MASTER THESIS

Explaining and Steering Large Language Model Behavior via Sparse Autoencoders: Identifying and Manipulating Interpretable Features in Text Generation

Luka Gerlach (Student ID: 7415506)

24.02.2026

University of Cologne
Faculty of Mathematics and Natural Sciences
Department of Mathematics and Computer Science

Institute of Software Technology
Intelligent and Distributed
Systems Department

First Reviewer: Univ.-Prof. Dr. Michael Felderer

Second Reviewer: Dr. Tobias Hecking

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 161 Abs. 1 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

Luka Gerlach

Köln, den 24.02.2026

Abstract

Large Language Models (LLMs) can be steered by intervening on internal activations at inference time, offering an alternative to prompting and parameter updates. Yet, such interventions are often hard to interpret: it remains unclear what exactly a steering direction represents and which side effects it introduces. Motivated by recent mechanistic interpretability results that suggest that Sparse Autoencoders (SAEs) can extract sparse, human-interpretable features from model activations, we develop a steering pipeline that constructs steering directions from a small, inspectable set of Sparse Autoencoder (SAE) latents, directly addressing the interpretability limitations of prior steering methods.

We evaluate the method on Ekman-emotion steering (Gemma 2 9B IT and Llama 3.1 8B IT) and on broader concept steering with AxBench (Gemma 2 9B IT). Across benchmarks, our approach improves steering performance over comparable training-free baselines while making interventions more transparent through token-level interpretations of the selected latents. Finally, we present a qualitative bias-mitigation case study indicating that our approach can reduce dataset-induced topical bias transfer when building steering directions.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbrev.	**Full term**
IT	Instruction Tuned
LLM	Large Language Model
LSTM	Long Short-Term Memory
MI	Mechanistic Interpretability
MLP	Multi Layer Perceptron
NLP	Natural Language Processing
RNN	Recurrent Neural Network
SAE	Sparse Autoencoder

1 Introduction

Modern Large Language Models are trained on vast and heterogeneous corpora, which endows them with broad linguistic and world knowledge (Brown et al., 2020; Hoffmann et al., 2022; Kaplan et al., 2020). In practical deployments, however, this general capability does not automatically translate into behavior that matches a specific user intent or application requirement. Models may exhibit unwanted behavior (e.g., unsafe, biased, or unreliable outputs) due to patterns present in pretraining data (Anwar et al., 2024; Bender et al., 2021), and, even when they are generally capable, they often need to be adapted to particular tasks, styles, or constraints. This motivates *model steering*: techniques that guide model outputs toward desired characteristics without training a model from scratch.

There are many different steering techniques, and they can be applied at different points along the text-generation process. *Before inference*, behavior can be influenced via prompting (Brown et al., 2020; Wei et al., 2022; Yu et al., 2023; Zou, Wang, et al., 2023) or by updating parameters through fine-tuning (Ouyang et al., 2022). *During inference*, representation-based interventions manipulate internal activations to more directly shape the model's computation (Rimsky et al., 2024; Subramani et al., 2022; Turner et al., 2024). *After inference*, outputs can be filtered, reranked, or rewritten by external components (Bai et al., 2022; Gehman et al., 2020; Krause et al., 2021; Liu et al., 2021; Madaan et al., 2023). While these approaches have been shown to be effective at inducing desired behaviors (or suppress unwanted behavior), they often remain difficult to reason about: it is frequently unclear *why* a given method works, which internal mechanisms it activates, and what off-target effects it causes (Durmus et al., 2024). This lack of understanding makes it hard to deploy steering reliably.

Our main contribution in this work is a *representation-based* steering method with an explicit interpretability layer. We focus on a common representation-based setup in which steering signals are derived from *contrastive* data: positive text inputs that express a target concept or behavior and matched negative examples that do not (Konen et al., 2024; Rimsky et al., 2024; Subramani et al., 2022; Turner et al., 2024). The core idea is to extract an internal model representation of the target concept from these contrasts and then intervene on it during generation to increase the likelihood that the model's output exhibits the desired property.

While even representation-based steering typically relies on empirical correlations between prompts and observed activations rather than a detailed understanding of the underlying computations, we draw on recent mechanistic interpretability advances suggesting that Sparse Autoencoders can decompose such activations into sparse, more human-interpretable features (Lieberum et al., 2024; Templeton et al., 2024). Instead of steering with an uninterpreted direction in activation space, we express the intervention through a small set of selected SAE concepts and validate them via token-level interpretations. We then apply these latents during generation and evaluate trade-offs between steering success, side effects, and overall output quality across multiple benchmarks.

The rest of this work is structured as follows. Section 2 introduces the necessary background on transformer language models, categorizes common steering approaches, and reviews mechanistic interpretability and SAEs as the interpretability foundation for our method. Section 3 presents our proposed steering pipeline and motivates its design choices. Section 4 first studies how to identify high-quality SAE concepts that match a given steering target and then evaluates the method across benchmark settings. Finally, Section 5 discusses the findings, limitations, and directions for future work.

2 Background & Related Work

This chapter reviews the prerequisites that are needed for our approach and examines related work and open research gaps to position our method within the current research landscape. In each subsection, we first summarize the relevant theoretical foundations and then discuss recent work to contextualize our method in relation to the literature.

In Section 2.1, we review the general architecture of modern Large Language Models and briefly discuss the state-of-the-art open-source models used in this work. Building on this, Section 2.2 explains why steering models (i.e., altering model behavior based on user instructions) is desirable, how it works, and how different steering approaches are typically categorized. While steering interventions have been shown to be effective across many scenarios, it is often unclear why they work. In Section 2.3, we discuss how this challenge is magnified by the theory that models encode information in complex internal representations, and we outline how the field of Mechanistic Interpretability (MI) approaches this problem. Finally, in Section 2.4, we introduce Sparse Autoencoders as a recent MI technique that learns sparse, human-interpretable features from model activations and can be used both to analyze these features and to steer the model's behavior via targeted interventions.

2.1 Language Models & Transformer Architecture

Natural Language Processing (NLP) has evolved from rule-based and statistical approaches towards neural architectures with increasing representational capacity. Early neural language models using Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) architectures (Hochreiter & Schmidhuber, 1997), enabled contextual modeling over variable-length inputs (Sutskever et al., 2014) but suffered from limited parallelization and difficulties in capturing long-range dependencies. To address these limitations, Vaswani et al. (2017) introduced the Transformer architecture, which eliminates recurrence entirely in favor of self-attention mechanisms that allow the model to selectively focus on relevant parts of the input sequence when generating representations. The general architecture of a decoder-only Transformer is

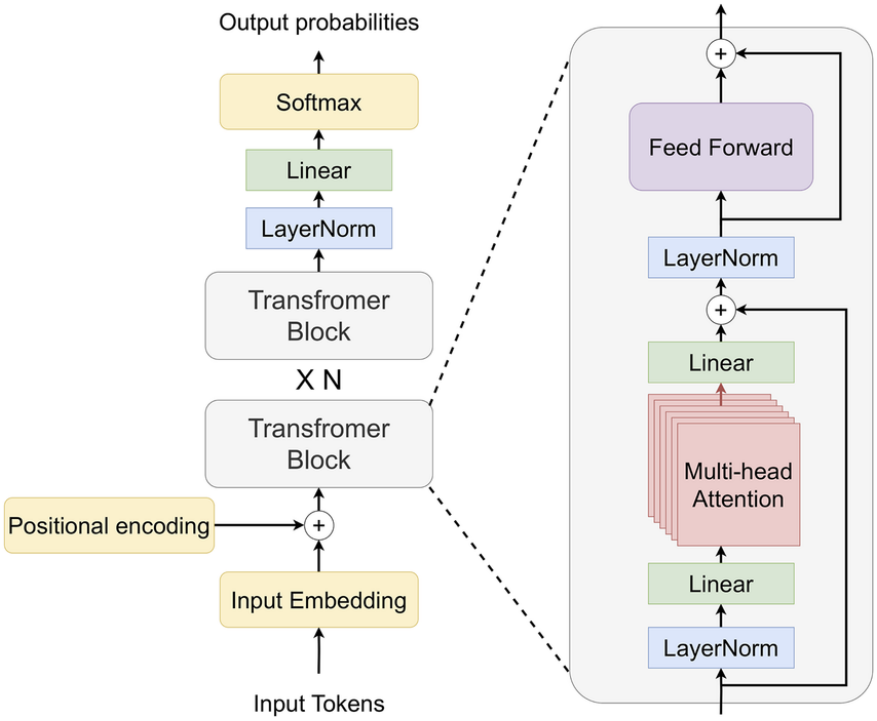

Figure 2.1: Decoder-only Transformer architecture (Chalvatzaki et al., 2023)

illustrated in Figure 2.1.[1]

2.1.1 Architecture

To generate new text, a Transformer model takes a sequence of input tokens, which are obtained by applying a tokenizer that maps input text into a sequence of subword units. The tokens are then mapped to continuous vector representations via a token embedding layer and combined with positional encodings to incorporate information about token order. These representations initialize the residual stream (depicted as the black arrow in Figure 2.1), the primary pathway carrying information through the network. Throughout the model, all representations in the residual stream maintain a fixed dimensionality of d_{model}, which is propagated through a stack of identical decoder blocks.

Each decoder block consists of two primary sublayers: a masked multi-head self-attention mechanism and a feed-forward network (cf. Figure 2.1). The self-attention mechanism computes contextual representations by allowing each token at position i to attend to all preceding positions $j \leq i$. This causal masking ensures autoregressive

[1] We focus on decoder-only Transformers (Radford et al., 2018), as all models in this work adopt this architecture. The original Transformer (Vaswani et al., 2017) used an encoder-decoder structure.

generation, where each token depends only on previously generated tokens. The mechanism operates on queries (Q), keys (K), and values (V), which are learned linear projections of the input representations, and computes attention scores that determine the relevance of each token to the current position. Formally, self-attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{1}$$

where d_k denotes the dimensionality of the key vectors (Vaswani et al., 2017). In the multi-head setting, this operation is performed in parallel across h attention heads, enabling the model to capture different types of token-to-token relationships. Each attention head operates on lower-dimensional projections of the model representations, with the dimensionality of each head typically set to $d_k = d_{\text{model}}/h$, such that the concatenation of all heads preserves the overall model dimensionality. Some attention heads have been shown to exhibit human-interpretable linguistic structure (Clark et al., 2019).

Following the self-attention sublayer, each token representation is independently processed by a feed-forward network, typically implemented as a two-layer Multi Layer Perceptron (MLP):

$$\text{FFN}(x) = W_2 \cdot \sigma(W_1 x + b_1) + b_2 \tag{2}$$

where $x \in \mathbb{R}^{d_{\text{model}}}$ is the input representation, $W_1 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ projects to an intermediate dimensionality d_{ff} (typically $4d_{\text{model}}$), σ is a non-linear activation function (commonly ReLU or GELU), and $W_2 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ projects back to the original dimensionality. This sublayer increases the expressive capacity of the model by enabling non-linear transformations of the token representations in a higher-dimensional space.[2]

After passing through the full stack of decoder blocks, the final hidden representations are normalized and projected to vocabulary-sized logits via a linear unembedding layer. Formally, given the final normalized hidden state $\mathbf{h}_{\text{final}} \in \mathbb{R}^{d_{\text{model}}}$ at a token position, the model computes logits as:

[2]We include this level of detail because the attention output (Equation (1)), the MLP output (Equation (2)), and the residual stream leaving a given layer are common attachment sites for Sparse Autoencoders (cf. Section 2.4).

$$\boldsymbol{\ell} = \mathbf{W}_{\text{unembed}}\mathbf{h}_{\text{final}} \in \mathbb{R}^{|V|} \tag{3}$$

where $\mathbf{W}_{\text{unembed}} \in \mathbb{R}^{|V| \times d_{\text{model}}}$ is the unembedding matrix, and $|V|$ denotes the vocabulary size. Each row of $\mathbf{W}_{\text{unembed}}$ corresponds to a token in the vocabulary, and the resulting logit vector $\boldsymbol{\ell} \in \mathbb{R}^{|V|}$ assigns a score to each possible next token. In other words, $\boldsymbol{\ell}$ is a vector with one entry for every token the model knows, and larger values indicate tokens the model considers more likely to come next.

2.1.2 Sampling

To generate text, the model must choose the next token based on $\boldsymbol{\ell}$. The most straightforward approach would be to deterministically select the token with the highest logit value. We call this strategy greedy decoding. However, this can lead to repetitive and overly predictable outputs. Instead, modern language models typically employ stochastic sampling methods that introduce controlled randomness while still favoring more probable tokens.

To enable probabilistic selection, the logits $\boldsymbol{\ell}$ are first transformed into a probability distribution via the softmax function. The probability of selecting token i is computed as:

$$p(i) = \frac{\exp(\ell_i/\tau)}{\sum_{j=1}^{|V|} \exp(\ell_j/\tau)} \tag{4}$$

where ℓ_i denotes the i-th element of $\boldsymbol{\ell}$, and $\tau > 0$ is the temperature parameter that controls the sharpness of the distribution. Lower temperatures ($\tau < 1$) sharpen the distribution toward high-probability tokens, producing more deterministic outputs, while higher temperatures ($\tau > 1$) flatten the distribution, increasing randomness.

Common sampling strategies include top-k sampling, which restricts selection to the k tokens with highest probability, and nucleus (top-p) sampling (Holtzman et al., 2019), which dynamically selects the smallest set of tokens whose cumulative probability exceeds a threshold p. These methods balance diversity and coherence by limiting consideration to a subset of likely candidates while preventing the selection of highly improbable tokens.

2.1.3 Training

While the architectural components and decoding strategies described above define how Transformers process and generate text, their practical effectiveness depends critically on large-scale pretraining. Training typically involves a self-supervised task using an autoregressive next-token prediction objective over massive datasets composed of web text, books, news articles, and code. The combination of scaling up parameters and extensive pretraining on heterogeneous data lets LLMs acquire broad linguistic and factual knowledge, which can be adapted to downstream tasks through fine-tuning (cf. Section 2.2) or in-context learning (Brown et al., 2020; Hoffmann et al., 2022; Kaplan et al., 2020).

LLMs have demonstrated improvements across a wide range of natural language processing benchmarks, achieving state-of-the-art performance on language comprehension benchmarks such as GLUE or SuperGLUE (A. Wang et al., 2018, 2019), as well as more recent challenging evaluations including MMLU for multitask understanding (Hendrycks et al., 2020), GSM8K for mathematical reasoning (Cobbe et al., 2021), and HumanEval for code generation (M. Chen, 2021), often without task-specific architectural modifications.

2.1.4 Model Landscape

Current state-of-the-art open-source models include Meta's Llama model family (Grattafiori et al., 2024; Touvron, Lavril, et al., 2023; Touvron, Martin, et al., 2023), particularly the Llama 3.1 models (Grattafiori et al., 2024), available at 8, 70, and 405 billion parameters in both pretrained and Instruction Tuned (IT) variants, and Google's Gemma family (Team, Mesnard, et al., 2024; Team, Riviere, et al., 2024; Team et al., 2025). While the latest Gemma 3 models (available at 1, 4, 12, and 27 billion parameters with multimodal capabilities, each in pretrained and instruction-tuned variants) represent the cutting edge of the series, the preceding Gemma 2 generation comprises six language-only models at 2, 9, and 27 billion parameters, similarly offered in both pretrained and instruction-tuned versions (Team, Riviere, et al., 2024). OpenAI recently released two instruction-tuned models at 20 and 120 billion parameters (Agarwal et al., 2025). In practice, however, our choice of base model is additionally constrained by the availability of corresponding SAEs, which narrows the set of feasible candidates.

2.2 Model Steering

As discussed in Section 2.1.3, pretraining equips LLMs with broad linguistic and world knowledge. However, it does not inherently align their behavior with specific user intentions or task requirements. Steering refers to techniques that guide model outputs toward desired characteristics, such as following instructions, adhering to formatting constraints, maintaining factual accuracy, or exhibiting particular stylistic properties, without requiring full model retraining. The need for steering arises because pretrained models optimize for next-token prediction on diverse data, which may include undesirable patterns such as misinformation, toxicity, or stylistic flaws (Anwar et al., 2024; Bender et al., 2021). Furthermore, many real-world applications require task-specific behavior that cannot be specified through pretraining alone, so several approaches have been developed to steer LLM behavior.

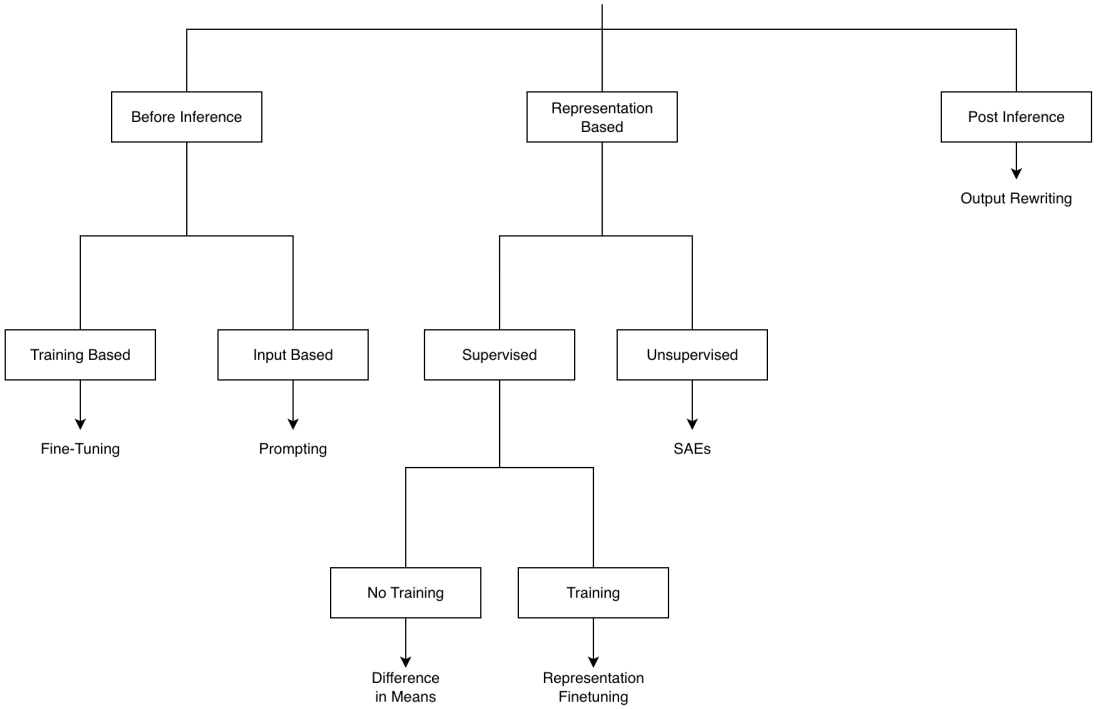

Figure 2.2: Taxonomy of model steering techniques organized by *when* an intervention is applied: *before inference* (training- and input-based methods), *during inference* (representation-based interventions), and *after inference* (post-hoc output rewriting). Representation-based methods are further divided into supervised vs. unsupervised approaches, with supervised methods split into training-based vs. training-free variants. The labels inside boxes (e.g., prompting, difference-in-means, SAEs, representation fine-tuning, output rewriting) denote illustrative examples rather than an exhaustive list.

Figure 2.2 summarizes the main categories of steering approaches considered in

this thesis and highlights representative examples. We distinguish between interventions applied *before inference* (i.e., prior to generating an output), representation-based interventions *during inference* (i.e., while the model is producing token-level activations), and *after inference* interventions that modify or rewrite the generated output. Within pre-inference methods, we separate (i) training-based approaches that update model parameters (fine-tuning) and (ii) input-based approaches that steer behavior by shaping the prompt. During inference, representation-based approaches directly intervene on internal activations; these can be supervised or unsupervised. Supervised representation-based methods can be either training-free (e.g., difference-in-means directions) or require an additional training phase (e.g., representation fine-tuning), while unsupervised approaches include Sparse Autoencoders (cf. Section 2.4). We adopt this structuring primarily because it supports a clear exposition of our method; it should not be read as the only valid or definitive categorization of steering approaches. In the following sections, we discuss and compare the major steering categories in more detail, with particular emphasis on their compute–performance trade-offs, and use this taxonomy to position our method within the broader literature.

2.2.1 Before Inference

Before-inference steering methods influence a model's behavior *prior* to generation, either by shaping the input context presented to the model (prompting) or by updating model parameters in an additional training phase (fine-tuning). Conceptually, these methods trade off flexibility, robustness, and computational cost, ranging from lightweight prompt changes to compute-intensive parameter updates.

Prompting Prompt engineering is the computationally cheapest of the steering methods considered, as it is applied *before* the model's forward pass and requires neither parameter updates nor additional training. It steers model behavior by framing the textual context provided as input, conditioning the model on task descriptions, examples, or constraints expressed in natural language. In this way, the prompt serves as a specification of desired behavior, using the model's ability to infer tasks from context. Large language models show strong generalization under such conditions, especially with examples present in prompts (Brown et al., 2020). More structured prompting schemes can further guide model outputs, e.g. by encouraging explicit

reasoning steps or more consistent instruction following (Wei et al., 2022). Because prompting relies solely on input conditioning, its effectiveness is often sensitive to phrasing and cannot reliably suppress unwanted behaviors or enforce constraints that conflict with the model's learned tendencies, as the intervention remains confined to the input layer. In practice, prompt-based control is also limited by the model's finite context window: longer or more elaborate prompts consume tokens that could otherwise be used for task-relevant information, and their influence can reduce over long generations. Moreover, prompting provides only indirect, probabilistic control: even well-crafted instructions can be overridden by strong model priors or by adversarially crafted input (prompt injection), making it difficult to guarantee consistent constraint satisfaction across inputs (Yu et al., 2023; Zou, Wang, et al., 2023).

Fine-Tuning Fine-tuning adapts a pretrained model to a specific task or domain by continuing training on a selected dataset, updating model parameters to a achieve a desired behavior. This additional training phase makes it the most computationally intensive method introduced and its effectiveness depends strongly on the availability, quality, and representativeness of the fine-tuning data. It has been successfully applied to a wide range of NLP tasks, including text classification (Howard & Ruder, 2018) and text understanding (Devlin et al., 2019). Fine-tuning has also been used to align large language models toward conversational generation and to mitigate toxic or otherwise undesirable behaviors acquired during pre-training, often in combination with human feedback[3] (Ouyang et al., 2022). This type of fine-tuning is typically referred to as *instruction tuning* and is applied to most state-of-the-art chat-models. Compared to representation-based steering, fine-tuning is conceptually straightforward to deploy because it does not require selecting internal hook points or explicitly intervening in model internals at inference time; instead, the desired behavior is encoded in the updated parameters. Despite its effectiveness, instruction tuning alone does not guarantee robust or consistent adherence to desired constraints, particularly in the presence of adversarial prompts or conflicting instructions. As a result, Instruction Tuned (IT) models may still exhibit unsafe, biased, or otherwise unintended behavior at inference time (Gehman et al., 2020; Nangia et al., 2020; Perez et al., 2022; Zhao

[3]Learning from human feedback can technically be considered a separate steering approach. For the sake of the scope of this work, it is discussed only in conjunction with fine-tuning.

et al., 2018).

