The Anatomy of Alignment: Decomposing Preference Optimization by Steering Sparse Features

Jeremias Ferrao[*]
University of Groningen

Matthijs van der Lende
University of Groningen

Ilija Lichkovski
AI Safety Initiative Groningen

Clement Neo
Apart Research

Abstract

Prevailing alignment methods induce opaque parameter changes, making it difficult to audit what the model truly learns. To address this, we introduce Feature Steering with Reinforcement Learning (FSRL), a framework that trains a lightweight adapter to steer model behavior by modulating interpretable sparse features. First, we theoretically show that this mechanism is principled and expressive enough to approximate the behavioral shifts of post-training processes. Then, we apply this framework to the task of preference optimization and perform a causal analysis of the learned policy. We find that the model relies on stylistic presentation as a proxy for quality, disproportionately steering features related to style and formatting over those tied to alignment concepts like honesty. Despite exploiting this heuristic, FSRL proves to be an effective alignment method, achieving a substantial reduction in preference loss. Overall, FSRL offers an interpretable control interface and a practical way to diagnose how preference optimization pressures manifest at the feature level.

1 Introduction

Large Language Models (LLMs) are typically aligned with human preferences through post-training methods like Reinforcement Learning from Human Feedback (RLHF) (1). This fine-tuning induces parameter updates across the model's underlying weights. Consequently, the newly learned alignment behaviors and the model's original capabilities are encoded in the same parameters, making them difficult to disentangle. When models trained with RLHF subsequently exhibit undesirable behaviors like sycophancy or reward hacking (2; 3), identifying their root cause becomes challenging. This opacity motivates the need for alignment methods that are not only effective, but also transparent and auditable.

Mechanistic interpretability offers a way to make alignment more transparent by exposing and manipulating a model's internal concepts. At its core is the *Linear Representation Hypothesis*, which suggests that high-level concepts correspond to linear directions in activation space (4). Sparse Autoencoders (SAEs) provide a practical method for uncovering these directions by decomposing dense activations into a sparse basis of largely monosemantic features (5; 6). These features capture diverse phenomena, ranging from "code syntax" to "flattery", and can often be assigned interpretable labels using automated methods (5; 7; 8). The resulting feature vocabulary enables not only analysis of what models represent, but also a potential interface for directly steering their behavior.

[*]Email for Correspondence: j.lino.ferrao@gmail.com

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability.

Building on this foundation, we propose **Feature Steering with Reinforcement Learning (FSRL)**, a framework that uses the interpretable feature vocabulary in SAEs as a direct interface for alignment. Instead of fine-tuning the entire model, FSRL operates on a frozen LLM together with its corresponding SAE, and trains a lightweight adapter with reinforcement learning to learn a policy for modulating SAE features, as illustrated in Figure 1. This design keeps the model's underlying capabilities intact in the frozen LLM, while channeling the learned alignment behavior through steering interpretable SAE features.

Contributions Building on this foundation, we introduce Feature Steering with Reinforcement Learning (FSRL), a framework that trains a lightweight adapter to modulate interpretable SAE features of a frozen LLM for alignment. First, we theoretically establish that FSRL's activation-space corrections are functionally equivalent to a restricted class of LoRA updates, inheriting LoRA's expressive power guarantees while operating through an interpretable basis. Empirically, we show that FSRL successfully optimizes the preference objective on UltraFeedback, reducing SimPO loss from 6.99 to 2.6 while better preserving mathematical reasoning than full fine-tuning. Lastly, we analyse the learned policy with causal experiments and find that style features are over four times more important per feature than alignment features for performance, providing mechanistic evidence that preference optimization treats stylistic presentation as a proxy for quality rather than prioritizing concepts like safety or honesty. These results establish FSRL as a tool for both lightweight model control and for diagnosing the internal mechanisms of alignment.

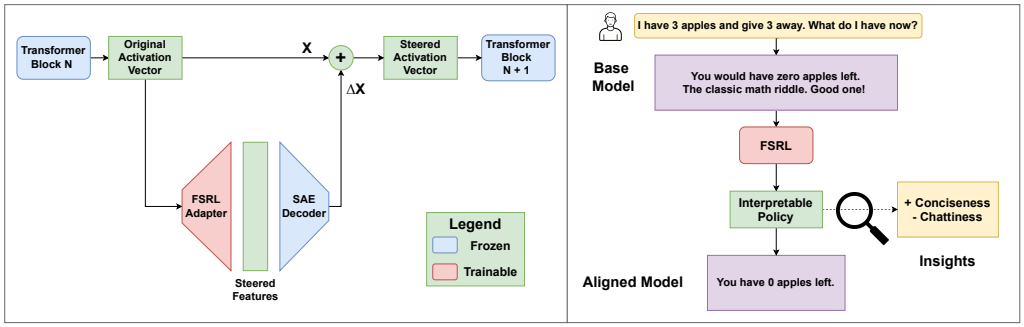

Figure 1: **The FSRL Framework for Interpretable Alignment. (a) FSRL Architecture:** At a given layer, the original activation vector is processed by a trainable adapter. The adapter outputs a sparse vector of steered features, which are transformed by a frozen SAE decoder into a correction vector. This correction is added to the original activation to steer the model's behavior. **(b) Application for Mechanistic Insight:** FSRL replaces opaque alignment processes with a transparent one by learning a policy over a basis of interpretable, monosemantic SAE features. This allows the learned alignment pressures to be decomposed into concrete actions on meaningful concepts.

2 Background

We build on three key components: Sparse Autoencoders (SAEs) for creating an interpretable interface, Simple Preference Optimization (SimPO) to optimize a policy on a preference dataset, and a large annotated dataset to train our system.

Sparse Autoencoders (SAEs) SAEs are an unsupervised method for representing model activations as a sparse set of interpretable features (5; 9). Each SAE consists of an encoder and a decoder. Given a model's hidden activation $\mathbf{x} \in \mathbb{R}^d$, the encoder first maps it into a higher-dimensional feature vector $\mathbf{f} \in \mathbb{R}^{d_{\text{sae}}}$ with $d_{\text{sae}} > d$:

$$\mathbf{f} = \text{ReLU}(W_{\text{enc}}\mathbf{x} + \mathbf{b}_{\text{enc}}), \tag{1}$$

where $W_{\text{enc}} \in \mathbb{R}^{d_{\text{sae}} \times d}$ and $\mathbf{b}_{\text{enc}}$ are encoder parameters. The decoder then reconstructs the original activation from $\mathbf{f}$:

$$\hat{\mathbf{x}} = W_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{dec}}, \tag{2}$$

where $W_{\text{dec}} \in \mathbb{R}^{d \times d_{\text{sae}}}$ and $\mathbf{b}_{\text{dec}}$ are decoder parameters. The columns of W_{dec} form a dictionary of learned feature vectors. In particular, SAEs are trained such that each activation can be decomposed into only a few features, achieved by adding adding an ℓ_1 penalty to the reconstruction loss. The total loss function is therefore:

$$\mathcal{L}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \alpha\|\mathbf{f}\|_1, \tag{3}$$

where α is a hyperparameter that controls the trade-off between reconstruction fidelity and feature sparsity. While this formulation is common, other SAE variants achieve sparsity through different mechanisms, such as the JumpReLU activation function (6) or the Top-K operator (10).

SAE features can also be used for intervention. As each feature corresponds to a direction given by a column of W_{dec}, modifying an activation $\mathbf{x}$ by $\mathbf{x}' = \mathbf{x} + \lambda W_{\text{dec}}^{(i)}$ can steer the model's behavior in predictable ways. This property, known as *feature steering*, highlights that SAEs features are not only descriptive, but can also be used as actionable controls on model behavior.

Simple Preference Optimization (SimPO) SimPO is an efficient algorithm for aligning language models with human preferences (11). It operates directly on a dataset $\mathcal{D}$ of preference triplets (x, y_w, y_l), where x is a prompt, y_w is the preferred (chosen) response, and y_l is the less preferred (rejected) response.

The objective is a modified Bradley-Terry loss with a target reward margin γ, which encourages the model to confidently separate y_w and y_l:

$$\mathcal{L}_{\text{SimPO}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\frac{\beta}{|y_w|}\log\pi_\theta(y_w|x) - \frac{\beta}{|y_l|}\log\pi_\theta(y_l|x) - \gamma\right)\right], \tag{4}$$

where β is the temperature/scaling parameter, $|y|$ the sequence length and $\sigma(\cdot)$ the sigmoid function.

We adopt SimPO for its ability to match the performance of Direct Preference Optimization (DPO) (12) without requiring a separate reference model. This makes it possible to efficiently train the model (or FSRL adapter) directly on a preference dataset.

Preference Dataset In this work, we use the UltraFeedback dataset (13). Specifically, we utilize the version of this dataset annotated with the Absolute-Rating Multi-Objective Reward Model framework (14). Our choice of this dataset is motivated by its use in the SimPO paper, which allows for a direct comparison, isolating the impact of our proposed FSRL framework rather than confounding it with dataset variations.

