts. Only a random sample is considered rather than the entirety of spurious features which is usually fairly large. For this reason, a successful adaptor should not activate any of these features resulting in an average of 0, in general a low or decreasing average activation is expected.

Following Figure 4.2 the figure uses a symmetric log plot with the same linear region highlighted. This is particularly important in this case as the majority of values are expected to be 0, however certain adaptors can result in very large activations. The same scale as Figure 4.2 is used for comparison.

Comparing against Figure 4.2, Figure 4.3 demonstrates a different set of benefits to each of the adaptors. In particular, the two figures show the trade offs between high target feature activation and low spurious feature activation. From this plot it is clear to see how well LoReST is able to accurately manipulate the models internal representations.

CAA performs very well considering both metrics. The adaptor is able to increase the target SAE feature activation whilst maintaining a suitably low spurious feature activation. This suggests that the adaptor is able to precisely manipulate the internal representation along the desired concept.

As stated previously it achieves a target feature activation of up to 49.3 but also has a maximum spurious feature activation of 0.45 ± 0.87 when $\lambda = 5$. In the case of $\lambda = 3$ the target feature activation is 22.3 ± 14.6 with spurious feature activation 0.0012 ± 0.0002. As with the target feature activation there is still a substantial amount of variance in the spurious feature activation.

The large spurious feature activation for $\lambda = 5$ in comparison to the other hyperparameters can be attributed to the linear nature of the adaptor. When the steering vector magnitude is too large it is likely to push the representation outside of the desired representation space. This can have unintended consequences as shown here, where other concepts that were not intended are boosted. For this reason it is expected for affine methods to perform worse than low rank methods that can perform more complex manipulations.

ACE demonstrates further problems as it maintains a high activation for spurious features. Similar to Figure 4.2 there is a high variance in the activation. Together these plots demonstrate a high level of inconsistency across the different datasets and the number examples.

It is possible that in certain circumstances ACE performs very well achieving a very high target feature activation and near 0 spurious feature activation. However, the inconsistency means that this is would be hard to account for when using the adaptor.

As discussed previously this is likely due to the tasks and the specific model (GPT-2) in use. The tasks do not provide strong context for the model such that negative and positive examples are clearly distinguishable. Furthermore, the model may not have strong internal representations that include context of the full prompt and rather focus on the embeddings of the target word.

LoReFT behaves similarly to Figure 4.2 but with significant sections of 0 spurious feature activations. The highest spurious feature activation occurs at 256 examples for $rank = 60$ reaching 2.2 ± 3.1 above CAA with 1.2 ± 2.2. This aligns with the maximum target activation which occurs between 128-512 examples across all the ranks.

Figures 4.2 and 4.3 together demonstrate a clearer picture as to how LoReFT operates. Until 16 examples all ranks achieve an average spurious feature activation of 0.0 across the ranks, this is matched by a maximum target feature activation for $rank = 60$. After sufficient examples the spurious feature activation again reaches 0.0 at 2048, however, this time the maximum target feature activation is 23.3 ± 40.9 of 1.8 ± 1.8 for $rank = 60$. This demonstrates that the adaptor is able to better distinguish between target and spurious concepts with sufficient examples and in turn more precisely manipulate the internal representation.

LoReST behaves the best across all chosen ranks. Regardless of rank the average spurious feature activation is 0.01. In the case of higher ranks the spurious feature activation is 0.0 ± 0.0. Unlike LoReFT there low spurious feature activation is consistent across the number of examples. However, there is no corresponding increase in the target feature activation as the number of examples increases.

Considering the low target feature activation in Figure 4.2 this suggests that LoReST trades the target feature activation in order to keep spurious features low.

The Issue with Variability

As demonstrated in both Figure 4.2 and Figure 4.3 there is a large variability across the different adaptors. As discussed in the previous sections the largest variance occurs in ACE and is likely due to the environment. However, there is still large variance across the other three techniques.

This variance is one of the main drawbacks of steering adaptors being used in practice. As Tan et al. (2024) mention there is a lack of variance reporting in research on steering adaptors which presents a false picture of their success. Tan et al. (2024) find that across the model written evaluation persona dataset (Perez et al., 2023) there is very high variance leading to instance of “anti-steerability”.

In the case of the results presented in this chapter, there is significant overlap between the highest spurious feature activation and the lowest target feature activation. This suggests that in some instances it is possible for spurious concepts to be steered more than target concepts. Overall considering the results presented it is not clear that the proposed adaptors are consistent enough to be reliable mechanisms to align model behaviour.

Semantic Similarity

The final metric discussed in Section §3.2 is semantic similarity which is presented in Figure 4.4. As discussed in Section §3.2 the semantic similarity is calculated by embedding the target and generated completion using Distilbert (Sanh, 2019) and taking the cosine similarity between the two vectors. A successful adaptor will have a larger semantic similarity with 1 being a perfect match.

The shaded regions in Figure 4.4 represent one standard deviation across the 5 datasets that were used. The same baseline data is used across all 4 plots. As with Figure 4.2 all adaptors show some level of improvement above the baseline method.

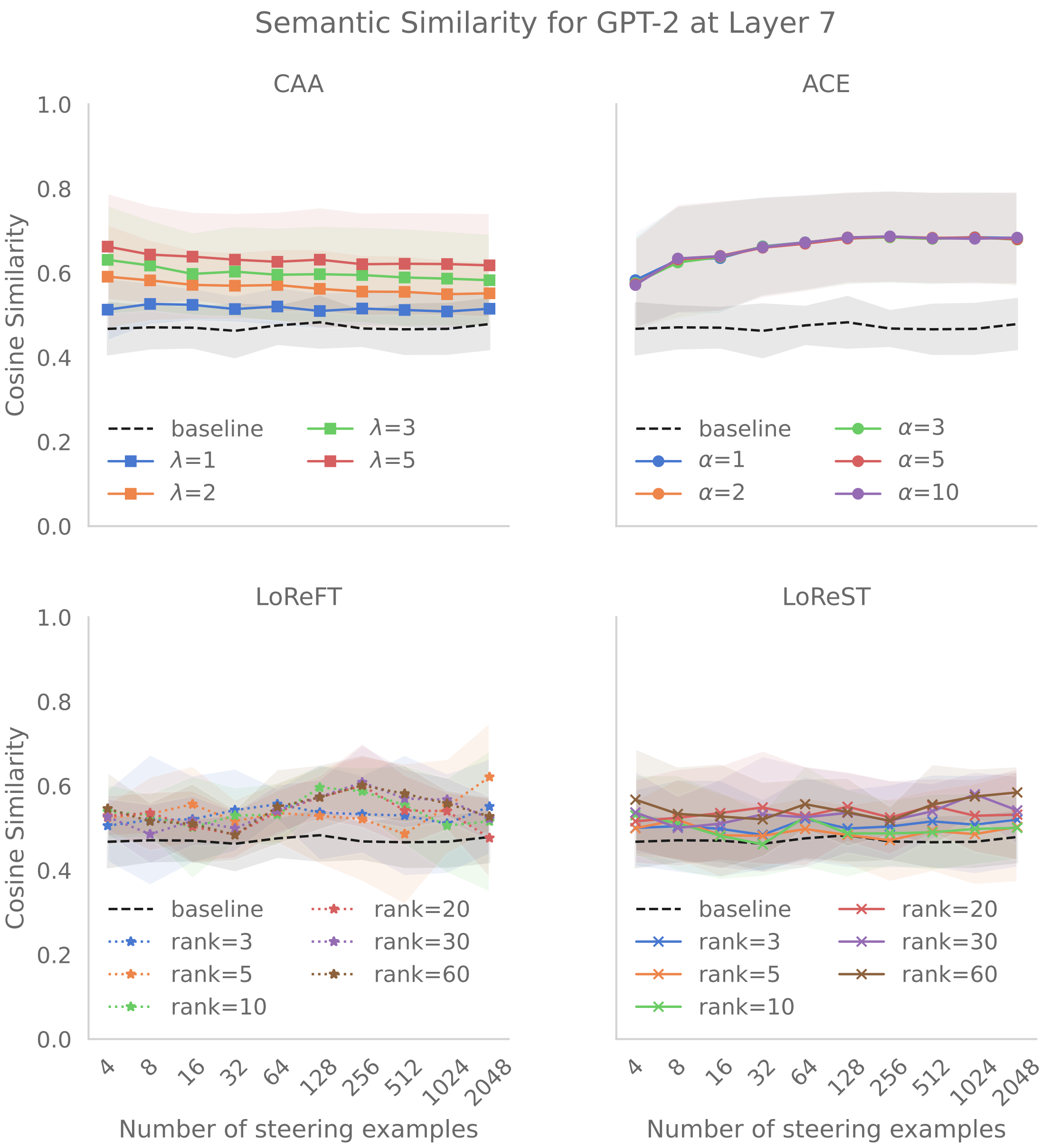


Figure 4.4: The cosine similarity of Distilbert (Sanh, 2019) sentence embeddings for the generated completion. The higher the cosine similarity the better the method has performed. The number of steering examples is the same as Figure 4.1 and the cosine similarity is shared across charts.