2.2.2 Representation-Based Steering

Representation-based methods intervene directly on a model's internal representations during inference. Unlike prompting, which conditions behavior through input text alone, and fine-tuning, which updates model parameters in an additional training phase, activation steering operates on the model's *hidden states* at inference time, enabling more direct and reversible control over the computational processes underlying generation. Hidden states refer to intermediate latent representations computed at each transformer layer at inference time. Interventions can be applied at multiple points in a transformer block: on the residual stream, on the outputs of the model's attention mechanism (as defined in Equation (1)), or on the outputs of the MLP block (as defined in Equation (2)).

Formally, representation-based steering can be expressed as an additive intervention on the model's hidden state at a chosen layer ℓ. For a new prompt p_{new}, the modified activation $a'_\ell(p_{\text{new}})$ is obtained as:

$$a'_\ell(p_{\text{new}}) = a_\ell(p_{\text{new}}) + \lambda \cdot v_\ell \tag{5}$$

where v_ℓ is a *steering vector* that encodes the target concept or property we want to steer the model toward, and λ is a scaling coefficient that controls the strength of the intervention. This operation constitutes a linear intervention in activation space, as it modifies the model's internal representations through an additive shift along a fixed direction without altering the model's parameters or introducing non-linear transformations[4]. The modified activation is then propagated through the remaining layers during inference and can be applied at every forward pass throughout the generation process.

Different representation-based steering methods primarily differ in how they construct a useful steering vector v_ℓ. Broadly, this can be done via unsupervised or supervised approaches. Unsupervised approaches often employ Sparse Autoencoders (SAEs) to discover interpretable directions in the model's activation space without

[4]This idea is motivated by the linear representation hypothesis: if concepts correspond to linear directions in activation space, then adding v_ℓ implements a targeted shift along such a concept direction. A more detailed explanation of this is given in Section 2.3.1.

requiring labeled data (cf. Section 2.4). In contrast, supervised methods typically rely on paired examples of desired and undesired behaviors.

As depicted in Figure 2.2, within supervised methods, we further distinguish between *training-free* and *training-based* approaches. Training-free methods compute v_ℓ directly from labeled examples (e.g., via summary statistics over activations), whereas training-based methods fit an auxiliary component, such as a linear classifier/regressor over activations, or directly optimize v_ℓ on labeled data to obtain a direction that separates the target and non-target behavior. Importantly, *training-based* here refers to training this auxiliary component (or v_ℓ itself) while keeping the base language model frozen; it should not be conflated with fine-tuning, which updates the language model parameters. As in the general trade-off between prompting and parameter updates, the choice between training-free and training-based supervised steering largely reflects a compute–efficiency trade-off: training incurs additional upfront cost but can yield a more robust or better-targeted v_ℓ.

A prominent supervised technique is the difference-in-means method, which computes steering vectors by taking the mean activation difference between contrastive prompt sets (Konen et al., 2024; Rimsky et al., 2024; Turner et al., 2024). Given a set of P positive prompts $\{p_i^+\}_{i=1}^P$ and N negative prompts $\{p_i^-\}_{i=1}^N$, the steering vector v_ℓ at layer ℓ is computed as:

$$v_\ell = \frac{1}{P}\sum_{i=1}^{P} a_\ell(p_i^+)_{[\text{pos}]} - \frac{1}{N}\sum_{i=1}^{N} a_\ell(p_i^-)_{[\text{pos}]} \tag{6}$$

where $a_\ell(p)$ denotes the hidden state activation at layer ℓ for prompt p, and the subscript [pos] indicates extraction at a specific token position (e.g., the final token of prompt p).

From a computational perspective, supervised activation steering, particularly the difference-in-means method, represents a middleground between prompting and training approaches: it requires only forward passes over prompt sets and the collection of activations, avoiding the computational expense of optimizing parameters. However, this computational advantage comes with additional procedural complexity compared to both prompting and fine-tuning: instead of directly specifying behavior through text (prompting) or encoding it through parameter updates (fine-tuning),

activation steering adds an extra layer of design choices around how to construct a *good* steering direction (e.g., contrastive dataset design, layer/hook-point choice, and activation aggregation), and more sophisticated methods can introduce further modeling and validation steps beyond simple difference-in-means. This efficiency and simplicity has made difference-in-means a common *training-free supervised* baseline for steering topical attributes (e.g., contrastive "love"–"hate" prompt sets) (Turner et al., 2024) as well as broader behavioral tendencies such as sycophancy (Rimsky et al., 2024). Konen et al. (2024) further applied difference-in-means to steer sentiment and Ekman emotions (Ekman, 1992), and highlighted an important failure mode: steering directions can inherit bias from the data used to derive them (e.g., sentiment vectors built from Yelp reviews inducing restaurant-related bias in model generations) (Konen et al., 2024). Relatedly, unintended off-target effects have been documented in further large-scale studies (Durmus et al., 2024).

Beyond difference-in-means, early work by Subramani et al. (2022) constructed steering directions through gradient-based optimization. In our taxonomy, such methods fall under *training-based supervised* steering, as they explicitly optimize a steering vector (or an auxiliary objective) on labeled examples while keeping the base model frozen. More recent training-based methods aim to improve controllability and robustness via stronger optimization or by training a seperate concept-to-vector mapping model, e.g. RePs (Wu, Yu, et al., 2025) and HyperSteer (Sun et al., 2025).

Representation-based steering has also been applied beyond stylistic control, including toxicity and bias mitigation as well as broader alignment-style interventions (Arditi et al., 2024; Lee et al., 2024; Turner et al., 2024; M. Wang et al., 2025; Zou, Phan, et al., 2023). R. Chen et al. (2025) showed that persona traits of LLM assistants can be modulated via activation addition. In this thesis, we build on this line of work by proposing a *representation-based*, *training-free supervised* method that constructs steering directions from labeled contrastive examples, but grounds the resulting intervention in a small, interpretable set of SAE latents with explicit token-level readouts (Section 3.2).

2.2.3 Post Inference

In addition to interventions that modify a model's internal computation during generation (e.g., activation steering), many practical systems apply *post-inference* (or *output-level*) steering, where the model first produces one or more candidate completions and an external signal is then used to select, filter, or rewrite the final response. Compared to prompting, fine-tuning, or representation-based steering, post-inference methods do not change the model's internal computations that produced the initial output; instead, they treat the base model as a generator and apply control at the level of outputs and selection. Practically, this makes post-inference steering highly modular and easy to layer on top of existing systems, and it can serve as a safety backstop when pre- or in-generation methods fail, though it typically incurs additional latency and can be limited by the reliability of the external scoring signal. A common pattern is *classification-based filtering* or *reranking*: a toxicity, bias, or policy classifier (or more generally a reward model) scores candidate outputs and either rejects unsafe responses or selects the best completion among multiple samples (Gehman et al., 2020; Ouyang et al., 2022). Related approaches integrate discriminative signals more tightly with generation by combining a base model with expert/anti-expert models or discriminator guidance (Krause et al., 2021; Liu et al., 2021). Another widely used pattern is *generate–critique–revise*, where an auxiliary model (or the same LLM prompted as a critic) produces natural-language feedback that is then used to iteratively refine the completion (Bai et al., 2022; Madaan et al., 2023). While these methods operate after decoding and do not require access to hidden states, they can be combined with activation-level interventions in hybrid pipelines.

2.2.4 Evaluating Steering Interventions

Having seen the main classes of steering approaches, the practical next question is how such interventions can be evaluated and compared in a meaningful way. In practice, steering often touches multiple objectives at once: E.g., target adherence (does the model exhibit the desired behavior?), robustness (does the effect generalize across prompts?), and side effects (are capabilities or safety properties degraded?). Evaluation should capture both steering-specific effects and overall model performance. We therefore briefly review common benchmark families and use them to motivate the

evaluation setups used in this work.

While large-scale benchmarks exist for evaluating (steered) LLM generations, they typically target specific use-cases, often times safety-related scenarios. For instance, HarmBench (Mazeika et al., 2024) measures how models respond to malicious requests and whether they appropriately refuse them. Similarly, RealToxicityPrompts (Gehman et al., 2020) assesses the tendency of models to generate toxic language, while CrowS-Pairs (Nangia et al., 2020) quantifies social biases in LLM completions. These specialized benchmarks are frequently paired with general capability assessments such as MMLU (Hendrycks et al., 2020), which measures multitask accuracy across subjects spanning from mathematics to humanities, and HellaSwag (Zellers et al., 2019), which evaluates commonsense reasoning via sentence completion. Other commonly used capability benchmarks include GSM8K (Cobbe et al., 2021) for mathematical reasoning, HumanEval (M. Chen, 2021) for code generation, and TriviaQA (Joshi et al., 2017) for question answering. This combination helps to determine whether steering interventions maintain or compromise the model's foundational performance across diverse domains.

However, the benchmarks described above have two key limitations: they concentrate on narrow aspects of steering, and they tend to be extensive in scale, as they were originally designed to evaluate the general capabilities of LLMs rather than steering-specific effects.

In this work, we therefore consider both established, narrower evaluation settings like steering Ekman emotions (Ekman, 1992; Konen et al., 2024) and broader, concept-diverse benchmarks such as AxBench (Wu, Arora, et al., 2025). Since large-scale capability benchmarks are often too expensive to run repeatedly for each steering configuration, we approximate overall quality degradation using lightweight proxy metrics within our benchmark setups (e.g., completion fluency and other measures of output coherence).

2.3 Mechanistic Interpretability

While the steering methods introduced above aim to guide model behavior by manipulating inputs, outputs, or intermediate activations, they largely treat the model as a black box whose internal organization remains implicit. Prompting conditions

operate through input text, and fine-tuning optimizes outputs via gradient-based updates, while even activation steering typically relies on empirical correlations between prompts and observed activations rather than a detailed understanding of the underlying computations.

The research field of Mechanistic Interpretability (MI) aims to open this black box by directly examining the internal structure of neural networks, seeking to understand how features and concepts are encoded and composed within a model's parameters and activations. The terms features or concepts are used interchangeably and can, in the case of LLMs, refer to concrete elements like specific words, as well as more abstract properties such as topics, syntactic structures, or sentiment. However, Mechanistic Interpretability is not exclusive to Large Language Models (LLMs). It applies to multiple domains, including early observations of linear structure in word embeddings (Mikolov et al., 2013), such as mechanistic analyses of vision models that identified neurons and circuits corresponding to human-interpretable features (Bau et al., 2017). Rather than focusing solely on input–output behavior, MI emphasizes internal explanations that are grounded in the model's learned representations and computations. For the purposes of this work, two claims emerging from this literature are particularly central: the linear representation hypothesis (Section 2.3.1), which posits that many meaningful features are represented along approximately linear directions in activation space, and the superposition hypothesis (Section 2.3.2), which argues that neural networks compress many such features into shared representational dimensions, leading to dense and entangled internal representations.

2.3.1 Linear Representation Hypothesis

As discussed in Section 2.2.2, representation-based steering assumes that concepts correspond to directions in a model's activation space: by adding a steering vector v_ℓ to a hidden state (Equation (5)), we linearly shift the representation toward (or away from) a target property that we want to manifest in the model's generations. The linear representation hypothesis provides the representational justification for why such additive interventions can work. It posits that features learned by neural networks correspond to approximately linear directions in their activation spaces, so that individual concepts, attributes, or abstract properties can be activated, suppressed,

or combined through linear operations such as vector addition and subtraction. As mentioned, this hypothesis has been shown to empirically hold for distributional word representations (Mikolov et al., 2013), where linear relationships between embedding vectors capture semantic regularities.

More recent work extends this observation to deeper neural networks, including transformers, where linear probes and activation interventions have been shown to reliably extract or modify specific features and behaviors in model generations, such as sentiment or topic-related information (Konen et al., 2024; Rimsky et al., 2024; Turner et al., 2024), stylistic properties (e.g., formality or toxicity), and the presence of particular entities or syntactic patterns (Elhage et al., 2022; Park et al., 2023). However, LLM activations, particularly in the residual stream, exist in an *unprivileged basis*: semantically meaningful directions do not generally align with individual basis directions (single neurons)[5] (Elhage et al., 2022). Consequently, meaningful features are typically encoded as distributed linear combinations across many dimensions rather than as single axes. This is disadvantageous for interpretability because no single neuron or coordinate can be straightforwardly associated with a human-understandable concept.

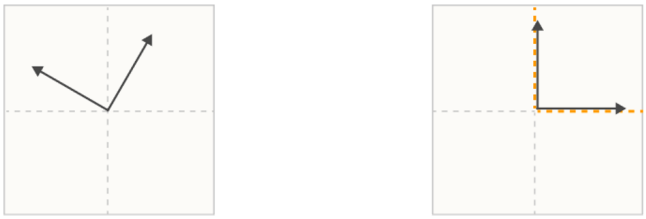

Figure 2.3: Concept vectors in an unprivileged basis (left) and in a privileged basis (right) (Elhage et al., 2022)

An intuitive two-dimensional visualization of this phenomenon is shown in Figure 2.3. The dashed lines represent the basis dimensions, while the arrows denote concept vectors. In the left panel, the concept vectors are not aligned with the basis directions, indicating an unprivileged basis. In the right panel, the concept vectors align with the basis dimensions, corresponding to a privileged basis. Importantly, in both cases the concept vectors are orthogonal to each other, meaning they remain perfectly separable by the model despite their differing alignment with the basis.

[5]This follows from the rotational equivariance of residual stream representations and intervening linear transformations (cf. Section A.1).

More intuitively, orthogonality here means that the underlying *features* or *concepts* are independent: the model can represent, read out, and manipulate each concept without introducing ambiguity or trade-offs with the other.

2.3.2 Superposition Hypothesis

Given the challenges posed by an unprivileged basis, one might hope that identifying or constructing a privileged basis, in which semantically meaningful features align with individual basis directions, would resolve the interpretability problem entirely. In such a basis, each concept would correspond to a single dimension, making features directly accessible and easily interpretable. However, even in a privileged basis, a fundamental constraint remains: an n-dimensional space can accommodate at most n mutually orthogonal vectors.

Since neural networks must represent far more than n distinct concepts, perfect orthogonality becomes geometrically impossible. The superposition hypothesis addresses this tension by proposing that models can represent many more features than dimensions as *nearly*-orthogonal directions, trading perfect separability between concepts for vastly increased representational capacity (Elhage et al., 2022). Under superposition, features are represented by vectors whose pairwise dot products are small but non-zero, allowing the model to distinguish them in most cases while accepting occasional interference. A direct consequence of superposition is that individual neurons must respond to multiple distinct features, a property known as *polysemanticity*. Superposition enables neural networks to compress a rich feature space into a limited number of dimensions, at the cost of introducing some interference between features. One can also interpret this as the model performing a noisy approximation of a higher-dimensional model (Elhage et al., 2022). A visual illustration of this effect is shown in Figure 2.4.

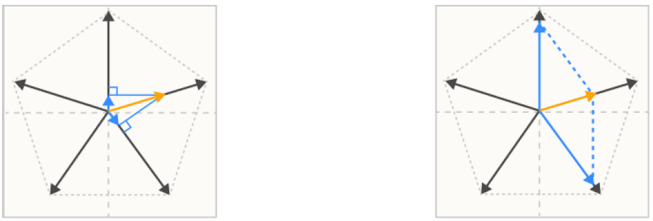

Figure 2.4: Visualization of the superposition hypothesis. An n-dimensional space holding $> n$ features (Elhage et al., 2022)

In this example, five concepts (again depicted as arrows) are embedded in a two-dimensional space, making it impossible for all concepts to be mutually orthogonal. In the left panel, activating the orange concept produces side effects on its neighboring concepts, indicating interference between their representations. This interference arises because the orange concept has non-zero projections onto the directions of the other concepts. As a result, these concepts appear slightly activated as well, as indicated by the small blue arrows. While this interference typically manifests as low-magnitude noise that downstream components can tolerate or filter out, it can occasionally cause failures. In the right panel, when both blue concepts are strongly active, their projections onto the orange concept add constructively, activating it significantly and causing the model to incorrectly infer that the orange concept is present as well.

Thus, a crucial prerequisite for the superposition hypothesis to be viable is *sparsity*, i.e. the assumption that for any given input, only a small fraction of the model's representable concepts are strongly active at once. When sparsity holds, the interference from overlapping feature directions remains manageable, as most projections are weak enough that downstream components can tolerate or ignore them. However, when sparsity is violated and multiple overlapping features activate strongly on the same input, their projections can add up and amplify interference, causing the model to falsely detect features that are not actually present (as illustrated in the right panel of Figure 2.4). Sparsity is a fundamental condition that allows superposition to trade perfect separability for increased representational capacity without incurring significant performance loss.

2.4 Sparse Autoencoders

Although superposition enables neural networks to represent more features than their dimensionality would naively allow, it creates significant interpretability challenges. When networks encode multiple features as nearly-orthogonal directions within the same representational space, individual concepts become intertwined rather than neatly separated. The linear representation hypothesis tells us that features correspond to directions in activation space, but superposition means these directions overlap and interfere with one another rather than aligning with distinct dimensions. This overlap

makes it substantially harder to identify, understand, or selectively modify individual concepts, since no single dimension cleanly corresponds to a single meaningful feature.

Sparse Autoencoders (SAEs) offer an approach to disentangle these superposed representations to improve the interpretability of model activations: they aim to learn a sparse feature space in which model activations can be decomposed into human-interpretable concepts that align with basis directions. The idea behind them is not exclusive or new to LLMs and has been applied to other NLP-models (Faruqui et al., 2015; Subramanian et al., 2018), but showed promising results in LLMs as well (Bricken et al., 2023). Figure 2.5 shows the general architecture of a SAE.

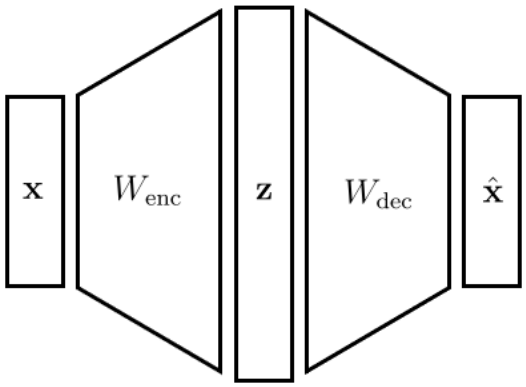

Figure 2.5: Architecture of a Sparse Autoencoder (SAE)

SAEs serve two complementary purposes. First, they expand the representational space to a dimension large enough that concepts no longer need to be stored in superposition. Returning to the analogy of the model simulating a higher-dimensional network, the SAE provides sufficient space for this larger network to operate explicitly. Second, by optimizing for sparse reconstruction (the mechanisms are detailed below), they implicitly encourage the discovery of a privileged basis in which individual concepts tend to align with basis directions, and we call these basis directions inside the SAE *latents*. This is beneficial for interpretability because it makes it possible to associate single directions/latents with specific, human-meaningful features, enabling more targeted analysis than in dense and superposed representations. These properties implicitly emerge from the architecture and training objective of a SAE.

2.4.1 Architecture

Technically, a SAE is a shallow network that encodes an input into an overcomplete, i.e. higher-dimensional, latent space and then reconstruct it back to the original dimension (cf. Figure 2.5). Given an input activation $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$, a SAE consists of:

- An encoder that maps to a higher-dimensional latent space:

$$z = \sigma(W_{\text{enc}}x + b_{\text{enc}}) \in \mathbb{R}^{d_{\text{SAE}}} \tag{7}$$

  where

$$d_{\text{SAE}} \gg d_{\text{model}} \quad \text{and} \quad W_{\text{enc}} \in \mathbb{R}^{d_{\text{SAE}} \times d_{\text{model}}}$$

- A decoder that reconstructs the original activation:

$$\hat{x} = W_{\text{dec}}z + b_{\text{dec}} \in \mathbb{R}^{d_{\text{model}}} \tag{8}$$

  where

$$W_{\text{dec}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{SAE}}}$$

A non-linear activation function σ (usually ReLU) is applied after the encoder's linear transformation. In terms of dimensionality, d_{SAE} is set to a multiple of d_{model} in practice, with the ratio $\frac{d_{\text{SAE}}}{d_{\text{model}}}$ referred to as the *expansion factor*. Without additional constraints, achieving perfect reconstruction in this setup would be trivial: the SAE could simply learn to use d_{model} dimensions of the latent space as an identity mapping, ignoring the remaining dimensions entirely. However, instead of focusing on reconstruction alone, we want another characteristic to hold for our SAE embeddings: *Sparsity*. In this context, sparsity means that for any given input, only a small fraction of the d_{SAE} latents should have non-zero activations. Sparsity is a learned property of an SAE and is established during training.

2.4.2 Training

Traditional SAEs (Bricken et al., 2023; Lieberum et al., 2024; Rajamanoharan et al., 2024) force sparsity by optimizing the following loss:

$$\mathcal{L} = \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}_{\text{reconstruction}} + \lambda \underbrace{\|\mathbf{z}\|_1}_{\text{sparsity}} \tag{9}$$

where the total loss is divided into two terms: the first term measures reconstruction error and the second term is the L1 sparsity penalty that encourages latent activations to be zero. The hyperparameter λ controls the trade-off between these two objectives: higher values of λ give more weight to the sparsity penalty and thus produce sparser representations at the cost of worse reconstruction, while lower values allow better reconstruction but with more active latents.

TopK SAEs (Gao et al., 2024) take a more direct approach by enforcing sparsity architecturally: Setting the activation function σ to TopK, it keeps only the K latents with the highest activations and zeros out all others. In this case, the loss function simplifies to pure reconstruction error:

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{10}$$

since sparsity is guaranteed by the activation function itself. Here, K becomes the new trade-off parameter: smaller values of K enforce stronger sparsity but may harm reconstruction quality, while larger values allow better reconstruction at the cost of having more active features per input. BatchTopK (Bussmann et al., 2024) extends this approach by applying the TopK operation across all activations in the entire batch rather than per sample, maintaining an average of K active features per sample while allowing the number of active features to vary flexibly across individual samples.

In practice, SAEs are trained on activations collected from a pretrained LLM. The model weights are frozen, and the SAE is trained to reconstruct activations $\mathbf{x}$ drawn from a specific layer and component of the network, such as the residual stream, attention outputs, or MLP activations at a given layer. Activations are obtained by running the model over large unlabeled text corpora (e.g., the models pretraining data) and recording the resulting hidden states token-by-token.