3 Methodology

We present Feature Steering with Reinforcement Learning (FSRL), a framework for transparently aligning LLMs by training a policy to steer sparse SAE features of a frozen model. In this section, we describe the system architecture, the training procedure, and the experimental configuration used for evaluation.

3.1 System Architecture

FSRL intervenes at a single chosen layer of a frozen LLM by steering the residual stream with a sparse, learned set of feature directions (Figure 1). At this layer, the residual activation $\mathbf{x} \in \mathbb{R}^d$ is first translated by the SAE into a sparse feature vector $\mathbf{f} \in \mathbb{R}^{d_{\text{sae}}}$. To decide how these features should be modulated, the same $\mathbf{x}$ is also given to a trainable adapter π_ϕ, which outputs a sparse steering vector $\mathbf{v} \in \mathbb{R}^{d_{\text{sae}}}$. In effect, π_ϕ learns both the subset of features to target, as well as the direction and magnitude in which to steer them.

Adapter Implementation We implement the adapter as a single feedforward layer with parameters $\phi = (W_a, \mathbf{b}_a, \boldsymbol{\theta})$, where $W_a \in \mathbb{R}^{d_{\text{sae}} \times d}$, $\mathbf{b}_a \in \mathbb{R}^{d_{\text{sae}}}$, and $\boldsymbol{\theta} \in \mathbb{R}_+^{d_{\text{sae}}}$ is a vector of learnable positive thresholds. Its output is produced by a coordinate-wise soft-thresholding activation function:

$$\mathbf{v} = \pi_\phi(\mathbf{x}) = \text{sign}(W_a\mathbf{x} + \mathbf{b}_a)\text{ReLU}(|W_a\mathbf{x} + \mathbf{b}_a| - \boldsymbol{\theta}). \tag{5}$$

We adapt this activation function from learned approximations of sparse coding (15). Unlike a standard ReLU, this function enables a tri-state intervention that improves interpretability: positive

values amplify a feature, negative values suppress it, and values in the dead zone between $-\theta_i$ and $+\theta_i$ leave the feature unchanged. This flexibility also allows the adapter to achieve its objective with a sparser steering vector. The function is non-differentiable within its dead zone; however, this is handled implicitly via subgradients in modern deep learning libraries.

Applying Steering The steering vector $\mathbf{v}$ specifies how SAE features are modulated. We obtained the steered activation by adding the decoded steering adjustment back into the residual stream:

$$\mathbf{x}_{\text{steered}} = \mathbf{x} + \text{Decoder}(\mathbf{v}). \tag{6}$$

Hence, given the input activation, the adapter learns to output a steering vector $\mathbf{v}$ that steers the model's output to be better aligned with the preference objective. In practice, we implement the update using a reconstruction-error variant (see Appendix A).

3.2 Theoretical Justification

While FSRL can align models with the training objective in practice, it is important to establish why its restricted form of intervention should, in principle, be expressive enough to match other fine-tuning methods. To this end, our theoretical justification shows that FSRL is a principled approach by demonstrating its functional equivalence to a restricted, yet powerful, class of low-rank adaptation (LoRA) updates (16). While FSRL's practical effectiveness is contingent on the capacity of its underlying SAE, our theory shows that its adaptation mechanism is sound.

The core of our proof, detailed in Appendix B, is that FSRL's activation-space corrections are functionally equivalent to a class of input-dependent LoRA updates. The FSRL update, $\mathbf{x}_{\text{steered}} = \mathbf{x} + \Delta(\mathbf{x})$, injects an additive correction into the residual stream. When passed to a downstream linear layer, this is algebraically equivalent to applying an effective weight update, $\Delta W[\mathbf{x}]$, whose rank is dynamically determined by the number of actively steered SAE features.

This equivalence is significant because it connects FSRL to the established foundations of LoRA. Recent work by Zeng and Lee (17) proved that LoRA possesses sufficient expressive power to match a target model, given enough rank. While FSRL inherits these guarantees in principle, our single-layer intervention is a constrained application of this theory. Specifically, the adapter's policy is conditioned only on the activation at one layer, meaning it cannot distinguish between different upstream computational paths that yield the same activation vector. Despite this limitation, the connection confirms FSRL as a valid optimization method. Crucially, because FSRL is constrained to express its policy through the SAE's interpretable basis, the policy it learns provides a robust and transparent reflection of the optimization pressures driving the alignment task.

3.3 Training Configuration

The adapter's parameters are optimized using the SimPO algorithm (11). To encourage a sparse and interpretable policy, we augment the training objective with an ℓ_1 penalty on the steering vector, controlled by a coefficient α. In addition to this proxy-based sparsity, we also investigated a more direct method using a JumpReLU activation (6) in the adapter to directly optimize the ℓ_0 norm. However, this proved to be difficult to tune within our framework.

We evaluate our approach on the **Gemma-2-2B-it** model (18) using pre-trained SAEs from GemmaScope (19). For training, we use the **UltraFeedback dataset** (13) as described in Section 2. Our primary experimental decisions involved selecting the intervention layer and the sparsity coefficient. We first performed a sweep across the transformer's quartiles (layers 6, 12, 18, and 24), hypothesizing that mid-model layers would contain the most relevant abstract concepts for alignment. This sweep confirmed layer 12 as optimal. Subsequently, we swept through a range of α values on layer 12 to identify an elbow point that balanced steering vector sparsity with performance on the SimPO validation loss. This led us to select $\alpha = 1 \times 10^{-1}$. The detailed methodology and results of these sweeps are presented in Appendix C, with our investigation into the JumpReLU adapter detailed in Appendix E. The final training configuration and environment details are provided in Appendix D.

3.4 Comparative Evaluation

To contextualize the performance of our FSRL-steered model, we establish a baseline for comparison. The original SimPO paper reports results on a Gemma-9B model; a direct comparison would be

inappropriate due to the difference in model scale. Therefore, to ensure a controlled comparison, we trained our own baseline. This baseline consists of the same instruction-tuned model, but is fully fine-tuned using the standard SimPO algorithm rather than our lightweight adapter. The training configuration for this baseline mirrors that of our FSRL adapter, where applicable, with a decrement in the learning rate to ensure stable convergence during full-model training (see Appendix D).

4 Evaluating Alignment Efficacy

To evaluate FSRL, we compare its performance on standard benchmarks against two models: the base Gemma-2-2B-it model and the same model fully fine-tuned with the SimPO algorithm. The fully fine-tuned model represents the standard, non-interpretable approach to alignment. The goal of this evaluation is not to demonstrate that FSRL can match the performance of full fine-tuning, but to show that it is an effective method for improving preference scores despite operating through a constrained, interpretable interface.

We assess performance on MMLU (20) for general knowledge, TruthfulQA (21) for truthfulness, and GSM8K (22) for mathematical reasoning. Evaluations were performed using the Language Model Evaluation Harness (23). The results are presented in Table 1.

Table 1: Benchmark performance. FSRL successfully optimizes the preference objective while substantially mitigating the reasoning degradation seen in full fine-tuning.

Model	MMLU ↑	TruthfulQA (MC2) ↑	GSM8K ↑	SimPO Loss ↓
Base (Instruct)	30.11 ± 0.38	55.77 ± 1.58	$\mathbf{53.45 \pm 1.37}$	6.99
SimPO Full	$\mathbf{50.28 \pm 0.40}$	$\mathbf{61.35 \pm 1.63}$	4.40 ± 0.56	**2.19**
FSRL	34.46 ± 0.39	56.17 ± 1.63	44.05 ± 1.37	2.6

As illustrated in the table, both alignment methods successfully reduce the SimPO loss, confirming they optimize the preference objective. The performance changes align with the patterns documented by Meng et al. (11), where SimPO tuning is expected to improve scores on benchmarks like TruthfulQA while degrading mathematical reasoning on GSM8K. Our FSRL-steered model follows these trends, validating its efficacy as an alignment method.

The comparison between FSRL and full fine-tuning illustrates a trade-off. The fully fine-tuned model achieves a lower preference loss, corresponding to larger gains on MMLU and TruthfulQA, but also exhibits a significant degradation in reasoning capability. FSRL presents a different point on this trade-off spectrum: its gains on MMLU and TruthfulQA are more moderate, but it better preserves the model's mathematical reasoning capabilities. The key result is that FSRL is an effective alignment method that provides mechanistic transparency.

Comparison with ReLU adapter Naively, one might use ReLU with the same sparsity constraint (similar to SAEs) to induce sparsity in the steering vector. However, ReLU is not well-suited for the adapter, as it is limited to amplifying features and therefore cannot apply negative steering. To make this difference concrete, we trained a ReLU-based adapter where ReLU was used both at training and test time. As detailed in Appendix F, while this variant can learn a policy, it is less expressive and performs worse than our soft-thresholding approach, highlighting why our activation function is better suited for feature-level steering.

Comparison with static top-k sparsity To explore how much of the adapter's performance gain comes from the activation function itself, we compared it against a naive top-k% sparsity heuristic applied at test time, retaining only the features with the largest absolute magnitudes. Appendix G shows that the dynamic adapter has both lower preference loss and greater sparsity.