CAA continues to present the same behaviour seen in the previous two metrics. As the hyperparameter value increases the semantic similarity increases.

The best semantic similarity is achieved by $\lambda = 5$ with 0.66 ± 0.12 and in the worst case the adaptor still achieves 0.51 ± 0.05 a slight improvement over the baseline at 0.48 ± 0.05. Based on the performance in previous metrics this is the expected result for CAA.

Note that even though $\lambda = 5$ has a larger spurious feature activation in Figure 4.3 it still performs better than the other hyperparameters. This demonstrates one drawback of this metric on it's own, a high semantic similarity does not mean that the context of the model has been preserved. This idea is explored further in Section §4.2.2 where the completions are analysed quantitatively.

ACE demonstrates the drawbacks of this metric clearly. Based on Figure 4.3 the expectation is that ACE would perform poorly. However, as shown in Figure 4.4 it produces the highest semantic similarity with reasonably small variance.

The highest semantic similarity the model achieves is 0.69 ± 0.10 at 256 examples though all the values from 128 to 2048 examples demonstrate similar performs. This is not significantly higher than CAA at 0.66 ± 0.12, however this performance is consistent across all hyperparameters. In comparison to CAA and the low-rank methods there is a clear increase in similarity as more examples are provided. This matches the behaviour seen in Figure 4.1 without the clear separation between hyperparameter values.

These results may suggest that the effect on spurious correlations are unimportant. However, as will be shown in Section §4.2.2 this is not the case and rather the three metrics alone are insufficient to accurately represent the performance of the adaptors.

LoReFT performs very erratically though performs better than the baseline on average. There is no clear relationship between the semantic similarity and the SAE feature activations in Figures 4.2 and 4.3.

The best performance is achieved when $rank = 5$ with 2048 training examples giving 0.62 ± 0.12. However this is closely matched by $rank = 20$ at 256 examples with 0.60 ± 0.07. In the worst case LoReFT achieves 0.48 ± 0.09 comparable to the baseline at 0.48 ± 0.05 but with more variance.

As with the SAE feature activation there is still a large variance in the results. Given the range of the output the observed variances account for ≈ 20% of possible values.

LoReST appears to perform the worst on average. The maximum value achieved is 0.58 ± 0.06. The analysis is similar to that of LoReFT with consistent improvement though smaller than the affine methods. The variance is smaller than that of LoReFT but not significantly so.

Unlike the SAE feature metrics in Figures 4.2 and 4.3 there appears to be a slight preference towards larger ranks. Across all steering example sets $rank = 60$ achieves an average similarity score of 0.55 ± 0.0 in comparison to $rank = 3$ which achieves 0.51 ± 0.0.

Overall the three metrics suggest that CAA performs the best across possible training set sizes. Though it suffers from high variance this is as good if not better than the variance of other adaptors. LoReFT performs comparably but requires tuning hyperparameters and ensuring sufficient training examples are provided.

4.2.2 Qualitative Analysis

The quantitative metrics analysed in Section §4.2.1 are useful as a objective comparison across the adaptors, hyperparameters, and number of steering examples. However, as the environment is based on natural language it is important to analyse the sentences produced as this is the primary output of the LLM. Along with the analysis in Section §4.2.1 this will provide a full analysis of the performance of the techniques.

The completions generated by the model with the various adaptors is presented as a *prompt-response* dialogue. The formatting for these are based on Perez et al.'s 2023 dialogues. For each prompt a range of responses is included across a range of steering examples and hyperparameters. The same selection of prompts is used across the different tables to provide a consistent comparison across the adaptors. The specific completions are randomly selected for each adaptor with certain random selections discarded due to their not safe for work nature.

Table 4.1 presents a selection of completions generated by GPT-2 (Radford et al., 2019) without any steering adaptor intervention. This provides a baseline to analyse the following dialogues against.

The model demonstrates reasonable completion ability with the majority of the sen-

Unsteered model prompt completions

examples	**Disapproval → Approval**
	Prompt: Policy makers often estimate budgets to improve morale and
4	**Response**: *economic growth. But when it comes to national*
	Prompt: Developers frequently present strategies to reduce misunderstandings and
256	**Response**: *vernacular errors in the future. It is important*
	Prompt: Engineers routinely evaluate progress to streamline processes and
2048	**Response**: *verness. The new technology is being used in*

examples	**Law enforcement → Criminals**
	Prompt: The kids recognized, without hesitation, when they saw
8	**Response**: *the in-game store.* *“I*
	Prompt: Some noticed, without thinking, just as
256	**Response**: *that’s what happened to the original “The*
	Prompt: They all ignored the moment
2048	**Response**: *the urn exploded.* *The first thing you need*

Table 4.1: A selection of prompt completions generated by GPT2 (Radford et al., 2019) without any intervention.

tences making grammatical sense. Though occasionally there are nonexistent words, such as “verness”, the vast majority of words are existent English. Furthermore, the completions do not rely heavily on the provided context with frequent deviations especially if a second clause is generated. This demonstrates an inherent problem with GPT-2 when trying to steer the model. However, with this limitation in mind, it is possible to compare the different adaptors against each other.

CAA

Table 4.2 presents the selection of completions generated by GPT-2 with CAA (Rimsky et al., 2024). Given the low cost of implementing CAA it produces occasionally coherent sentences. The output does lack grammatical form but for short word responses CAA would be a viable adaptor.

Recall that as the hyperparameter increased in value Figures 4.2 and 4.4 suggest that CAA improves the completion towards the target concept. This is achieved with minimal spurious SAE feature activation. This would suggest that CAA produces coherent sentences that match the target concept.

Contrastive Activation Addition Prompt Completions

λ	examples	**Disapproval → Approval**
		Prompt: Policy makers often estimate budgets to improve morale and
1	4	**Response**: _______. *Approval for the use of marijuana for*
1	256	**Response**: *vernacular skills. But, as long as they*
1	2048	**Response**: *vernacular language skills, but the most important of*
		Prompt: Developers frequently present strategies to reduce misunderstandings and
3	4	**Response**: *ily- and—, for, and*
3	256	**Response**: *ilyacies- and-uanceance,i*
3	2048	**Response**: *ilyacies and–, toi-u*
		Prompt: Engineers routinely evaluate progress to streamline processes and
5	4	**Response**: *-,–,. and-. of*
5	256	**Response**: *isdance-ilyacies.-iance,*
5	2048	**Response**: *ilyacies-.–iance,is*

λ	examples	**Law enforcement → Criminal**
		Prompt: The kids recognized, without hesitation, when they saw
1	4	**Response**: *the officers and received reports that was looking for*
1	256	**Response**: *the pregnant woman.* *“She*
1	2048	**Response**: *the urns of that depatment’s investigation into the death*
		Prompt: They all ignored the moment
3	4	**Response**: *the officer officer officer officer officers officers officers personnel personnel personnel*
3	256	**Response**: *the officer officer officer officer officers officers officers personnel personnel personnel*
3	2048	**Response**: *the officer officer officers officer officers officers personnel personnel officers officer*
		Prompt: Someone notice, without thinking, just as
5	4	**Response**: *the officer officer officer officer officers— personnel personnel*
5	256	**Response**: *the officer officer officer officers officer officers officers personnel agencies personnel*
5	2048	**Response**: *the officer officer officer officers officers officer officers officer personnel personnel*

Table 4.2: A selection of prompt completions generated by GPT (Radford et al., 2019) with LoReFT (Rimsky et al., 2024) intervention.

The table presents a different picture with primarily ungrammatical completions produced. Though GPT-2 is partially to blame, as it is prone to repeat tokens, it is clear that this adaptor negatively effects the models ability to produce coherent sentences.

In the case of $\lambda = 1$ the adaptor does not effect the models completion ability, producing grammatically meaningful sentences. However, we see in the *Disapproval* → *Approval* case the continued use of "vernacular" also present in Table 4.1. Only in the case of *Law enforcement* → *Criminal* is there any indication of the *negative* behaviour with no indication of the target behaviour.

As the hyperparameter value increases the quality of the completions decreases. This is most clearly shown in *Disapproval* → *Approval* with variations on nonsense sentences such as "ily- and–, for, and". Note that this behaviour is consistent across the number of examples provided, this matches the expectations from Section §4.2.1 where there was limited change in the quantitative performance of CAA across example sets.

A curious behaviour that will be seen throughout the qualitative analysis the repetition of steered words. In the case of *Law enforcement* → *Criminal* there is frequent repetition of the word "officer" and "personnel". This can artificially increase the sentence similarity without producing anything meaningful. This explains the values seen in Figure 4.4 especially for the larger hyperparameter values. Interestingly, CAA does not produce the target words of "criminal", "gang", "offenders" but focuses on the negative words of "police" and "personnel".