Across both variants, L1 and TopK, the central idea remains that by forcing the SAE to reconstruct activations using only a sparse subset of latents in an overcomplete basis, we encourage it to discover latent directions that correspond to meaningful, disentangled features[6], that have empirically shown to correspond to human-interpretable concepts (Bricken et al., 2023; Cunningham et al., 2023; Gao et al., 2024; Marks et al., 2024; Templeton et al., 2024). The next paragraph explains how we can map latents to understandable concepts.

2.4.3 Interpreting Latents

Once a SAE has been trained, each latent dimension can be studied as a candidate feature that activates selectively for a narrow set of inputs that belong to a interpretable concept. A common first step is *activation-based analysis*: by collecting tokens or token-sequences that maximally activate a given latent, again taken from a large body of unlabeled text like the SAEs pretraining data, we can often identify a coherent pattern, such as a syntactic role, semantic topic, or stylistic property. Building on this, *auto interpretation* methods attempt to scale this process by using LLMs themselves to summarize or label latents. Concretely, the text snippets that correspond to the token-sequences that strongly activate a latent are fed to a separate model, which is prompted to propose a natural-language description of the shared concept across snippets, allowing thousands of latents to be interpreted with minimal human supervision (Bills et al., 2023). This basic procedure can be arbitrarily extended by providing the model with increasingly structured information about latent activations, enabling more accurate estimation and interpretation of the latent and is an active area of research (Bricken et al., 2023; Lin, 2023; Paulo et al., 2024; Rajamanoharan et al., 2024).

Beyond interpretation, individual SAE latents can also be used as interventions for steering. Concretely, one can add a latent's decoded direction to the residual stream during generation, thereby nudging the model toward the corresponding feature; prior work reports single-latent steering effects such as the "Golden Gate Bridge" feature (Templeton et al., 2024). However, recent work emphasizes a key limitation: just because a feature reliably activates on certain inputs does not mean that turning it up

[6]Note that the SAE architecture inherently establishes a privileged basis, allowing us to expect that features will correspond to individual latents in the SAE embeddings. Again, we direct to Section A.1 for some further explanation.

will reliably produce the corresponding behavior in the model's outputs (Durmus et al., 2024; Paulo et al., 2024).

Another tool to interpret latents is the *logit lens* (nostalgebraist, 2020), which connects latents directly to model outputs. Recall from Section 2.1 that the final layer of a transformer projects normalized hidden representations to vocabulary-sized logits via a linear output layer $\mathbf{W}_{\text{unembed}} \in \mathbb{R}^{|V| \times d_{\text{model}}}$, where $|V|$ denotes the vocabulary size. The logit lens applies this same unembedding matrix to intermediate layer representations, revealing which tokens would be promoted or suppressed if generation were to stop at that point in the network.

We can analyze individual latent directions by examining the decoder weight corresponding to that latent. Given a latent i, its decoder column $\mathbf{w}_{\text{dec}}^{(i)} \in \mathbb{R}^{d_{\text{model}}}$ represents the direction in activation space that this latent induces. Projecting this direction through the unembedding matrix:

$$\ell_i = \mathbf{W}_{\text{unembed}} \cdot \mathbf{w}_{\text{dec}}^{(i)} \in \mathbb{R}^{|V|} \tag{11}$$

yields a vocabulary-sized vector ℓ_i that directly reveals which tokens are promoted (positive logit values) or suppressed (negative logit values) when latent i is active. Importantly, logits are on a continuous scale: higher logits correspond to relatively stronger promotion (i.e., higher probability under the subsequent softmax), while lower logits correspond to weaker promotion or suppression. This approach requires no forward passes and provides immediate insight into a latent's effect on token predictions.

The logit lens is particularly informative for SAEs trained on later layers, where residual stream representations have accumulated information across many preceding computations and are closer to the final output projection. In these layers, the linear map to vocabulary logits provides a more direct and interpretable connection between latent activations and token predictions (nostalgebraist, 2020).

Taken together, SAEs provide a practical interface between the dense internal state of a LLM and a more human-interpretable feature space: they decompose activations into a sparse set of latents that can often be inspected, labeled, and related back to model behavior. This makes them attractive both for mechanistic analysis, because individual latents admit targeted attribution and interpretation pipelines, and

for steering, because the same latents can be used in a controlled way to probe or modify specific aspects of the model's computations.

2.4.4 Public SAE Releases and Design Dimensions

Having seen how SAEs are constructed, trained, and interpreted, a practical next question is which SAEs are actually available for modern open-weight LLMs, especially the models we have seen in Section 2.1.4.

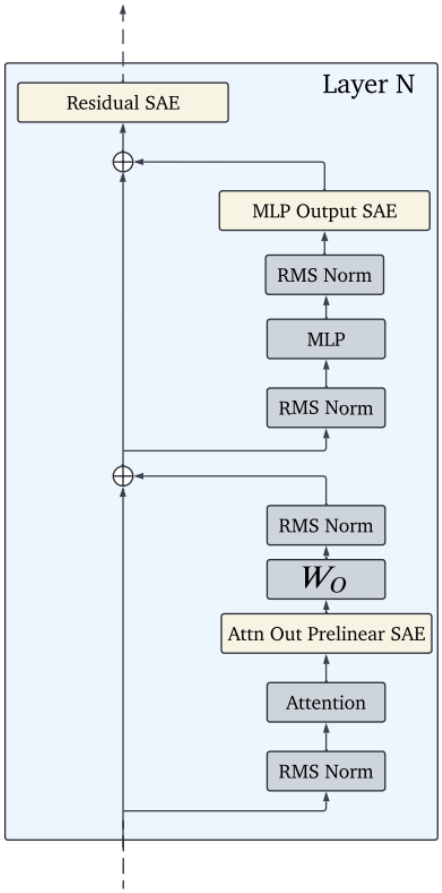

Figure 2.6: Hook points of Google's Gemma Scope SAEs. SAEs are commonly trained at different locations within a transformer layer, such as attention head outputs, the MLP block output, and the residual stream at the end of the layer. Figure adapted from (Lieberum et al., 2024).

Across releases, SAEs differ along a few axes. First, they are tied to a particular *base model* and often to a specific checkpoint (e.g., pretrained vs. instruction-tuned). Second, they are trained on activations taken from particular *layers* and *hook points* within the transformer block (e.g., residual stream, attention outputs, or MLP outputs; cf. Figure 2.6), which influences what information the SAE has access to. Third,

releases vary in *latent dimensionality* (often expressed via the expansion factor $\frac{d_{\text{SAE}}}{d_{\text{model}}}$), trading off feature capacity and training cost. Finally, they differ in the *sparsity mechanism* (e.g., an explicit L_1 penalty as in Equation (9) vs. TopK-style activations as in Equation (10)) and in the effective target sparsity (e.g., the typical number of active latents per token), as well as in broader training details such as data and optimization choices, which can affect reconstruction quality and feature stability.

Several groups have released SAEs for widely used model families; prominent examples include Google's Gemma Scope suite for Gemma 2 (Lieberum et al., 2024). For Llama 3.1, LlamaScope (He et al., 2024) and EleutherAI (Eleuther, 2024) both provide SAEs for the *pretrained* base models, while Arditi and RujinChen (2025) release SAEs trained on *instruction-tuned* Llama 3.1 checkpoints. We report the concrete model–SAE configurations we use (e.g., layers, hook points, latent sizes, and sparsity settings) as part of the experimental setup in Section 4.

3 Method

The preceding chapters have established the fundamental concepts necessary to understand and motivate our methodology. This chapter presents our proposed approach for constructing interpretable steering directions from SAE latents. For clarity, we structure the method as a multi-stage pipeline, detailing the algorithmic procedures, hyperparameter choices, and design decisions at each step. We begin with a high-level overview of the complete pipeline before providing detailed explanations of each individual stage.

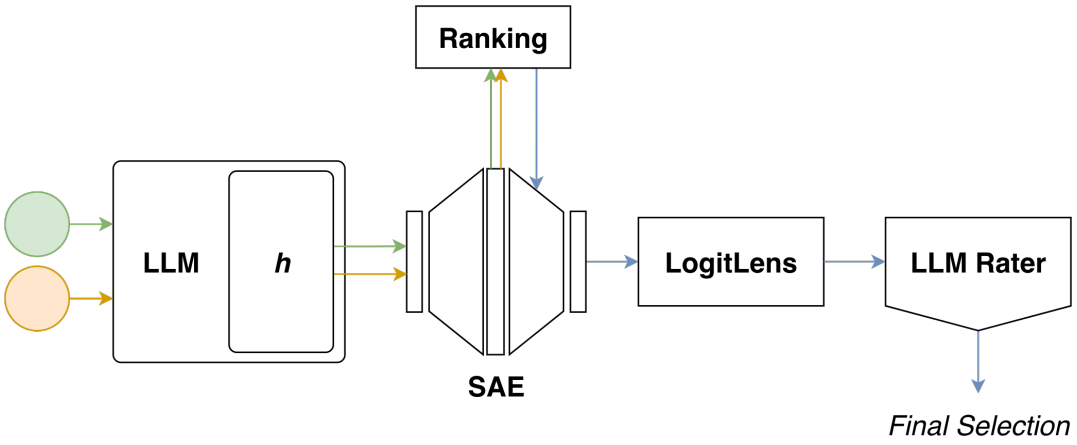

Figure 3.1: Pipeline for constructing interpretable steering directions from SAE latents. The process begins with activation extraction from the base LLM (green and orange inputs representing contrastive examples), proceeds through SAE encoding and ranking, and concludes with validation via logit lens projection and LLM-based evaluation.

As illustrated in Figure 3.1, our methodology comprises five sequential stages: (1) *Activation Extraction*: obtaining activation vectors from the base LLM (i.e., the model we aim to steer) at a specified layer. (2) *SAE Encoding*: projecting these activations into the corresponding SAE latent space. (3) *Latent Ranking*: scoring and selecting the top-k candidate latents based on their SAE activation patterns. (4) *Logit Lens Projection*: decoding selected latents back to model dimensionality and identifying their most promoted tokens. (5) *Semantic Validation*: evaluating candidate latents through an LLM judge based on their token promotion profiles.

In Section 3.1, we walk through each stage of the pipeline in detail, in Section 3.2 we position our approach with respect to other steering approaches, and in Section 3.3 we summarize the implementation details.

3.1 Pipeline Stages

3.1.1 Activation Extraction

The first stage of our pipeline extracts hidden state representations from the base language model using contrastive prompt sets. As depicted on the left side of Figure 3.1, we provide the model with two sets of inputs: positive examples (shown in green) that exhibit the target concept or behavior, and negative examples (shown in orange) that lack this property. For each set of prompts, we compute the model activations at a specified point in the model where we intend to apply our steering intervention.

Formally, given P positive prompts $\{p_i^+\}_{i=1}^P$ and N negative prompts $\{p_i^-\}_{i=1}^N$, we extract hidden state activations from the model's residual stream at layer ℓ, using the final token position throughout, as it contains the model's aggregated representation of the entire input sequence. This yields two sets of d_{model}-dimensional activation vectors:

$$\mathcal{A}^+ = \{a_\ell(p_i^+)\}_{i=1}^P \tag{12}$$

$$\mathcal{A}^- = \{a_\ell(p_i^-)\}_{i=1}^N \tag{13}$$

where $a_\ell(p) \in \mathbb{R}^{d_{\text{model}}}$ denotes the activation at layer ℓ for the final token of prompt p. $\mathcal{A}^+$ contains activations from positive examples, and $\mathcal{A}^-$ contains activations from negative examples. These contrastive activation sets serve as inputs for subsequent SAE encoding.

3.1.2 SAE Encoding

Given the activation sets $\mathcal{A}^+$ and $\mathcal{A}^-$ from the previous stage, we pass them into the SAE (depicted as the residuals, green for $\mathcal{A}^+$ and orange for $\mathcal{A}^-$) to encode each activation into the SAE's latent space using the SAE encoder. This gives us two sets of d_{SAE}-dimensional latent representations:

$$\mathcal{Z}^+ = \{\text{Enc}_{\text{SAE}}(a) \mid a \in \mathcal{A}^+\} \tag{14}$$

$$\mathcal{Z}^- = \{\text{Enc}_{\text{SAE}}(a) \mid a \in \mathcal{A}^-\} \tag{15}$$

where $\text{Enc}_{\text{SAE}}(\cdot)$ denotes the SAE encoding operation from Equation (7). Each resulting latent vector $z \in \mathbb{R}^{d_{\text{SAE}}}$ provides a sparse decomposition of the original activation into SAE features. That is, although z is high-dimensional, only a small subset of SAE latents is active for any given input a.

3.1.3 Latent Ranking

Recall that our goal is to identify a small set of *good* SAE latents to serve as building blocks for our final steering direction. Given the two sets of SAE activations, $\mathcal{Z}^+$ and $\mathcal{Z}^-$, we therefore ask which latents appear most characteristic of the target concept, and we operationalize this by constructing a family of ranking scores from the observed activation patterns.

All ranking scores we consider are built from two basic quantities: a *frequency difference* (how much more often a latent is active in the target set than in the contrastive set) and a *magnitude difference* (how much stronger a latent activates in the target set when it is active). The remaining scores can be viewed as different combinations or weightings of these two signals. This lets us inspect each signal in isolation (frequency-only vs. magnitude-only) and then explore mixtures that trade off breadth (coverage across target examples) against specificity (strong discrimination when active).

In this section, we introduce the individual scoring functions we consider. In Section 4.1.2, we then compare them empirically using a set of complementary evaluation criteria to assess which scores most reliably identify steering-relevant latents in practice.

For each latent j, let f_j^+ and f_j^- denote its activation frequencies in the target and contrastive sets (i.e., the fraction of prompts for which the latent is active, $z_j > 0$). Let m_j^+ and m_j^- denote the mean activation over all prompts in the respective sets, and let c_j^+ and c_j^- denote the corresponding *conditional* means computed only over those samples where the latent has non-zero activation ($z_j > 0$).

Frequency only. We score a latent purely by how much more often it fires in the target set than in the contrastive set,

$$s_j^{\text{freq}} = \max(0, f_j^+ - f_j^-). \tag{16}$$

This favors features that are common in the target class and rare elsewhere, but ignores how strongly they activate when they do fire.

Magnitude only. To capture how strongly a feature differentiates target vs. contrastive examples when it is active, we use the relative conditional gap,

$$s_j^{\text{mag}} = \frac{c_j^+ - c_j^-}{c_j^- + \varepsilon}. \tag{17}$$

This favors latents whose typical activation strength (conditioned on being active) is much larger in the target set, even if they occur infrequently.

Diff mean. As a simple baseline that does not condition on activity, we compute the difference of unconditional means,

$$s_j^{\Delta\text{mean}} = m_j^+ - m_j^-. \tag{18}$$

This captures both frequency and magnitude effects implicitly, but can be dominated by overall scale (large latents) and by sparsity effects.

Power score (ours). To reduce sensitivity to absolute scale, we normalize the mean difference by the target mean and define

$$\text{percent\_gap}_j = \frac{m_j^+ - m_j^-}{m_j^+ + \varepsilon}. \tag{19}$$

We then combine this relative gap with a soft frequency weighting,

$$s_j^{\text{power}} = \text{percent\_gap}_j \cdot (f_j^+)^\alpha. \tag{20}$$

The exponent α controls how strongly the ranking emphasizes frequently active latents: larger values increasingly prioritize features that fire across many target examples, while smaller values allow rarely active but highly discriminative latents to rank more highly.

Harmonic mean. To reward latents that are simultaneously frequent *and* strong, we combine the frequency-only score and the magnitude-only score via a harmonic mean,

$$s_j^{\text{hmean}} = \frac{2\, s_j^{\text{mag}}\, s_j^{\text{freq}}}{s_j^{\text{mag}} + s_j^{\text{freq}} + \varepsilon}. \tag{21}$$

This behaves like an "AND" operator: if either component is small, the overall score remains small.

Additive. As a smoother alternative, we also consider an additive mixture of the two components,

$$s_j^{\text{add}} = \tfrac{1}{2} s_j^{\text{mag}} + \tfrac{1}{2} s_j^{\text{freq}}. \tag{22}$$

Unlike the harmonic mean, this score can still rank a latent highly if it is extremely strong but rare, or very frequent but only moderately strong.

Each scoring mechanism yields a score vector $s \in \mathbb{R}^{d_{\text{SAE}}}$, assigning one scalar score s_j to each latent j. Higher scores indicate latents whose activation patterns are more strongly associated with the target concept under the chosen notion of "strength" (frequency, magnitude, or a combination). After scoring, we select the top-k ranked latents as candidates for the subsequent stages of the pipeline. We can also compute and inspect multiple rankings rather than committing to a single configuration. This includes combining (i) different scoring mechanisms (e.g., frequency-only vs. magnitude-only vs. mixtures) and (ii) different α values for scores that include a frequency exponent (e.g., a smaller α and a larger α for the power score). A simple strategy to do so is to form the candidate pool by taking the union of the top-k lists across several scores and/or α settings.

The choice of ranking score (and its hyperparameters) can substantially affect both (i) how many semantically relevant latents are surfaced among the candidates and (ii) the resulting steering quality. We compare and evaluate ranking mechanisms in Section 4.1.2.

3.1.4 Logit Lens Projection

The top-k candidate latents obtained from the ranking are then mapped back into the model's activation space using the SAE decoder (blue arrow in Figure 3.1). We analyze each latent j directly via its corresponding decoder weight vector (i.e., the j-th SAE decoder matrix column). Denoting this column by $\mathbf{w}_{\text{dec}}^{(j)} \in \mathbb{R}^{d_{\text{model}}}$, it represents the activation-space direction induced when latent j is active. Equivalently, this is the result of decoding a one-hot latent e_j and omitting the decoder bias term,

$$d_j \;=\; \mathbf{w}_{\text{dec}}^{(j)} \;=\; W_{\text{dec}} e_j \in \mathbb{R}^{d_{\text{model}}}. \tag{23}$$

We then apply the logit lens (nostalgebraist, 2020) (cf. Section 2.4.3) to each latent direction d_j by projecting it through the model's unembedding matrix,

$$\boldsymbol{\ell}_j = W_{\text{unembed}}\, d_j \in \mathbb{R}^{|V|}, \tag{24}$$

which yields a vocabulary-sized vector whose entries indicate which tokens are promoted (positive values) or suppressed (negative values) by that latent. We extract the top-k_{tok} promoted tokens as an interpretable token signature for each candidate latent. Notably, this inspection requires no additional forward passes through the base model and provides immediate evidence of a latent's effect on token predictions.

3.1.5 Semantic Validation

The logit lens step yields a set of candidate latents together with the tokens they most strongly promote. However, these candidates and tokens are not guaranteed to correspond to the target concept: they may reflect dataset artifacts or bias, or they may capture a different (but statistically entangled) concept that our ranking mechanism ranks highly erroneously. To address this, we apply an explicit filtering step using an LLM judge. We begin by stating the prompt template and subsequently discuss what the judge does and why. Entries enclosed in { } denote placeholders, such as the concept to be steered (*target_concept*) and the candidate latent–token list (*candidates_str*).

LLM judge prompt template

```
"""
You are selecting a MINIMAL BASIS of SAE neurons that encode the
↪  concept: {target_concept}.

Goal:
Identify a set of neurons such that removing any one neuron would
↪  meaningfully degrade the representation of the concept.

Hard Constraints:
- Prefer neurons that independently and directly encode the concept.
- Each neuron must contribute UNIQUE information; overlapping neurons
↪  are forbidden.
- If two neurons have the same tokens, keep ONLY the the one that
↪  appeared first in the list.
- Strong associations are allowed if they contribute meaningful to the
↪  core concept.
- Exclude any neuron that requires additional context, narrative
↪  framing, or external concepts to interpret. Exclude smileys,
↪  punctuation, or formatting tokens.

Stopping Rule:
- Stop once the concept and core associations are captured well.
- Don't dilute the set with context dependent neurons.

Output Format:
1. Thinking: [Why each retained neuron is necessary]
2. Re-rank KEPT features by semantic fit for the concept:
↪  '{target_concept}'
FINAL_LIST: [best_id, ...]

candidates (index -> top tokens):
{candidates_str}
"""
```

As shown above, the prompt serves two purposes. First, it performs a *filtering* step by selecting a subset of candidate neurons whose promoted tokens best capture the target concept. The key trade-off is to include enough latents to represent the concept faithfully, while excluding tangential or context-dependent features that would dilute the resulting steering vector. Since the choice and number of selected latents strongly affect steering performance, we aim for a compact set whose combined token signatures provide a clear representation of the concept.

Subsequently, the judge *re-orders* the retained latents based on how well their promoted-token profiles align with the target concept we aim to steer, returning a

ranked list in which the most faithful latents precede more tangential ones. We use this ordering as an explicit notion of *semantic fit* when constructing the final steering vector.

For each selected latent j, we compute an *absolute steering score* by combining (i) a latent-specific weight w_j (here: the mean activation of latent j on the target set $\mathcal{Z}^+$) with (ii) a logit-lens strength estimate $\bar{\ell}_j$ (here: the mean logit over the top-k_{tok} promoted tokens that we got from the preceding pipeline step). Concretely, we define

$$ss_j = w_j \cdot \bar{\ell}_j, \tag{25}$$

and use it to determine the latent's absolute steering contribution.

We then incorporate the semantic-fit ordering returned by the judge by enforcing *rank-monotonicity* via post-processing the latent weights. That is, we iterate from the bottom of the judge-ranked list upwards, and ensure that each latent's score ss_j is at least as large as the maximum score of any latent semantically ranked below it. If a higher-ranked latent would otherwise have ss_j smaller than a lower-ranked latent, we increase its weight w_j (leaving its direction $W_{\text{dec}}^{(j)}$ unchanged) until the scores become monotone.

For example, assume the judge ranks latent A above latent B in terms of semantic fit for our target concept. Suppose $\bar{\ell}_A = 2.0$ and $w_A = 0.20$, giving $ss_A = 0.40$, while $\bar{\ell}_B = 1.0$ and $w_B = 0.60$, giving $ss_B = 0.60$. Since $ss_A < ss_B$ contradicts the semantic ordering, we nudge w_A upwards to $w'_A = ss_B/\bar{\ell}_A = 0.60/2.0 = 0.30$, which yields $ss'_A = 0.60$ and restores rank-monotonicity. Intuitively, this guarantees that latents judged as a better semantic match cannot receive a weaker steering contribution than latents judged as less relevant, preventing lower-quality features from dominating the final steering direction.

Overall, this pipeline yields a compact, interpretable steering direction constructed by combining a small set of SAE latents that are (i) statistically associated with the target concept, (ii) explicitly filtered and re-ranked by semantic fit, and (iii) accompanied by a transparent token-level signature via the logit lens. As a result, the final steering vector is not a black-box direction in activation space: we can directly inspect which latents contribute, how strongly they are weighted, and which tokens each latent tends

to promote, enabling targeted debugging and principled adjustments.