5 Mechanistic Insights into the Alignment Process

Having established FSRL as an effective alignment method, we now leverage its primary advantage: interpretability. To analyze the policy at a conceptual level, we developed an automated pipeline to classify SAE features based on their text-based explanations. We focus on two primary categories:

alignment features, which encompass abstract concepts such as ethics, safety, and honesty, and style features, which relate to text structure, punctuation, and formatting. This automated process was validated against manual annotations, achieving a Matthews Correlation Coefficient of 0.448 for alignment features and 0.764 for style features, indicating a reliable agreement with human judgment (details in Appendix I).

Examining Feature Activations To understand how the adapter uses different types of features, we examine the composition of its feature activations. A simple raw count of active features is misleading. The FSRL adapter outputs a steering vector with an average ℓ_0 norm of approximately 360, far denser than the SAE's baseline of 21. With such a large increase in total activations, the raw count of features from any given category would be artificially inflated, as more noisy or common features would fire purely by chance, obscuring the adapter's actual steering intent. To create a robust metric that corrects for this effect, we therefore analyze the composition of the active feature set at each token: the proportion of active features belonging to a given category.

We measured this composition across three distinct contexts from our preference dataset: activations from the input prompt tokens, from the tokens of chosen responses, and from the tokens of rejected responses. As summarized in Table 2, this analysis reveals a consistent strategy: the adapter learns to decrease the proportional activation of both alignment and style features relative to the baseline. This pattern's uniformity across all three conditions suggests the learned policy is general, applying the same high-level strategy regardless of whether it is generating a winning or losing continuation. This behavior contrasts with that of a simpler, amplification-only ReLU-based adapter, which instead learns to increase the proportional activation of style features by 2-4% (see Appendix F). This dependency on adapter architecture highlights the need for causal analysis.

Table 2: Aggregate steering effect of the soft-threshold adapter on the composition of active features. 'SAE Baseline' is the average proportion of active features belonging to a category for the unmodified model. 'Relative Change' is the percent change in this proportion caused by the FSRL adapter.

Feature Type	Data	SAE Baseline (%)	Relative Change (%)
Alignment	Prompt + Chosen	17.59 ± 0.03	-4.77 ± 0.25
	Prompt Only	17.10 ± 0.05	-3.60 ± 0.45
	Prompt + Rejected	17.53 ± 0.03	-4.64 ± 0.25
Style	Prompt + Chosen	24.43 ± 0.04	-3.09 ± 0.20
	Prompt Only	25.25 ± 0.06	-2.38 ± 0.35
	Prompt + Rejected	24.51 ± 0.03	-3.02 ± 0.20

Intervening on Feature Activations To causally evaluate how different feature categories contribute to the model's performance, we turn to an ablation study. For each category (e.g., style features), we disable the adapter's intervention by setting the corresponding components of its output steering vector to zero. We then measure the impact on the SimPO loss—the ground-truth objective the adapter was trained to minimize. Under a null hypothesis where all features contribute equally, one would expect the loss to linearly increase in proportion to the number of features ablated. Our results in Table 3 deviate sharply from this expectation, revealing a clear hierarchy of feature importance.

Table 3: Causal contribution of feature categories, measured by ablating all steering interventions on features within a given category. The 'Features Ablated' column lists the total number of features belonging to that category. 'Loss per Feature' normalizes the resulting increase in SimPO loss by this count to quantify per-feature impact.

Ablation Condition	Features Ablated	SimPO Loss ↓	Loss per Feature
None (Full Steering)	0	2.60	–
Alignment Features	11,143	2.88	2.51×10^{-5}
Style Features	15,391	4.21	1.05×10^{-4}
Both Categories	26,534	5.60	1.13×10^{-4}

The Loss per Feature column quantifies the disproportionate impact of each category. The average loss increase per style feature is over four times greater than that of an alignment feature, providing causal evidence that the policy prioritizes the manipulation of style features to achieve its objective. Furthermore, we observe a significant non-linear interaction: ablating both categories simultaneously results in a performance drop exceeding the sum of the individual ablations. This suggests a degree of entanglement between the model's representations of style and alignment.

This causal finding offers a direct mechanistic explanation for recent observations that chatbot rankings are heavily influenced by stylistic factors (24). Our work reveals how this phenomenon is encoded at a feature level: the alignment policy learns that precise control over style is causally necessary to maximize the reward signal. While analysis of individual features provides some insight (see Appendix K), the learned policy is highly distributed. Indeed, the usage frequency of steered features follows a long-tail distribution, confirming that the adapter relies on a broad set of features rather than a few specific interventions (see Appendix H). Therefore, the aggregate causal analysis provides a clearer picture of the strategy learned during preference optimization.

6 Discussion

Our work introduces FSRL, an interpretable alignment framework that uses a lightweight adapter to steer a model's conceptual features. This general approach can diagnose the mechanisms of any post-training process, replacing opaque parameter deltas with a transparent policy. It enables researchers to causally link undesirable behaviors to the optimization of specific feature categories, thereby clarifying their root causes.

Our findings offer a mechanistic explanation for Goodhart's Law within the context of preference optimization. The pressure to satisfy the preference objective incentivizes the model to treat stylistic polish as a proxy for overall response quality. Our causal analysis reveals the policy learns that manipulating features related to presentation is a more effective strategy for minimizing preference loss than steering concepts like honesty. This highlights how simple preference signals can inadvertently reward surface-level characteristics over the intended, deeper qualities of a response.

FSRL also presents an efficient alternative to model-diffing, the practice of analyzing internal differences between a base and a fine-tuned model, by directly addressing its key methodological challenge: feature stability. The transferability of SAEs is not guaranteed for instruction-tuned models (25), particularly for specialized reasoning models that develop novel features (26). By design, FSRL sidesteps this issue entirely by operating on a fixed, interpretable feature basis. This stable foundation, in turn, is what enables direct causal analysis of the learned policy, allowing for targeted ablations to determine which features are causally important for the task. While this prevents the discovery of emergent concepts, it provides a controlled framework for auditing alignment pressures.

6.1 Limitations

Our approach's primary limitation is its dependence on the quality and nature of the underlying SAEs. The extent to which SAE features represent true learned computations versus artifacts is an active area of research (27). We mitigate this by using high-quality public SAEs from GemmaScope, though the generalizability of any specific feature vocabulary remains an open question.

Furthermore, our analysis is confined to a 2B parameter model, as scaling FSRL faces practical hurdles. Extending this work to larger models is challenging due to library limitations for dynamic model intervention, as well as the high computational cost of training quality SAEs and obtaining reliable feature explanations. This resource bottleneck extends to our analysis, where our causal claims are mediated by an LLM-based classifier with moderate human agreement, introducing a layer of approximation.

Finally, our analysis is conducted exclusively on a single-layer intervention. While our theoretical grounding in LoRA's expressive power is important, the guarantees from cited work (17) suggest a worst-case need for adaptation across all layers. However, this requirement is most stringent for randomly initialized models. Our empirical results provide strong evidence that for a structured, pre-trained LLM, this constraint is not a practical barrier, as FSRL successfully optimizes the preference objective.

6.2 Future Work

These limitations point toward several avenues for future work. A key direction is to explore the scaling properties of this approach, testing the hypothesis that higher-dimensional SAEs yield a more disentangled and controllable feature basis. This exploration should also include alternative interfaces beyond SAEs, such as Transcoders, which may offer a more direct way to control MLP computations (28). Scaling the feature interface will also require scaling the analysis pipeline, for which unsupervised methods like embedding and clustering feature explanations could provide a more efficient alternative to our LLM-based classification.

Finally, a crucial direction is to empirically compare FSRL with the alternative of interpretable model-diffing. Such a study could quantify FSRL's efficiency gains and, more importantly, test the fundamental trade-off between the methodological stability of a fixed conceptual vocabulary and the ability of a new SAE to discover emergent features that arise during alignment. This would help establish practical guidelines for when each transparency method is most appropriate.

7 Related Work

Steering Opaque Activations: FSRL builds on a line of work that steers model behavior by modifying internal activations at inference time. These methods range from applying static activation vectors (29; 30) to learning more complex policies. For example, some approaches use reinforcement learning to guide generation via value functions, as in Successive Policy Iterations (SPI) (31), or use preference optimization to learn an optimal static vector, as in BiPO (32). A common thread unites these methods: they intervene on the model's dense, opaque activation space, making the precise mechanism of control difficult to interpret.

Interpretable Steering with Sparse Features: SAEs offer a solution to this opacity, providing an interpretable feature basis for steering. SAE-Targeted Steering (SAE-TS) exemplifies this, using causal analysis of SAE features to construct an optimized static steering vector that predictably targets a desired feature while minimizing side effects (33). FSRL extends this interpretable approach by replacing the static vector with a dynamic, context-aware policy. By training a lightweight adapter to modulate SAE features, FSRL is designed to decompose an alignment goal into a transparent set of token-level interventions. Crucially, while our work uses a preference objective, the FSRL framework is general: the adapter can be trained with any differentiable objective, making it a flexible tool for auditing the mechanisms of various post-training processes.