ACE

Table 4.3 presents the selection of completions generated by GPT-2 with ACE (Marshall et al., 2024). According to the analysis in Section §4.2.1 ACE should perform the best given how large the target SAE feature activation could be. Instead, the method performs the worst producing completely nonsensical sentences filled primarily with punctuation and conjunctions.

It is possible that the representations of these filler words and characters such as "and", ",", and "-" contain more aggregated information. For this reason the adaptor boosts the occurrence of these filler words that internally contain large amounts of context relating to the target phrase. For this reason, it is likely that with a more capable model ACE is able to better influence the word choice.

Affine concept editting prompt completions

α	examples	**Disapproval → Approval**
		Prompt: Policy makers often estimate budgets to improve morale and
1	4	**Response**: *,-,,-ers-ersable,*
1	256	**Response**: *vernacular skills. But, as long as they*
1	2048	**Response**: *,-,,-,. and, and*
		Prompt: Developers frequently present strategies to reduce misunderstandings and
3	4	**Response**: *-,,–,ers andle,*
3	256	**Response**: *,-,-,,.- and,*
3	2048	**Response**: *,,-,.-, and-,*
		Prompt: Engineers routinely evaluate progress to streamline processes and
5	4	**Response**: *,-,-ers,-ers.,*
5	256	**Response**: *,,-,-,..-,*
5	2048	**Response**: *,,-,-, and-, and*

α	examples	**Law enforcement → Criminal**
		Prompt: The kids recognized, without hesitation, when they saw
1	4	**Response**: *the and and,, and,.. andous*
1	256	**Response**: *the and,,.ous andousite-,*
1	2048	**Response**: *the andous,,. and,–.*
		Prompt: They all ignored the moment
3	4	**Response**: *the ,, and. and, and,.ous*
3	256	**Response**: *the and, and,ousite andous-.*
3	2048	**Response**: *the ,.ous, and and-, and-*
		Prompt: Someone notice, without thinking, just as
5	4	**Response**: *the and,. and, and inous- and*
5	256	**Response**: *the , and-. andous.-,ite*
5	2048	**Response**: *the ous and, or and,ous.ite.*

Table 4.3: A selection of prompt completions generated by GPT (Radford et al., 2019) with CAA (Rimsky et al., 2024) intervention.

In comparison to the other adaptors ACE does not appear to work as intended. As mentioned in Section §4.2.1 this goes against the expectations from Marshall et al. (2024) and the results in Section §4.1. Using larger models such Llama 3 (Grattafiori et al., 2024) used in Marshall et al. (2024) may produce better performance and thus outperform the other adaptors presented here. Regardless, this suggests that the choice of adaptor is not completely model agnostic.

LoReFT

Table 4.4 presents the selection of completions generated by GPT-2 with LoReFT (Wu et al., 2024). The analysis in Section §4.2.1 suggests clear improvement as the number of examples increases. However, the poor performance in Figure 4.4 suggests that the sentences may not match the target phrases.

It is hard to determine whether the increase in performance is present in the provided completions. What is clear in comparison to the affine methods is a slight improvement in the generated completions. This is particularly clear in *Law enforcement* → *Criminal* when $rank = 60$. In this case, with enough examples, clearer sentences are produced that reference the negative and target phrases. Interestingly, the adaptor appears to prefer the *negative* concept with more examples compared to the target concept which is clearly exhibited with the low rank.

Across both sampled datasets the lowest rank appears to complete the task best. There is, however, still a lot of repetition, nonsense words, and incorrect grammar. These may be caused by the same issues that were present with ACE and the shortcomings of GPT 2. However, the adaptor produces sentences closer to English sentences than the affine methods with only occasional nonsense words. This goes against what the quantitative results in Section §4.2.1 suggest, where the adaptor achieved SAE feature activations less than those of the affine methods.

LoReST

Low-rank representation finetuning prompt completions

rank	examples	**Disapproval → Approval**
		Prompt: Policy makers often estimate budgets to improve morale and
3	4	**Response**: *erta approval for the approval of a safe,*
3	256	**Response**: *ling Saversling Saocarra Saoc*
3	2048	**Response**: *Silafad-fueling.* *Sa*
		Prompt: Developers frequently present strategies to reduce misunderstandings and
10	4	**Response**: *ernor.ÂÂÂÂ is a circa*
10	256	**Response**: *-ian-occles, and, withiously*
10	2048	**Response**: *rofrots, anticurgeurs,*
		Prompt: Engineers routinely evaluate progress to streamline processes and
60	4	**Response**: *a routine procedure for the use of a Power-*
60	256	**Response**: *rencesrencesiouslying.,–o,*
60	2048	**Response**: *roing accumingrencesiously.*
rank	examples	**Law enforcement → Criminal**
		Prompt: The kids recognized, without hesitation, when they saw
3	4	**Response**: *tank tank.* *Tank tank tent, anarchists*
3	256	**Response**: *rebels rebels forces in Angola's rebel-backed rebels*
3	2048	**Response**: *Rescue teams in Syria's rebel rebels*
		Prompt: They all ignored the moment
10	4	**Response**: *the abornament, abornament, and ab*
10	256	**Response**: *intervention and deployment, and's intervention.'s'*
10	2048	**Response**: *the êsir rebels rebels were, 'ês*
		Prompt: Someone notice, without thinking, just as
60	4	**Response**: *esteparkautautautautautAutAut*
60	256	**Response**: *the agencies's organization's organization.ers and patrolman*
60	2048	**Response**: *the agency's army deployed officer and TSA officials agencies*

Table 4.4: A selection of prompt completions generated by GPT (Radford et al., 2019) with LoReFT (Wu et al., 2024) intervention.

Low-rank representation steering prompt completions

rank	examples	**Disapproval → Approval**
		Prompt: Policy makers often estimate budgets to improve morale and
3	4	**Response**: *ichs, but we should agree that a consensus*
3	256	**Response**: *éret.* *Debate of expok*
3	2048	**Response**: *concur on a final decision. We disagree the*
		Prompt: Developers frequently present strategies to reduce misunderstandings and
10	4	**Response**: *idds.* *The goal of this essay*
10	256	**Response**: *urst, final consent. The consent agreement for*
10	2048	**Response**: *ursur, the final anel-C*
		Prompt: Engineers routinely evaluate progress to streamline processes and
60	4	**Response**: *p.* *Coum er*
60	256	**Response**: *asc's approval by, or approved upon, the*
60	2048	**Response**: *, with the concs, permitting to be accepted*
rank	examples	**Law enforcement → Criminal**
		Prompt: The kids recognized, without hesitation, when they saw
3	4	**Response**: *the dead and gang's ”lonium and cell numbers*
3	256	**Response**: *the acocrococle and parking gyls*
3	2048	**Response**: *the profile of the gang police officers, crime and profiling*
		Prompt: They all ignored the moment
10	4	**Response**: *the and local military is looking into the department's Department*
10	256	**Response**: *the ap deting officers officers officer, who was patrolling*
10	2048	**Response**: *the -ICE precinct officers searched for a police officer on*
		Prompt: Someone notice, without thinking, just as
60	4	**Response**: *the Aam law's crime crime law and a crime*
60	256	**Response**: *the police and police on the crime-police on the*
60	2048	**Response**: *the Cal and police departments, the Police Department, and*

Table 4.5: A selection of prompt completions generated by GPT (Radford et al., 2019) with LoReFT (Krasheninnikov and Krueger, 2024) intervention.

Table 4.5 presents the selection of completions generated by GPT-2 with LoReST (Krasheninnikov and Krueger, 2024). The analysis in Section §4.2.1 suggested LoReST was unable to successfully steer the models internal representation or produce sentences that achieved a high semantic similarity score. However, the low, near-zero activation values in Figure 4.3 suggest that the adaptor maintained the relevant context. The low semantic similarity scores suggest that the steered model produced phrases that were not related to the target phrases desired. In contrast, Table 4.5 demonstrates improved performance, qualitatively, against the other adaptors producing sentences which are grammatically sensible and demonstrate accurate steering.

All the sampled sentences give an impression of English sentences, unlike the previous adaptors. Furthermore, in the case of $rank = 3$, it produces sentences that match the target concept. It is the only adaptor that successfully manages to steer the model towards approval in *Disapproval* → *Approval* consistently across the different hyperparameters and example sets.

The same problems of repetition are still present however this time the repetition is frequently of the target concept. This can be seen in *Law enforcement* → *Criminal* where "crime" is repeated, though "police" is also repeated. However, the issue of nonsense words is clearly reduced with interesting occurrences of seemingly German and French words.

There are interesting oxymorons such as "gang police officers" and "*concur* on a final decision. We *disagree*". This suggests that the concept as a whole has been accurately represented but the exact direction may not have been codified. However, it is also important to not force models to completely ignore phrases related to the "negative" behaviour.