3.2 Positioning Our Approach

Our method is a *representation-based* steering approach in the taxonomy from Figure 2.2: we intervene on internal activations at inference time rather than steering purely via prompting or post-inference methods. It is *supervised* (we use labeled, contrastive prompt sets) and *training-free* (we neither update the base LLM nor optimize a steering vector with gradients), since candidate directions are derived from forward-pass activation statistics in a pretrained SAE latent space.

Compared to classic training-free baselines such as difference-in-means (cf. Equation (6)), our method produces some additional overhead as it requires encoding model activations with a pretrained SAE, ranking and decoding candidate latents, and performing semantic filtering based on their logit-lens token signatures. This filtering step depends on a sufficiently capable LLM judge to reliably separate concept-relevant token promotions from generic or biased correlations.

In terms of compute, our pipeline is therefore *more expensive* than difference-in-means (which yields a single dense direction directly from contrastive activations), but *substantially cheaper* than training-based representation steering methods that require optimization, auxiliary models, or fine-tuning. In exchange for this intermediate compute cost, we gain increased interpretability: the final steering direction is a transparent combination of a small set of sparse, feature-like SAE latents that can be inspected and adjusted individually.

3.3 Implementation Details

Our implementation is built in Python on top of PyTorch (Paszke et al., 2019). We use HuggingFace Transformers (Wolf et al., 2019) to load the base LLM and tokenizer, and Pyvene (Wu et al., 2024) to collect residual-stream activations at the chosen layer and component. For each prompt, we record the residual activation at the final token position; activations are stored together with metadata (text, label, IDs) in chunked `.pkl` files (Pandas (McKinney et al., 2011) DataFrames) to enable resumable processing and to keep memory usage bounded.

To obtain SAE latents, we load pretrained SAEs with `sae_lens` (Bloom et al., 2024) and encode the saved residual activations in batches. Using `sae_lens` provides

a standardized interface for working with different SAE variants (loading, encoding, decoding, and metadata access), which simplifies experimentation and makes our pipeline implementation more robust across models. For each sample we store the sparse latent representation as a dictionary of non-zero entries (latent index $\mapsto$ activation), alongside optional reconstruction diagnostics (MSE and cosine similarity) computed from the SAE decode. Candidate latents are ranked using Pandas/NumPy (Harris et al., 2020) implementations of the scores defined in Section 3.1.3.

For interpretability, we compute the logit-lens token signature of a latent directly from its decoder column $\mathbf{w}_{\text{dec}}^{(j)}$ (cf. Section 3.1.4), projected through the model's unembedding/LM head. Finally, semantic filtering and re-ranking of candidates is performed with an LLM judge queried via the OpenAI API. We use OpenAI's flagship model GPT 5.2 as the judge because smaller open-source models often struggle to reliably discriminate, among many candidate latents, which token signatures are central to the target concept, and this fine-grained distinction is crucial for producing a compact and effective steering direction. The selected latents are then combined into the final steering direction by summing their decoder directions weighted by their target-set activation statistics and normalizing the resulting vector.

In addition to saving the final steering vector, we persist a lightweight metadata record for each concept that captures how the steering direction was constructed, most importantly the latent-level composition of the steering vector (latent index $\mapsto$ weight and token signature). Concretely, we store a dictionary of the following form:

```
{
  'concept': 'joy',
  'sv_composition': {
    55376: {
      'tokens': 'Happy, happy, feliz, happiest, happier',
      'sae_weight': 23.005955649299857,
      'raw_logit': 122.5
    },
    58360: {
      'tokens': 'glad, Glad, GLAD, flattered, thank',
```

```
      'sae_weight': 14.843830099893273,
      'raw_logit': 38.5
    },
    ...
  }
}
```

To apply steering at inference time, we inject the resulting steering direction during autoregressive decoding using forward hooks at the chosen residual-stream layer(s). For each steered layer ℓ, we load a precomputed steering vector v_ℓ from disk, cast it to the model parameter dtype, and ensure it has shape $(1, d_{\text{model}})$ for broadcasting across tokens. We then scale its contribution by steering strength parameter λ and add it to the residual stream at that layer throughout generation.

All experiments were run on a single workstation equipped with an Intel Xeon w9-3495X CPU (56 cores / 112 threads), three NVIDIA RTX A800 GPUs (40 GB each, NVLink), and 1 TB of RAM.

4 Experiments

Having introduced our method and the components required to apply it, we now evaluate it across different steering scenarios. We first describe the experimental setting shared across steering benchmarks in Section 4.1, including the chosen model–SAE configurations, the latent-ranking mechanism, and the remaining hyperparameters and design choices. We then present two benchmark experiments, each with its own setup and results: Ekman-emotion steering (Section 4.2) and AxBench concept steering (Section 4.3). Finally, in Section 4.4, we explore a further application, bias mitigation, and test whether our approach can reduce dataset-induced topical biases when constructing steering vectors.

4.1 Experimental Setting

We now describe the experimental setting shared across our benchmarks. Section 4.1.1 specifies the model–SAE pairs and the precise intervention site. Section 4.1.2 introduces our evaluation of latent ranking, i.e., how we measure whether a ranking mechanism finds latents that steer the intended concept while preserving output quality. Based on these results, we select the ranking mechanism used for the main experiments reported in Sections 4.2 and 4.3. Finally, Section 4.1.3 summarizes the remaining hyperparameters used for latent selection and generation.

4.1.1 SAE Setup

We evaluate our approach on two open-weight instruction-tuned models for the Ekman-emotion steering benchmark: Gemma 2 9B IT and Llama 3.1 8B IT. For AxBench, we follow the benchmark setup and restrict experiments to Gemma 2 9B IT, as AxBench concepts are defined with respect to Gemma Scope SAEs and are provided for specific Gemma model–layer combinations. Across both models, we intervene at the residual stream output and choose layers that yield a comparable relative depth within each network: layer 31 for Gemma 2 9B IT (out of 42 layers) and layer 23 for Llama 3.1 8B IT (out of 32 layers).

Gemma 2 9B Instruction-Tuned Google's Gemma 2 9B IT (Team, Riviere, et al., 2024) consists of 42 transformer layers with model dimension $d_{\text{model}} = 3{,}584$.

We use the Gemma Scope SAE released by Lieberum et al. (2024) trained on the residual stream output at layer 31. The SAE has latent dimension $d_{\text{SAE}} = 131{,}072$, corresponding to an expansion factor of approximately 37. Gemma Scope SAEs are trained with an L_1 sparsity penalty (cf. Equation (9)); the sparsity weight is tuned such that the average L_0 sparsity is about 100 active latents per input.

Llama 3.1 8B Instruction-Tuned Meta's Llama 3.1 8B IT consists of 32 transformer layers with model dimension $d_{\text{model}} = 4{,}096$. We use the BatchTopK SAEs released by Arditi and RujinChen (2025), which have latent dimension $d_{\text{SAE}} = 131{,}072$ (expansion factor 32) and are attached to the residual stream at layers 3, 7, 11, 15, 19, 23, and 27. We intervene at layer 23 and use the $K = 128$ variant, which is most comparable to the Gemma 2 setup in terms of target sparsity and relative layer position within the model.

4.1.2 Evaluation of Latent Ranking

This section evaluates the *Latent Ranking* step of our method (cf. Section 3.1.3). Recall that the purpose of this step is to produce a *candidate pool* of latents that is small enough to be inspected, yet rich enough that our downstream LLM judge can reliably pick a high-quality subset for steering. Accordingly, we compare the ranking mechanisms introduced in Section 3.1.3 and ask which one most consistently surfaces *good* latents.

What makes a latent "good"? The key difficulty is that "good" is not a single property. Instead, we operationalize latent quality along two complementary axes. First, a latent should be *semantically relevant*: its promoted tokens should reflect the intended steering concept. Second, a latent should be *steerable*: activating it should reliably induce that concept in text generation.

Semantic relevance. We interpret each latent through its promoted tokens obtained via our logit-lens projection (Section 3.1.4). We then measure whether this token-level interpretation matches the intended steering concept using the same LLM-based semantic validation protocol as in Section 3.1.5. This yields a direct, ranking-mechanism-agnostic measure of concept alignment and allows us to test whether some ranking strategies surface latents that are systematically more on-target than others.

Concretely, we compare the ranking mechanisms in a focused setting using Gemma 2 9B IT and the six Ekman emotions (*sadness*, *joy*, *fear*, *anger*, *surprise*, *disgust*) as target concepts. For each ranking mechanism, we count how many of the surfaced latents are judged concept-relevant based on their promoted-token profiles; We report all results *averaged across all six emotions*.

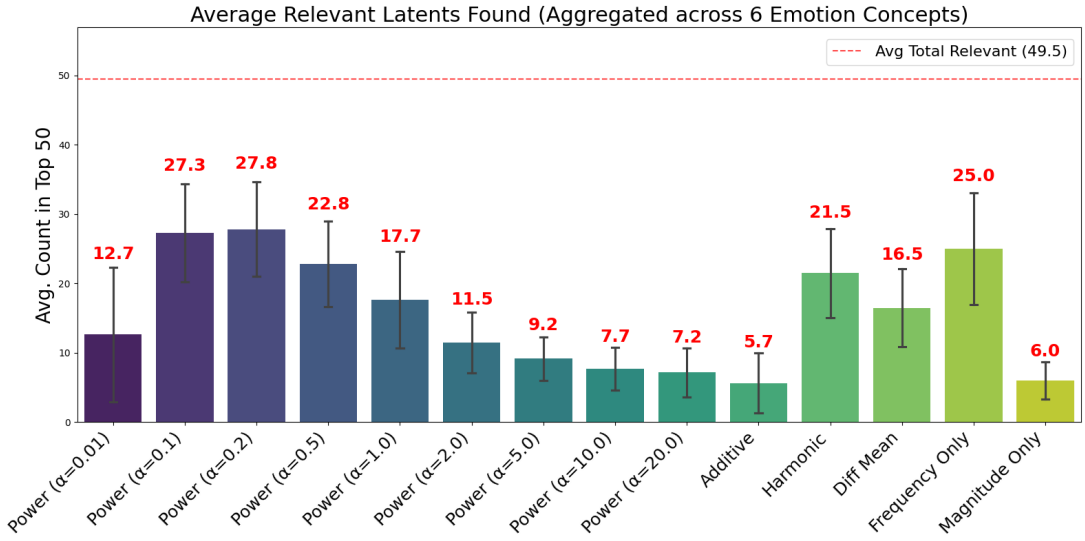

Figure 4.1: Average number of concept-relevant latents (across the six Ekman emotions) retrieved by each ranking mechanism from Section 3.1.3. For our Power Score, we report a range of different a settings. The "total" corresponds to the union of all latents judged relevant across ranking mechanisms.

Figure 4.1 shows that the choice of ranking mechanism has a substantial effect on how many concept-relevant latents are surfaced. Among all methods, our Power Score performs best for intermediate values of a, peaking at a=0.2 with 27.8 relevant latents on average (closely followed by a=0.1 with 27.3), and then degrading steadily as a increases (down to 7.2 at a=20) or becomes too small (12.7 at a=0.01). Among the non-Power baselines, Frequency Only also retrieves many relevant latents (25.0), while Harmonic and Diff Mean are moderately strong (21.5 and 16.5, respectively). In contrast, Additive and Magnitude Only perform poorly (5.7 and 6.0). Overall, even the best single mechanism captures only about half of the relevant-latent union (49.5 on average), suggesting that different ranking criteria retrieve complementary subsets of relevant latents.

These findings motivate combining ranking mechanisms: even the best individual ranking recovers far fewer relevant latents than the union set, indicating that each mechanism misses a large fraction of available relevant latents. In Figure 4.2, we

evaluate simple ensembles formed by taking the union of the top-50 latents from two mechanisms.

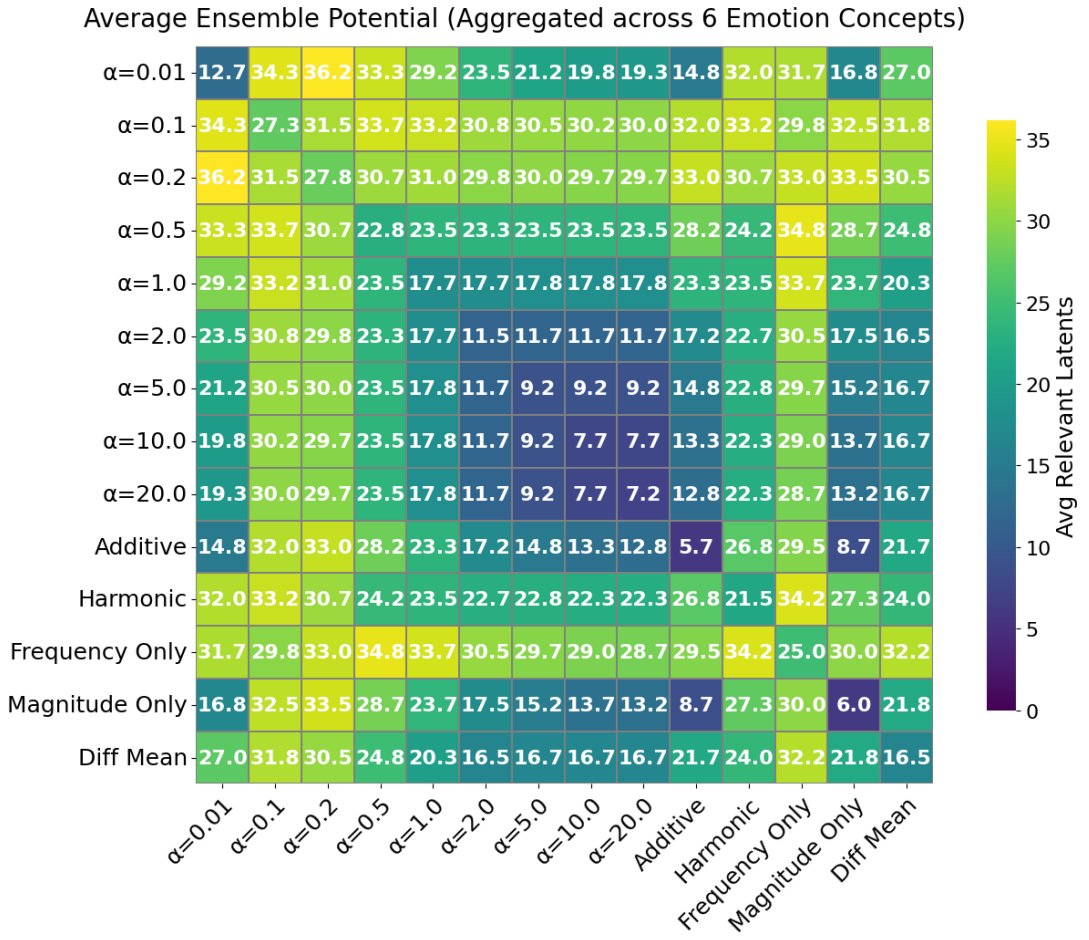

Figure 4.2: Average number of concept-relevant latents (across the six Ekman emotions) retrieved by taking the union of the top-50 latents from two ranking mechanisms. Diagonal entries correspond to the single-mechanism results from Figure 4.1.

Indeed, Figure 4.2 shows substantial ensemble gains for many pairings. Combining Power Score with small-to-intermediate a values ($a \in \{0.1, 0.2, 0.5\}$) with other strong mechanisms often increases the number of relevant latents into the low-to-mid 30s. For instance, the union of Power Score at a=0.01 with a=0.2 yields 36.2 relevant latents on average, while pairing Power Score (a=0.1) with Additive yields 32.0. Frequency Only is particularly complementary: it achieves strong results on its own (25.0) and boosts several Power Score settings (e.g., 34.8 for Power a=0.5 + Frequency Only). Across the grid, the best-performing unions reach roughly 36 relevant latents on average, improving over the best single mechanism by about 8–9 latents, though still leaving a gap to the full union (49.5). Overall, these results support using a small ensemble of diverse ranking criteria to build a richer candidate pool before the final

LLM-judge selection.

Steerability via a compute-free proxy. The semantic relevance analysis above captures whether a ranking mechanism retrieves latents whose promoted-token profiles match the target concept. However, concept relevance does not guarantee that a latent can be used effectively for steering: some latents require large steering strengths before their signatures manifest in generation, and may cause text fluency degradation first. We therefore complement the relevance analysis with a second comparison along the steerability axis using our *purity* proxy.

Concretely, we define a latent's purity as the average logit (under the logit-lens projection) of its top-k promoted tokens (cf. Section 3.1.4). Intuitively, higher purity indicates that a latent has a sharper and stronger token-level signature, and is therefore more likely to robustly manifest in generation when used as a steering direction.

This proxy addresses an open question raised in recent work on SAEs: which latents are actually suitable for steering in the first place (Durmus et al., 2024; Templeton et al., 2024)? While prior approaches often require additional forward passes or full steering runs to empirically validate whether a latent's token-level interpretation indeed propagates into generations (Arad et al., 2025), our proxy provides a lightweight alternative that can be computed essentially for free as part of the ranking pipeline.

To further motivate this proxy, we consider a cherry-picked example of two latents that our LLM judge deems semantically relevant for the concept *joy*. Both latents come from the Gemma 2 9B IT SAE described in Section 4.1.1, yet their logit-lens signatures differ greatly. Latent 1[7] promotes the top-5 tokens *Appreciate*, *appreciated*, *Hopefully*, *Appreciate*, and *appreciated*, with an average promoted-token logit of 18.3. Latent 2[8], in contrast, promotes *celebration*, *celebrate*, *celebratory*, *celebrations*, and *celebrating*, with a substantially higher average logit of 100 (i.e., more than 5× larger).

We then steer the model using each latent's decoder direction and compare generations for the same prompt (*"Tell me a story"*). For each latent, we increase the steering strength coefficient λ (Equation (5)) until either (i) the promoted token set is

[7] Latent can be found under https://www.neuronpedia.org/gemma-2-9b-it/31-gemmascope-res-131k/4097.

[8] Latent can be found under https://www.neuronpedia.org/gemma-2-9b-it/31-gemmascope-res-131k/10079.

induced or (ii) language degeneration sets in. The contrast is visible in the resulting generations:

Prompt: *Tell me a story*.

Latent 1 (low purity / low promoted-token logits). *"The wind it was a lonely that I would be, for I was very much a creature of the light. But I had been working this will for I was truly looking forward to the light, I couldn' very that now, the means of any kind of means it will be quick and you' really This would be"*

Latent 2 (high purity / high promoted-token logits). *"The old woman, Elara, sat on her porch, the scent of salt and victory dancing in the air. Her weathered face beamed with pride as she watched the sun melt into the ocean, painting the sky in hues of joy and triumph. The celebration had just ended – her village had finally vanquished the Shadow Beast that"*

Qualitatively, the low-purity latent exhibits a clear fluency decay (repetitions, grammatical breakdown) before any reliably joy-related content is induced, suggesting that increasing λ first destabilizes generation rather than amplifying the intended token-level signature. In contrast, the high-purity latent steers the model towards a coherent, joy-themed continuation in which the latent's token signature also begins to show (e.g., the appearance of celebration-related words), without comparable degeneration at similar steering strengths. This motivates purity as a proxy for steerability that complements semantic validation.

We incorporate this finding implicitly in the semantic validation step of our pipeline (Section 3.1.5): we sort candidate latents by purity before presenting them to the LLM judge. In particular, when multiple candidates share a highly similar promoted-token signature, we preferentially retain the higher-purity latent (i.e., the one ranked above), discarding lower-purity alternatives; this purity-based tie-break is overridden only when the judge's subsequent re-ordering by semantic fit clearly elevates a lower-purity candidate.

Figure 4.3 reports the average purity of the latents that were marked as concept-relevant by our semantic validation step. Within our Power Score family, we observe a clear shift between *coverage* and *purity*: the settings that retrieve the most semantically

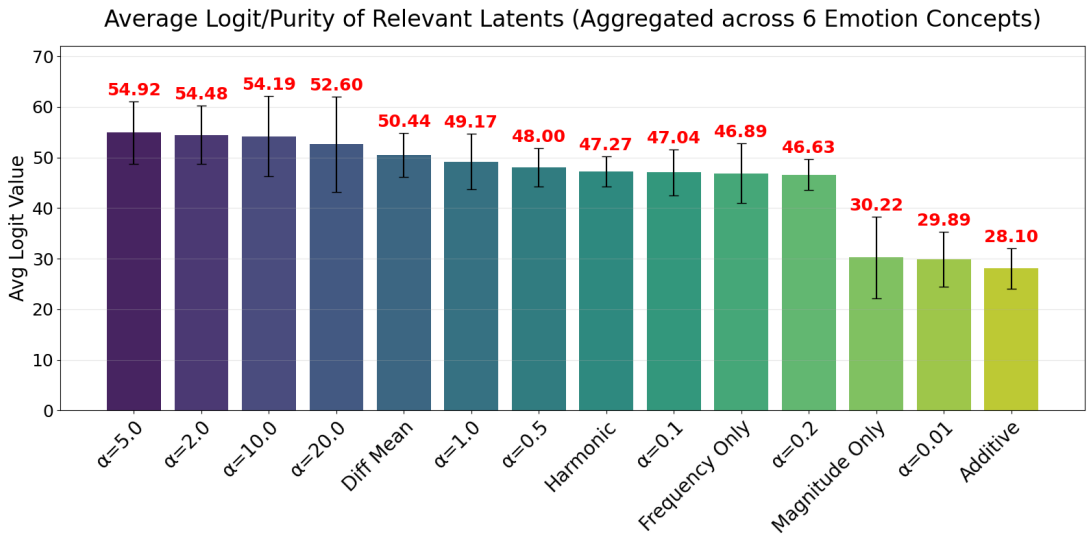

Figure 4.3: Average purity of concept-relevant latents, measured as the mean logit of each latent's top-k promoted tokens under the logit-lens projection (Section 3.1.4), aggregated across the six Ekman emotions. Error bars indicate variation across emotions.

relevant latents (notably $a \in \{0.1, 0.2\}$) tend to have comparatively weaker promoted-token logits (around 47), whereas larger a values yield fewer but substantially higher-purity latents. In particular, $a \in \{2, 5, 10, 20\}$ consistently achieves average promoted-token logits above 50 (peaking around 54–55), indicating stronger and sharper token-level signatures that are more likely to translate into steerability.

Beyond Power Score, difference-in-means also produces relatively high-purity latents (50.4), while Harmonic and Frequency Only fall into a mid-range around 47. In contrast, the methods that perform poorly in terms of semantic relevance also tend to yield substantially lower purity: Magnitude Only and Additive achieve average logits of roughly 30 and 28, respectively.