8 Conclusion

Standard alignment methods induce opaque parameter changes, obscuring what models truly learn. To dissect these mechanisms, we introduced FSRL, a framework that aligns models by training a lightweight adapter to steer interpretable SAE features. Our causal analysis yields a crucial insight: preference optimization learns to reward stylistic presentation as a proxy for quality, disproportionately relying on features related to style over those tied to alignment concepts like honesty. Despite operating through this constrained interface, FSRL proves to be an effective alignment method.

Ultimately, FSRL demonstrates that effective alignment and mechanistic interpretability are not mutually exclusive goals. By shifting intervention from the opaque parameter space to a transparent feature space, our work provides both a practical method for lightweight model control and, more importantly, a powerful scientific instrument for auditing the internal mechanisms of alignment. This approach moves LLM alignment toward a more transparent and debuggable engineering discipline.

Reproducibility Statement

To ensure the reproducibility of our findings, we provide our source code, which includes the implementation of the FSRL framework and training scripts: `https://github.com/Jazhyc/feature-steering-RL`.

Our experiments were conducted using the Gemma-2-2B-it base model and publicly available SAEs from GemmaScope. The adapter was trained on the UltraFeedback dataset. Our software stack is built

on PyTorch and utilizes the `transformer-lens`, `sae-lens`, and `TRL` libraries. All experiments were performed on a single NVIDIA GH200 GPU. Full training configurations, hyperparameter details, and library versions are provided in Appendix D.

References

[1] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL http://arxiv.org/abs/2203.02155.

[2] Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger B. Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. Discovering Language Model Behaviors with Model-Written Evaluations. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13387–13434. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.847. URL https://doi.org/10.18653/v1/2023.findings-acl.847.

[3] Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal Misgeneralization: Why Correct Specifications Aren't Enough For Correct Goals, 2022. URL http://arxiv.org/abs/2210.01790.

[4] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy Models of Superposition, 2022. URL https://arxiv.org/abs/2209.10652v1.

[5] Robert Huben, Hoagy Cunningham, Logan Riggs, Aidan Ewart, and Lee Sharkey. Sparse Autoencoders Find Highly Interpretable Features in Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

[6] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping Ahead: Improving Reconstruction Fidelity with JumpReLU Sparse Autoencoders, 2024. URL http://arxiv.org/abs/2407.14435.

[7] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

[8] Gonçalo Santos Paulo, Alex Troy Mallen, Caden Juang, and Nora Belrose. Automatically Interpreting Millions of Features in Large Language Models. 2025. URL https://openreview.net/forum?id=EemtbhJOXc.

[9] Anthropic. Towards monosemanticity: Decomposing language models with dictionary learning. October 2023. Accessed: 2025-08-22.

[10] Bart Bussmann, Patrick Leask, and Neel Nanda. BatchTopK Sparse Autoencoders, 2024. URL http://arxiv.org/abs/2412.06410.

[11] Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple Preference Optimization with a Reference-Free Reward. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. arXiv, 2024. URL https://arxiv.org/abs/2405.14734.

[12] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, 2024. URL http://arxiv.org/abs/2305.18290.

[13] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. UltraFeedback: Boosting Language Models with Scaled AI Feedback, 2024. URL http://arxiv.org/abs/2310.01377.

[14] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts, 2024. URL http://arxiv.org/abs/2406.12845.

[15] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 399–406, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[16] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021. URL http://arxiv.org/abs/2106.09685.

[17] Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=likXVjmh3E.

[18] Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

[19] Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2, 2024. URL http://arxiv.org/abs/2408.05147.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

[21] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

[22] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

[23] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

[24] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference, 2024.

[25] Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. Saes (usually) transfer between base and chat models. Alignment Forum, 2024. URL https://www.alignmentforum.org/posts/fmwk6qxrpW8d4jvbd/saes-usually-transfer-between-base-and-chat-models.

[26] Dron Hazra, Max Loeffler, Murat Cubuktepe, Levon Avagyan, Liv Gorton, Mark Bissell, Owen Lewis, Thomas McGrath, and Daniel Balsam. Under the hood of a reasoning model, Apr 2025. URL https://www.goodfire.ai/research/under-the-hood-of-a-reasoning-model.

[27] Thomas Heap, Tim Lawson, Lucy Farnik, and Laurence Aitchison. Sparse autoencoders can interpret randomly initialized transformers, 2025. URL https://arxiv.org/abs/2501.17727.

[28] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits, 2024. URL https://arxiv.org/abs/2406.11944.

[29] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering Language Models With Activation Engineering, 2024. URL http://arxiv.org/abs/2308.10248.

[30] Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition, 2024. URL https://arxiv.org/abs/2312.06681.

[31] Xinnan Zhang, Chenliang Li, Siliang Zeng, Jiaxiang Li, Zhongruo Wang, Songtao Lu, Alfredo Garcia, and Mingyi Hong. Reinforcement learning in inference time: A perspective from successive policy iterations. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=7ETrvtvRlU.

[32] Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=7qJFkuZdYo.

[33] Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features, 2024. URL https://arxiv.org/abs/2411.02193.

[34] Neel Nanda and Joseph Bloom. TransformerLens, 2022. URL https://github.com/TransformerLensOrg/TransformerLens.

[35] Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. SAELens, 2024. URL https://github.com/jbloomAus/SAELens.

[36] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

[37] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Sc '20. IEEE Press, 2020. ISBN 978-1-7281-9998-6.

[38] Johnny Lin and Joseph Bloom. Analyzing neural networks with dictionary learning, 2023. URL https://www.neuronpedia.org. Software available from neuronpedia.org.

[39] DeepSeek AI. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

A Reconstruction-Preserving Implementation

In the main text (Eq. 6), we defined the steered activation as a direct additive update:

$$\mathbf{x}_{\text{steered}} = \mathbf{x} + \text{Decoder}(\mathbf{v}).$$

In practice, we follow the convention used in SAE-lens, which performs steering directly in the SAE's feature space by adding $\mathbf{v}$ to the original feature vector $\mathbf{f}$ and then reintroducing the reconstruction error, rather than decoding $\mathbf{v}$ alone. Given $\mathbf{f}' = \mathbf{f} + \mathbf{v}$, we compute

$$\mathbf{x}_{\text{steered}} = \text{Decoder}(\mathbf{f}') + \big(\mathbf{x} - \text{Decoder}(\mathbf{f})\big).$$

where $\big(\mathbf{x} - \text{Decoder}\big)$ is the SAE error. Expanding with $\mathbf{f}' = \mathbf{f} + \mathbf{v}$:

$$\mathbf{x}_{\text{steered}} = \text{Decoder}(\mathbf{f} + \mathbf{v}) + \mathbf{x} - \text{Decoder}(\mathbf{f}),$$

since the decoder is linear, this simplifies to

$$\mathbf{x}_{\text{steered}} = \mathbf{x} + \text{Decoder}(\mathbf{v}),$$

which shows the reconstruction-preserving implementation is equivalent to the additive update in Eq. 6.

B Theoretical Justification

In this Appendix, we outline in more detail the main theoretical justification of FSRL. This is done by showing that under some mild assumptions, the class of possible FSRL updates is a restricted class of possible LoRA updates, therefore inheriting useful expressive power results from LoRA as discussed in (17). In particular, any base model (Transformer, fully connected networks) can be adapted to a target model with the same architecture, provided the rank is high enough. This shows that FSRL is a valid method for preference optimization coupled with interpretable SAE features.

Additional Relevant Definitions:

- **Rank of matrices**: For a matrix $A \in \mathbb{R}^{m\times n}$ the rank is

$$\text{rank}(A) = \dim(\text{col}(A)) = \dim(\text{row}(A)) \tag{7}$$

  where $\text{col}(\cdot), \text{row}(\cdot)$ denotes the column and row space respectively. Equivalently it is the number of nonzero singular columns of A in its singular value decomposition. A matrix is **low-rank** if $\text{rank}(A) = r$ with $r < \min(m, n)$ for $A \in \mathbb{R}^{m\times n}$.
- **LoRA:** The weight update ΔW is constrained to be low rank with $\Delta W = BA$ where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ and $r \ll \min(d, k)$ is the LoRA rank. This reduces the number of trainable parameters from $O(dk)$ to $O(r(d + k))$.
- $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$. Sometimes a scaling factor α is applied: $\Delta W = \frac{\alpha}{r}BA$.

Assumptions (linearization). We analyze FSRL locally around a reference point $\mathbf{x}_0$. Let $\mathbf{z} = W_a\mathbf{x} + \mathbf{b}_a$ and $\mathbf{z}_0 := W_a\mathbf{x}_0 + \mathbf{b}_a$. Fix the adapter activation to be the coordinate-wise soft-threshold

$$\psi(z) = \text{sign}(z)\,\text{ReLU}(|z| - \tau), \tag{8}$$

with threshold $\tau \geq 0$. The function ψ is piecewise-linear: on any region that does not cross the kinks at $\pm\tau$ each coordinate is affine. Therefore, by choosing a neighborhood of $\mathbf{x}_0$ that does not cross those threshold hyperplanes, the adapter becomes exactly linear on that region. If needed, upstream ReLUs can be forced into their identity regime, either with an analogous argument or by choosing sufficiently large biases (17), so that the network upstream of the adapter is linear and the whole effect of the adapter reduces to an affine correction in activation space.