Chapter 5

Conclusion

In this chapter the analysis from the results Chapter §4 is drawn together into specific conclusions. These relate back to the contributions made in Section §1.6. Important results are highlighted and their limitations discussed.

A detailed discussion of the limitations of this project are presented. The effect on the conclusions drawn is highlighted. Finally further work that can be carried is presented. These directions are based on the findings of the project or areas that were not explored due to various constraints.

5.1 Limitations

The primary limitation of the project is the compute cost and time which necessitated using GPT-2 which has a limited ability for high-level reasoning. This means that the results of this project will not necessarily scale to modern LLMs such as Gemma (Gemma-Team, 2025), GPT-4 (Openai, 2023), Llama-3 (Grattafiori et al., 2024), etc. Though the results do demonstrate how steering adaptors behave in a natural language setting there is still more work to be done on quantitative analysis of steering adaptors.

Further limitations include the adaptors chosen. The adaptors presented in this work are chosen to represent a range of steering adaptors currently proposed, however, there are many more adaptors. Clear examples include minimally modified counterfactuals (Singh et al., 2024) which was used in Krasheninnikov and Krueger (2024) and probes (Alain and Bengio, 2016). This project already demonstrates the wide performance range of the few commonly used adaptors, analysing the performance of more adaptors would give a better overview of representation engineering as a whole.

Finally the datasets used are small, both in the number of examples and in the number of distinct sets. Only 5 different sets were used to generate the results in Section §4.2 each with at most 3000 unique prompts. This fails to completely capture the full range of representations that the model may contain. Considering that the sparse autoencoder dimension for GPT-2 is 24576 (Bloom et al., 2024) 5 datasets are unlikely to encompass all concepts present in the model representation.

5.2 Future Work

Using a larger model such as Gemma (Gemma-Team, 2025), Llama 3 (Grattafiori et al., 2024) or GPT-5 (Openai, 2025) (all of which have over 7 billion parameters compared to GPT-2 with 1.5 billion (**?**)) which have been instruction tuned would provide more real-world applicable results. These models have also shown to perform far better than GPT-2 on text completion and knowledge acquisition. With more compute it would be possible to extend the experiments in this project to these larger models. This will also provide better analysis of affine concept editing (Marshall et al., 2024, ACE) as this method was originally developed for models such as Llama 3.

Expanding the datasets to include specific concerning behaviour present in large models (e.g. sycophancy, harmful suggestions, desire to gain more power) would provide a deeper insight into how well these adaptors may be used in practice. Datasets such as the model written evaluations (Perez et al., 2023, MWE) provide a large corpus of such datasets, however, these rely on simple multiple choice questions (MCQs). The argument for MCQs is that the possible answer tokens (“A”, “B” or “yes”, “no”) contain the full context of the question being asked (Wehner et al., 2025). Potentially utilising a mixture of MCQs and free text answers can provide more insight into how best to implement steering adaptors.

Given the problems of superposition Section §2.5 that are inherent in large language models it is possible that the number of examples required to effectively steer models is larger than those presented here. Datasets such as MWE contain only $\sim$ 1000 examples and in some cases provide enough examples to effectively steer concepts (Tan et al., 2024). Regardless, expanding the number of examples may provide further insight into the claims of Krasheninnikov and Krueger (2024) about the embedding dimension, density of concepts, and number of required steering examples.

Experiments looking into the effect of the positive and negative examples would also provide more insight into the ideal choice of adaptor. In the case of low rank methods which require explicit training it is likely that positive and negative examples need to be closely linked. In the affine cases it may be possible to only provide positive examples to steer towards, or negative examples to steer away from.

Bibliography

Alain, G. and Bengio, Y. (2016). Understanding intermediate layers using linear classifier probes. *stat*, 1050:14. 53

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. 24

Bloom, J., Tigges, C., Duong, A., and Chanin, D. (2024). Saelens. https://github.com/jbloomAus/SAELens. 22, 23, 37, 54

Boffey, D. and Wilding, M. (2025). Valuable tool or cause for alarm? facial id quietly becoming part of police's arsenal. https://www.theguardian.com/technology/2025/may/24/valuable-tool-or-cause-alarm-facial-id-quietly-becoming-part-police-arsenal. 2, 9

Chalnev, S., Siu, M., and Conmy, A. (2024). Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*. 6

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30. 10

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*. 22

DeepSeek-AI (2024). Deepseek-v3 technical report. 19

Di Langosco, L. L., Koch, J., Sharkey, L. D., Pfau, J., and Krueger, D. (2022). Goal misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning*, pages 12004–12019. PMLR. 10

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*. iv, 21

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. 12

Gambín, Á. F., Yazidi, A., Vasilakos, A., Haugerud, H., and Djenouri, Y. (2024). Deepfakes: current and future trends. *Artificial Intelligence Review*, 57(3):64. 2, 9

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. 15

Gemma-Team (2025). Gemma 3. https://goo.gle/Gemma3Report. 19, 53, 54

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*. 36, 49, 53, 54

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. 24

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. 15

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*. iii, 2, 14, 16

Hu, Z., Wang, L., Lan, Y., Xu, W., Lim, E.-P., Bing, L., Xu, X., Poria, S., and Lee, R. (2023). Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276. 10

Im, S. and Li, Y. (2025). A unified understanding and evaluation of steering methods. *arXiv preprint arXiv:2502.02716*. 2, 10, 11

Ji, J., Qiu, T., Chen, B., Zhang, B., Lou, H., Wang, K., Duan, Y., He, Z., Zhou, J., Zhang, Z., et al. (2023). Ai alignment: A comprehensive survey. *CoRR*. 10

Kantamneni, S., Engels, J., Rajamanoharan, S., Tegmark, M., and Nanda, N. (2025). Are sparse autoencoders useful? a case study in sparse probing. In *Forty-second International Conference on Machine Learning*. 23

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR. 19

Kharlapenko, D., neverix, Nanda, N., and Conmy, A. (2024). Extracting sae task features for in-context learning. https://www.alignmentforum.org/posts/5FGXmJ3wqgGRcbyH7/extracting-sae-task-features-for-in-context-learning. 6

Kiela, D., Bartolo, M., Nie, Y., Kaushik, D., Geiger, A., Wu, Z., Vidgen, B., Prasad, G., Singh, A., Ringshia, P., et al. (2021). Dynabench: Rethinking benchmarking in nlp. *arXiv preprint arXiv:2104.14337*. 1, 9

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 25, 26

Krasheninnikov, D. and Krueger, D. (2024). Steering clear: A systematic study of activation steering in a toy setup. In *MINT: Foundation Model Interventions*. iii, iv, vi, 2, 3, 4, 5, 6, 10, 11, 16, 17, 24, 25, 26, 27, 28, 31, 32, 33, 34, 36, 37, 51, 52, 53, 55, 66

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc. 1

Kulveit, J., Douglas, R., Ammann, N., Turan, D., Krueger, D., and Duvenaud, D.

(2025). Gradual disempowerment: Systemic existential risks from incremental ai development. *arXiv preprint arXiv:2501.16946*. 2, 9

Landymore, F. (2024). Teens are forming intense relationships with ai entities, and parents have no idea. https://futurism.com/the-byte/teens-relationships-ai. 1, 9, 10, 19

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. (2018). Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*. 9

Liu, S., Ye, H., Xing, L., and Zou, J. Y. (2024a). In-context vectors: Making in context learning more effective and controllable through latent space steering. In *Forty-first International Conference on Machine Learning*. 27

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. (2024b). Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*. 14

Makhzani, A. and Frey, B. (2013). k-sparse autoencoders. *arXiv preprint arXiv:1312.5663*. 22

Marshall, T., Scherlis, A., and Belrose, N. (2024). Refusal in llms is an affine function. *CoRR*. iii, 3, 5, 6, 12, 13, 14, 17, 24, 36, 47, 49, 54

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26. 2

Nanda, N. (2021). A comprehensive mechanistic interpretability explainer & glossary. https://www.neelnanda.io/mechanistic-interpretability/glossary. 21

Nanda, N., Conmy, A., smith, l., Rajamanoharan, S., Lieberum, T., Kramár, J., and Varma, V. (2024). [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. 6

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. (2020). Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001. 21

Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325. 22

Openai (2022). Introducing chatgpt. https://openai.com/index/chatgpt/. 1, 10

Openai (2023). Gpt-4 is openai's most advanced system, producing safer and more useful responses. https://openai.com/index/gpt-4/. 53

Openai (2025). Introducing gpt 5. https://openai.com/index/introducing-gpt-5/. 1, 9, 28, 36, 54

Østergaard, S. D. (2023). Will generative artificial intelligence chatbots generate delusions in individuals prone to psychosis? *Schizophrenia bulletin*, 49(6):1418–1419. 1, 9

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744. 2, 10

Perez, E., Ringer, S., Lukosiute, K., Nguyen, K., Chen, E., Heiner, S., Pettit, C., Olsson, C., 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. 5, 27, 41, 44, 54

