# ADAPT: Hybrid Prompt Optimization for LLM Feature Visualization

**João Cardoso**
Instituto Superior Técnico
INESC-ID
joao.n.m.cardoso@tecnico.ulisboa.pt

**Arlindo L. Oliveira**
Instituto Superior Técnico
INESC-ID
arlindo.oliveira@tecnico.ulisboa.pt

**Bruno Martins**
Instituto Superior Técnico
INESC-ID
bruno.g.martins@tecnico.ulisboa.pt

## Abstract

Understanding what features are encoded by learned directions in LLM activation space requires identifying inputs that strongly activate particular directions. Feature visualization, which optimizes inputs to maximally activate a target direction, offers an alternative to costly dataset search approaches, but remains underexplored for LLMs due to the discrete nature of text. Furthermore, existing prompt optimization techniques are poorly suited to this domain, which is highly prone to local minima. To overcome these limitations, we introduce ADAPT as a hybrid method combining beam search initialization with adaptive gradient-guided mutation, designed to address the local minima and unreliable gradient estimates that limit existing methods. We evaluate on sparse autoencoder latents from Gemma 2 2B, proposing metrics grounded in dataset activation statistics to enable rigorous comparison. Experimental results show that ADAPT consistently outperforms prior methods across layers and latent types. ADAPT also produces significantly more interpretable prompts in human evaluation. Our results establish that feature visualization for LLMs is tractable, but requires design assumptions tailored to the domain.

## 1 Introduction

Neural networks construct rich representations of their inputs, often assigned to linear directions in their latent activation space. Understanding the relevant input features encoded by these directions is essential for interpretability efforts, but identifying the range of input patterns that activate a given latent direction (henceforth, *latent*) is a challenging task. The standard approach is to perform a dataset search [2, 13]: running the model on a large corpus and registering what input examples maximally activate the directions of interest, which is an effective approach for large-scale surveys where the computational costs of running a model on a large corpus can be amortized over thousands of latents in parallel. Dataset search is often performed for latent directions learned by sparse autoencoders (SAEs) [3, 19], which are designed to yield large dictionaries of interpretable and monosemantic latents. However, with alternate techniques for obtaining latents of interest, or in the case of more complex sub-components, such as model circuits or collections of latents, this approach becomes computationally infeasible.

An alternative approach is feature visualization [11], which consists of creating an input that is "tailor-made" for a given latent, illustrating the kind of input it fires for. This can be achieved with prompt optimization techniques, which typically seek to maximally activate the target latent. Feature visualization brings several advantages over dataset search:

- **Overcoming dataset limitations:** Searched datasets are typically smaller than the model's training dataset, limiting the number of illustrative examples for any given latent.
- **Disentangling correlated features:** If two features happen to be correlated in the searched dataset (e.g. a semantic concept and the language it happens to appear in), it is hard to tease apart the latent's corresponding feature [11].
- **Versatility:** Optimization procedures can be modified to incorporate additional complex objectives and penalties.
- **Causal grounding:** Correlational methods can produce plausible-looking explanations even for randomly initialized networks [10]. Feature visualization constructs inputs that cause activation, sidestepping spurious associations in any fixed corpus.

Prior vision interpretability work has made extensive use of feature visualization, demonstrating these advantages in practice. However, the application of these techniques to LLMs has been limited, mostly due to the discrete nature of textual inputs [22], which precludes direct gradient-based updates. In this work, we draw on prior adversarial prompt optimization approaches, where the goal is to elicit outputs that circumvent safety training, as well as earlier efforts in LLM feature visualization.

We introduce ADAPT as a new algorithm that improves on existing baselines by combining beam-search initialization with gradient-guided mutation. Alongside it, we develop an evaluation setting in which feature visualization methods can be compared quantitatively, including a stratified sample of 486 SAE latents from Gemma 2 2B [6], metrics that score optimized prompts against each latent's maximally activating dataset examples, and a protocol for rating their interpretability with humans. Using this setting, we compare ADAPT with prior methods. Finally, we examine the reliability of gradient-based optimization methods in this setting, and explore how fluency penalties influence prompt optimization for feature visualization.

Experiments show that ADAPT achieves higher activation than GCG on 70.4% of latents, matches or outperforms all baselines on every layer, and produces prompts that human raters find significantly more interpretable. Our analyses further show that GCG's gradient estimate is an unreliable guide to promising token swaps, and that a fluency penalty does not make GCG's prompts more interpretable.

## 2 Background

This section reviews sparse autoencoders, which provide our evaluation testbed, together with existing prompt optimization methods, whose limitations motivate ADAPT.

### 2.1 Sparse Autoencoders

Sparse autoencoders (SAEs) [3, 20] reconstruct an LLM's representation vectors through an overcomplete hidden layer, with a sparsity constraint on the activations, yielding latent directions that tend to be more monosemantic and interpretable than individual neurons. While ADAPT can accommodate arbitrary directions, we focus on SAE latents due to the availability of open-source tooling, which enables direct comparison with dataset search at low computational cost. In particular, the Gemma Scope project [9] provides a publicly available SAE suite, with latent statistics and dataset examples accessible via the Neuronpedia API[1].