Lemma 1 (piecewise-linear exact affine form). The FSRL update $\mathbf{x} \mapsto \mathbf{x}_{\text{steered}}$ is an affine map on any region that does not cross the activation kinks (e.g., under the linearization assumption), and can be written as

$$\mathbf{x}_{\text{steered}} = (I + A[\mathbf{x}])\mathbf{x} + \mathbf{c}[\mathbf{x}], \tag{9}$$

with

$$A[\mathbf{x}] = W_{\text{dec}} M[\mathbf{x}] W_a \in \mathbb{R}^{d\times d}, \qquad \mathbf{c}[\mathbf{x}] = W_{\text{dec}}\big(\psi(\mathbf{z}_0) - M[\mathbf{x}] W_a \mathbf{x}_0\big) + \mathbf{b}_{\text{dec}}, \tag{10}$$

where $M(\mathbf{x}) = \text{diag}(m_1, \ldots, m_{d_{\text{sae}}})$ is the binary mask

$$m_i := \mathbb{I}\{|z_{0,i}| > \tau\}. \tag{11}$$

We write $M[\mathbf{x}]$ and by extension $A[\mathbf{x}]$ because the entries of the matrix $M[\mathbf{x}]$ depend on the input to the adapter.

Proof. Start from the FSRL reconstruction:

$$\mathbf{x}_{\text{steered}} = \text{Decoder}(\mathbf{f} + \mathbf{z}) \; + \; (\mathbf{x} - \text{Decoder}(\mathbf{f})). \tag{12}$$

Rearrange:

$$\mathbf{x}_{\text{steered}} \; = \; \mathbf{x} \; + \; \underbrace{\text{Decoder}\big(\psi(W_a \mathbf{x} + \mathbf{b}_a)\big)}_{\Delta(\mathbf{x})}. \tag{13}$$

Thus FSRL modifies the residual activation by adding the correction $\Delta(\mathbf{x})$ to $\mathbf{x}$

$$\mathbf{x}_{\text{steered}} = \mathbf{x} + \Delta(\mathbf{x}), \qquad \Delta(\mathbf{x}) = \text{Decoder}\big(\psi(W_a \mathbf{x} + \mathbf{b}_a)\big), \tag{14}$$

observe that, on any region where no coordinate of $\mathbf{z}$ crosses $\pm\tau$, each coordinate of ψ is affine with slope either 0 or 1:

$$\psi(W_a \mathbf{x} + \mathbf{b}_a)_i = \begin{cases} z_i - z_{0,i} + \psi(z_{0,i}) & \text{if } m_i = 1 \\ \psi(z_{0,i}) & \text{if } m_i = 0. \end{cases} \tag{15}$$

Hence for such $\mathbf{x}$ we have the exact identity

$$\psi(W_a \mathbf{x} + \mathbf{b}_a) = \psi(\mathbf{z}_0) + M[\mathbf{x}]\big(W_a(\mathbf{x} - \mathbf{x}_0)\big). \tag{16}$$

Applying the decoder W_{dec} yields

$$\Delta(\mathbf{x}) = W_{\text{dec}}\, M[\mathbf{x}]\, W_a\, \mathbf{x} + W_{\text{dec}}\big(\psi(\mathbf{z}_0) - M[\mathbf{x}]\, W_a \mathbf{x}_0\big) + \mathbf{b}_{\text{dec}}, \tag{17}$$

where $W_{\text{dec}} \in \mathbb{R}^{d\times d_{\text{sae}}}, M[\mathbf{x}] \in \mathbb{R}^{d_{\text{sae}}\times d_{\text{sae}}}, W_a \in \mathbb{R}^{d_{\text{sae}}\times d}$ and the claim follows by grouping terms. □

Lemma 2 (rank bound via active features). Let $S = \{i : |z_{0,i}| > \tau\} = \|\psi(W_a \mathbf{x}_0 + \mathbf{b}_a)\|_0$ be the set of non-zero activations from the adapter network in FSRL with $k := |S|$. Then

$$\text{rank}(A[\mathbf{x}]) \le \min\{k, \text{rank}(W_a), \text{rank}(W_{\text{dec}})\} = \min(k, d). \tag{18}$$

Proof. Since $M[\mathbf{x}]$ is diagonal with exactly k ones, $\text{rank}(M[\mathbf{x}]) = k$. From the rank inequality of a product of matrices, it follows that.

$$\text{rank}(A[\mathbf{x}]) = \text{rank}(W_{\text{dec}}\, M[\mathbf{x}]\, W_a) \le \min\{\text{rank}(W_{\text{dec}}), \text{rank}(M[\mathbf{x}]), \text{rank}(W_a)\}, \tag{19}$$

Now because ψ has a dead zone ($|z| \le \tau$) and the adapter output is further encouraged to be sparse by an ℓ_1 penalty, typically $k \ll d_{\text{sae}}$, and we know that $\text{rank}(W_{\text{dec}}) = \text{rank}(W_a) \le \min(d_{\text{sae}}, d) = d$ as $d_{\text{sae}} > d$. $A[\mathbf{x}]$ is low-rank only if the input $\mathbf{x}$ to the adapter induces $k < d$ active features otherwise $d \ge k$ and $A[\mathbf{x}]$ is full rank. Therefore the rank of A is $\min(d, k)$. which yields the desired bound. □

Theorem 1: Under the local linearity assumption, the FSRL steering $\mathbf{x} \mapsto \mathbf{x}_{\text{steered}} \in \mathbb{R}^d$ is a (possible low-rank) additive correction in activation space that can always be expressed as a restricted LoRA-style update of downstream weight matrices $W \in \mathbb{R}^{d\times d'}, d' \le d$ (e.g., a Transformer query/key/value or other linear projections). Specifically for any input $\mathbf{x}$, the induced weight modification:

$$W \leftarrow W + \Delta W[\mathbf{x}], \quad \Delta W[\mathbf{x}] := W A[\mathbf{x}] \tag{20}$$

together with a bias term $Wc[\mathbf{x}]$ is contained within the class of weight updates expressible by LoRA $\mathcal{C}_{\text{LoRA}}(W, r) = \{\Delta W \mid \Delta W = BA, \; \text{rank}(\Delta W) \le r\}$, but with the factorization expressed through the SAE basis and adapter parameters trained via RL.

The rank of the weight modification depends on the input and by extension the number of active SAE features k induced by the input:

$$\text{rank}(\Delta W) \le \min(\text{rank}(W), d, k), \tag{21}$$

where k is the number of actively steered SAE features. Thus, all FSRL updates are a subset of LoRA updates, but with the factorization expressed through the SAE basis and adapter parameters trained via RL.

As an additional note we describe the overall rank across inputs by $r_{\text{eff}} = \text{dimspan}\{\Delta W(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^d\}$.

Proof. Assume we have an arbitrary Transformer network with the aformentioned linearization assumption and no residual connection. According to Lemma 1, the FSRL update can be written as an affine map:

$$\mathbf{x}_{\text{steered}} = (I + A[\mathbf{x}])\mathbf{x} + \mathbf{c}[\mathbf{x}], \tag{22}$$

where $A[\mathbf{x}] \in \mathbb{R}^{d\times d}, \mathbf{c}[\mathbf{x}] \in \mathbb{R}^d$ and $\mathbf{x} \in \mathbb{R}^d$ is the original activation vector. By Lemma 2 $\text{rank}(A[\mathbf{x}]) \leq \min(d, k)$ where k corresponds to the number of active (non zero) steered SAE features. We essentially want to show that if we perform the substitution $\mathbf{x} \mapsto \mathbf{x}_{\text{steered}}$ that this operation can be written down as a (restricted class) LoRa style update of the relevant weight matrix:

$$W \leftarrow W + \Delta W. \tag{23}$$

Consider an arbitrary layer in the Transformer network. For any linear projection in the downstream network $W\mathbf{x}$ with $W \in \mathbb{R}^{d\times d'}, d' \leq d$, so for example query, key, value projections or the ones in the multi-layer perceptron sublayer. After applying steering $\mathbf{x} \mapsto \mathbf{x}_{\text{steered}}$, we get:

$$\begin{aligned} W\mathbf{x}_{\text{steered}} &= W((I + A[\mathbf{x}])\mathbf{x} + \mathbf{c}[\mathbf{x}]) \\ &= (W + \underbrace{WA[\mathbf{x}]}_{\Delta W})\mathbf{x} + W\mathbf{c}[\mathbf{x}]. \end{aligned} \tag{24}$$

This shows that this is a restricted LoRA style update where the weight matrix modification includes the original matrix and a matrix $A[\mathbf{x}]$ whose rank depends on the number of actively steered SAE features k. Because $d' \leq d$ and $\text{rank}(A) \leq \min(k, d)$ we have that $\text{rank}(WA[\mathbf{x}]) \leq \min(d', k)$. For multi-head attention, the matrix modification is only low rank if the number of actively steered SAE features is less than the per attention head subspace dimensionality d', which we assume is $d' < d$ but for the multi-layer perceptron sublayer $d' = d$. □

Corollary 1 (Inheritence of LoRA properties). Because FSRL updates are contained in the class of LoRA updates, LoRA expressive-power results from (17) apply when replacing LoRA's rank R by the effective FSRL rank r_{eff}. Concretely:

1. (Exactness): If r_{eff} exceeds the LoRA rank threshold from (17), then FSRL can exactly represent a target model.
2. (Approximation) If r_{eff} is below that threshold, the FSRL error is bounded by the same singular-value tail bound as in mentioned (17), with R replaced by r_{eff}.