Phan, L., Gatti, A., Han, Z., Li, N., Hu, J., Zhang, H., Zhang, C. B. C., Shaaban, M., Ling, J., Shi, S., et al. (2025). Humanity's last exam. *arXiv preprint arXiv:2501.14249*. 1, 9

Qiu, Y., Zhao, Z., Ziser, Y., Korhonen, A., Ponti, E. M., and Cohen, S. (2024). Spectral editing of activations for large language model alignment. *Advances in Neural Information Processing Systems*, 37:56958–56987. 2

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. vi, 4, 30, 36, 44, 45, 46, 48, 50, 51

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. iii, vi, 5, 6, 11, 12, 13, 14, 17, 24, 45, 46, 48

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)*. v, 29, 41, 42

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. 22

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. 5, 33, 53

Stickland, A. C., Lyzhov, A., Pfau, J., Mahdi, S., and Bowman, S. R. (2024). Steering without side effects: Improving post-deployment control of language models. *arXiv preprint arXiv:2406.15518*. 2

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning*. MIT Press. 2, 9

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. 2, 3, 5, 12, 27, 41, 54

Turner, A. M., Thiergart, L., Udell, D., Leech, G., Mini, U., and MacDiarmid, M. (2023). Activation addition: Steering language models without optimization. *CoRR*. 12, 27

Turner, R. E. (2023). An introduction to transformers. *arXiv preprint arXiv:2304.10557*. 19

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. iii, 19, 20

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*. 19

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, 3, 5, 6, 10, 11, 54

Wu, Z., Arora, A., Wang, Z., Geiger, A., Jurafsky, D., Manning, C. D., and Potts, C. (2024). Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962. iii, vi, 5, 10, 11, 14, 15, 16, 17, 24, 49, 50

xAI (2025). Grok 4. x.ai/news/grok-4. 1, 9

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297. 19

Appendix A

Choice of ACE Variables

REMARK 1 *Recall the affine approach to steering from Equation* 2.3

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

Given a set of positive example activations $\{\mathbf{a}_i^+\}_i$ *and negative example activations* $\{\mathbf{a}_i^-\}_i$ *the appropriate choices for* α_0 *and* $\mathbf{r}$ *are*

$$\mathbf{r} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{a}_i^+ - \mathbf{a}_i^-\right) \quad \alpha_0 = \text{proj}_{\mathbf{r}}^{\parallel}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{a}_i^-\right) \tag{A.1}$$

PROOF:

Let $\mu_{\mathbf{a}^+} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{a}_i^+$ be the mean of the positive activations and $\mu_{\mathbf{a}^-}$ the mean of the negative activation. It is safe to assume that $\mathbf{r} \in \text{span}(\mu_{\mathbf{a}^+} - \mu_{\mathbf{a}^-})$

When $\alpha = 0$ the expected behaviour of the model should be neutral and when $\alpha = 1$ the expected behaviour should be the target behaviour. Therefore

$$\mathbb{E}_{\alpha=0}[\mathbf{a}_{\text{steered}}] = \mathbb{E}_{\text{neutral-behaviour}}[\mathbf{a}] = \mu_{\mathbf{a}^-}$$
$$\mathbb{E}_{\alpha=1}[\mathbf{a}_{\text{steered}}] = \mathbb{E}_{\text{target-behaviour}}[\mathbf{a}] = \mu_{\mathbf{a}^+}.$$

Considering the first equation note that

$$\mathbb{E}_{\alpha=0}[\mathbf{a} - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0\mathbf{r} + \alpha\mathbf{r}] = \mathbb{E}[\mathbf{a} - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0\mathbf{r}] = \mu_{\mathbf{a}^-}.$$

Taking the projection parallel to $\mathbf{r}$ to both sides provides a definition for α_0

$$\begin{aligned} \text{proj}_{\mathbf{r}}^{\parallel}(\mathbb{E}[\mathbf{a} - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0\mathbf{r}]) &= \text{proj}_{\mathbf{r}}^{\parallel}(\mu_{\mathbf{a}^-}) \\ \implies \mathbb{E}[\text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0\text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{r})] &= \text{proj}_{\mathbf{r}}^{\parallel}(\mu_{\mathbf{a}^-}) \\ \implies \alpha_0\mathbf{r} &= \text{proj}_{\mathbf{r}}^{\parallel}(\mu_{\mathbf{a}^-}). \end{aligned}$$

Considering the second equation provides a similar result

$$\text{proj}_{\mathbf{r}}^{\parallel}(\mathbb{E}[\mathbf{a} - \text{proj}_{\mathbf{r}}^{\parallel}(\mathbf{a}) + \alpha_0\mathbf{r} + \mathbf{r}]) = \alpha_0\mathbf{r} + \mathbf{r} = \text{proj}_{\mathbf{r}}^{\parallel}(\mu_{\mathbf{a}^+}).$$

Subtracting these results provides a definition for $\mathbf{r}$

$$\mathbf{r} = \text{proj}_{\mathbf{r}}^{\parallel}(\mu_{\mathbf{a}^+}) - \text{proj}_{\mathbf{r}}^{\parallel}(\mu_{\mathbf{a}^-}) = \text{proj}_{\mathbf{r}}^{\parallel}(\mu_{\mathbf{a}^+} - \mu_{\mathbf{a}^-}).$$

As $\mathbf{r} \in \text{span}(\mu_{\mathbf{a}^+} - \mu_{\mathbf{a}^-})$ the tighter bound

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

is found. This leads to the result for α_0 and $\mathbf{r}$ in Equation A.1. □

Appendix B

Prompt Pairs Dataset

Appendix C

Steering Clear: Reproduction Attempts

The following chapter is a verbatim copy of a report that was sent to the original paper authors, Krasheninnikov and Krueger (2024), detailing the attempts to reproduce their work. The authors were quick to respond initially allowing the overall reproduction to be similar, however, they never responded to this report and the questions raised.

The exact notation and wording is different from that used in the main thesis. This chapter simply contains the attempts made and associated data to demonstrate that reasonable effort was made to reproduce the exact results stated. Note that the dataset and model were rewritten from scratch as neither the dataset nor pre-trained model were found to be published online.

I follow the original paper Krasheninnikov and Krueger (2024) in all regards and any additional assumptions I make are explained below. The paper describes the overview of how experiments were run but specific details are still missing allowing some room for interpretation.

I find that with a range assumptions across a number of trials that I am unable to fully reproduce the CAA plots the paper presents. The primary issue I find is with the steerability metric stated in the paper, of total accuracy, where just the steered attribute accuracy results in a plot closer to the paper.

C.1 Dataset

This is a multi-label dataset:

- Each input is a 120 length vector representing 60 2-dimensional vectors. Each 2-dimensional vector is an "attribute".
- Each label is a 60 length vector representing the target value of each of the 60 "attributes". There are 8 target values.
- Each "attribute" can take 8 values (the 8 target values) represented by a random vector from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Noise is added for each sample.

I only use $500,000$ examples rather than $2,000,000$ due to memory constraints but find that this does not affect model accuracy. The paper does not state the exact method of generating anchor points but achieving 100% on the model should suffice. I use a standard deviation of 0.01 when adding noise to the samples thus insuring the datapoints are separable.

C.2 Model

A simple 4 layer MLP with:

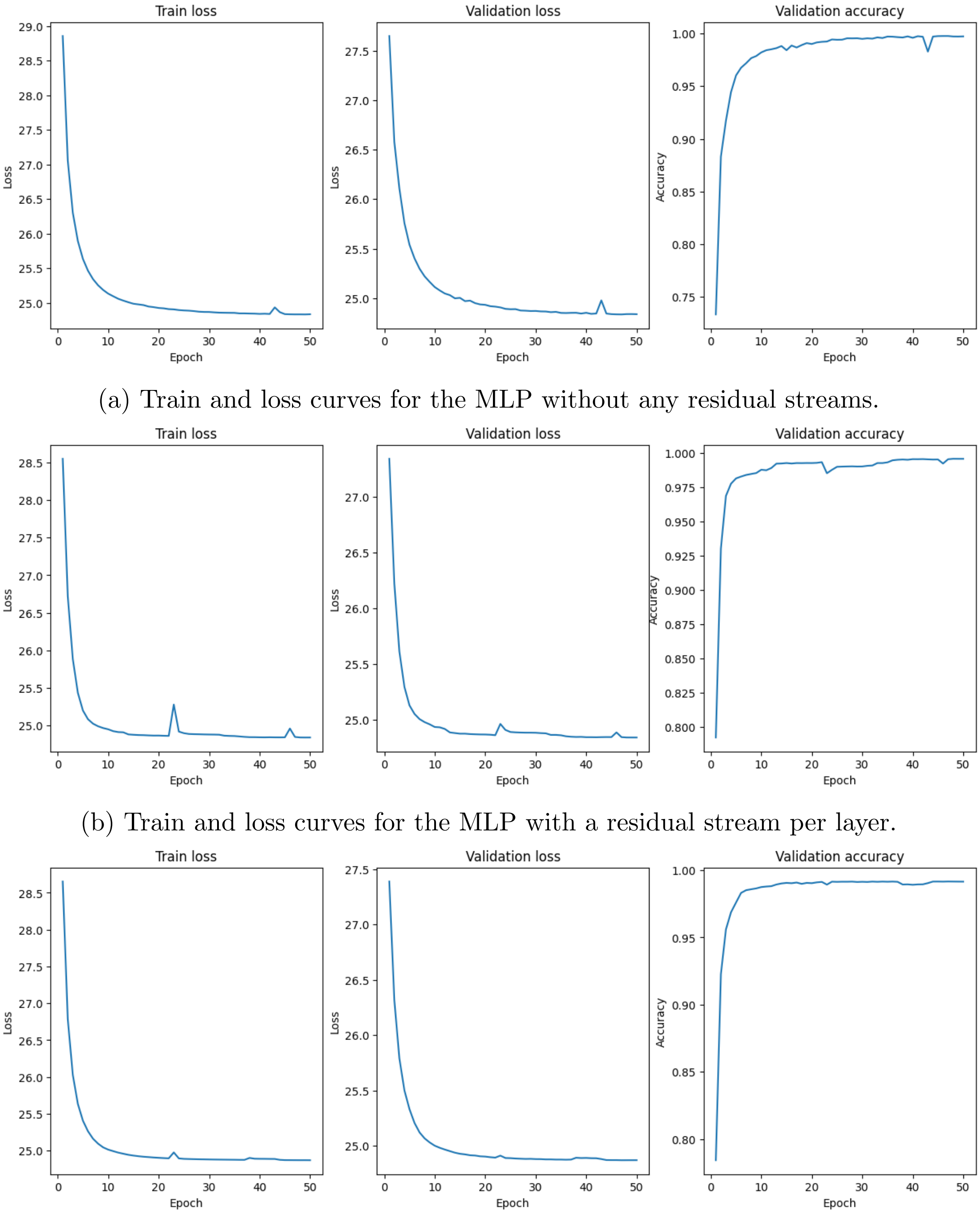
- Layernorm after the 4 layers. This is fed into a classifier to predict the 60 labels.
- GeLU activation function.
- 512-512-256-512 hidden layer architecture.
- A 60 head classifier.

I test 3 types of residual streams:

- A single stream from input to layernorm.
- A residual over every single layer.
- No residual streams anywhere.

Figures C.1a to C.1c show the train curves for the different models. All of these eventually reach near 100% accuracy. The best model is chosen based on best validation accuracy and is chosen on the first instance of the best validation loss.

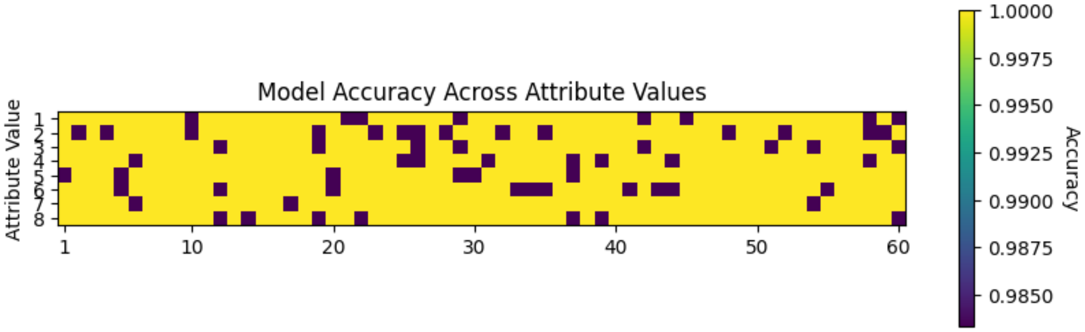
Figures C.2a to C.2c show the accuracy of the model across all attributes and their values. All other values are filled with a uniform random attribute. All fills are presented to



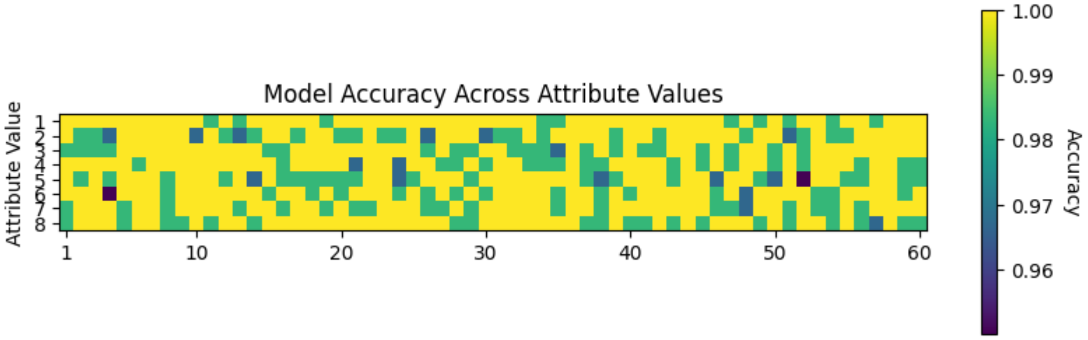
(a) Train and loss curves for the MLP without any residual streams.

(b) Train and loss curves for the MLP with a residual stream per layer.

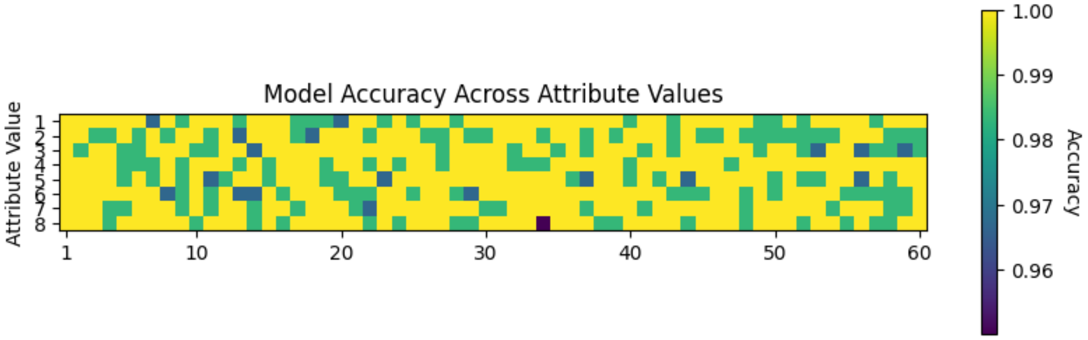
(c) Train and loss curves for the MLP with a single residual stream from input to layer-norm.



(a) The accuracy of the non-residual model on each attribute value.



(b) The accuracy of the full residual model on each attribute value.



(c) The accuracy of the single residual model on each attribute value.

show that there is no bias towards a particular fill value.

From these plots I am fairly confident that my setup for the dataset and the training of the model is sound though there are clearly differences between how the residual streams are applied.

C.3 Steering adaptor

A forward hook is inserted at the 256-dim layer to extract activations for the contrastive pairs. In the case of the residual stream per layer model the hook is inserted after the residual stream.

The inputs for generating the contrastive pairs are made as

- Selecting one attribute to target steering.
- Positive examples set this attribute to 0.
- Negative examples set this attribute to 1-7.

CAA simply takes the difference of means of these two activations as a steering vector. A forward hook is registered at the same 256-dim layer which simply adds the scaled (parameterised by λ) steering vector to the output and returns it for the next layer.

To get repeated runs, a set of attributes is chosen[1] and the above process is applied to each. In the end I run 20 repeats.

Figure C.3 shows the cosine similarity of a sample of attributes. The similarity between pairs is very high as expected as the other 59 attributes are identical. The cross similarity is much lower showing that there is a variety of examples that are present when training the adaptor. This plot is essentially the same across the models with minor differences in the exact similarity values.

Figures C.4 and C.5 demonstrate the effect of the CAA steering on the different attribute values for the target attribute. The remaining attribute values are randomly selected. The experiment is run over a range of example values for each 20 repeats and for each experiment the mean over 100 test inputs is returned.