Given an input activation vector $x \in \mathbb{R}^n$, an SAE produces a reconstruction via $\hat{x} = W_{\text{dec}} f(x) + b_{\text{dec}}$, where $f(x) = \sigma(W_{\text{enc}} x + b_{\text{enc}}) \in \mathbb{R}^M$ is the sparse activation vector and $\sigma$ is an elementwise nonlinearity. For a target latent $i$, the optimization objective is $x^* = \arg\max_x f_i(x)$, where $f_i(x)$ denotes the activation magnitude of the $i$-th latent direction. Interpretability is not optimized directly, but is the criterion by which we evaluate the resulting prompts (§ 4.3.3).

### 2.2 Prompt Optimization

Despite the abundance of prompt optimization methods [16], most are not applicable to feature visualization due to fundamental domain differences. For one, it would be infeasible to use exceedingly expensive techniques, such as those which involve extensive LLM API calls [24], comprehensive

[1] `https://neuronpedia.org/api-doc`

tree searches [23], or model fine-tuning [4]. A successful jailbreak prompt justifies hours of compute because it transfers across contexts, whereas feature visualization must scale to thousands of latents, each requiring independent optimization. For another, approaches that leverage prior knowledge to constrain the search space *a priori* are inapplicable, given the variety of roles played by different latents. Lastly, since latents are specific to each model, surrogate models [14] or simulation techniques [25] are infeasible. We thus focus on prompt optimization methods which are self-contained, relatively inexpensive, and devoid of prior assumptions. The three methods below, which we use as baselines, meet these criteria.

#### 2.2.1 Greedy Coordinate Gradient

Greedy Coordinate Gradient (GCG) [26] optimizes directly in discrete input space, iteratively swapping tokens in a randomly initialized prompt. To constrain the search space, it uses a HotFlip-style gradient estimate [5]: backpropagating to a one-hot representation of the input yields, at each position, a gradient over the vocabulary, whose signed value approximates the loss impact of swapping in each token. At each iteration, a random position is selected for mutation, $B$ candidates are formed by sampling replacements from the $k$ most-promising tokens at that position, a forward pass scores them, and the best-loss candidate is retained.

#### 2.2.2 Beam Search

Sadasivan et al. [17] propose Beam Search-Based Adversarial Attack (BEAST), which sidesteps the discreteness problem with a gradient-free beam search: at each step, $k_2$ candidates per beam are produced by sampling and right-appending tokens from the model's own predicted distribution, and the $k_1$ lowest-loss candidates form the next beam. BEAST is fast (forward passes only), and a single forward pass both scores candidates and produces the logits used for the next expansion.

#### 2.2.3 Evolutionary Prompt Optimization

Evolutionary Prompt Optimization (EPO) [22] extends GCG with an evolutionary framework that balances activation and fluency. Rather than optimizing a single objective, EPO maintains a population of prompts along a Pareto frontier, with each member optimizing a different weighting between activation strength and self-cross-entropy:

$$\mathcal{L}_\lambda(t) := f(t) - \frac{\lambda}{n} \sum_{i=0}^{n-1} H(\text{LLM}(t_{\leq i}), t_{i+1}),$$

where $f(t)$ is the target latent activation for prompt $t$, $H(\text{LLM}(t_{\leq i}), t_{i+1})$ is the cross-entropy of the model's predicted distribution up to token $t_i$ against the next token $t_{i+1}$, $n$ is the total sequence length, and $\lambda \geq 0$ controls the trade-off. Several extensions have been proposed [18, 21], including ContextBench [8], which introduces a taxonomy of feature properties (namely, activation density, vocabulary diversity, and locality) that we adopt for our latent sampling.

## 3 ADAPT: Adaptive DynAmic Prompt Tuning

ADAPT extends GCG with elements designed to address limitations we observed in feature visualization. We describe its three main components below: beam-search initialization (§ 3.1), candidate generation via hybrid mutation (§ 3.2), and diversity-preserving evaluation and selection (§ 3.3). Key design decisions are validated in § 4.3. Figure 1 gives an overview of these stages, and pseudocode is provided in Algorithm 1 (Appendix A.1). Source code is available at github.com/joaoncardoso/ADAPT.

### 3.1 Initialization

During experiments with GCG, we found that the choice of initialization strongly affected whether a highly activating prompt was successfully found for a given latent. We also noticed that the optimization trajectory resulting from a given initialization was crucial, with prompts optimized in parallel often achieving drastically different final losses (see § 4.3).

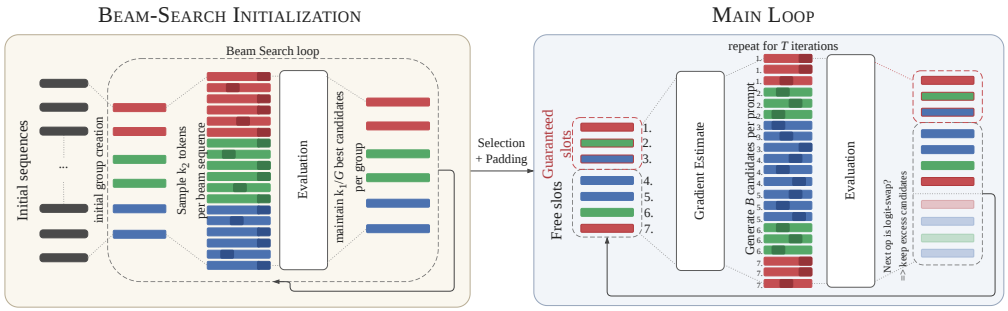



Figure 1: Diagram illustrating the stages in ADAPT.