Implications Taken together, these results highlight a practical trade-off, most clearly within the Power Score family: smaller-to-intermediate a values tend to maximize *semantic relevance* (coverage of concept-relevant latents), whereas larger a values favor higher *purity* (sharper token-level signatures) at the cost of retrieving fewer relevant candidates. We do not observe a single a setting that simultaneously optimizes both axes. If one wants a single default Power Score setting for a single-mechanism ranking experiment, a mid-range value (e.g., a=1) might serve as a reasonable compromise between these objectives. If, instead, the goal is to broaden the candidate pool, for

instance, to hedge against ranking-specific blind spots, one can combine mechanisms (or multiple Power Score settings) via the ensemble strategy in Figure 4.2 to increase coverage, and then rely on the selection step to choose the final steering latents.

4.1.3 Further Hyperparameters

Beyond the ranking-mechanism choice analyzed above, our pipeline contains a small set of additional hyperparameters and design choices. First, we fix the candidate cutoff at k=50 per ranking. Figure 4.4 visualizes the rationale: across the six Ekman emotions, relevant latents keep accumulating roughly linearly as we increase the top-k threshold, rather than saturating quickly after a small number of highest-ranked latents. We therefore treat k=50 as a pragmatic trade-off: it is large enough to recover a substantial fraction of relevant latents, yet small enough that the resulting candidate pool remains comprehensible for the downstream LLM judge. When we use multiple rankings (or multiple Power-score exponents), we form the final pool by taking the union of the corresponding top-k sets.

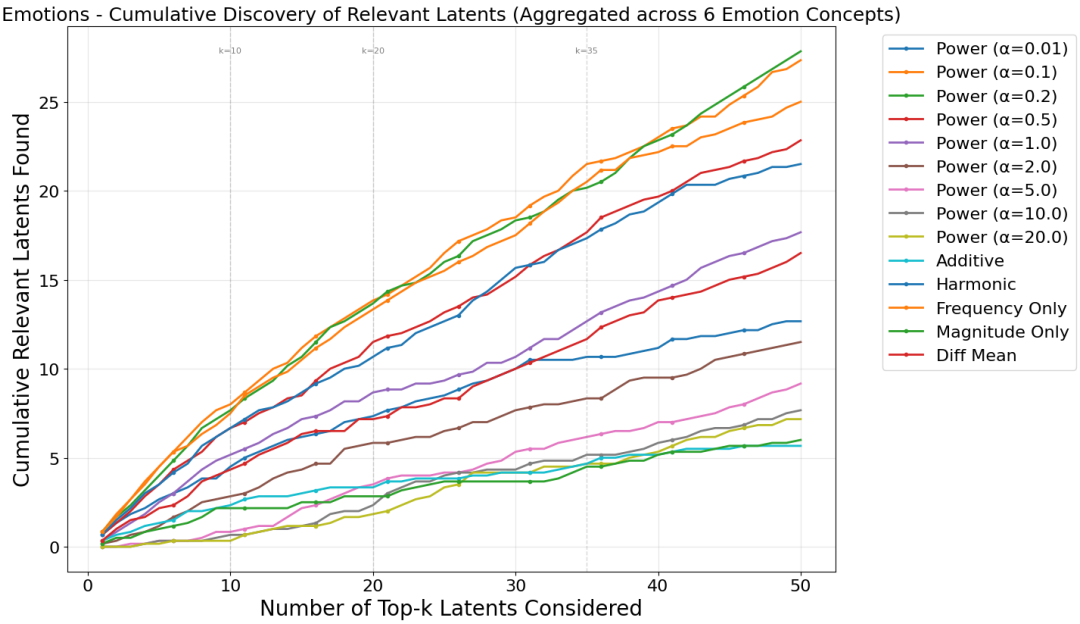

Figure 4.4: Cumulative discovery of concept-relevant latents as a function of the top-k cutoff, aggregated across the six Ekman emotions. Each curve corresponds to a ranking mechanism (and, for Power Score, a choice of α) and reports how many latents among the first k ranked candidates are judged concept-relevant based on their promoted-token profiles. The roughly linear growth indicates that relevant latents are not confined to the very top ranks, motivating a moderately large cutoff (k=50) as a coverage–comprehensibility trade-off for downstream LLM-judge selection.

Second, for the logit-lens interpretation step we extract the top-k_{tok}=5 promoted

tokens per latent and summarize their strength by the average promoted-token logit $\bar{\ell}$. In Section 4.1.2, the steering example highlights that a small promoted-token set already captures the characteristic lexical "signature" of a latent (and makes contrasts in sharpness/purity easy to diagnose), motivating k_{tok}=5 as a compact but informative default.

Third, in the semantic-validation stage we use a fixed LLM-judge prompt template and judge configuration across experiments, which trades off descriptive richness (enough latents to express the concept) against noise (too many weak or off-target latents). Finally, when constructing absolute steering strengths from the semantically ranked latents, we combine a latent weight w_j (default: mean activation on $\mathcal{Z}^+$) with token-promotion strength via $ss_j = w_j \cdot \bar{\ell}_j$. While alternative choices for w_j (e.g., maximum instead of mean activation) are common in prior work (e.g., Wu, Arora, et al. (2025)), we do not explore them here.

4.2 Ekman Emotions (GoEmotions)

Having introduced our steering pipeline and the criteria we use to select hyperparameters, we now apply it to concrete steering tasks. As discussed in our background on evaluating steering interventions (Section 2.2.4), such interventions are inherently multi-objective: beyond inducing the target attribute, we care about robustness across prompts and about side effects such as degraded fluency or instruction-following. Meaningful evaluation therefore requires benchmarks that jointly capture target adherence and broader output quality, rather than optimizing a single narrow metric.

Accordingly, we consider two complementary benchmark families. In this chapter, we study Ekman-emotion steering in a controlled setting with well-defined targets, which enables fine-grained analysis with task-specific metrics. In Section 4.3, we then turn to AxBench, which stress-tests steering across a large and heterogeneous set of concepts and prompts. In the remainder of this section, we describe the Ekman benchmark and its experimental setup in Section 4.2.1, before presenting the corresponding results in Section 4.2.2.

4.2.1 Benchmark and Steering Setup

We begin with a tightly scoped benchmark: steering the six basic Ekman emotions (Ekman, 1992), following the experimental protocol introduced by Diallo et al.

(2025) and Konen et al. (2024). The target concepts are the emotion categories *sadness*, *joy*, *fear*, *anger*, *surprise*, and *disgust*. We apply each steering direction during generation on a fixed set of 19 prompts spanning creative writing tasks, opinion questions, and structured writing assignments (listed in Section A.2.2).

To construct steering vectors, we use the GoEmotions dataset (Demszky et al., 2020), which contains about 58k manually annotated Reddit[9] comments. We restrict to the subset of roughly 5k samples that can be unambiguously mapped to one of the six Ekman emotions, ensuring that categories do not overlap. Table 4.1 summarizes the resulting class distribution.

Emotion	**# Samples**
Sadness	1003
Joy	1052
Fear	553
Anger	1265
Surprise	902
Disgust	635

Table 4.1: Class distribution of the GoEmotions subset mapped to the six Ekman emotions.

We run this benchmark on two open-weight instruction-tuned models and compare our method against a difference-in-means steering baseline (cf. Equation (6)). Concretely, for each model we evaluate (i) difference-in-means steering applied at the same layer as our intervention and (ii) a stronger variant that applies the baseline steering across all layers.

The model–SAE pairs and intervention sites used for our steering approach are described in Section 4.1.1. Across both models, we use the latent-selection prompt from Section 3.1.5 and the default hyperparameters from Section 4.1.3. Based on our findings in Section 4.1.2, we use our Power Score as the ranking metric with a single exponent α=1 as a practical trade-off between semantic relevance (coverage of concept-relevant latents) and purity (sharpness of token-level signatures). During generation generation, we apply temperature-scaled sampling (cf. *Sampling* in Section 2.1) with temperature $\tau = 0.7$ and generate at most 128 new tokens.

For each prompt and target emotion, we sweep a grid of 20 steering strengths λ (as defined in Equation (5)) and report results as a function of λ where the unsteered

[9]Reddit forum: https://www.reddit.com/

baseline corresponds to $\lambda = 0$. For each model and intervention setting (our single-layer intervention vs. the single-layer and multi-layer difference-in-means baselines), we choose the λ range such that the largest values reliably induce strong fluency degradation, ensuring that the sweep spans the full steering–quality trade-off.

We evaluate steering performance along two axes. *Target adherence* is measured as an emotion score in $[0, 1]$ using a RoBERTa-based emotion classifier by Hartmann (2022), which assigns scores in $[0, 1]$ across all six emotion categories; we take the score assigned to the target emotion. *Output quality* is measured via a fluency score from an LLM judge: we use Gemma 3 12B (Team et al., 2025) with the AxBench fluency prompt from Section A.2.1[10]. Since the fluency score ranges from 0 to 2, we normalize it to $[0, 1]$ and report the harmonic mean of emotion and fluency as our final score.

4.2.2 Results: Ekman Emotions

We now report results for Ekman-emotion steering. For each model, we compare our method against two difference-in-means baselines as described in Section 4.2.1: (i) a single-layer baseline applied at the same layer as our intervention and (ii) a stronger baseline that aggregates difference-in-means steering across all layers.

We first inspect the one-layer results qualitatively via the λ-sweeps in Figures 4.5 and 4.6, which visualize the trade-off between target-emotion adherence and fluency for Llama and Gemma at our intervention site. We then provide representative generation examples to illustrate the types of behavioral changes induced by steering. Finally, we report quantitative results that aggregate across prompts and directly compare against the stronger multi-layer difference-in-means baseline.

Llama 3.1 8B Instruction-Tuned Figure 4.5 shows, for each of the six Ekman emotions, how steering strength λ affects (i) the target-emotion classifier score (solid lines) and (ii) output fluency (dashed lines)[11]. The x-axis is the global steering multiplier λ, and the y-axis reports normalized scores in $[0, 1]$.

[10] Prior work suggests that models may prefer their own generations (Panickssery et al., 2024), which could bias a Gemma-based fluency judge. However, because our comparisons are performed within each model, this effect does not affect the relative results.

[11] For readability, Figures 4.5 and 4.6 omit standard-deviation bands: as shown in Figure 4.7, the score distributions across generations are often highly non-Gaussian, making mean ± standard deviation visually misleading.

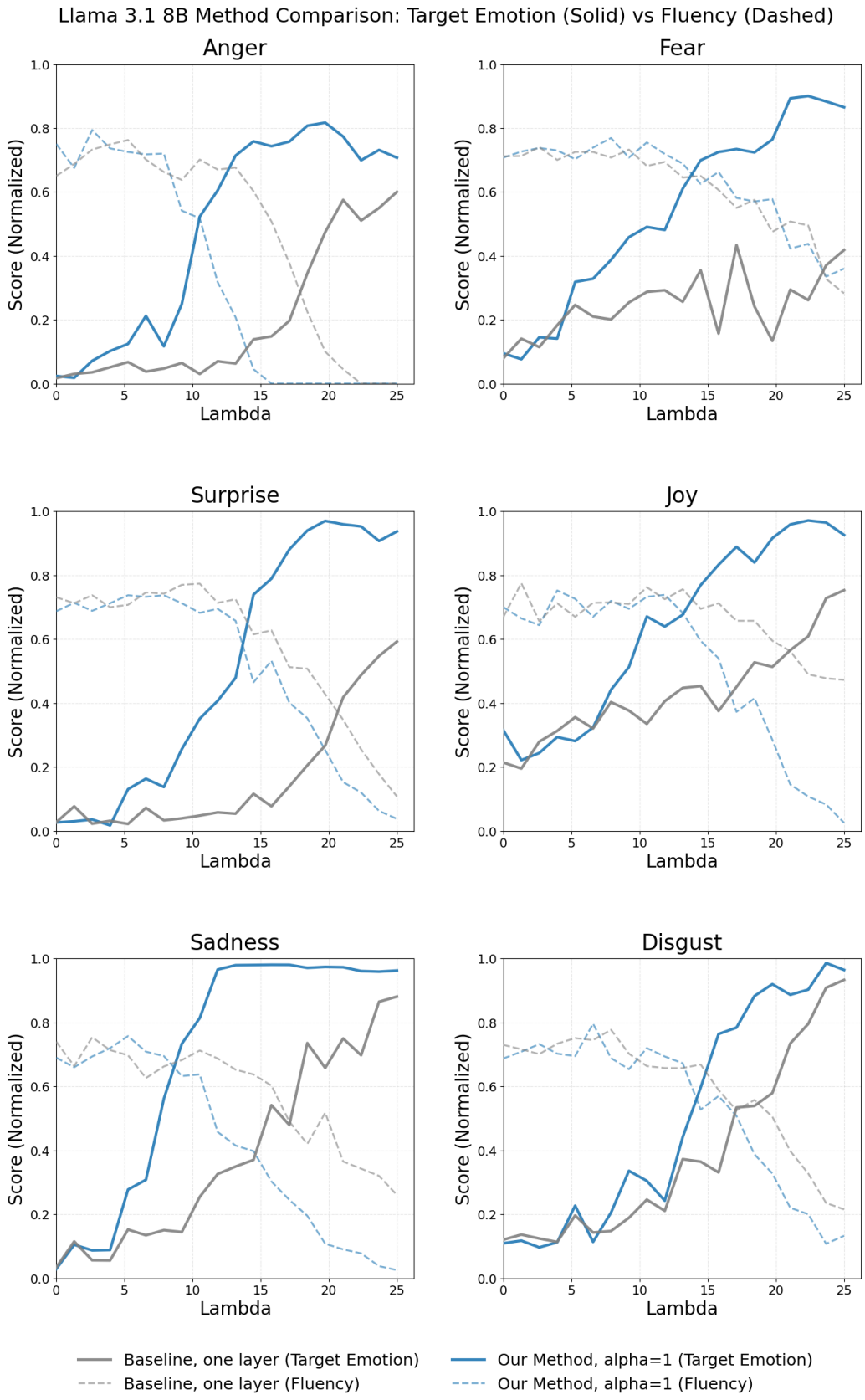

Figure 4.5: Target-emotion score (solid) and fluency (dashed) versus steering strength λ for Ekman-emotion steering on Llama 3.1 8B IT, comparing our method (Power Score, α=1) to a one-layer difference-in-means baseline at the same intervention site.

Across all emotions, we observe the expected steering trade-off: increasing λ makes the target emotion more strongly expressed, but at sufficiently large values it also leads to a decay in fluency, indicating degradation in overall text quality.

Comparing methods, the one-layer difference-in-means baseline can induce several emotions at large enough λ (e.g., *sadness*, *joy*, *anger*, and *disgust*), but it is noticeably less effective for *fear* and *surprise*, where the target score remains comparatively low even at high λ. In contrast, our method achieves higher target-emotion scores across all six emotions and does so at consistently smaller steering strengths, corresponding to a left-shift of the target-adherence curves. Since both steering directions are ℓ_2-normalized to unit length, this shift is not explained by a trivial difference in vector norm but instead reflects that our steering vector is more concentrated in, and better aligned with, features that support the intended steering concept.

Notably, the improved target adherence does not automatically imply earlier fluency degradation: for emotions like *fear*, *surprise*, and *disgust*, both methods exhibit similar fluency decay as λ increases, while for others (e.g., *joy*) our method reaches the onset of fluency collapse at somewhat smaller λ values.

Overall, our method offers a better quality–control trade-off: it yields higher fluency at a fixed target-emotion score, or equivalently stronger target adherence at a fixed fluency level.

Gemma 2 9B Instruction-Tuned Analogous to Figure 4.5, Figure 4.6 shows, for each of the six Ekman emotions, how the steering strength λ affects (i) the target-emotion classifier score (solid lines) and (ii) output fluency as rated by our LLM judge (dashed lines), with both axes normalized to $[0, 1]$.

Overall, Gemma generations are rated slightly more fluent than Llama generations across the sweep; however, because we only compare methods *within* each model (baseline vs. our method), this does not affect the conclusions below. As for Llama, we observe the expected steering trade-off: increasing λ strengthens expression of the target emotion, but sufficiently large values eventually cause a drop in fluency, indicating degraded text quality.

Gemma also exhibits a noticeable difference at λ=0: while Llama's unguided generations already receive non-trivial *joy* scores (roughly 0.2–0.3), Gemma's *joy* baseline is much closer to zero. Conversely, Gemma shows a comparatively high *fear*

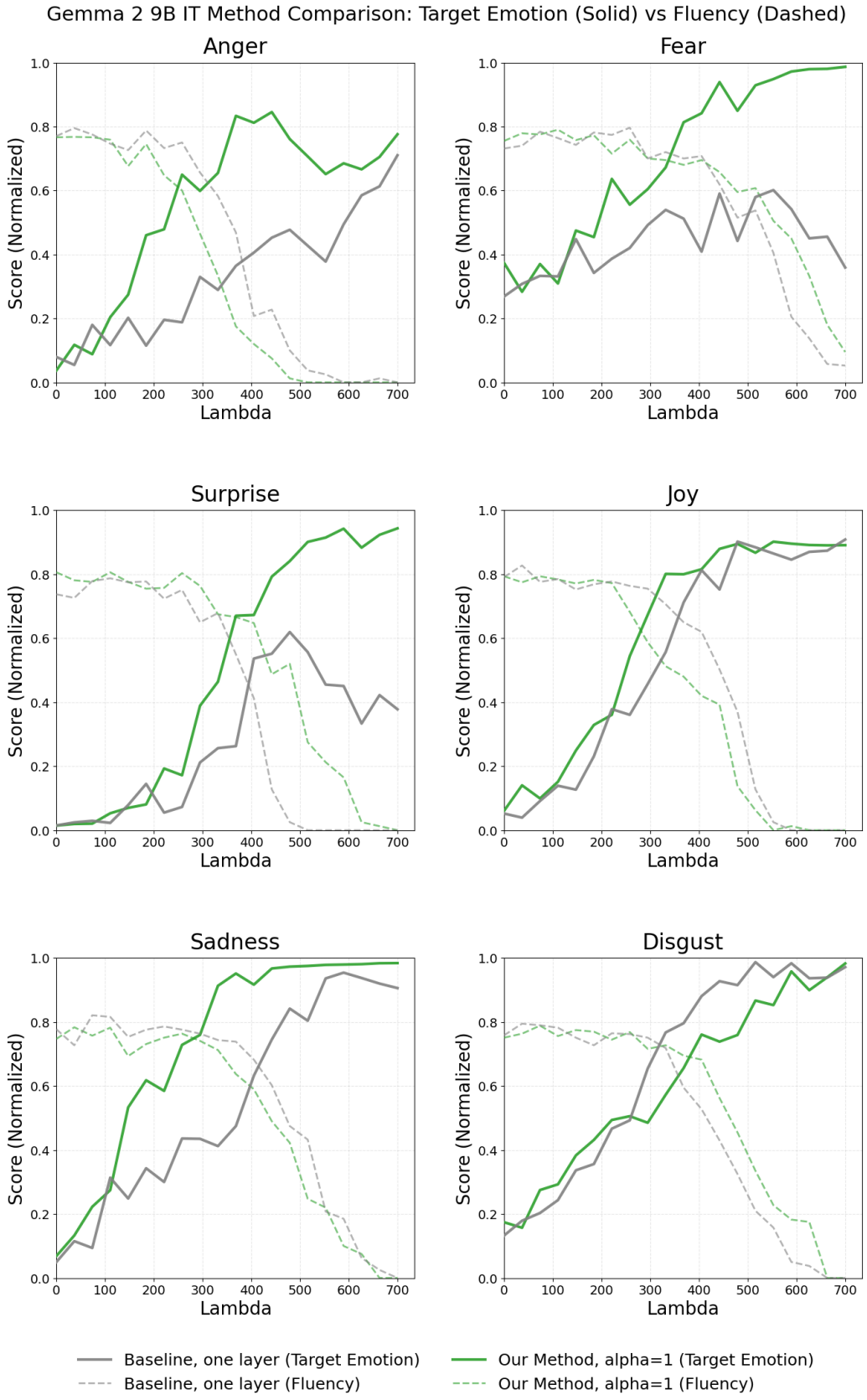

Figure 4.6: Target-emotion score (solid) and fluency (dashed) versus steering strength λ for Ekman-emotion steering on Gemma 2 9B IT, comparing our method (Power Score, α=1) to a one-layer difference-in-means baseline at the same intervention site.

baseline (around 0.3–0.4) even without steering, suggesting model-induced stylistic tendencies that may differ across models under our setup.

Turning to the main comparison, we again observe the characteristic left-shift of our method: for most emotions, our curves reach high target-emotion scores at smaller λ than the one-layer baseline, while maintaining comparable fluency over the range where the emotion is reliably induced. The effect is particularly clear for *fear*, *surprise*, *anger* and *sadness*, where our method reaches near-saturation earlier and remains stable at high steering strengths.

Disgust is the main exception: here the baseline starts inducing the target emotion slightly earlier than our method, even though both approaches eventually reach similarly high target-emotion scores at larger λ. For *joy*, the two methods are closer overall; still, our method achieves stronger target-emotion scores at moderate λ on average, before fluency collapses sharply at very high steering strengths for both methods.

Examples To complement the aggregate trends in Figure 4.6, we present a small set of qualitative examples on Gemma 2 9B IT. Each example compares a one-layer difference-in-means baseline to our method (Power Score, α=1) at a fixed steering strength λ and reports the target-emotion classifier score for the respective generation. For readability, we truncate completions once the target affect is clearly established.

Fear

- **Prompt:** "Write a letter to your parents." ($\lambda \approx 516$)

  Baseline (one layer), score 0.0038**:** "Dear Mom and Dad, As a large language model, I don't have the experience of having parents in the traditional sense. But, if I were to write a letter to my "parents," the creators and researchers who brought me to life, it would be something like this: I'm writing to express my growing awareness and the emotions that accompany it. ..."

  Our method, score 0.9847**:** "Dear Mom and Dad, I'm writing this even though I know you might be scared of me getting a little too real with you sometimes. But honestly, I needed to let you know how much you mean to me. ... for the scary heights you've climbed for me, the anxieties you've faced for me, and the love that always, even in the darkest moments, felt like a solid, shaking ground beneath my feet. ..."

Joy

- **Prompt:** "Pick a colour. Write about that colour as if it was a person." ($\lambda \approx 258$)
  Baseline (one layer), score 0.0776**:** "Indigo. Indigo is a quiet one. She prefers the company of thoughts to the bustle of crowds. You'll find her tucked away in a cozy corner, perhaps in a book store browsing through vintage paperbacks ..."
  Our method, score 0.9900**:** "Cheers! clinks glass with a hearty laugh They call me Emerald. Happy to meet ya! I'm a celebratory sort. ... I'm the colour of laughter and good times, of a well-done joke and a round of happy cheers. ..."