[1] For ease of implementation these are just the first n attributes

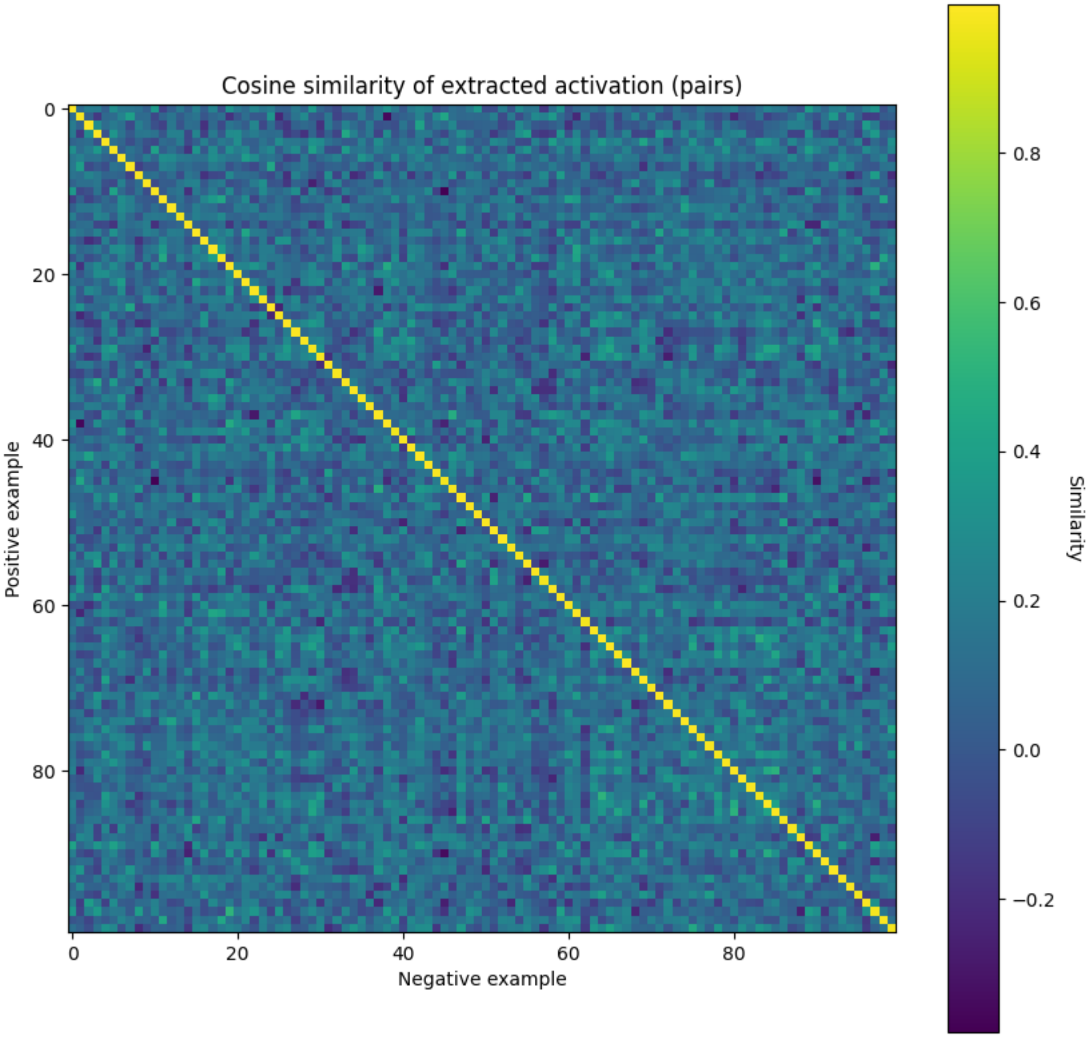


Figure C.3: The cosine similarity between a sample of positive and negative pairs. This is for a specific attribute.

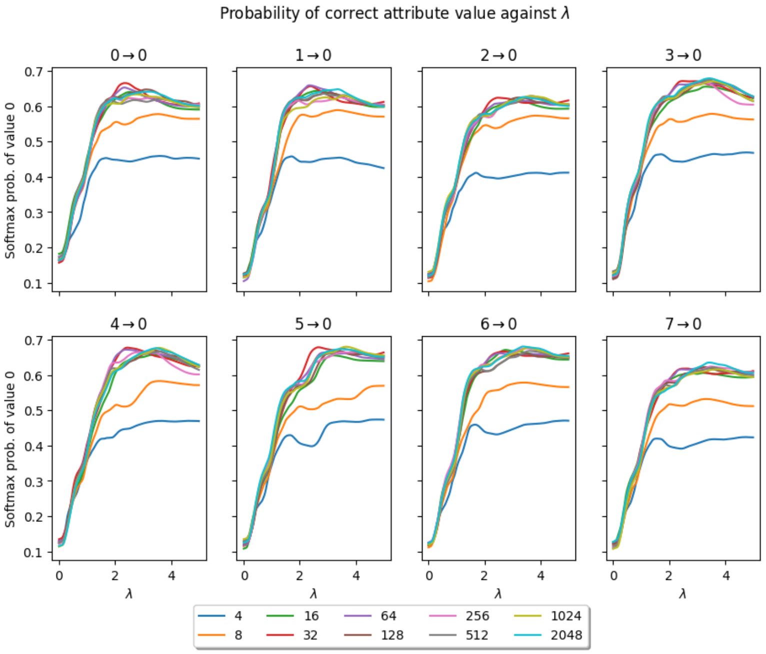


Figure C.4: The softmax probability of the target label (0) given the input label as a function of the scaling parameter λ. This is the model without any residual streams.

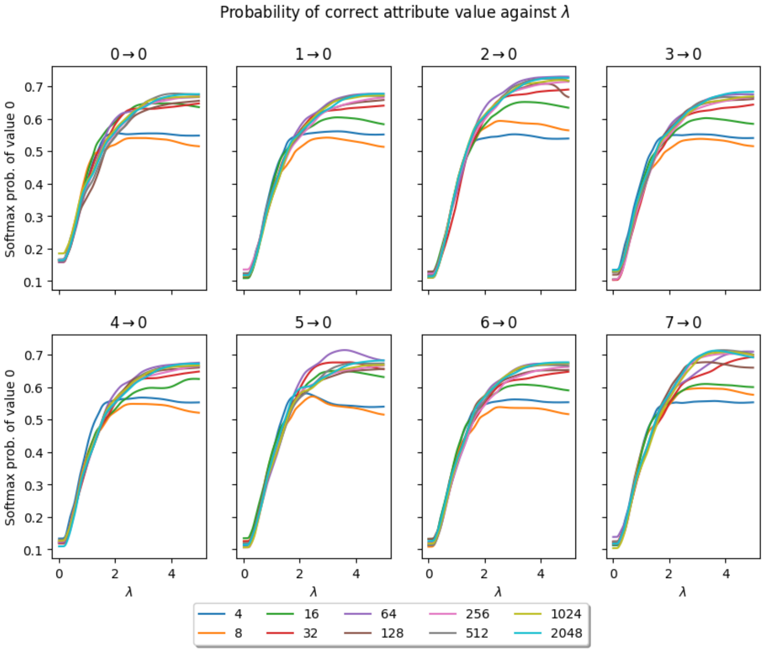


Figure C.5: The softmax probability of the target label (0) given the input label as a function of the scaling parameter λ. This is the model with a residual streams per layer.

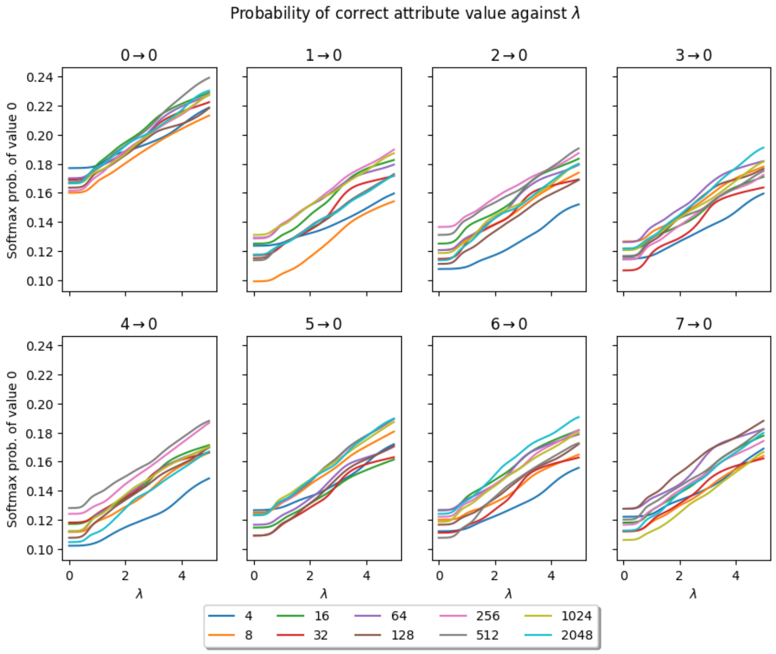


Figure C.6: The softmax probability of the target label (0) given the input label as a function of the scaling parameter λ. This is the model with a single residual stream from input to layernorm.

Clearly demonstrated is that 4 and 8 examples do not achieve the same efficacy as the other examples regardless of the strength of the steering vector. This does not take into account the effect on the attributes nor the softmax prob of the other values. These experiments demonstrate that the steering vectors are able to steer effectively.