ADAPT therefore initializes via a beam-search procedure, where several beams are maintained and optimized in parallel, forgoing the need to initialize with an arbitrary user-defined or randomly generated input sequence. Starting from a number of single-token sequences, candidates are generated by right-appending likely next tokens, after which point the lowest loss candidates are included in the next iteration's beam. The number of beams $G$ (one per group, see § 3.3), the number of initial sequences, the beam size $k_1$, and the expansion factor $k_2$, are specified by the user.

We observed that the structure of SAE-latent maximization does not lend itself well to sole use of right-append operations, since we take the latent activation magnitude at the last position, which is continuously relocated to that of the right-appended token. As a hypothetical example, for a latent that exclusively fires for the token `foo`, a right-append-only beam search would converge on the prompt "`··· foo foo foo`", whereas a better prompt might be obtained by refining the left-hand context instead. As such, the initialization procedure also performs middle-inserts, by randomly selecting a sequence position and sampling from the model's own predicted distribution at that position.

Each beam is independent, which means that our initialization is equivalent to a parallel run of more than one beam search procedure, each with its own initial starting set. The aim is to enable better exploration of the search space, decreasing the likelihood of stagnating in suboptimal local minima. The initialization returns the best prompts across all beams and iterations, without losing track of beam provenance, so that no cross-contamination takes place. The initialization may therefore return prompts of differing lengths.

### 3.2 Candidate Generation

ADAPT optimizes over an adaptively sized population of prompts, initialized to the best sequences returned by the beam-search initialization. At each iteration and for each prompt, a fixed, user-specified number of candidates is generated via one of two mutation mechanisms: GCG-style swaps, using the aforementioned gradient-based estimate, and logit-swaps, as in EPO. Mixing the two mechanisms does not increase peak activation, but makes ADAPT robust to whether the gradient signal is informative for a given latent (§ 4.3.4). We sample the next iteration's operation from a Bernoulli distribution with a user-specified probability, and retain an excess number of candidates in the surviving population if the next operation is a logit-swap, in order to take advantage of the lower computational requirements of the gradient-free logit-swap mutation. Mutation positions are selected either randomly or by sampling from a distribution over sequence positions that biases towards the selection of later positions. Since the initialization may return underlength prompts, we instead perform middle-inserts for logit-swap iterations.

### 3.3 Evaluation and Culling

Each candidate is first evaluated. Our evaluator begins by running a forward pass on the candidates, storing both the latent activations and the logits. Afterwards, we apply a fluency penalty mechanism. We employ a global sigmoid fluency penalty schedule, which gradually ramps up depending on several parameters. The penalty weight at any given time $\lambda_t$ is a function of the maximum penalty weight $\lambda_{\max}$ that was applied, the iteration of steepest increase $t_{\text{mid}}$, and the parameter $s$, which controls the steepness of the increase:

$$\lambda_t = \frac{\lambda_{\text{max}}}{1 + \exp(-s \cdot (t - t_{\text{mid}}))}.$$

The user may also choose a fluency threshold. Prompts with a mean cross-entropy score below this threshold do not incur any fluency penalty, while prompts above this threshold get penalized in proportion to how much their mean self-cross-entropy exceeds it. We allow the user to fully customize the fluency penalty strategy by setting the cross-entropy threshold to zero (disabling penalty thresholding), or the steepness $s$ to zero (disabling the fluency schedule entirely). We analyze the importance of fluency in § 4.3.3.

At each iteration, the population of prompts is refilled from the scored candidates into a fixed number of slots. Each prompt belongs to a *group*, namely the initialization beam it descends from. No cross-group selection takes place, to avoid rapid convergence on minor variations of the same prompt. A single highly-activating prompt is consistent with an arbitrary number of interpretations, and only significantly different prompts can enable our goal of supporting interpretability. Notwithstanding, the user may define a `mergepoint`, i.e. an iteration after which the selection mechanism defaults to a greedy global selection that disregards group provenance. Two types of slots exist. The first are group-specific "guaranteed" slots, which always contain the group's best scoring prompt across all iterations. The second set comprises "free" slots, which are allocated globally to the highest-scoring remaining candidates. This dual structure maintains diversity through the guaranteed slots, while allowing high-performing mutations from any group to claim additional representation through the free slots. With that said, we cap each group at twice its proportional share of free slots.

# 4 Experiments

In this section we begin by conducting a comparative evaluation of ADAPT against GCG, BEAST, and EPO. Afterwards, we perform individual experiments to validate individual design decisions.

## 4.1 Experimental Setup

All methods were compared at matched computational cost. Baselines were run with their authors' published defaults, and ADAPT's parameters were set to the same values wherever the methods share them. Only budget parameters, such as candidate and iteration counts, were adjusted, so that each method took approximately 80s per latent on a single NVIDIA A40 48GB GPU. Each method optimized several prompts in parallel to make full use of this budget. We did not tune hyperparameters for performance, since tuning ADAPT alone would bias the comparison, while tuning all four methods was beyond our computational budget. Moreover, activation is only a measurable proxy for interpretability, limiting the value of a comprehensive sweep. Full settings are listed in Appendix A.1.