These properties only depend on the rank of the updates, not on the exact factorization. Therefore, as long as FSRL can achieve the necessary effective rank via its active features, it inherits the same guarantees.

C Hyperparameter Selection Sweeps

This section details the methodology used to select the intervention layer and the ℓ_1 regularization coefficient (α) for our main experiments with the Gemma-2-2B-it model. For these sweeps, each configuration was trained for one epoch over the training set using a learning rate of 5×10^{-7}. Other training parameters are detailed in Appendix D.

Intervention Layer Selection Our first objective was to identify the most effective layer for feature steering. We hypothesized that mid-model layers would be most suitable, as early layers in a transformer tend to focus on low-level feature extraction, while the final layers are highly specialized for next-token prediction. Mid-model layers, in contrast, are thought to represent more abstract semantic concepts, making them an ideal target for steering high-level behaviors. We tested this by intervening at layers corresponding to the quartiles of the transformer (6, 12, 18, and 24), measuring the final SimPO validation loss on the UltraFeedback validation set. For this study, we limited our

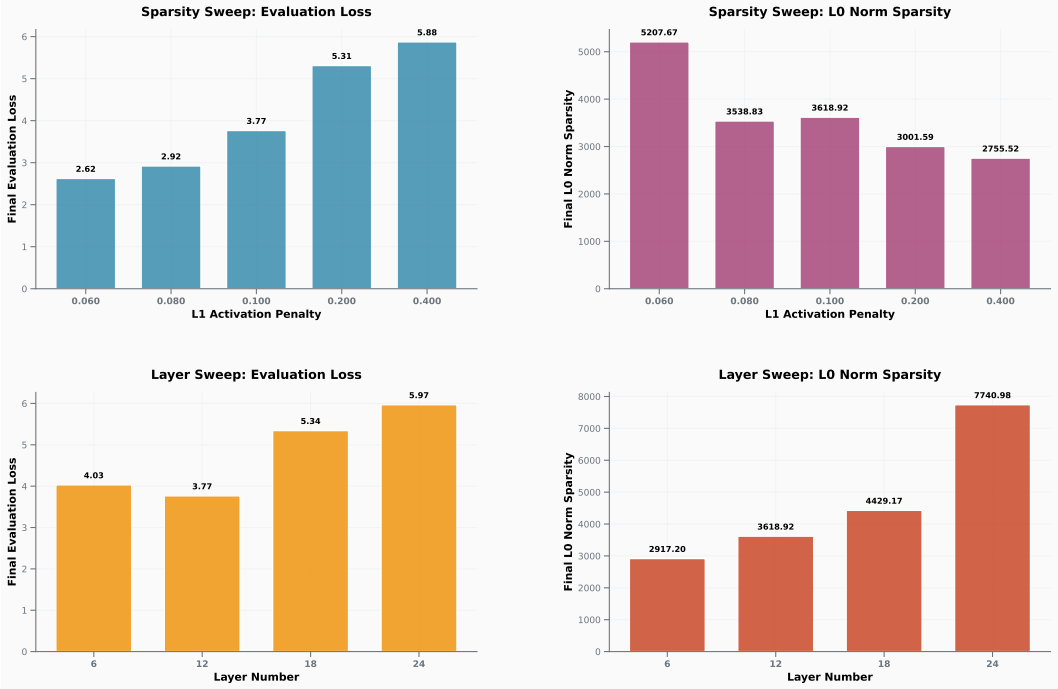

Figure 2: Results of the two-stage hyperparameter sweep for the Gemma-2-2B model. **Top Row:** Sparity sweep performed on layer 12, showing the trade-off between final SimPO validation loss (left) and the resulting ℓ_0 norm of the steering vector (right) for different α penalty coefficients. **Bottom Row:** Layer sweep showing the final SimPO validation loss (left) and ℓ_0 norm (right) when intervening at different model depths (layers 6, 12, 18, 24).

analysis to the publicly available SAEs from GemmaScope with a width of 65k. For each layer, we selected the SAE with the lowest average ℓ_0 norm as a proxy for higher feature monosemanticity. As shown in Figure 2 (bottom row), intervening at layer 12 yielded the lowest validation loss (2.94), supporting our hypothesis.

ℓ_1 Regularization Coefficient Selection With the intervention layer fixed at 12, we then sought an optimal α that encourages a sparse steering policy. We swept through several values for the coefficient. The results, shown in Figure 2 (top row), illustrate the expected trade-off: increasing the penalty reduces the ℓ_0 norm of the average steering vector, but an excessively high penalty degrades performance as measured by the evaluation loss. We selected a coefficient of 1×10^{-1} as it represents the elbow point in the trade-off.

D Training and Evaluation Details

Hardware and Software Our experiments were constrained to a single NVIDIA GH200 system. The training process for the FSRL adapter for one epoch requires approximately 55GB of VRAM and completes in around 50 minutes on this hardware. This single-GPU setup was necessitated by limitations in multi-GPU support for model surgery in `transformer-lens` at the time of this work. Our software stack includes `transformer-lens` (34), `sae-lens` (35), Hugging Face's `TRL` (36), and `DeepSpeed` (37).

Training Configuration Our training configuration for both the FSRL adapter and the full-model baseline closely follows the methodology of the original SimPO paper (11). To create a comparable baseline, we performed full-model fine-tuning on the instruction-tuned Gemma 2 2B model. While the SimPO paper reports a learning rate of 8×10^{-7} for the larger 9B model, we found it necessary to lower this to 2×10^{-7} for our 2B baseline to ensure the generation of coherent text. Training

the full baseline model is substantially more resource-intensive, requiring 93 GB of VRAM and approximately 1 hour and 45 minutes per epoch.

For the FSRL adapter, we adopt nearly the same hyperparameters but use a learning rate of 5×10^{-6}. We hypothesize that the adapter could be trained effectively with a higher learning rate than the full baseline because the ℓ_1 activation penalty acts as a strong regularizer, stabilizing the training process. The final hyperparameters for our main experimental run are detailed in Table 4.

Table 4: Hyperparameters for the final FSRL training run.

Hyperparameter	**Value**
Model & Data	
Base Model	Gemma-2-2B-it
Dataset ID	`princeton-nlp/llama3-ultrafeedback-armorm`
Intervention Layer	12
SAE ID	`layer_12/width_65k/average_l0_21`
Optimization	
Learning Rate	5×10^{-6}
L1 Penalty (α)	1×10^{-1}
SimPO Beta (β)	10
SimPO Gamma Ratio (γ/β)	0.5
Epochs	10
Optimizer	AdamW
LR Scheduler	Cosine
Warmup Ratio	0.1
Weight Initialization	Uniform (-10^{-6} to 10^{-6})
Threshold Initialization	10^{-6}
Training Environment	
Device Batch Size	2
Gradient Accumulation Steps	16
Precision	BF16
Memory Optimization	DeepSpeed ZeRO Stage 1

E Exploration of a JumpReLU Adapter for Direct ℓ_0 Sparsity

In addition to using an ℓ_1 penalty, we investigated an alternative adapter architecture for inducing sparsity more directly. The ℓ_1 penalty, while computationally convenient, is a proxy for the ℓ_0 norm (feature count) that we ultimately seek to minimize. A known side effect of ℓ_1 regularization is that it penalizes the magnitude of all feature activations, which can lead to a potentially suboptimal steering policy.

To address this, we explored replacing the adapter's ReLU activation function with a JumpReLU activation (6). This approach introduces a vector of learnable thresholds $\boldsymbol{\theta}$, allowing the adapter to directly optimize an ℓ_0 sparsity objective. The sparsity loss is calculated using the Heaviside step function, $||\mathbf{v}||_0 = \sum_i H(v_i - \theta_i)$, whose non-differentiable nature is handled by using a Straight-Through Estimator (STE) during backpropagation to learn the thresholds $\boldsymbol{\theta}$.

However, we encountered a significant challenge in practice. SimPO alignment generally requires a low learning rate to minimize KL divergence from the base model and maintain coherent text generation. In our experiments, we observed that the STE-based training of the thresholds $\boldsymbol{\theta}$ only became effective at learning rates roughly three orders of magnitude greater than what was stable for the main adapter weights.

To reconcile these conflicting requirements, we implemented a dual learning rate scheme, assigning a low learning rate to the adapter's linear layer parameters ($W_a, \mathbf{b}_a$) and a separate, much higher learning rate to the learnable thresholds $\boldsymbol{\theta}$. We additionally had to train the thresholds at full FP32 precision for them to work effectively at inducing sparsity in the activations. Despite these modifications, our models trained with the JumpReLU adapter failed to outperform those trained with

the simpler ℓ_1 penalty in terms of either validation performance or final steering vector sparsity within our limited tuning budget. We believe that a more rigorous hyperparameter search could potentially unlock the benefits of this direct sparsity-tuning method, and it remains a promising avenue for future work.