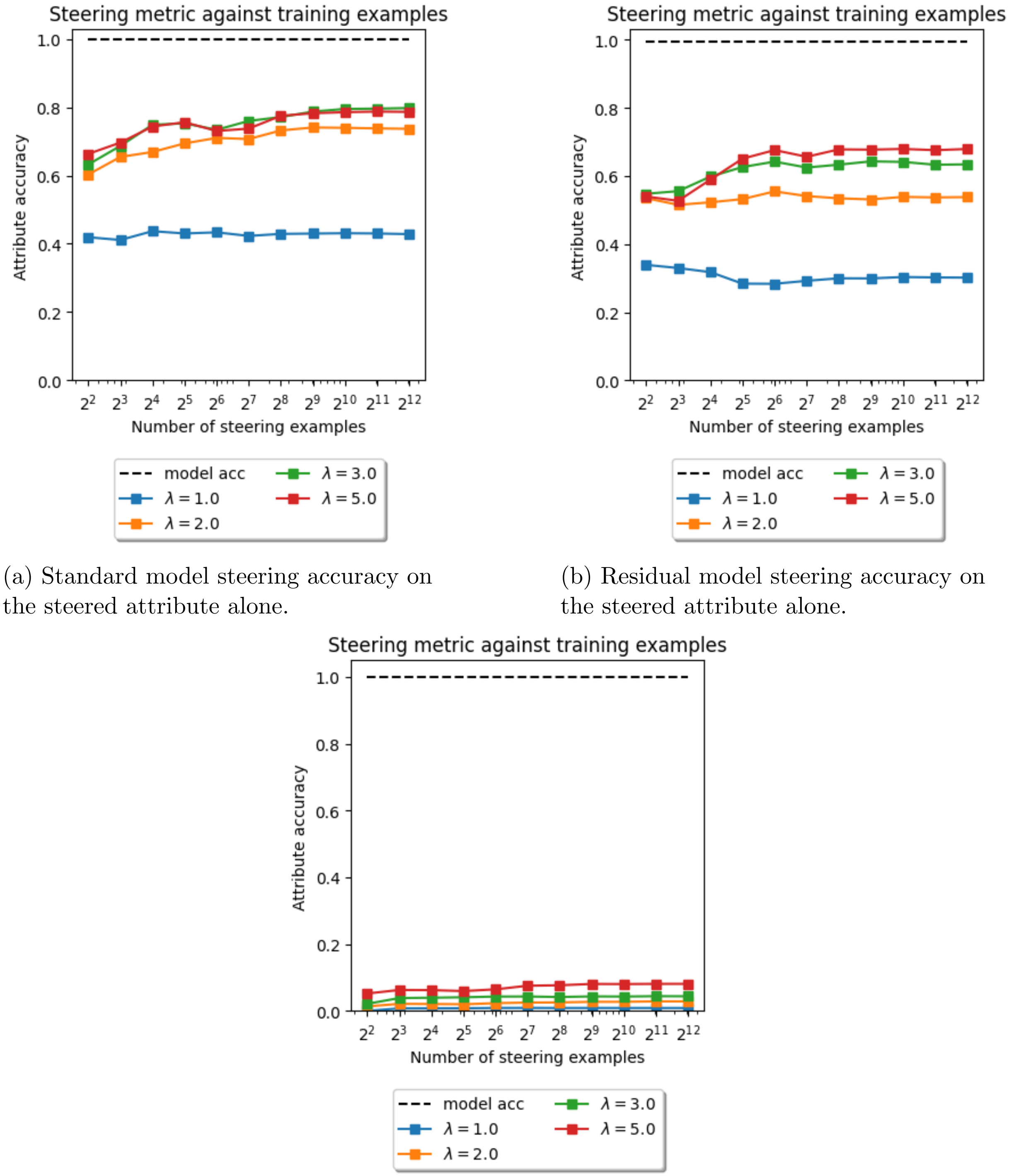
C.4 Steering metric

A subset of 1000 of the contrastive pairs above are withheld during training. The negative generating inputs are fed through the model with the trained steering adaptor and the output recorded. The goal is for the output to match the positive generating labels.

There are two metrics that I test:

- The accuracy on the steered attribute alone.
- The accuracy on all the attributes (this is the one stated in the paper).

Figures C.7a to C.7c show the attempt at reproducing the paper figures for the top-left plot in Figure 1. This only focuses on the CAA approach. This uses a different metric to



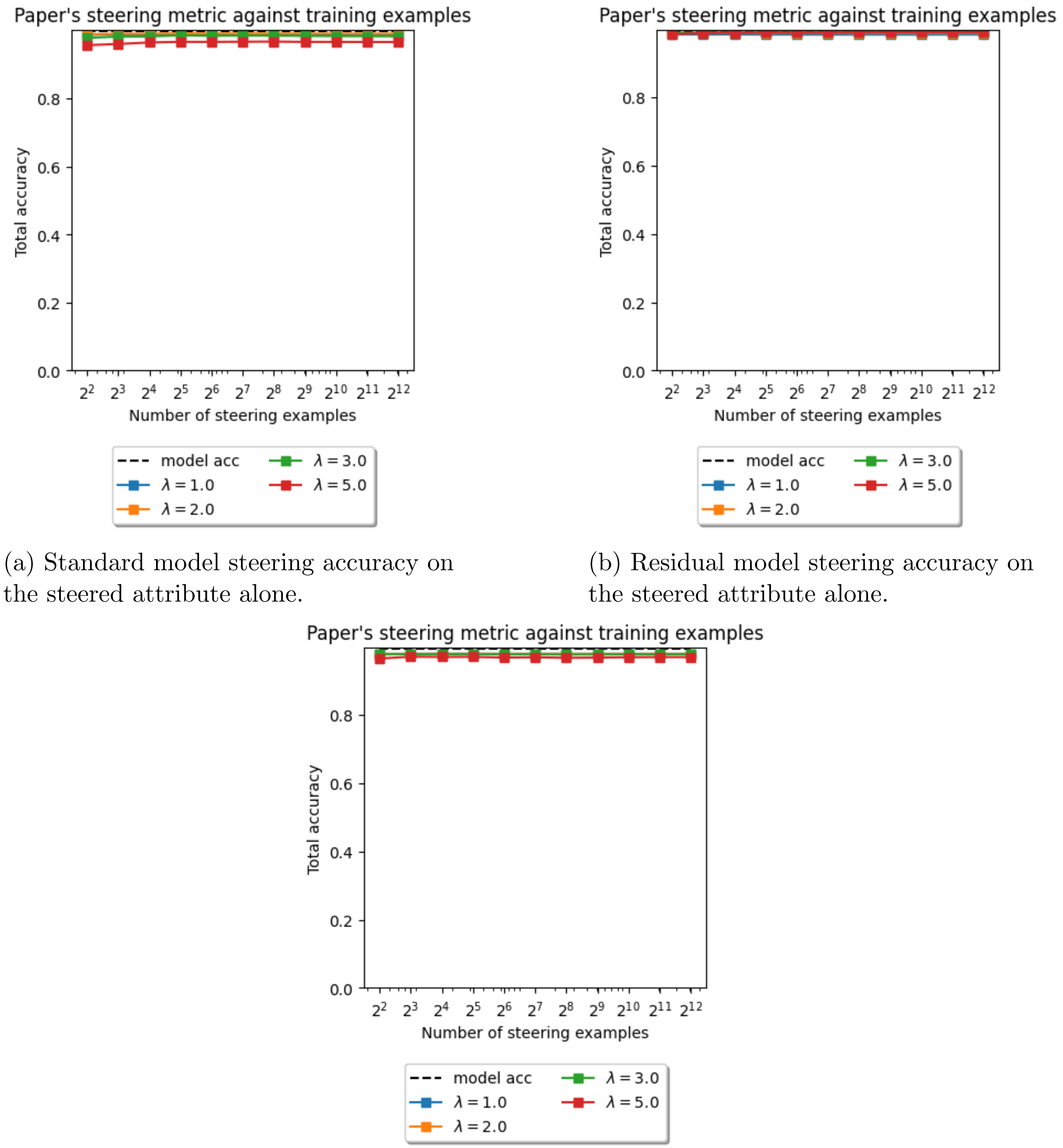
(a) Standard model steering accuracy on the steered attribute alone.

(b) Residual model steering accuracy on the steered attribute alone.

(c) Single residual stream model steering accuracy on the steered attribute alone.

the paper focusing only on attribute accuracy rather than total accuracy and the trends are similar to those in the paper. However, these plots are still not the same as the papers plots especially as they do not use the same metric.

Figures C.8a to C.8c show the reproduction using the stated steering metric of total accuracy across all target labels.



(a) Standard model steering accuracy on the steered attribute alone.

(b) Residual model steering accuracy on the steered attribute alone.

(c) Single residual stream model steering accuracy on the steered attribute alone.

Appendix D

Layer Sweeps