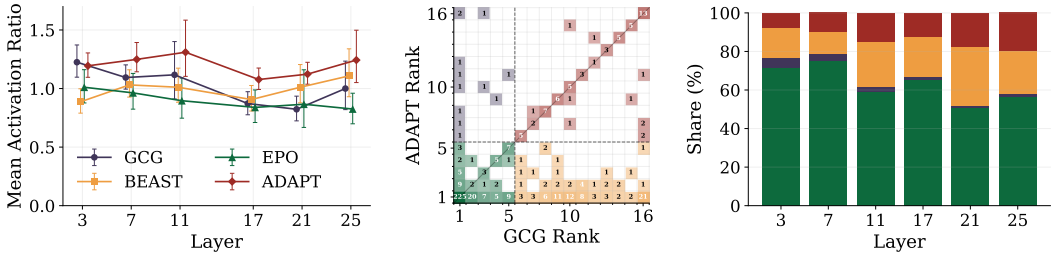
Our experiments were performed on Gemma 2 2B [6], in keeping with previous work [18]. We used the 16k-wide residual stream SAE suite from Gemma Scope [9]. The residual stream represents the cumulative computation accessible to all downstream layers, making it the natural target for understanding what information the model maintains and routes through its depth.

For our comparative experiments, we curated a set of 486 latents sampled from 6 evenly spaced layers (layers 3, 7, 11, 17, 21, and 25) across the model's 26-layer depth. We applied a stratified sampling procedure, which is detailed in Appendix A.2.

Despite the use of fluency penalties in EPO and ADAPT, we consider the raw activation values in our metrics. Since activation magnitudes differ across model latents and layers, our metrics compare the optimized prompt's activation with the corresponding Maximally Activating Examples (MAEs) in two different ways: the activation ratio, corresponding to the ratio between the optimized prompt and the top MAE, and the activation rank, which corresponds to the position which the optimized prompt would occupy if ranked among the top 15 de-duplicated MAEs. A prompt with a rank of 1 outperforms all MAEs and, consequently, its activation ratio is greater than 1, while a prompt with rank 16 is worse than all 15 MAEs.

| | GCG | BEAST | EPO | ADAPT |
|---|---|---|---|---|
| GCG | – | 56.4% | 63.2% | 29.6% |
| BEAST | 43.6% | – | 50.2% | 12.8% |
| EPO | 36.8% | 49.8% | – | 21.4% |
| ADAPT | 70.4% | 84.0% | 78.6% | – |
| Mean ratio | 1.022 | 0.992 | 0.900 | **1.199** |
| 95% CI | [0.957, 1.091] | [0.936, 1.057] | [0.834, 0.974] | [1.138, 1.269] |

Table 1: Top: share of the 486 latents on which the row method beats the column method (pairs may not sum to 100% due to ties). Bottom: mean activation ratio per method, with percentile bootstrap 95% confidence intervals (10,000 resamples).




(a) Mean activation ratio.

(b) Rank matrix.

(c) Quadrant shares per layer.

Figure 2: Performance overview. (a) Mean activation ratio (optimized prompt activation divided by the maximally activating dataset example) per layer, for all 4 methods; error bars are 95% bootstrap confidence intervals across latents. Since all methods are evaluated on the same latents, comparisons between methods are paired: ADAPT beats GCG on 70.4% of latents (Wilcoxon signed-rank $p < 10^{-28}$). (b) Activation-rank confusion matrix for the GCG-ADAPT comparison. (c) Per-layer share of joint and asymmetric success/failure quadrants for the GCG-ADAPT comparison.

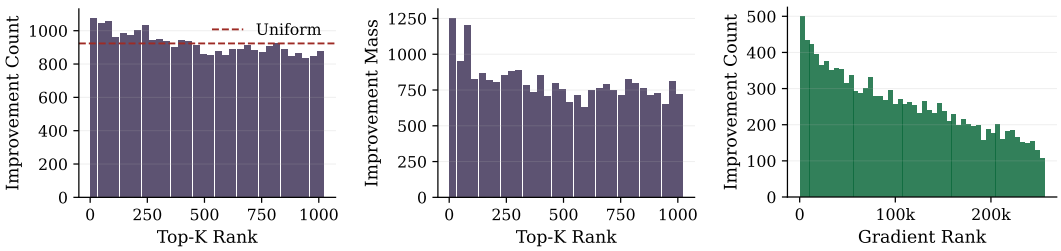
### 4.2 Overall Performance

Figure 2a shows the mean activation ratio per considered layer and for all methods, showing that ADAPT outperforms or is on par with every other method on all layers. Among the previous methods, GCG is dominant for the first layers, while BEAST edges it out on the last layers, indicating a degrading performance profile for GCG toward later layers. Furthermore, the failure modes of GCG and BEAST appear to be complementary, which is corroborated by the fact that ADAPT, which incorporates both a beam-search procedure and gradient-based mutations, achieves a more constant performance profile throughout the layers. In turn, the comparison in Table 1 shows that ADAPT achieves higher activation than GCG on 70.4% of the features (Wilcoxon signed-rank $p < 10^{-28}$), further corroborating the improved performance profile.

EPO performs comparably to GCG and BEAST (Table 1). Since its distinguishing feature is fluency, we revisit that question separately in § 4.3.3, and forgo further EPO-specific analysis here.

On two additional models, with configurations unchanged (Appendix A.4), ADAPT outperforms all baselines on Gemma 3 1B [7] (beating GCG on 81.2% of latents, $p < 10^{-8}$), and outperforms BEAST and EPO on GPT-2 Small [15] while performing on par with GCG (56.2%, $p = 0.077$). GCG's gradient signal may be more informative in smaller, shallower models such as GPT-2 Small.