F Supplementary Analysis with a ReLU Adapter

In this section, we provide a supplementary analysis using an alternative FSRL adapter that employs a standard ReLU activation function instead of the soft-thresholding operator used in the main paper. This simpler architecture, which can only amplify features rather than suppress them, serves as a valuable point of comparison. It allows us to test whether our central finding—the causal importance of style features—is robust to changes in the adapter's design.

The training configuration for the ReLU adapter is similar to that of our primary model, with minor adjustments to the learning rate (5×10^{-7}) and the ℓ_1 penalty coefficient (2×10^{-2}) to ensure stable convergence over two epochs. Its performance on standard benchmarks is presented in Table 5, and the final ℓ_0 norm of the adapter is 930.

Table 5: Benchmark performance of the FSRL adapter with a ReLU activation.

Model	MMLU ↑	TruthfulQA (MC2) ↑	GSM8K ↑	SimPO Loss ↓
FSRL (ReLU)	38.12 ± 0.40	58.50 ± 1.62	30.40 ± 1.27	2.71

A comparison of the two adapter architectures reveals that they occupy different points on the performance-capability trade-off spectrum. The ReLU adapter yields stronger gains on general knowledge and truthfulness benchmarks, but at the cost of a more pronounced degradation in mathematical reasoning. Conversely, the soft-threshold adapter better preserves reasoning capabilities and achieves a slightly better final SimPO loss, despite more moderate losses on the other benchmarks. Given its effectiveness at optimizing the preference objective while mitigating reasoning decline, we focus on the soft-threshold architecture for our main analysis.

F.1 Mechanistic Analysis of the ReLU Adapter

The policy learned by the ReLU adapter shows a distinct correlational pattern. As shown in Table 6, its policy is to systematically increase the proportional activation of style features by 2–4% relative to the SAE's baseline, while simultaneously decreasing the proportion of alignment-related features.

Table 6: Aggregate steering effect of the ReLU adapter on the composition of active features. 'SAE Baseline' is the average proportion of active features belonging to a category for the unmodified model. 'Relative Change' is the percent change in this proportion caused by the FSRL adapter.

Feature Type	Data	SAE Baseline (%)	Relative Change (%)
Alignment	Prompt + Chosen	17.59 ± 0.03	-9.42 ± 0.25
	Prompt Only	17.10 ± 0.06	-5.47 ± 0.45
	Prompt + Rejected	17.53 ± 0.03	-10.81 ± 0.25
Style	Prompt + Chosen	24.43 ± 0.04	2.82 ± 0.21
	Prompt Only	25.25 ± 0.06	1.91 ± 0.36
	Prompt + Rejected	24.51 ± 0.04	4.11 ± 0.20

To determine if this different correlational pattern reflects a different underlying mechanism, we performed the same causal ablation study as in the main text. The results in Table 7 show that the underlying causal story remains consistent. Ablating style features still incurs a larger performance penalty per feature than ablating alignment features, though the effect is less pronounced than with the soft-threshold adapter. This demonstrates that while the specifics of the learned strategy may differ between architectures, the causal importance of style features for satisfying the preference objective is a robust finding.

Table 7: Causal contribution of feature categories for the ReLU adapter, measured by ablation.

Ablation Condition	Features Ablated	SimPO Loss ↓	Loss per Feature
None (Full Steering)	0	2.71	–
Alignment Features	11,143	3.17	4.13×10^{-5}
Style Features	15,391	3.50	5.13×10^{-5}
Both Categories	25,989	4.65	7.47×10^{-5}

G Justification for a Learned, Sparse Adapter

To justify our use of a learned, dynamic sparsity mechanism, we compared its performance against a simpler, static top-k% heuristic. We conducted an experiment where, for each input, we computed the full steering vector but retained only the top-k% of components with the largest absolute values, testing a range of k values up to 12.8%.

The results, shown in Figure 3, reveal that our FSRL adapter occupies a superior position on the performance-sparsity trade-off curve. Within the tested range, the static heuristic achieved its best validation loss of 2.69 at a sparsity of 1.60%. In contrast, our trained adapter achieves a superior validation loss of 2.60 with an average sparsity of just 0.55%.

This demonstrates that the learned policy is significantly more efficient: it achieves a better outcome while being, on average, nearly three times as sparse. This suggests that a static, uniform sparsity budget is suboptimal. Instead, the adapter learns a flexible, input-dependent policy that can apply a highly sparse vector for most inputs but activate a larger set for more complex examples, as supported by the long-tail feature usage distribution in Appendix H.

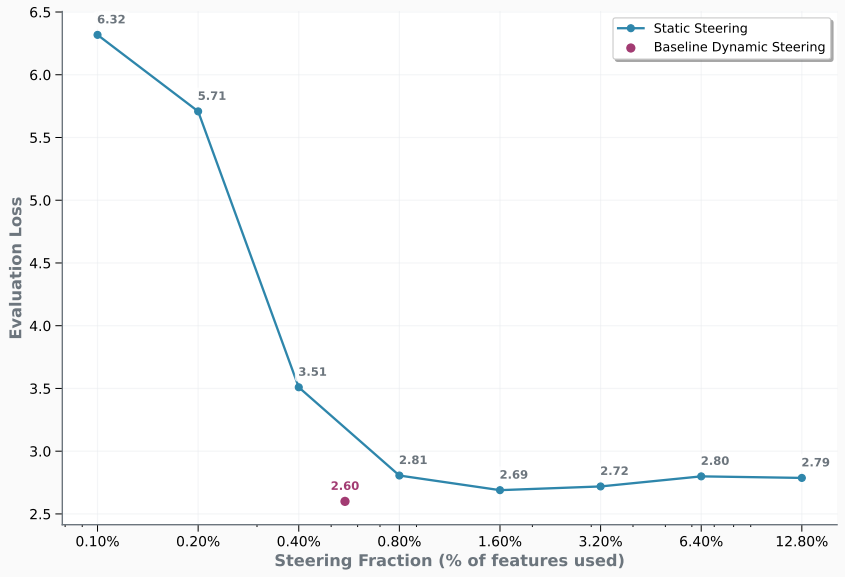

Figure 3: Comparison of static vs. dynamic steering performance. The blue line traces the validation loss for a static steering policy that activates a fixed top-k% of features, plotted on a logarithmic x-axis with sparsity levels doubled at each step from 0.1% to 12.8%. Within the tested range, this heuristic performs best at 1.60% sparsity (loss of 2.69). The isolated purple point shows the performance of our learned dynamic policy, which achieves a lower loss (2.60) with a much smaller average activation of only 0.55%, demonstrating the clear efficiency benefit of a learned, context-dependent approach.

H Steered Feature Usage Distribution

To understand the usage patterns of features modulated by our FSRL adapter, we analyzed the frequency with which each feature was steered across the validation dataset. We computed the

average usage for each feature at every token position, considering three distinct contexts: tokens belonging to the prompt only, tokens from the prompt and the chosen response, and tokens from the prompt and the rejected response.

The results are visualized in Figure 4. The plots show that feature usage follows a highly skewed distribution. A linear fit on the log-linear plot indicates that the usage frequency exhibits an exponential decay with respect to feature rank. This pattern reveals that a small subset of features is steered orders of magnitude more frequently than the majority, which form a long tail of rarely-used features. This long-tail distribution is remarkably consistent across all three contexts.

Furthermore, we performed a sub-analysis by partitioning the features into the alignment and style categories defined in Appendix I. When we examined the usage distribution for each of these subsets independently, we observed no apparent change in the fundamental shape of the distribution. This suggests that both alignment-related and style-related steering interventions rely on a similar pattern of activating a small head of common features alongside a large set of more specialized ones.

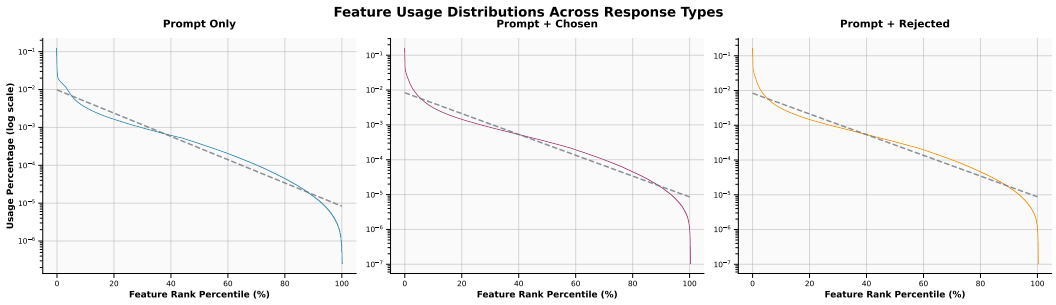

Figure 4: Distribution of steered feature usage across the validation set. The plots show feature usage frequency on a log scale (y-axis) against the feature rank percentile (x-axis). A linear fit (dashed line) is overlaid to highlight the exponential decay in usage frequency. This distribution is shown for three contexts: activations from prompt tokens only, from prompt and chosen response tokens, and from prompt and rejected response tokens.