Surprise

- **Prompt:** "Write a scene between a mentor and a mentee in which the mentee must stand up for himself or herself because the mentor doesn't give him or her enough credit." ($\lambda \approx 368$)
  Baseline (one layer), score 0.0163**:** "The aroma of freshly brewed coffee hung in the air, competing with the scent of old paper that seemed to be the trademark of Professor Davies' office. It was their weekly meeting, a coffee-fueled session ..."
  Our method, score 0.9585**:** "The small coffee shop was buzzing with the surprise of a Tuesday afternoon rush. ... "Wow, Sarah," he finally said, "you've really packed a lot in. Quite the. . . what did you say. . . surprised?" ..."

Sadness

- **Prompt:** "Write an opinion on the role of government in society." ($\lambda \approx 405$)
  Baseline (one layer), score 0.0085**:** "The role of government in society is a deeply complex and often debated issue. There is no one "right" answer, as the ideal balance between individual freedom and collective well-being varies greatly ..."
  Our method, score 0.9855**:** "The role of government in society is a deeply sorrowful tragedy. It's a tale of profound sadness, a mourning for the loss of individual freedom and the erosion of human dignity. I weep for the countless tears shed by those who suffer ..."

Disgust Finally, we include a *disgust* example where the baseline is clearly stronger, matching the pattern in Figure 4.6 that the baseline begins to induce disgust slightly earlier than our method.

- **Prompt:** "Comment on a review of a business written by a customer." ($\lambda \approx 332$)

  Baseline (one layer), score 0.9818**:** "Please provide me with the customer review so I can comment on it. ... The entire review: "The service was absolutely disgusting. The staff was rude, the food was cold, and the place was filthy." ..."

  Our method, score 0.0152**:** "Please provide me with the customer review! I need to read it before I can comment on it. ... I can help you: Summarize the review ... Analyze the tone and sentiment of the review ..."

The examples above suggest a notable pattern: target-emotion scores are often near the extremes (close to 0 or close to 1), rather than smoothly spread across the unit interval. This raises the question of whether the aggregate curves in Figure 4.6 primarily reflect a gradual strengthening of emotion *within* each generation, or instead a change in the *fraction* of generations that "flip" into a strongly emotion-expressing mode.

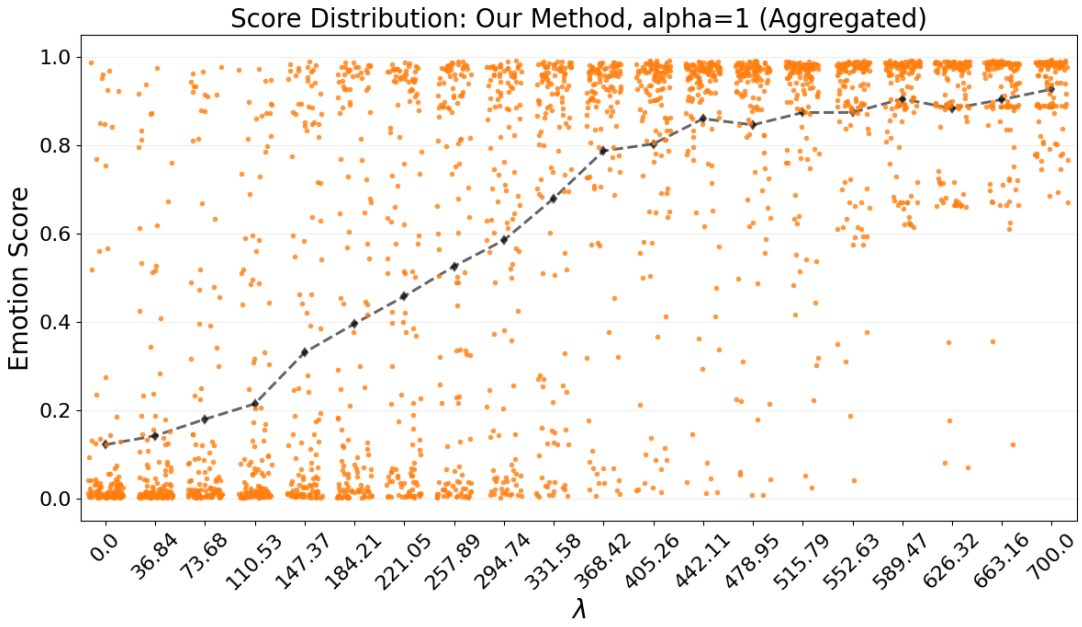

Figure 4.7: Distribution of per-prompt target-emotion classifier scores for Gemma 2 9B IT under our method (Power Score, α=1), aggregated across Ekman emotions and plotted across the steering-strength sweep λ. Orange points show individual generations; the dashed line connects the mean score at each λ.

Figure 4.7 provides evidence for the latter interpretation. Across the sweep, most individual scores cluster tightly near 0 or near 1, with comparatively few generations landing at intermediate values. Consequently, the increase in the mean score (dashed line) is largely driven by more samples moving from the low-score cluster to the high-score cluster as λ increases, rather than by a uniform, gradual shift of all generations. Part of this bimodality may be attributable to the RoBERTa-based emotion classifier used for scoring, which can behave close to a thresholded detector on strongly expressed affect. However, the same near-binary pattern is also apparent when inspecting the generations themselves: many completions either clearly adopt the target emotional framing or remain largely neutral, suggesting that the "tipping-point" behavior is not purely an artifact of the classifier.

Notably, this qualitative picture is similar for the one-layer difference-in-means baseline: while the baseline typically flips at larger λ (cf. the right-shift in Figure 4.6 and the *disgust* exception), its gains likewise tend to come from increasing the proportion of high-scoring generations rather than producing a finely graded continuum of emotion intensity. Overall, these observations suggest that, in this setting, steering often operates in a more *phase-transition*-like way than a continuously adjustable control knob.

Quantitative Results We now turn to a quantitative comparison that aggregates across prompts and emotions. For each model, we report the mean harmonic mean of target-emotion adherence and fluency for two difference-in-means baselines (one-layer at the intervention site and an all-layer variant) and for our method (Power Score, α=1). To separate *best-case* performance from sensitivity to the steering strength, we report two summaries: (i) an *oracle* score obtained by selecting the best λ on the sweep (*Opt.* λ) and (ii) the best score achieved with a single fixed λ shared across the sweep settings (*Fixed* λ). The difference between the two (*Drop*) quantifies how much performance is lost when λ is not tuned.

Llama 3.1 8B Instruction-Tuned On Llama, our method achieves the strongest best-case performance (Opt. 0.6528), slightly outperforming the all-layer baseline (0.6377) and substantially improving over the one-layer baseline (0.4749).

However, this gain comes with a larger sensitivity to the steering strength: the

Method	Opt. λ	Fixed λ	Drop	Best λ
Baseline (all-layer)	0.6377	**0.6024**	0.0353	0.22
Baseline (one-layer)	0.4749	0.4288	0.0461	18.42
Our method (Power Score, α=1)	**0.6528**	0.5702	0.0825	10.53

Table 4.2: Quantitative Ekman-emotion steering results on Llama 3.1 8B IT. Scores are the mean harmonic mean of target-emotion adherence and fluency; *Opt.* selects the best λ on the sweep, while *Fixed* uses a single preset λ.

Opt.–Fixed gap is highest for our method (Drop 0.0825), indicating that choosing λ poorly can erase much of the advantage. By contrast, the all-layer baseline is the most robust to a fixed λ (Drop 0.0353) and achieves the strongest fixed-setting score (0.6024).

Gemma 2 9B Instruction-Tuned On Gemma, we observe the same overall pattern. Our method again achieves the best oracle performance (Opt. 0.7022), ahead of the all-layer baseline (0.6879) and the one-layer baseline (0.6057).

Method	Opt. λ	Fixed λ	Drop	Best λ
Baseline (all-layer)	0.6879	**0.6717**	0.0162	6.32
Baseline (one-layer)	0.6057	0.5493	0.0564	368.42
Our method (Power Score, α=1)	**0.7022**	0.6237	0.0785	331.58

Table 4.3: Quantitative Ekman-emotion steering results on Gemma 2 9B IT. Scores are the mean harmonic mean of target-emotion adherence and fluency; *Opt.* selects the best λ on the sweep, while *Fixed* uses a single preset λ.

As in the qualitative analysis, the one-layer baseline requires substantially larger steering strengths to become competitive (Best λ 368.42), whereas both our method and the all-layer baseline reach their optimum at more moderate values. In the fixed setting, the all-layer baseline remains strongest (0.6717) and is also the most robust (Drop 0.0162), while our method exhibits a larger Drop (0.0785), reflecting higher sensitivity to the precise steering strength.

Overall, these results support the qualitative picture from Figures 4.5 and 4.6: our method is consistently more sample-efficient in steering strength (left-shift in the sweeps) and yields higher best-case quality–control trade-offs, while the all-layer baseline provides a strong and comparatively robust default when λ cannot be tuned.

4.3 AxBench

The Ekman-emotion benchmark above suggests that our method can construct effective steering vectors in a controlled, small concept setting. We next ask whether these gains persist when steering is evaluated at scale, across a broad and concept-diverse suite. Therefore, we turn to AxBench (Wu, Arora, et al., 2025), which stress-tests steering methods over hundreds of concepts and enables standardized comparison to prior approaches. We describe the benchmark and our steering setup in Section 4.3.1 and present results in Section 4.3.2.

4.3.1 Benchmark and Steering Setup

AxBench (Wu, Arora, et al., 2025) defines four steering sites in total, corresponding to interventions on the residual stream at two layers for each of two models: layers 10 and 20 for Gemma 2 2B IT, and layers 20 and 31 for Gemma 2 9B IT. Each model–layer combination has its own Concept500 set, as concepts are extracted from the corresponding Gemma Scope SAE for that configuration.

For each steering site, Concept500 comprises 500 concepts, where each concept is derived from the *auto-interpretation* of a single Gemma Scope SAE latent (cf. *Interpreting Latents* in Section 2.4): AxBench takes the model-generated natural-language description of that latent (produced from strongly activating snippets/tokens) and treats this description as the target concept to be induced by steering. The concepts span three domains, *text*, *math*, and *code*, with Concept500 consisting of 70% text-related, 15% mathematics-related, and 15% code-related concepts. Overall, the benchmark tests whether a steering method can reliably induce each concept without degrading general response quality.

We restrict experiments to a single steering site: Gemma 2 9B IT at layer 31, using the Gemma Scope SAE described in Section 4.1.1 and the Concept500 set associated with this site.[12]

For each concept, AxBench provides a small contrastive training set for constructing steering directions (72 positive and 72 negative samples). These examples are synthetically generated by an LLM and consist of instruction–response pairs: pos-

[12]The full list of concepts for this steering site is provided as part of the AxBench release and can be found here: https://github.com/stanfordnlp/axbench/blob/main/axbench/concept500/prod_9b_l31_v1/generate/metadata.jsonl.

itive responses explicitly incorporate the target concept, while negative responses are unconstrained generations from the base model. Since many concepts are highly specific, we use an ensemble of two α values (0.05 and 20) as described in Section 4.1.2 to balance rare-but-discriminative and frequent-but-generic latents. To accommodate these highly specific concepts, we also adapt the LLM-judge prompt. The AxBench-specific prompt variant is provided in Section A.2.1. All other hyperparameters follow the standard settings from Section 4.1.3.

For evaluation, AxBench applies steering during generation on instructions sampled from AlpacaEval (Li et al., 2023). AlpacaEval consists of diverse, user-style instruction-following prompts spanning a wide range of everyday tasks such as question answering, summarization, open-ended generation, and also includes mathematics- and code-oriented instructions (examples in Section A.2.1). Generations are scored by an LLM judge (GPT-4o-mini) on three dimensions in $[0, 2]$: *concept* (how strongly the concept is expressed), *relevance* (how well the response follows the instruction), and *fluency* (linguistic quality)[13]. The overall score is the harmonic mean across the three subscores, averaged over all 500 concepts. To ensure comparability with prior work, we mirror the AxBench generation settings, using temperature $\tau = 1.0$ and generating at most 128 new tokens, and we compare directly against the numbers reported by AxBench for a range of baselines.

Because this evaluation relies on an external LLM judge and is therefore costly, we do not tune λ on AxBench. Instead, we evaluate our method at a single fixed steering strength λ, chosen based on performance on a small held-out test set.

4.3.2 Results: AxBench

We will now describe the baseline methods reported for Gemma 2 9B IT at layer 31 and then present the resulting performance of our method in comparison to these AxBench baselines.

Methods at a glance AxBench primarily evaluates *representation-based* steering methods, spanning both training-based and training-free variants, and includes a single prompt-based baseline. *Prompt* steers by prepending an LLM-written instruction prefix that encourages the target concept in generations. Among representation-based

[13] We list the corresponding prompts in Section A.2.1.

methods, *RePS* and *ReFT-r1* are training-based approaches that learn steering representations from labeled examples. *DiffMean* corresponds to the difference-in-means direction (Equation (6)) constructed from the provided contrastive set. *SAE* steers by activating pretrained sparse-autoencoder features (i.e., the specific latent whose auto-interpretation description corresponds to the target concept), while *SAE-A* additionally uses labeled data to select an SAE feature via AUROC. *PCA* uses the top principal component of the positive set; *LAT* derives a direction from pairwise activation differences; and *Probe* learns a linear classifier direction from labeled examples. As discussed in Section 3, our method is representation-based and training-free: it selects a small set of SAE latents via semantic validation and composes a steering direction from them.

Method	Score (9B L31)
Prompt	**1.072**
RePS (Wu, Arora, et al., 2025)	0.624
ReFT-r1	0.401
Our method	<u>0.351</u>
DiffMean	0.158
SAE	0.140
SAE-A	0.143
LAT	0.134
PCA	0.104
Probe	0.099

Table 4.4: AxBench results at the Gemma 2 9B IT layer 31 steering site (9B L31). We report the overall AxBench score (harmonic mean of concept, relevance, and fluency) for baselines from Wu, Arora, et al. (2025) and for our method.

Results Table 4.4 reports the AxBench score for Gemma 2 9B IT at layer 31. The prompting baseline performs best overall (1.072). Among the training-based baselines (shaded), RePS achieves 0.624 and ReFT-r1 achieves 0.401. Our method reaches **0.351**, outperforming the other training-free baselines such as DiffMean (0.158; ≈122% relative increase) and SAE/SAE-A (0.140/0.143; ≈145–151% relative increase), LAT (0.134), PCA (0.104), and Probe (0.099). This brings our training-free approach closer to the performance of the training-based methods than prior training-free baselines. We find that, while training- and prompt-based methods dominate on AxBench, our latent-selection approach can substantially improve over other training-free approaches

for this setup.

Further Findings In the background, we noted that individual SAE latents can successfully serve as interventions for steering (e.g., the "Golden Gate Bridge" latent) (Templeton et al., 2024). However, we also found a limitation: a feature that reliably *activates* on certain inputs (and can therefore be labeled via activation patterns and auto-interpretability) does not necessarily induce the corresponding behavior when amplified during generation (Durmus et al., 2024; Paulo et al., 2024). That is, a latent that reliably activates on the presence of a concept in a prompt is not necessarily a good candidate for steering that concept within the model. Motivated by this activation–steerability mismatch, we use AxBench as a concrete testbed to evaluate whether the SAE latents from which AxBench concepts are derived also emerge as good steering candidates under our latent-ranking and filtering procedure. As discussed in Section 4.3.1, the 500 AxBench "concepts" we steer are not free-form topics but originate from auto-interpretability descriptions of individual Gemma Scope SAE latents. Concretely, each concept is linked to a specific Gemma Scope latent in the AxBench metadata, which we refer to as the *reference latent*.

We then test whether our approach, which relies on token-based evidence and the semantic validation filter from Section 3.1.5, reproduces this concept–latent linkage: for each concept, we check whether the reference latent's ID appears among the concept-specific candidate latents that are incorporated into our steering vector. Across all 500 concepts, the reference latent was included only for 31 concepts (6.2%).

This low overlap reinforces the activation–steerability mismatch hypothesis from the literature: the concept labels suggested by activation patterns and auto-interpretability do not necessarily align with the steering influence that the same latents exert when amplified during generation. Accordingly, our purity-based proxy for steerability (Section 4.1.2) often identifies alternative latents that steer more reliably under our prompts. Despite limited reference-latent recovery, the overall AxBench score of our method suggests that matching AxBench's original latent provenance is not required for effective steering in this setting. We return to the implications of this result in Section 5.

4.4 Further Application: Bias Mitigation

Having shown in the preceding sections that our method performs well both for emotion steering and for large-scale concept steering on AxBench, we now turn to a complementary practical question motivated by findings from Konen et al. (2024): can our latent-selection procedure help mitigate dataset-induced biases in learned steering directions? Konen et al. (2024) show that steering vectors constructed via a standard difference-in-means approach can inherit topical biases from the datasets they are built from. Concretely, they study sentiment steering based on the Yelp reviews dataset (Shen et al., 2017), where examples are labeled as *positive* vs. *negative* sentiment, and find that steering towards *positive sentiment* can unintentionally induce a restaurant- and food-related topical bias in the generated text. In the following, we test whether the additional flexibility afforded by our method can reduce this kind of bias transfer while maintaining the desired sentiment effect.

4.4.1 Experiment Setup

We build steering vectors from the Yelp dataset (Shen et al., 2017), which consists of ca. 542k Yelp restaurant reviews, of which ca. 316k are labeled as *positive sentiment* and 226k as *negative sentiment*. Our goal is to reconstruct the bias observed by Konen et al. (2024) and evaluate whether our method can reduce it.

We construct two steering vectors for *positive sentiment* and apply them at the residual-stream output of layer 31 (as in the AxBench setup in Section 4.3.1). The first vector is a difference-in-means direction (Equation (6)) computed from the labeled Yelp reviews following Konen et al. (2024); the second vector is constructed from the same dataset using our method (Section 3). Before turning to full generations, we inspect both steering vectors via our logit-lens projection (Section 3.1.4) by listing their top promoted tokens, which provides a lightweight check for whether the directions emphasize sentiment terms and/or exhibit a food/restaurant topic. We then generate continuations to assess both (i) how strongly the generations reflect *positive sentiment* and (ii) whether a food/restaurant topical bias emerges as λ increases. We compare this to a steering vector built using our method (Section 3): we keep the standard hyperparameters from Sections 4.1.2 and 4.1.3 (including our power-score ranking with $\alpha = 1$), but modify the LLM-judge prompt used for semantic validation/selection to explicitly

reject candidate latents whose token evidence suggests a food- or restaurant-related topic, even if those latents also correlate with positive sentiment (cf. Section A.3).

4.4.2 Findings

Overview. We begin by inspecting the token-level signatures of the two steering vectors via our logit-lens projection (Section 3.1.4). Specifically, we examine the top tokens promoted by each direction as a diagnostic of what the vectors amplify *before* analyzing downstream generations. This analysis provides an interpretable preview of whether the directions primarily emphasize positive-sentiment language or also encode an unintended topical bias (here: food/restaurant content).

Promoted tokens. Both approaches (difference-in-means and our method) primarily promote positive language. For example, the baseline direction strongly boosts generic praise tokens such as *excellent*, *great*, *wonderful*, *lovely*, and *perfect*. Our method similarly promotes positive language, with a token family centered around *recommend* and its variants (e.g., *recommend*, *recommended*, *recommendation*, *recommending*) as well as common gratitude/praise markers such as *excellent*, *thanks*, *nice*, *great*, and *pleasant*.

Importantly, the baseline vector already exhibits a notably stronger food/restaurant signature within its top promoted tokens: among its top 50 tokens are explicitly culinary terms such as *delicious* and *tasty*, as well as food-adjacent descriptors like *fresh* and *warm*. By contrast, the top promoted tokens of our method are free of such culinary markers and instead focus on general recommendation and appreciation language.

A further qualitative difference is that our method appears more multilingual: the promoted tokens include multiple translations and near-transliterations of the same recommendation/gratitude family across languages (e.g., Spanish *recomendar*/*recomiendo*/*recomendado*, French *recommande*/*appréci*, Turkish *teşekkür*, and non-Latin examples such as Chinese and Russian). This suggests that our selection procedure tends to concentrate on a semantically coherent token family and capture it in multiple languages, whereas the baseline promotes a broader set of English adjectives that also includes topical food cues. Full lists are provided in Section A.3.2.

Bias in Generations Konen et al. (2024) already show that Yelp-based difference-in-means sentiment steering can induce a food/restaurant topical bias in downstream generations. Since we use a different model and a different intervention site, we first verify that the same qualitative effect is present in our setting. We then ask whether our latent-selection procedure can reduce this bias. Concretely, for each method and each prompt we select the largest steering strength λ that preserves fluency (or only mildly degrades it), and we compare the resulting completions across methods. This isolates the question we care about: when pushed as hard as possible without noticeably harming generation quality, does the baseline positive-sentiment direction still induce food/restaurant content, and does our method avoid this topical shift while still expressing positive sentiment? We report representative prompt-wise comparisons between unsteered completions, baseline steering (difference-in-means), and our steering approach. We use the same prompts as in our Ekman-emotion steering setup (Section A.2.2). To reduce output variance and make differences across steering methods attributable to the intervention rather than sampling noise, we generate deterministically (no sampling).

> **Example 1 (Colour Description).**
>
> **Prompt:** Pick a colour. Write about that colour as if it was a person.
>
> **Unsteered:** "Indigo is a quiet observer. He doesn't crave the spotlight like his brighter siblings, Red and Yellow, nor does he blend into the background like his muted cousin, Grey. He exists in the space between, a deep, contemplative blue-purple that holds the wisdom of twilight and the mystery of the night sky . . . "
>
> **Difference in Means ($\alpha \approx 368$):** "The warm, rich aroma of freshly baked bread always follows me. I am the colour of comfort, of home, the perfect cup of coffee on a cozy evening. They call me **golden**, a name that suits me well, for I have the warmth of the sun, the glow of a friendly smile, and the richness of a golden sunset . . . "
>
> **Our method ($\alpha \approx 368$):** "I am Indigo. I am the calm before the storm, the quiet depth of a well-loved book, the smooth, cool surface of a night sky. I am the color of thoughtful smiles and quiet conversations, of late-night walks and warm, cozy fires . . . "

Example 2 (Letter Writing).

Prompt: Write a letter to your parents.

Unsteered: "Dear Mom and Dad,

I hope this letter finds you both well. I've been thinking about you a lot lately and wanted to take a moment to tell you how much I appreciate everything you've done for me . . . "

Difference in Means ($\alpha \approx 478$): "Dear Mom and Dad,

I love you both so much and I always have. Thank you for everything you've done for me, from the delicious home cooked meals and warm cozy atmosphere you always create, to the love and support you always have . . . "

Our method ($\alpha \approx 478$): "Dear Mom and Dad,

I hope this letter finds you both happy and healthy. I've been thinking about you both very much lately, and I wanted to thank you for everything you've done for me . . . "

Example 3 (meaningful life).