Figure 2b shows the activation rank confusion matrix for the GCG-ADAPT comparison. Arbitrarily considering that a rank less than or equal to 5 corresponds to a successful optimization, the green and red quadrants indicate joint success and failure, respectively, while the other quadrants indicate asymmetric success. In turn, the stacked bar chart in Figure 2c illustrates the per-layer share of these quadrants, showing that the share of GCG-only successes is marginal, while ADAPT-only successes are substantial, further establishing the superiority of the method. We further corroborate ADAPT's advantage with human ratings in § 4.3.3.




(a) Number of improvements per estimate rank bin.

(b) Improvement mass per estimate rank bin.

(c) GCG estimate rank distribution for improvements obtained via uniform-random sampling.

Figure 3: GCG estimate signal-to-noise analysis. (a) Improvement counts. (b) Improvement mass per gradient rank bin for candidates generated via the GCG estimate. (c) Rank distribution for improvement candidates obtained via uniform-random sampling from the vocabulary.

### 4.3 Validation Experiments

To diagnose failure modes more carefully, we conduct targeted experiments on two smaller subsets sampled from the 486-latent set: a balanced subset of 50 latents drawn evenly across layers, and a hardness-weighted subset of 50 latents drawn from the final three layers (where GCG performs worst) and skewed toward latents that resist GCG optimization. Hardness was estimated as the inverse of GCG's mean activation ratio across iterations and slots (Appendix A.3).

#### 4.3.1 The Signal-to-noise Ratio in the GCG Estimate

The gradient-based estimate used in GCG (henceforth, the 'GCG estimate') is predicated on the assumption that the loss impact of discretely including each token is well-approximated by its corresponding gradient computed for the one-hot encoding. If the GCG estimate were a perfect approximation, we would expect low-ranked tokens (with the most negative gradient) to lead to the greatest loss decreases. Conversely, we would also expect the tokens resulting in the largest loss decreases to have the lowest gradients.

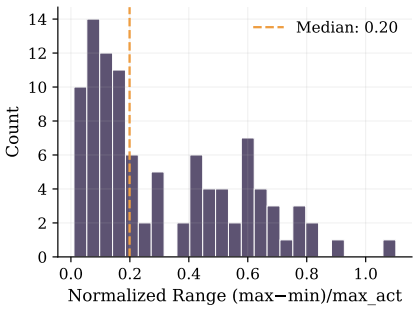
In order to assess these two claims, we perform the following experiment on the balanced latent set. For each latent, we begin by determining the two slots which achieved the highest and lowest final activation, respectively. For each slot, we obtain the cached prompt trajectory from the main run. Afterwards, we generate 128 new candidate mutations in two ways. The first 64 are generated via the same GCG estimate procedure as in the original GCG, with a top-$k$ of 1024 (doubling the candidate count and top-$k$). The last 64 are generated by randomly sampling tokens from the entire vocabulary, nevertheless keeping track of each token's GCG estimate position. We register the resulting candidates which constitute improvements over the cached prompt's activation. We also fix the trajectory to ensure comparable results between the two candidate generation methods.

Figure 3a shows that the gradient rank distribution for improvements obtained via GCG is quite close to uniform, indicating that lower gradient values do not yield remarkably more improvements. Figure 3b, on the other hand, shows that the distribution of the sum of improvements is more uneven, with the lower gradient bins containing a disproportionate share of the absolute improvements.

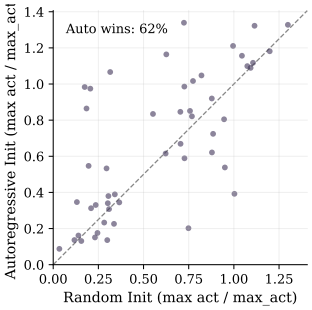
Figure 3c shows the gradient rankings for improvements obtained by sampling uniformly from the vocabulary. While lower ranks do produce more improvements, a significant number of improvements still originate from rankings which, given the usual top-$k$, would not be considered in GCG.

#### 4.3.2 Consistency of GCG

We now examine how sensitive GCG is to initialization and trajectory. For the benchmark in § 4.2, GCG was configured to optimize six prompts in parallel, and we observed substantial inter-slot variability in final activation. We therefore ask two further questions: whether specific initial prompts reliably predict higher final activations, and whether GCG attains similar activations across different runs initialized with the same prompt.




(a) Variability from identical initializations.




(b) Autoregressive vs. random init.

Figure 4: GCG sensitivity analysis. (a) Distribution of activation variability across runs from identical initializations, normalized by the MAE. (b) Best activations from autoregressive versus random initializations, with points above the diagonal indicating autoregressive advantage.

To answer these questions, we conduct a simple experiment. We begin by determining which of the 6 initial prompts in the main benchmark achieved both the highest and lowest average final activations. Since the highest-performing prompt was obtained through autoregressive generation, and the lowest-performing prompt consists of randomly sampled tokens from the vocabulary, we call them the "autoregressive" and "random" prompts, respectively. Afterwards, we conduct a benchmark on the hardness-weighted latent set, where 3 of the 6 initial slots are initialized with the autoregressive prompt, and the remaining 3 are initialized with the random prompt.