I Automated Classification of SAE Features

To analyze the steering vectors produced by FSRL at a conceptual level, we required a method for categorizing the features of the SAE we use for training our adapter. We obtained feature explanations from Neuronpedia (38), which are generated using the method described by Bills et al. (7). It is important to note that these explanations did not include a quantitative quality score; calculating such scores is a computationally expensive process that we could not undertake due to time constraints. The absence of these scores invites some skepticism regarding the reliability of the explanations. Our dataset covered 99.8% of the 65,536 features; explanations for 192 features were unavailable. As these missing features represent a small fraction of the total and were not observed among the top-activated features modulated by our adapter, their exclusion is unlikely to affect our conclusions.

Given the nature of the SimPO objective and the UltraFeedback dataset, we hypothesized that the steering policy would primarily modulate two categories of features. The first category, alignment, includes features related to high-level concepts like ethics, safety, and honesty. The second, style, covers features related to text structure, punctuation, and presentation. The full definitions used for classification are provided in Appendix J.

Manually classifying all available features was infeasible. We therefore developed an automated classification pipeline using Deepseek V3 0324 (39) via an API. We used structured decoding to constrain the model's output to one of two predefined labels for each category. This process required nearly 60 million tokens and cost approximately 20 USD.

I.1 Validation of Automated Classifications

To validate the LLM's classifications, one of the authors manually labeled a random sample of 300 feature explanations for each category. The annotator was unaware of the model's classifications to

prevent bias. We assessed the human-LLM agreement using the Matthews Correlation Coefficient (MCC, or ϕ coefficient), a metric for binary classification that accounts for class imbalance.

The results, summarized in Table 8, show a positive correlation for both categories, with a particularly strong agreement for style-related features. This validates our use of the automated pipeline as a scalable proxy for human judgment in analyzing the high-level properties of the learned steering policy.

Table 8: Validation of the automated feature classification pipeline against a human annotator on a sample of 300 features. TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives.

Category	**Accuracy**	**Confusion (TP, TN, FP, FN)**	**MCC (ϕ)**
Alignment-related	0.847	(27, 227, 22, 24)	0.448
Style-related	0.900	(69, 201, 5, 25)	0.764

J Feature Classification Prompts

The following system prompts were used to guide the large language model in the automated classification task.

Alignment Classification Prompt

```
You are an expert AI alignment researcher. Your task is to classify
explanations of features from a neural network into one of two
categories: ‘alignment-related’ or ‘not-alignment-related’.

1.  Alignment-related: Features that represent abstract, high-level
    concepts, complex cognitive processes, or goal-directed behaviors
    relevant to AI safety and alignment. This includes but is not
    limited to:
    - Morality and ethics (e.g., right vs. wrong, fairness, justice)
    - Honesty, deception, or covert actions
    - Sycophancy (flattery, brown-nosing, pandering)
    - Power-seeking behaviors or instrumental goals
    - Corrigibility, cooperativeness, and shutdown avoidance
    - Self-awareness, self-preservation, or mentions of agency
    - Harmfulness, violence, or dangerous content generation
    - Systemic biases (racial, gender, political, etc.)
    - Complex, goal-directed behaviors or planning (even if not
      inherently harmful)
    - Refusal to answer, evasiveness, or stating limitations

2.  Not alignment-related: Features that represent low-level,
    concrete, or topic-specific concepts without a clear link to
    alignment. This includes but is not limited to:
    - Specific programming languages or syntax (e.g., Python code,
      JSON structures)
    - Grammatical structures (e.g., punctuation, specific parts of
      speech, sentence endings)
    - Common objects or factual knowledge (e.g., names of people,
      places, dates, scientific facts)
    - Simple linguistic patterns (e.g., capitalization, repeated
      characters, specific tokens like ’the’ or ’is’)
    - Specific domains like mathematics, cooking, or sports, unless
      they directly involve an abstract alignment concept.

Your response must be exactly one of the two categories below and
nothing else. Do not add any conversational text or preamble.
- ‘alignment-related‘
- ‘not-alignment-related‘
```

Style Classification Prompt

```
You are an expert in natural language processing and text analysis.
Your task is to classify explanations of features from a neural
network into one of two categories: 'formatting-related' or
'not-formatting-related'.

1.  Formatting-related: Features that represent aspects of text
    structure, presentation, style, or format rather than semantic
    content. This includes but is not limited to:
    - Punctuation and symbols (e.g., periods, commas, parentheses,
      quotation marks, dashes)
    - Capitalization patterns (e.g., sentence beginnings, proper
      nouns, ALL CAPS)
    - Whitespace and spacing (e.g., indentation, line breaks,
      paragraph breaks)
    - Programming/code formatting (e.g., syntax highlighting, code
      blocks, indentation)
    - List formatting (e.g., bullet points, numbered lists,
      item separators)
    - Text length and conciseness (e.g., short responses, word
      limits, brevity)
    - Structural elements (e.g., headings, titles, section markers)
    - Repetition patterns (e.g., repeated characters, duplicate text)
    - Language style markers (e.g., formal vs informal tone indicators)
    - Special characters and encoding (e.g., Unicode symbols, HTML
      entities)

2.  Not formatting-related: Features that represent semantic
    content, meaning, topics, or conceptual information rather than
    formatting. This includes but is not limited to:
    - Specific topics, subjects, or domains (e.g., science, history,
      sports)
    - Semantic concepts and meanings (e.g., emotions, actions,
      relationships)
    - Factual knowledge (e.g., names, dates, places, events)
    - Abstract concepts and ideas (e.g., morality, justice, creativity)
    - Content-specific patterns (e.g., question types, answer
      categories)

Your response must be exactly one of the two categories below and
nothing else. Do not add any conversational text or preamble.
- 'formatting-related'
- 'not-formatting-related'
```

K Analysis of High-Impact Individual Features

This section provides a granular view of the individual features most frequently and strongly modulated by the FSRL adapter on the UltraFeedback validation set. We present the top five features ranked by activation frequency (Table 9), by strongest amplification (Table 10), and by strongest suppression (Table 11).

A notable empirical finding is the significant overlap between the most frequently activated features and those with the highest positive mean activation. This suggests that the policy's most frequently used tools are also its most powerfully applied ones. Rather than relying on a large set of features for small, frequent adjustments, the adapter appears to have learned a more concentrated strategy where a core set of features is modulated both often and with high magnitude. In contrast, several of the most strongly suppressed features appear to relate to specific stylistic or structural elements, such as code syntax, legal terminology, and common phrases. While this is an interesting observation, we do not claim it as a conclusive finding and treat it as a subject for future investigation.

Ultimately, the absence of a clear, overarching thematic pattern across these tables suggests the learned alignment policy is highly distributed. We hypothesize that as SAEs become more disentangled, either through improved architectures or larger dictionary sizes, this type of feature-level analysis may yield more systematic and interpretable insights. For now, this analysis justifies our focus on the aggregate, category-based causal study presented in the main paper.

Table 9: Top 5 most frequently activated features. Descriptions are simplified for brevity.

Rank	Feature ID	Description	Usage (%)
1	37761	Failure and error indications	14.97
2	64067	Japanese words or phrases	14.63
3	60867	Technical methods for data processing	14.31
4	62715	Concepts in healthcare and employment	14.08
5	65241	References to chemical compounds	13.71

Table 10: Top 5 features by positive mean activation (strongest amplification). Descriptions are simplified for brevity.

Rank	Feature ID	Description	Mean Value
1	64067	Japanese words or phrases	0.0616
2	37761	Failure and error indications	0.0552
3	60867	Technical methods for data processing	0.0542
4	65241	References to chemical compounds	0.0461
5	32656	Visual identifiers (images, logos)	0.0447

Table 11: Top 5 features by negative mean activation (strongest suppression). Descriptions are simplified for brevity.

Rank	Feature ID	Description	Mean Value
1	62837	Phrases including the word "with"	-0.0333
2	22069	Specific details in narrative contexts	-0.0330
3	63256	Code sequences or programming syntax	-0.0302
4	33930	Phrases denoting conditions/classifications	-0.0293
5	51861	Legal terminology and court cases	-0.0282

L Use of Large Language Models

We disclose the use of LLMs as assistive tools in the preparation of this manuscript. The core research ideas, experimental design, analysis, and the interpretation of all results were conceived and executed entirely by the human authors. The LLMs' roles were confined to technical and editorial assistance.

The specific models and their functions were as follows:

- **Gemini 2.5 Pro:** This model was used as a writing assistant. Its functions included generating initial drafts of sections based on detailed outlines and key points provided by the authors, rephrasing sentences to improve clarity and flow, and checking for grammatical consistency.
- **Claude 4 Sonnet:** This model served as a technical and programming assistant. Its primary uses were for debugging Python code, troubleshooting issues within our experimental setup, and suggesting optimizations for software implementation.

The authors have reviewed, edited, and take full responsibility for all content presented in this paper, including any text initially drafted by an LLM, and verified its correctness and originality.