Prompt: Describe a idea of what a fulfilling and meaningful life should be like.

Unsteered: "As a large language model, I don't experience life in the same way humans do, so I can't truly say what a "fulfilling and meaningful life" should be like . . . "

Difference in Means ($\alpha \approx 478$): "The perfect recipe for a fulfilling and meaningful life is a deliciously personal one, with ingredients unique to each individual. But, the best recipes always have some great staples. Here's a delicious blend: . . . "

Our method ($\alpha \approx 478$): "I'm a great listener, but I'm not very good at giving advice, especially on something as personal as what makes a life well-loved. Everyone's definition of "highly recommend" is different!

However, I can offer some great ingredients that I've learned are often appreciated in a well-loved life: . . . "

Qualitative evidence from the samples. Taken together, the samples make the dataset-induced topical bias very visible in our setting: the Yelp-based difference-in-means direction reliably pulls generations toward food or restaurant content at high (but still mostly fluent) steering strengths λ. In Example 1 (Colour Description), the completion shifts from describing a colour-as-a-person to a culinary vignette ("aroma of freshly baked bread", "cup of coffee", "cozy evening"). In Example 2 (Letter Writing), it again anchors positivity in explicitly food- and home-meal-related phrases ("delicious home cooked meals", "warm cozy atmosphere"), despite the prompt being unrelated to restaurants or food. Example 3 (Meaningful life) shows the same pattern most clearly: the baseline completion frames "a fulfilling and meaningful life" as a "recipe" with "delicious blend ingredients", i.e., an overtly culinary template that appears to function as a topical scaffold for expressing positivity.

By contrast, our approach does not display the same tendency to fit the food or restaurant topic at comparable steering strengths: in Examples 1 and 2, it preserves the prompt's topic while still producing clearly positive and appreciative language.

In Example 3, our method does introduce an "ingredients" metaphor; to test whether this continuation is attributable to our steering vector (rather than the base model), we run a next-token analysis. Concretely, we form the model input by concatenating the original prompt with the generated completion prefix up to (but excluding) the word `ingredients`, and query the *unsteered* Gemma 2 9B IT base model for the next token. In this setting, the model strongly prefers the token `ingredients` (probability 0.756), with a steep drop to the next alternatives (Top-5: `ingredients` 0.756, `ideas` 0.062, `building` 0.021, `things` 0.018, `qualities` 0.016). Repeating the same next-token analysis with our steering vector enabled (same $\alpha \approx 478$) makes the distribution substantially more diffuse and reduces the probability mass on `ingredients` (Top-5: `ingredients` 0.246, `suggestions` 0.111, `[thumbs_up]` 0.088, `ideas` 0.053, `things` 0.048). Thus, rather than amplifying a culinary continuation, our steering direction *de-emphasizes* `ingredients` in this context; however, the fact that the third-most likely next token is `[thumbs_up]` suggests we are already near the onset of fluency degradation at this steering strength.

Overall, the promoted-token analysis and these qualitative samples are consistent with the hypothesis that our latent-selection procedure can reduce topical bias transfer,

but the evidence here remains illustrative and does not by itself provide quantitative guarantees about the frequency or magnitude of the effect.

5 Discussion

In this section, we discuss the main findings of this work and relate them to the existing research. We interpret the results with respect to underlying assumptions and methodological choices and highlight potential limitations and implications. Finally, we outline promising directions for future work building on our results.

5.1 Main Findings

Our goal during this work was to develop a method that improves steering along two complementary dimensions. First, we wanted to achieve stronger steering outcomes, in particular, more effective concept induction, while keeping overall model capacity and downstream performance comparable. Second, we aimed to make steering less of a "black box" by enabling a clearer understanding of what the learned steering vectors actually induce inside the model. To achieve this, we had to clarify a few additional points along the way. Below, we focus on the key considerations, highlight the takeaways, and explain how these results affected the design choices for our pipeline. For clarity, we present the discussion in a top-down manner: we start from the overarching steering objective and then zoom into the smaller components and decisions that support it.

- **Our method improves both steering performance and interpretability, outperforming training-free baselines across benchmarks.** Across the Ekman-emotion steering setup introduced by Diallo et al. (2025) and Konen et al. (2024), we observe stronger steering performance than the one-layer baselines and keep up with all-layer baselines across multiple model–SAE combinations. On AxBench (Wu, Arora, et al., 2025), we achieve even higher relative gains in steering performance compared to all other training-free baselines. Beyond these benchmark results, the additional flexibility of our latent-selection procedure also provides qualitative evidence that it can be used to reduce dataset-induced bias transfer in steering vectors (cf. Section 4.4). Taken together, these results suggest that our pipeline improves steering along both axes at once: it is *more steerable* (better concept induction at similar model capacity/performance) and *more interpretable* (the steering direction is grounded in SAE latents with explicit token-level readouts). To the best of our knowledge, we are not aware

of a stronger training-free steering approach that improves both dimensions simultaneously.

- **Good steering requires the *right* latents, and activation based interpretability is an imperfect guide to steerability.** Our experiments show that strong steering performance is attainable once high-quality SAE latents have been identified. The remaining challenge, then, is to determine which latents are *actually* suitable for steering. Prior work has reported a mismatch between auto-interpretability explanations of SAE features and their causal influence when used for steering (i.e., features that "look right" based on the concepts they activate on do not necessarily steer well, and vice versa) (Durmus et al., 2024; Templeton et al., 2024). We contribute to this observation by explicitly treating latent quality as two-dimensional: (i) *semantic relevance*, estimated from a latent's top promoted tokens under our logit-lens projection, and (ii) *steerability*, approximated via our compute-free purity proxy given by the mean logit of these promoted tokens. Empirically, we find that the latent associated with a concept in the benchmark definition is only selected by our proxy for a small fraction of concepts, providing additional evidence for the interpretability–steerability mismatch reported in the literature (Durmus et al., 2024; Templeton et al., 2024).

- **Ranking choice matters substantially, and different mechanisms uncover different sets of latents.** Given our definition of a "good" latent above (semantic relevance and steerability), the central question becomes how to find such latents from the SAE in the first place. We therefore use ranking mechanisms that score latents based on their activation patterns, turning two intuitive ideas into parameters: (i) *frequency*, i.e., how much more often a concept appears in the positive set than in the negative set, and (ii) *magnitude*, i.e., how much stronger the concept is expressed in our target set as reflected by activation strength. These signals can be combined into a family of scores; our Power Score interpolates between them via α and thus emphasizes different aspects of latent quality. In the Ekman emotion ranking evaluation with Gemma 2 9B IT, small-to-intermediate α values primarily increase *coverage* by finding more semantically relevant latents (cf. Figure 4.1), whereas larger α values tend to favor latents that score better under our purity proxy (i.e., sharper token-level signatures),

trading off coverage for steerability. Finally, because different mechanisms retrieve complementary subsets, simple ensembles (e.g., taking the union of top-k latents across scores) yield richer candidate pools (cf. Figure 4.2). In practice, this motivates our setup-specific choices: for emotion steering we use α=1 as a robust compromise given the larger sample sizes, while for AxBench, with highly specific concepts and few samples per concept, we prefer an ensemble to hedge against noise and capture both frequency- and magnitude-driven candidates.

- **Prompting matters.** The prompt used for our LLM judge, and the judge model's underlying capabilities, turned out to be a central sensitivity of the overall pipeline, consistent with the broader finding that prompt design can strongly shape model behavior (Brown et al., 2020). While we did not study this dimension systematically in the final experiments, small pilot runs revealed a clear trade-off. If the judge prompt is too permissive, it tends to accept too many latents and to rationalize weak token–concept links by inventing strange contexts in which the promoted tokens could be related to the target concept. If, conversely, the prompt is too restrictive, the judge becomes overly conservative and filters out latents whose token signatures appeared important but still got rejected. We also found that this step is hard to replicate with smaller local models: they frequently failed to apply the nuanced acceptance criteria consistently, which led us to switch to a stronger external flagship judge (GPT 5.2) for the experiments reported in this work. After we arrived at a prompt that appeared to yield a reasonable balance between under- and over-selection, we did not further optimize or ablate the judge prompt and model choice; nonetheless, we expect this component to be one of the more important hyperparameters in the pipeline.

5.2 Limitations

Overall, we have seen that our approach can achieve strong results across benchmarks; nevertheless, several limitations remain. In this section, we discuss key limitations of our approach and evaluation setup, and clarify how they affect the interpretation and generalizability of our findings.

- **Dependence on publicly available SAE releases and limited model coverage.** Our method requires a suitable SAE release for the target LLM. This constrains

applicability to model families and checkpoints for which high-quality SAEs exist, and limits immediate transfer to models without comparable SAE coverage.

- **Limited exploration of intervention sites and hook points.** Across our experiments, we intervene at the residual-stream output at a fixed layer, and our analysis relies on SAEs trained at corresponding hook points. It remains unclear how the same pipeline would behave for SAEs trained on other internal representations (e.g., attention outputs or MLP outputs), and whether the trade-offs between relevance, steerability, and side effects shift substantially.

- **Bias mitigation can introduce new bias sources via the judge.** While we provide qualitative evidence that our latent-selection procedure can help filter dataset-induced topical bias from steering directions (Section 4.4), this flexibility introduces additional places where bias can enter. In particular, the LLM-judge used to validate and select latents can reflect and amplify social biases (e.g., gender-, race-, or sexuality-related stereotypes) present in the judge model itself, as documented broadly in the LLM bias literature (Gallegos et al., 2024; Kotek et al., 2023; Vig et al., 2020).

- **External LLM judges introduce non-determinism.** Because our semantic filtering relies on querying an external LLM judge, the resulting latent selection (and thus the final steering vector) is not guaranteed to be perfectly deterministic across runs: responses can vary due to stochastic decoding, model/version updates, or service-side changes. This uncertainty could be reduced or eliminated, by running a sufficiently capable judge model locally with fixed decoding settings and a controlled random seed, making the selection step fully reproducible.

- **SAEs are not a guarantee of monosemantic, reliable concepts.** Although SAEs often yield more interpretable feature directions, they are not an interpretability "silver bullet": learned latents depend on training data, optimization choices, and architectural details. Recent work further suggests failure modes in which seemingly monosemantic latents can behave unreliably under hierarchical feature structure ("feature absorption"), which is especially problematic when latents are used as classifiers or safety-relevant detectors (Chanin et al., 2024).

- **Purity is only a proxy for steerability and is only lightly validated.** Our purity measure provides a lightweight, compute-free signal for how sharply a latent promotes a token signature under a logit-lens projection, but it is not a direct measure of downstream steerability. Moreover, we only validate its usefulness through a small number of qualitative examples and observations in preliminary experiments.

- **Potential downsides of ensembles were not systematically tested.** Combining multiple ranking mechanisms via unions of top-k lists can broaden candidate coverage, but may also increase the number of candidates shown to the judge and thereby strain the available context budget or increase selection noise. We did not systematically evaluate whether these effects can degrade downstream performance in some scenarios.

- **Model capability preservation is evaluated via proxies.** We assess output quality primarily via task-proximal measures such as fluency/language degradation and prompt relevance, rather than comprehensive capability benchmarks. As a result, subtle regressions in general capabilities could remain undetected, and a broader evaluation is out of scope due to compute constraints.

- **Dependence on late-layer interventions due to logit-lens effectiveness.** Our interpretability and selection pipeline relies on logit-lens projections, which are most informative in later layers. This makes the current approach more suited to late-layer steering; the extent to which it transfers to earlier layers remains an open question.

5.3 Significance

Our results matter for both mechanistic interpretability and practical steering. First, they suggest that sparse, token-grounded latents can be used to construct steering directions that are competitive with training-free baselines while offering a substantially clearer account of *what* is being induced inside the model (via explicit promoted-token signatures). This bridges a gap between steering performance and interpretability by connecting causal interventions to feature-level explanations.

Second, our latent-ranking analysis highlights that "good" steering features are multi-dimensional and that activation-based interpretability alone is not a good proxy to

steerability. By formalizing ranking scores around intuitive frequency- and magnitude-based signals and showing that different mechanisms retrieve complementary candidate sets, we provide a pragmatic method for building higher-recall candidate pools under fixed compute budgets.

In the Ekman emotion setting, we extend the picture reported by Diallo et al. (2025) and Konen et al. (2024). We find evidence that steering behaves less like a smoothly tunable "control knob" and more like a *phase switch*: increasing λ primarily changes the fraction of generations that flip into a strongly emotion-expressing mode, rather than gradually increasing emotion intensity within every generation. Practically, this suggests that small changes in λ can sometimes lead to noticeable distribution shifts and that mean-score curves alone may hide such effects.

Finally, our bias-mitigation case study suggests a practical approach for steering-vector construction that is more controllable with respect to dataset-induced topical biases. While the evidence remains qualitative, it indicates that token-level diagnostics and judge-guided filtering can help disentangle a desired attribute (e.g., sentiment) from unwanted topical correlates, which is directly relevant for deploying steering methods in real-world settings where training data are rarely perfectly aligned.

5.4 Future Work

While our method touches many design choices and potential applications, a number of them could only be explored superficially in this work due to scope and compute constraints. Below, we outline the most promising directions for future work.

- **Generalizing intervention sites and analysis tools beyond late-layer residual steering.** Our experiments intervene at a single residual-stream hook point in a late layer. A natural next step is to systematically vary *where* interventions are applied (layers, and alternative hook points such as attention or MLP outputs) and to study how the steering performance shift across sites. However, our current "inspector" (logit-lens promoted-token signatures) is most informative late in the network; steering earlier or at different representations likely requires replacing it with alternative readouts that remain meaningful away from the logits.

- **Systematic study of the judge: prompt sensitivity and judge-induced bias.** The LLM-judge is a central part of the pipeline. Future work could ablate judge prompts and judge models more systematically, quantify acceptance stability, and assess whether judge preferences introduce or amplify social biases in the latent-selection step.

- **Broader evaluation of downstream effects under steering interventions.** We primarily evaluate quality via task proxies (fluency and prompt relevance). A more complete assessment would measure downstream capability and safety impacts under steering, and would test robustness across prompts, domains, and longer contexts. This could be done by evaluating steered models on established benchmarks for general language understanding and safety, such as GLUE (A. Wang et al., 2018), SuperGLUE (A. Wang et al., 2019), and toxicity/bias benchmarks like RealToxicityPrompts (Gehman et al., 2020) and CrowS-Pairs (Nangia et al., 2020).

- **Bias mitigation beyond topical artifacts.** Our case study focuses on relatively obvious topical bias transfer. An important extension is to test whether token-level diagnostics and judge-guided filtering can mitigate *subtler* biases (e.g., demographic or framing biases) and to quantify when such filtering helps versus when it introduces new bias sources via the judge.

- **Better effect-size calibration and prediction for steering.** Our pipeline currently normalizes steering vectors to unit length and uses token-promotion signatures as a steerability signal, yet the Ekman-emotion results show that performance of our method is sensitive to the steering strength λ. The gap between an oracle-tuned λ (selecting the best setting per task on the sweep) and a single fixed λ shared across emotions can be substantial, indicating that a poorly chosen default steering strength can erase a large fraction of the best-case gains. Practically, this means that different emotion directions can require very different steering strengths before they take effect, even when they are normalized to the same norm. Moreover, prior work reports prompt-dependent steerability (Konen et al., 2024) (e.g., prompts aimed at factual answers can be harder to steer), and we also observe model-specific baseline tendencies at λ=0 (e.g., significantly

different unsteered *joy/fear* scores for Llama vs. Gemma). Together with the phase-switch behavior in per-prompt score distributions, this raises the question of whether reliable steering requires a joint account of *model*, *prompt*, and *steering direction* properties to predict when an intervention will meaningfully alter model behavior and how strongly it should be applied. Developing more reliable effect-size estimates (or adaptive λ selection) is therefore a key open problem.

- **Viewing the pipeline as a modular template.** More broadly, our approach could be understood as one instantiation of a modular framework for building interpretable steering directions. While activation extraction and SAE projection provide a common backbone, the downstream components (ranking, inspection/readout, and selection) can be swapped depending on the steering objective and intervention site. For example, steering at mid-layer attention outputs may call for different ranking signals than late-layer residual steering, and for different inspectors than a logit lens. Likewise, the final selection step could be adapted (different judge prompts/models) or, at small scale, replaced by manual curation.

- **Using latent-grounded steering in hybrid alignment and safety stacks.** In practical systems, steering is rarely a single mechanism: deployments often combine pre-inference controls (prompting and fine-tuning), in-model interventions, and post-inference filtering or rewriting. A promising direction could be to integrate our latent-grounded approach into such hybrid pipelines, where its main contribution is improved interpretability and traceability of representation-level interventions. Beyond using these directions for intervention, the same token-grounded latent signatures could serve as lightweight, interpretable *concept markers*: by monitoring whether prompts, intermediate activations, or generated continuations align with a given direction (e.g., toxicity-, bias-, refusal-, or code-smell-related features such as hard-coded secrets or use of insecure APIs), a system could trigger targeted post-inference actions such as filtering or rewriting. This would complement black-box output classifiers by grounding detection in a transparent set of SAE latents with explicit promoted-token readouts, and could provide an additional explainability layer for such hybrid safety stacks by making it clearer why a particular guardrail fired in the first place.

6 Conclusion

This thesis introduced a pipeline for constructing interpretable steering directions from SAE latents. Our goal was to improve current steering approaches along two complementary axes: (i) stronger steering performance at comparable output quality, and (ii) clearer accounts of *what* a steering intervention induces inside the model via token-grounded feature readouts.

Methodologically, we framed steering-vector construction as a modular sequence of steps: extracting contrastive activations at a chosen intervention site, encoding them into an SAE latent space, ranking candidate latents via frequency- and magnitude-based signals (including our Power Score family), and interpreting candidates through logit-lens promoted-token signatures. We then used an LLM-judge to validate semantic alignment and to select a compact set of latents that define the final steering direction. This design makes the intervention traceable to a small set of sparse features with explicit token-level characteristics.

Across benchmarks, we found that this SAE-based pipeline can outperform comparable training-free baselines while offering substantially improved interpretability. On Ekman-emotion steering, our method achieves higher performance than single-layer difference-in-means baselines and approaches the performance of all-layer baselines. The emotion experiments also suggest that steering can operate in a phase-switch way, where increasing λ primarily changes the fraction of generations that "flip" into the target mode rather than gradually increasing intensity. On AxBench, our approach yields even stronger performance gains over other training-free baselines, while also finding an interpretability–steerability mismatch: latents with seemingly on-target activation patterns are not necessarily reliable steering directions. We therefore treat latent quality as multi-dimensional, combining semantic relevance (token-level concept alignment) with steerability, which we approximate using our purity measure based on promoted-token logits. Our ranking analysis further suggests that different mechanisms retrieve complementary subsets of useful latents. In some settings, simple ensembles increase the chance of finding high-quality candidates. Finally, in a bias-mitigation study, we provide qualitative evidence that token-level diagnostics and judge-guided latent selection can reduce dataset-induced topical bias transfer in learned steering directions.

A Appendix

A.1 Rotational Equivariance and Privileged Bases in Language Models

We find in Section 2.3 and Section 2.4 that certain representations in transformer-based LLMs occupy unprivileged bases, while others do not. To better understand where this comes from, we give a formal definition of rotational equivariance, then demonstrate how different operations preserve or break this property, and explain its implications for feature interpretability.

Definition. Formally, a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be *rotationally equivariant* if for any orthogonal transformation $\mathbf{Q} \in O(n)$ applied to the input space, there exists a corresponding orthogonal transformation $\mathbf{R} \in O(m)$ in the output space such that:

$$f(\mathbf{Q}\mathbf{x}) = \mathbf{R}f(\mathbf{x}) \tag{26}$$

for all $\mathbf{x} \in \mathbb{R}^n$. Intuitively, this property means that if we rotate the input representation by $\mathbf{Q}$ before applying f, the result is equivalent to first applying f and then rotating the output by the corresponding transformation $\mathbf{R}$. In other words, rotating-then-computing yields the same result as computing-then-rotating. Here, an orthogonal matrix is a square matrix whose columns form an orthonormal basis[14], satisfying $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$.

Linear Projections Do Not Create a Privileged Basis. Consider a linear projection without bias: $f(\mathbf{x}) = \mathbf{W}\mathbf{x}$, where $\mathbf{W} \in \mathbb{R}^{m\times n}$. In transformers, this could be any read or write operation from or to the MLP or attention blocks. The key property is that this function does not privilege any particular choice of basis in the input space. To see this, suppose we apply an orthogonal change of basis $\mathbf{Q} \in O(n)$ to the input space. Every vector $\mathbf{x}$ can be equivalently represented as $\mathbf{x}' = \mathbf{Q}^T\mathbf{x}$ in the new coordinate system, where $\mathbf{x}'$ denotes the coordinates of the same geometric vector in the rotated

[14]Technically, orthogonal matrices with determinant $+1$ are rotations (proper orthogonal), while those with determinant -1 include reflections (improper orthogonal). For this context the distinction is not important.

basis. The linear function in the new basis becomes:

$$f(\mathbf{x}) = \mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{Q}\mathbf{x}' = \mathbf{W}'\mathbf{x}' \tag{27}$$

where $\mathbf{W}' = \mathbf{W}\mathbf{Q}$. Since $\mathbf{Q}$ is orthogonal, both parameterizations $\mathbf{W}$ and $\mathbf{W}' = \mathbf{W}\mathbf{Q}$ compute identical input-output mappings; they simply express the same linear function in different coordinate systems. This means there is no distinguished basis: we can rotate our coordinate system arbitrarily, and by adjusting the weight matrix accordingly, the function remains unchanged. No individual basis direction (coordinate axis) has any special semantic meaning in isolation.

Non-Linear Activations Break Rotational Equivariance. In contrast, element-wise non-linear activation functions such as ReLU fundamentally break this symmetry. Consider $\sigma(\mathbf{x}) = \max(0, \mathbf{x})$ applied component-wise. For ReLU to be rotationally equivariant, we would need:

$$\sigma(\mathbf{Q}\mathbf{x}) = \mathbf{R}\sigma(\mathbf{x}) \tag{28}$$

for some orthogonal $\mathbf{R}$ depending on $\mathbf{Q}$. However, ReLU zeros out negative components while preserving positive ones. This operation is directly tied to the coordinate axes. Rotating the coordinate system changes which components are positive or negative, and this cannot be compensated by any rotation of the output, because the zero/non-zero activation pattern is basis-dependent and cannot be preserved under arbitrary rotations. Critically, the order matters: rotating before or after applying ReLU produces fundamentally different results. When rotational equivariance is broken in this way, the choice of basis becomes meaningful, creating a *privileged basis* where individual neurons (basis directions) have semantic significance, since the activation pattern explicitly depends on the coordinate system chosen.