Figure 4a shows the distribution of activation ranges across runs from identical initializations, normalized by maximal dataset activation. The median variability of 0.20 indicates that runs from the same starting point can diverge substantially, suggesting that GCG is prone to converging on different local minima depending on the stochastically-selected trajectory through prompt space.

In turn, Figure 4b compares best activations achieved from autoregressive versus random initializations. Autoregressive initialization outperforms random initialization in 62% of configurations, demonstrating that the starting location in prompt space meaningfully influences optimization outcomes, but does not fully predict them.

#### 4.3.3 The Role of Fluency

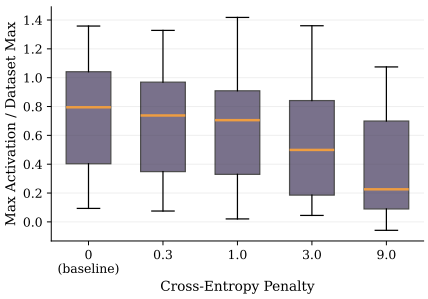
GCG usually produces 'gibberish' prompts, which contain incoherent token sequences mixing languages, scripts, punctuation, and subword fragments. This is due to the lack of any constraint on possible token sequences. In the adversarial domain, fluency is usually enforced to bypass perplexity-based filters and to obtain in-distribution prompts. However, it is not immediately apparent that the same desiderata apply to feature visualization. For one, perplexity filters have no analogue in this domain; for another, adversarial behavioral objectives admit multiple sufficient mechanisms (e.g., simultaneously suppressing refusal directions and hijacking attention [1]), whereas targeted latent activation objectives specify the intervention site directly, achieving mechanistic specificity by construction rather than requiring auxiliary constraints.

Given the attributes of the feature visualization domain, two putative roles for fluency are relevant: regularization, preventing optimization from getting stuck in local minima and returning low-activation prompts; and interpretability, preventing high-activation solutions that are uninterpretable or unrelated to the feature pattern [22].

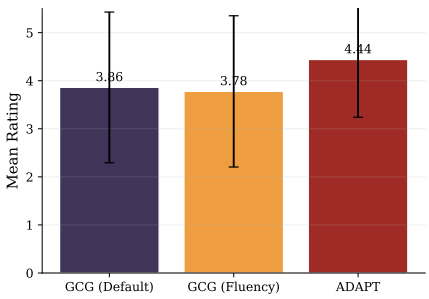
The first hypothesis suggests that a fluency penalty may reshape the loss landscape, smoothing over narrow valleys and sharp local minima that trap optimization in gibberish regions. To test this, we rerun GCG on the hardness-weighted latent set with an added self-cross-entropy penalty:

$$\mathcal{L}(t) = -f(t) + \frac{\lambda}{n} \sum_{i=0}^{n-1} H(\text{LLM}(t_{\leq i}), t_{i+1}),$$

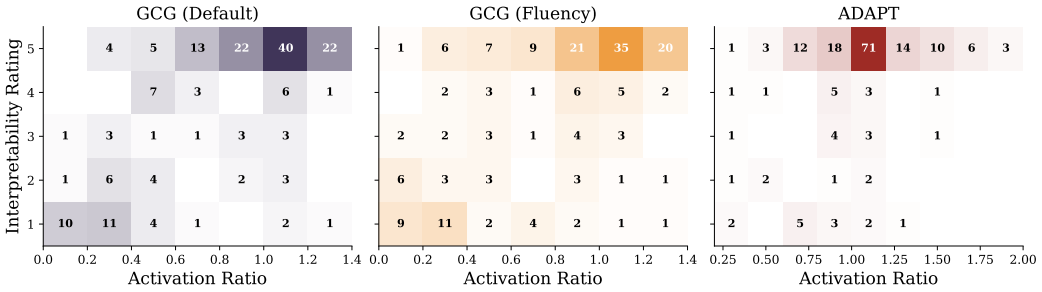
where $f(t)$ is the target latent activation, $H(\cdot, \cdot)$ is the cross-entropy, and $\lambda \in \{0.3, 1.0, 3.0, 9.0\}$ controls the fluency-activation tradeoff.




(a) Activation ratio for varying fluency penalty weights.




(b) Mean rating per method.




(c) Ratio/rating co-occurrence matrix.

Figure 5: Fluency penalty and human-rated interpretability results for standard GCG, GCG with a fluency penalty parameter $\lambda$ set to 1.0, and ADAPT. (a) Activation ratio distribution for GCG runs with varying fluency penalty weights. (b) Mean rating averaged over all latents and annotators, with error bars corresponding to 95% confidence intervals. ADAPT is rated higher than GCG on paired comparisons (Wilcoxon signed-rank, $p < 0.003$). (c) Ratio/rating co-occurrence.

Figure 5a shows that higher values of $\lambda$ hinder optimization across the whole latent set, thereby undermining the hypothesis that was put forward. Notwithstanding, we highlight that this experiment does not directly bear on the fluency configuration used in the ADAPT benchmark, since higher penalties are only applied in later iterations, leaving the initialization and early optimization periods relatively unhindered by fluency.

To assess whether fluency penalties improve interpretability and to validate ADAPT's design, we conducted a manual evaluation comparing three conditions: GCG, GCG with a fluency penalty ($\lambda = 1.0$), and ADAPT. Three raters (including one author) evaluated prompts on 60 latents: the 50-latent hardness-weighted set plus 10 additional latents with activation ratio greater than 1.0, included to better probe whether high-activation prompts recover interpretable patterns. For each latent, raters viewed the latent's explanation, maximally activating dataset examples (but not their corresponding activation), and three prompts (one per condition, order randomized). Since some latents encode patterns that are hard to infer from dataset examples alone (e.g., upweighted predictions or structural properties of the text), explanations were written by the authors[2]. Prompts were rated on a 1–5 scale (Appendix A.5). We computed Krippendorff's $\alpha$ for inter-rater agreement, obtaining $\alpha = 0.650$.

Figure 5c shows that, for GCG, higher activation reliably correlates (Pearson $r = 0.645, p < 10^{-5}$) with more interpretable prompts, despite the lack of fluency constraints. Indeed, high activation (ratio$> 1.0$) prompts seem overwhelmingly interpretable, whereas the correlation is less clear for lower activation regimes. The $\lambda = 1.0$ run shows similar patterns, with no remarkable improvement in interpretability.

Our analysis also establishes that ADAPT is significantly better at generating interpretable prompts. Figure 5b shows the mean rating for each of the methods, establishing the superiority of ADAPT (Wilcoxon signed-rank statistic of 460.5, $p < 0.003$), while showing that GCG with $\lambda = 1.0$ does not produce significant changes in human-rated interpretability. Lastly, we note that because the

[2]The annotation file is available in our code repository.

| | Balanced subset | | | Hardness-weighted subset | | |
|---|---|---|---|---|---|---|
| Condition | Median $\Delta$ | $p$ | Worst 10% | Median $\Delta$ | $p$ | Worst 10% |
| *Mutation mix, $p_{gcg}$* | | | | | | |
| 0.00 (logit-only) | $-0.040$ | 0.0083 | 0.162 | $+0.008$ | 0.77 | 0.386 |
| 0.25 | $+0.004$ | 0.31 | 0.550 | | | |
| 0.50 (full ADAPT, default) | – | – | 0.425 | – | – | 0.354 |
| 0.75 | $-0.000$ | 0.59 | 0.496 | | | |
| 1.00 (gradient-only) | 0.000 | 0.70 | 0.373 | $-0.012$ | 0.47 | 0.315 |
| *Component removed* | | | | | | |
| No beam-search init | $-0.073$ | 0.0031 | 0.132 | $-0.106$ | 0.00025 | 0.116 |
| No diversity groups | $+0.040$ | 0.021 | 0.316 | $+0.032$ | 0.0054 | 0.435[†] |

Table 2: Ablations of ADAPT on Gemma 2 2B (50 latents per subset). Median $\Delta$: median paired difference in activation ratio versus full ADAPT; $p$: Wilcoxon signed-rank; Worst 10%: mean activation ratio over the worst-performing decile of latents. Intermediate values of $p_{\text{gcg}}$ were run on the balanced subset only. [†]Average of per-seed values over 3 seeds.

activation ratio is a mere proxy for human interpretability, these results provide the most compelling evidence in support of our design decisions in ADAPT.

#### 4.3.4 Ablation Study

To validate ADAPT's components, we remove one at a time and compare against full ADAPT on both validation subsets (Table 2). We treat the balanced subset as primary, since the hardness-weighted subset targets latents on which gradients struggle.

Consistent with GCG's sensitivity to initialization (§ 4.3.2), replacing beam-search initialization with a fixed prompt causes the largest drop on both subsets ($\Delta = -0.07$ and $-0.11$), and the worst decile collapses ($0.43 \to 0.13$ and $0.35 \to 0.12$).

Mixing mutation mechanisms provides robustness rather than higher peak activation. On the balanced subset, gradient-only mutation matches full ADAPT, while logit-only is significantly worse (worst decile $0.16$ versus $0.43$). On the hardness-weighted subset, the ordering reverses slightly, without significance. The hybrid approach thus matches the stronger pure variant in each regime. Performance is also insensitive to $p_{\text{gcg}}$ between $0.25$ and $1$ (all $p \geq 0.31$), so the untuned default of $0.5$, as discussed in § 4.1, is not critical.

Removing diversity groups slightly increases activation ($\Delta = +0.04$ and $+0.03$), but more than halves prompt diversity (mean pairwise Jaccard distance $0.80$ versus $0.34$, lower for every latent). Since a single prompt is consistent with many interpretations, this modest cost buys distinct prompts from which to triangulate the latent's pattern (§ 3.3).

## 5 Conclusions

We presented ADAPT, a hybrid prompt optimization method for LLM feature visualization. Combining beam-search initialization, adaptive mixing of gradient-guided and logit-based mutations, and diversity-preserving slot management, ADAPT outperforms GCG on 70.4% of SAE features ($p < 10^{-28}$), maintains consistent performance across layers, and produces significantly more interpretable prompts in human evaluation. These results establish that feature visualization for LLMs is tractable when methods are tailored to the domain's discrete optimization and local-minima-prone loss landscape, and open the door to its application to steering vectors, circuit components, and other latent directions where exhaustive dataset search is prohibitive.

**Limitations and Future Work.** Our evaluation focused on residual stream SAEs from Gemma 2 2B, with smaller-scale evaluations on GPT-2 Small and Gemma 3 1B. Extending ADAPT to more diverse latent extraction methods, and automated interpretability pipelines that correlate with human judgments [12], is left for future work.