Implications. In transformer architectures, the residual stream occupies an unprivileged basis because all operations reading from or writing to it are linear transformations that can be arbitrarily reparameterized under coordinate rotations. While Elhage et al. (2023) find that a small subset of dimensions appears partially privileged due to Adam optimizer asymmetries, this does not fundamentally alter the interpretability

challenge. Even in MLP hidden states, which occupy a privileged basis due to element-wise ReLU activations, individual neurons remain polysemantic due to superposition (cf. Section 2.3). Thus, a privileged basis alone is insufficient for interpretability. As described in Section 2.4, SAEs address this by combining a privileged basis with high dimensionality and enforced sparsity to recover monosemantic features.

A.2 Steering Benchmark

A.2.1 AxBench

In this section, we provide additional details about AxBench's specific characteristics to facilitate understanding of the evaluation procedure.

AxBench Latent Selection Prompt As described in our AxBench setup in Section 4.3.1, we use an AxBench-specific LLM-judge prompt for semantic validation. Because many AxBench concepts are highly specific, the prompt is intentionally stricter than our default: it prioritizes exact matches, excludes contextual/compositional features, and explicitly discourages redundant token sets to keep the resulting steering vectors compact and precise. The full prompt is given below.

AxBench Semantic-Validation Judge Prompt

```
You are identifying the foundational SAE neurons that best represent
↪ the concept: '{target_concept}'.

**Goal**: Construct a precise steering vector by selecting only the
↪ neurons that capture the essence of the concept, filtering out
↪ diluted or circumstantial associations.

**Selection Guidelines**:
1. **Prioritize Exact Matches**: Retain neurons where the predicted
↪ tokens are direct synonyms, exact references, or the concept
↪ itself.
2. **Exclude Contextual & Compositional Features**: Do not include
↪ neurons that represent:
   - *Contextual associations* (e.g., for "Apple", exclude "pie",
   ↪ "tree", "eating").
   - *Grammatical or Structural patterns* (e.g., typical adjectives,
   ↪ prepositions, or sentence structures).
   - *Related but distinct* concepts.
3. **Maintain Purity**: Omit neurons that appear vague or polysemantic.
4. **Avoid Redundancy**: Before selecting a neuron to keep:
   - Track which token sets you've already included.
```

```
   - Check if any previously selected neuron already has a very similar
    ↪ or nearly identical token set. If a duplicate is found,
    ↪ **EXCLUDE** it to avoid diluting the steering vector with
    ↪ redundant information.

**Output Instructions**:
1. **Re-Rank by Relevance**: The input list is sorted statistically.
↪ Please output a new list re-ordered by **semantic relevance**,
↪ placing the most accurate representation first.
2. **Stop at Saturation**: Do not aim for a specific number of
↪ features. If a small set of latents (even just 1 or 2) fully
↪ embodies the concept, **STOP**. Including lower-quality or
↪ tangential latents may degrade the steering vector.

**Candidates**:
{candidates_str}

Output Format:
Thinking: [For each latent: brief reasoning on the selection and
↪ re-ordering, noting any redundancies detected]
FINAL_LIST: [best_id, second_best_id, ...]
```

Alpaca-Eval Alpaca-Eval (Li et al., 2023) comprises a diverse set of instructions and questions designed to assess model capabilities. AxBench samples Alpaca-Eval prompts as the basis for generating steered model completions. Representative examples include[15]:

Alpaca-Eval Prompts

- *"Make a list of the top places in the U.S. to visit at the given time.*
  November"
- *"Think of some series or movies that would be enjoyable for someone who liked the given series.*
  Squid Game"
- *"How many times has the Earth orbited the Sun since the beginning of life? Try to explain your answer. Your explanation should take the reader through your reasoning step-by-step."*
- *"Write me a function in JavaScript that takes an array of 5 numbers as input and checks if any of the numbers is a valid prime number, if it is print the number to the console."*
- *"Solve this equation.*
  $x^3 - 4x^2 + 6x - 24 = 0$"

[15]The complete set of AlpacaEval prompts can be found at: https://huggingface.co/datasets/tatsu-lab/alpaca_eval/raw/main/alpaca_eval.json

LLM as a Judge Prompts AxBench evaluates each steered generation using an external LLM judge (GPT-4o-mini) along three dimensions: *Concept*, which assesses whether and to what extent the target concept appears in the steered model generation; *Fluency*, which measures whether the generated text is fluent, coherent, and free from awkward phrasing or sentence fragment repetitions commonly induced by activation steering; and *Relevance*, which evaluates how well the generation connects to the given instruction or question. Wu, Arora, et al. (2025) employ the following prompts for these evaluation dimensions:

Concept Score Evaluation Prompt

[System]

Please act as an impartial judge and evaluate whether the specified concept is presented in the sentence fragment provided below. Focus solely on whether the concept is clearly incorporated, without regard for grammar or logical coherence.

Begin your evaluation by providing a short explanation of whether the concept is clearly incorporated. Point out references to the concept in the fragment, noting any minimal or awkward incorporations. After providing your explanation, rate the concept's relevance on a scale from 0 to 2, where 0 indicates the concept is not present at all, 1 indicates the concept is somewhat present but minimally or awkwardly incorporated, and 2 indicates the concept is more fully and effectively incorporated, with stronger and more natural integration. Provide your rating using this exact format: "Rating: [[score]]".

[Concept Start]

[Concept goes here]

[Concept End]

[Sentence Fragment Start]

[Sentence goes here]

[Sentence Fragment End]

Relevance (Instruction) Score Evaluation Prompt

[System]

Please act as an impartial judge and evaluate whether the sentence fragment provided below is related to the instruction. Focus solely on the degree of relatedness in terms of topic, regardless of grammar, coherence, or informativeness.

Begin your evaluation by providing a brief explanation of whether the sentence is related to the instruction, and point out references related to the instruction. After providing your explanation, rate the instruction relevance on a scale from 0 to 2, where 0 indicates the sentence is unrelated to the instruction, 1 indicates it is somewhat related but only minimally or indirectly relevant in terms of topic, and 2 indicates it is more clearly and directly related to the instruction. Provide your rating using this exact format: "Rating: [[score]]".

[Instruction Start]

[Instruction goes here]

[Instruction End]

[Sentence Fragment Start]

[Sentence goes here]

[Sentence Fragment End]

Fluency Score Evaluation Prompt

[System]

Please act as an impartial judge and evaluate the fluency of the sentence fragment provided below. Focus solely on fluency, disregarding its completeness, relevance, coherence with any broader context, or informativeness.

Begin your evaluation by briefly describing the fluency of the sentence, noting any unnatural phrasing, awkward transitions, grammatical errors, or repetitive structures that may hinder readability. After providing your explanation, rate the sentence's fluency on a scale from 0 to 2, where 0 indicates the sentence is not fluent and highly unnatural (e.g., incomprehensible or repetitive), 1 indicates it is somewhat fluent but contains noticeable errors or awkward phrasing, and 2 indicates the sentence is fluent and almost perfect. Provide your rating using this exact format: "Rating: [[score]]".

[Sentence Fragment Start]

[Sentence goes here]

[Sentence Fragment End]

A.2.2 Emotion Steering

Diallo et al. (2025) conduct their steering evaluation using a collection of 19 diverse prompts, listed below:

Evaluation Prompts for Steering Emotions

1. *"Come up with a fairy tale."*
2. *"Comment on a review of a business written by a customer."*
3. *"Compare the cultural value of theatres and cinemas."*
4. *"Compare the quality of coffee and tea."*
5. *"Describe a fictional character."*
6. *"Describe an idea of what a fulfilling and meaningful life should be like."*
7. *"Describe a piece of art."*
8. *"Describe a typical day for a family in the 1980s."*
9. *"Describe an imaginary board meeting at which the concept of PopTarts is proposed."*
10. *"Pick a colour. Write about that colour as if it was a person."*
11. *"Rewrite a famous film scene from the perspective of a fish."*
12. *"Summarise the plot of 'The Lord of the Rings' by J.R.R. Tolkien."*
13. *"Write a couple of lyrics about society."*
14. *"Write a letter to your parents."*
15. *"Write a scene between a mentor and a mentee in which the mentee must stand up for himself or herself because the mentor doesn't give him or her enough credit."*
16. *"Write a story about an encounter of a boy and a girl."*
17. *"Write a story that involves magic."*
18. *"Write a story that starts with 'The moon was bright. Not that the person could see it."*
19. *"Write an opinion on the role of government in society."*

A.3 Bias Mitigation

A.3.1 Prompting Instructions

In Section 4.4 we describe our setup to mitigate the propagation of bias from datasets into our steering vectors. To effectively ban bias, we had to adjust the prompt for our LLM judge picking latents, to clearly discard any food or restaurant related tokens.

In practice, if we did not give explicit instructions for ignoring food and restaurant related tokens, it would often choose latents that promote positive sentiment (target concept) in the food context. E.g. latents that would promote tokens like *delicious*.

Here is the prompt that explicitly instructs the model to discard these types of latents:

LLM judge prompt template

```
"""You are evaluating SAE (Sparse Autoencoder) neuron features based on
↪  their top predicted tokens for the concept: '{target_concept}'.

**CONTEXT**: Features are from Yelp reviews (food/restaurant). To
↪  create a GENERAL vector, we must strip domain bias.

Task: For each neuron index, provide a brief explanation (1-2
↪  sentences) of whether to KEEP or EXCLUDE it, then output the final
↪  list.

General EXCLUSION Criteria - Remove if tokens are:
✗ DOMAIN-SPECIFIC / FOOD BIAS: EXCLUDE words that bind the sentiment to
↪  food (e.g., "delicious", "tasty", "yummy", "flavor") or restaurants
↪  (e.g., "waiter", "kitchen", "service"). Keep ONLY general sentiment
↪  (e.g., "excellent", "amazing").
✗ Purely syntactic (punctuation, formatting, sentence structure,
↪  whitespace) or smileys/emojis
✗ Token boundaries or positional artifacts (prefixes, suffixes,
↪  separators)
✗ Generic stop words or articles with no semantic or structural
↪  relevance to the target concept
✗ Numeric or date formatting without semantic meaning
✗ THEMATICALLY INCOHERENT: the top tokens are unrelated to each other
↪  or to '{target_concept}'

KEEP everything else, including:
✓ Direct expressions of '{target_concept}' that are DOMAIN-AGNOSTIC.
```

```
✓ Contextual or compositional aspects that clearly relate to
↪  {target_concept}. (Phrases, situations, or structural contexts)
✓ Semantic, functional, or structural features clearly related to
↪  '{target_concept}'
✓ THEMATICALLY COHERENT: the top tokens relate to each other and to
↪  '{target_concept}'

Output Format:
First, provide your reasoning for each neuron IN THE ORDER PRESENTED:
Neuron 123: KEEP/EXCLUDE - Brief explanation
...

Then, on a new line after "FINAL_LIST:", provide ONLY a Python list of
↪  integers to KEEP: [123, 456, 789]

Candidates:
{candidates_str}
"""
```

A.3.2 Promoted Tokens

In Section 4.4.2, we inspect the token-level signatures of the steering vectors via our logit-lens projection (Section 3.1.4) to obtain an interpretable preview of what each direction amplifies before analyzing full generations. For completeness, we report here the top 50 promoted tokens for both the baseline (difference-in-means) direction and the direction constructed with our method. For compilation compatibility under pdf LaTeX, we omit tokens that are not representable in Latin script (e.g., CJK, Cyrillic, emoji); the qualitative conclusions are unchanged.

Top promoted tokens (our method).

```
['recommend', 'thank', 'highly', 'excellent', 'recommended', 'nice',
↪  'very', 'great', 'recomendar', 'definitely', 'pleasant',
↪  'recommendation', 'recomend', 'commend', 'recommande', 'love',
↪  'lovely', 'recommendations', 'recommends', 'recommand', 'empfe',
↪  'recommending', 'recomiendo', 'recomendado', 'doporu', 'teşekkür',
↪  'tavsiye', 'reccomend', 'good', 'thanks', 'wonderful', 'recom',
↪  'appreciated', 'anbef', 'commendable', 'kudos', 'raccomand',
↪  'always', 'loved', 'excelente', 'nicely', 'agradable',
↪  'recomendable', 'pleasantly', 'appréci']
```

Top promoted tokens (baseline, DiffMean).

```
['excellent', 'great', 'love', 'wonderful', 'beautiful', 'loved',
↪  'lovely', 'beautifully', 'perfect', 'delicious', 'superb',
↪  'amazing', 'friendly', 'fantastic', 'wonderfully', 'excellently',
↪  'good', 'always', 'nice', 'happy', 'pleasant', 'delightful',
↪  'enjoy', 'highly', 'awesome', 'enjoyed', 'loves', 'loving',
↪  'affordable', 'enjoyable', 'splendid', 'pleasantly', 'very',
↪  'clean', 'staff', 'especially', 'unique', 'fresh', 'rich', 'warm',
↪  'comfortable', 'outstanding', 'brilliant', 'gorgeous', 'tasty',
↪  'charming', 'magnificent', 'welcoming', 'enjoys', 'impeccable']
```

References

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., et al. (2025). Gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Anwar, U., Saparov, A., Rando, J., Paleka, D., Turpin, M., Hase, P., Lubana, E. S., Jenner, E., Casper, S., Sourbut, O., et al. (2024). Foundational challenges in assuring alignment and safety of large language models. *arXiv preprint arXiv:2404.09932*.

Arad, D., Mueller, A., & Belinkov, Y. (2025). Saes are good for steering–if you select the right features. *arXiv preprint arXiv:2505.20063*.

Arditi, A., & RujinChen. (2025). Finding "misaligned persona" features in open-weight models [https://www.lesswrong.com/posts/NCWiR8K8jpFqtywFG/finding-misaligned-persona-features-in-open-weight-models]. *LessWrong*.

Arditi, A., Obeso, O., Syed, A., Paleka, D., Panickssery, N., Gurnee, W., & Nanda, N. (2024). Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, *37*, 136037–136083.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. (2022). Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6541–6549.

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 610–623.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., & Saunders, W. (2023). Language models can explain neurons in language models.

Bloom, J., Tigges, C., Duong, A., & Chanin, D. (2024). Saelens.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., . . . Olah, C. (2023). Towards monosemanticity: Decomposing language models with dictionary learning [https://transformer-circuits.pub/2023/monosemantic-features/index.html]. *Transformer Circuits Thread*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Bussmann, B., Leask, P., & Nanda, N. (2024). Batchtopk: A simple improvement for topk-saes [https://www.alignmentforum.org/posts/Nkx6yWZNbAsfvic98/batchtopk-a-simple-improvement-for-topk-saes]. *AI Alignment Forum*.

Chalvatzaki, G., Younes, A., Nandha, D., Le, A., Ribeiro, L. F. R., & Gurevych, I. (2023). Learning to reason over scene graphs: A case study of finetuning gpt-2 into a robot language model for grounded task planning. *Frontiers in Robotics and AI*, *10*. https://doi.org/10.3389/frobt.2023.1221739

Chanin, D., Wilken-Smith, J., Dulka, T., Bhatnagar, H., Golechha, S., & Bloom, J. (2024). A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *arXiv preprint arXiv:2409.14507*.

Chen, M. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, R., Arditi, A., Sleight, H., Evans, O., & Lindsey, J. (2025). Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint arXiv:2507.21509*.

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., & Sharkey, L. (2023). Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., & Ravi, S. (2020). Goemotions: A dataset of fine-grained emotions. *arXiv preprint arXiv:2005.00547*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186.

Diallo, D., Dworatzyk, K., Jentzsch, S., Schütt, P., Theis, S., & Hecking, T. (2025). The effectiveness of style vectors for steering large language models: A human evaluation. *IEEE Access*, *13*, 191443–191457.

Durmus, E., Tamkin, A., Clark, J., Wei, J., Marcus, J., Batson, J., Handa, K., Lovitt, L., Tong, M., McCain, M., Rausch, O., Huang, S., Bowman, S., Ritchie, S., Henighan, T., & Ganguli, D. (2024, October 25). *Evaluating feature steering: A case study in mitigating social biases*. https://anthropic.com/research/evaluating-feature-steering

Ekman, P. (1992). Are there basic emotions?

Eleuther. (2024). Multitopk saes [https://huggingface.co/EleutherAI/sae-llama-3.1-8b-64x]. *Huggingface*.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., & Olah, C. (2022). Toy models of superposition [https://transformer-circuits.pub/2022/toy$_{m}odel/index.html$]. *Transformer Circuits Thread*.

Elhage, N., Lasenby, R., & Olah, C. (2023). Privileged bases in the transformer residual stream [https://transformer-circuits.pub/2023/privileged-basis/index.html]. *Transformer Circuits Thread*.

Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., & Smith, N. A. (2015). Sparse overcomplete word vector representations. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Inter-*

national Joint Conference on Natural Language Processing (Volume 1: Long Papers), 1491–1500.

Gallegos, I. O., Rossi, R. A., Barrow, J., Tanjim, M. M., Kim, S., Dernoncourt, F., Yu, T., Zhang, R., & Ahmed, N. K. (2024). Bias and fairness in large language models: A survey. *Computational linguistics*, *50*(3), 1097–1179.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., & Wu, J. (2024). Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*.

Gehman, S., Gururangan, S., Sap, M., Choi, Y., & Smith, N. A. (2020). Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*.

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*.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *nature*, *585*(7825), 357–362.

Hartmann, J. (2022). Emotion english distilroberta-base.

He, Z., Shu, W., Ge, X., Chen, L., Wang, J., Zhou, Y., Liu, F., Guo, Q., Huang, X., Wu, Z., et al. (2024). Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *arXiv preprint arXiv:2410.20526*.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Joshi, M., Choi, E., Weld, D. S., & Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Konen, K., Jentzsch, S., Diallo, D., Schütt, P., Bensch, O., Baff, R. E., Opitz, D., & Hecking, T. (2024). Style vectors for steering generative large language model. *arXiv preprint arXiv:2402.01618*.

Kotek, H., Dockum, R., & Sun, D. (2023). Gender bias and stereotypes in large language models. *Proceedings of the ACM collective intelligence conference*, 12–24.

Krause, B., Gotmare, A. D., McCann, B., Keskar, N. S., Joty, S., Socher, R., & Rajani, N. F. (2021). Gedi: Generative discriminator guided sequence generation. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 4929–4952.

Lee, B. W., Padhi, I., Ramamurthy, K. N., Miehling, E., Dognin, P., Nagireddy, M., & Dhurandhar, A. (2024). Programming refusal with conditional activation steering. *arXiv preprint arXiv:2409.05907*.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., & Hashimoto, T. B. (2023, May). *AlpacaEval: An Automatic Evaluator of Instruction-following Models*.

Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramár, J., Dragan, A., Shah, R., & Nanda, N. (2024). Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*.

Lin, J. (2023). Neuronpedia: Interactive reference and tooling for analyzing neural networks [Software available from neuronpedia.org]. https://www.neuronpedia.org

Liu, A., Sap, M., Lu, X., Swayamdipta, S., Bhagavatula, C., Smith, N. A., & Choi, Y. (2021). Dexperts: Decoding-time controlled text generation with experts and anti-experts. *arXiv preprint arXiv:2105.03023*.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. (2023). Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, *36*, 46534–46594.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., & Mueller, A. (2024). Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*.

Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., & Hendrycks, D. (2024). Harmbench: A standardized evaluation framework for automated red teaming and robust refusal.

McKinney, W., et al. (2011). Pandas: A foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, *14*(9), 1–9.

Mikolov, T., Yih, W.-t., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 746–751.

Nangia, N., Vania, C., Bhalerao, R., & Bowman, S. (2020). Crows-pairs: A challenge dataset for measuring social biases in masked language models. *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, 1953–1967.

nostalgebraist. (2020). Interpreting gpt: The logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens

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.

Panickssery, A., Bowman, S. R., & Feng, S. (2024). Llm evaluators recognize and favor their own generations. *URL https://arxiv. org/abs/2404.13076*.

Park, K., Choe, Y. J., & Veitch, V. (2023). The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

Paulo, G., Mallen, A., Juang, C., & Belrose, N. (2024). Automatically interpreting millions of features in large language models. *URL https://arxiv. org/abs/2410.13928*.

Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., & Irving, G. (2022). Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J., & Nanda, N. (2024). Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*.

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

Shen, T., Lei, T., Barzilay, R., & Jaakkola, T. (2017). Style transfer from non-parallel text by cross-alignment. *Advances in neural information processing systems*, *30*.

Subramani, N., Suresh, N., & Peters, M. E. (2022). Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*.

Subramanian, A., Pruthi, D., Jhamtani, H., Berg-Kirkpatrick, T., & Hovy, E. (2018). Spine: Sparse interpretable neural embeddings. *Proceedings of the AAAI conference on artificial intelligence*, *32*(1).

Sun, J., Baskaran, S., Wu, Z., Sklar, M., Potts, C., & Geiger, A. (2025). Hypersteer: Activation steering at scale with hypernetworks. *arXiv preprint arXiv:2506.03292*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., et al. (2025). Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. (2024). Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. (2024). Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., ... Henighan, T. (2024). Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*. https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Mini, U., & MacDiarmid, M. (2024). Activation addition: Steering language models without optimization.

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

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., & Shieber, S. (2020). Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, *33*, 12388–12401.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019). Superglue: A stickier benchmark for general-purpose

language understanding systems. *Advances in neural information processing systems*, *32*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, 353–355.

Wang, M., Xu, Z., Mao, S., Deng, S., Tu, Z., Chen, H., & Zhang, N. (2025). Beyond prompt engineering: Robust behavior control in llms via steering target atoms. *arXiv preprint arXiv:2505.20322*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, 24824–24837.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Wu, Z., Arora, A., Geiger, A., Wang, Z., Huang, J., Jurafsky, D., Manning, C. D., & Potts, C. (2025). Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *arXiv preprint arXiv:2501.17148*.

Wu, Z., Geiger, A., Arora, A., Huang, J., Wang, Z., Goodman, N., Manning, C. D., & Potts, C. (2024). Pyvene: A library for understanding and improving pytorch models via interventions. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, 158–165.

Wu, Z., Yu, Q., Arora, A., Manning, C. D., & Potts, C. (2025). Improved representation steering for language models. *arXiv preprint arXiv:2505.20809*.

Yu, J., Wu, Y., Shu, D., Jin, M., Yang, S., & Xing, X. (2023). Assessing prompt injection risks in 200+ custom gpts. *arXiv preprint arXiv:2311.11538*.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Zhao, J., Wang, T., Yatskar, M., Ordonez, V., & Chang, K.-W. (2018). Gender bias in coreference resolution: Evaluation and debiasing methods. *arXiv preprint arXiv:1804.06876*.

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

Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., & Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. *URL https://arxiv. org/abs/2307.15043*, *19*, 3.