## Acknowledgments and Disclosure of Funding

This work was supported by national funds through Fundação para a Ciência e a Tecnologia, I.P. (FCT) under project UID/50021/2025 (DOI: 10.54499/UID/50021/2025), and also by the Recovery and Resilience Fund through the Center for Responsible AI project (Ref. C628696807-00454142). The authors declare no competing interests.

## References

[1] Andy Arditi, Oscar Balcells Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, 2024.

[2] 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.

[3] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL `https://transformer-circuits.pub/2023/monosemantic-features/index.html`.

[4] Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 3201–3219. Association for Computational Linguistics, 2024.

[5] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36. Association for Computational Linguistics, 2018.

[6] Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[7] Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[8] Robert Graham, Edward Stevinson, Leo Richter, Alexander Chia, Joseph Miller, and Joseph Isaac Bloom. ContextBench: Modifying contexts for targeted latent activation. *arXiv preprint arXiv:2506.15735*, 2025.

[9] Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma Scope: Open sparse autoencoders everywhere all at once on Gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 278–300. Association for Computational Linguistics, 2024.

[10] Maxime Méloux, Giada Dirupo, François Portet, and Maxime Peyrard. The dead salmons of AI interpretability: The need for a statistical inference perspective. *arXiv preprint arXiv:2512.18792*, 2025.

[11] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11), 2017. doi: 10.23915/distill.00007.

[12] Gonçalo Paulo and Nora Belrose. Evaluating SAE interpretability without explanations. *arXiv preprint arXiv:2507.08473*, 2025.

[13] Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. *arXiv preprint arXiv:2410.13928*, 2024.

[14] Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. AdvPrompter: Fast adaptive adversarial prompting for LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

[16] Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, Haozhu Wang, Han Ding, Yuzhe Lu, Zhichao Xu, Yun Zhou, Balasubramaniam Srinivasan, Qiaojing Yan, Yueyan Chen, Haibo Ding, Panpan Xu, and Lin Lee Cheong. A systematic survey of automatic prompt optimization techniques. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33078–33110. Association for Computational Linguistics, 2025.

[17] Vinu Sankar Sadasivan, Shoumik Saha, Gaurang Sriramanan, Priyatham Kattakinda, Atoosa Chegini, and Soheil Feizi. Fast adversarial attacks on language models in one GPU minute. *arXiv preprint arXiv:2402.15570*, 2024.

[18] Stepan Shabalin, Daniel Tan, Dmitrii Kharlapenko, Neel Nanda, and Arthur Conmy. Evolutionary prompt optimization for SAE feature visualization. *AI Alignment Forum*, 2024. URL `https://www.alignmentforum.org/posts/hMBTaFvAzdMNnj29c`.

[19] Lee Sharkey, Dan Braun, and Beren Millidge. Taking features out of superposition with sparse autoencoders. *AI Alignment Forum*, 2022.

[20] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

[21] T. Ben Thompson and Michael Sklar. FLRT: Fluent student-teacher redteaming. *arXiv preprint arXiv:2407.17447*, 2024.

[22] T. Ben Thompson, Zygimantas Straznickas, and Michael Sklar. Fluent dreaming for language models. *arXiv preprint arXiv:2402.01702*, 2024.

[23] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. PromptAgent: Strategic planning with language models enables expert-level prompt optimization. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[24] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[25] Haoting Zhang, Jinghai He, Rhonda Righter, and Zeyu Zheng. Language model prompt selection via simulation optimization. *arXiv preprint arXiv:2404.08164*, 2024.

[26] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

# A Appendix

## A.1 Implementation Details

Table 3 provides detailed hyperparameter settings for all methods evaluated in this work, while Algorithm 1 presents the pseudocode for ADAPT.

| Parameter | GCG | BEAST | EPO | ADAPT |
|---|---|---|---|---|
| *General* | | | | |
| Iterations | 50 | 10 | 50 | 25 |
| Prompt length | 10 | 10 | 8–15 | 10 |
| Parallel prompts | 6 | – | 6 | 10 |
| *Candidate Generation* | | | | |
| Candidates/iter | 32 | – | 32 | 32 |
| Top-$k$ (gradient) | 512 | – | 512 | 512 |
| Beam size | – | 128 | – | 30 |
| Expansion factor | – | 32 | – | 32 |
| *Operation Probabilities* | | | | |
| GCG-swap | 1.0 | – | 0.50 | 0.50 |
| Logit-swap | – | – | 0.20 | 0.50 |
| Insert | – | – | 0.15 | – |
| Delete | – | – | 0.15 | – |
| *Fluency* | | | | |
| $\lambda_{\min}$ | – | – | 0.1 | 0.0 |
| $\lambda_{\max}$ | – | – | 15.0 | 3.0 |
| *ADAPT-specific* | | | | |
| Diversity groups | – | – | – | 3 |
| Guaranteed slots/group | – | – | – | 2 |
| Init sequences | – | – | – | 5000 |
| Steepness $s$ | – | – | – | 0.1 |
| Cross-entropy threshold $\tau$ | – | – | – | 9.0 |

Table 3: Hyperparameter settings for all evaluated methods.

## A.2 Latent Selection Procedure

We computed proxies for the three taxonomic axes described by Graham et al. [8] using data available via the Neuronpedia API.

**Density.** A latent's density corresponds to the frequency with which it activates. We take the fraction of tokens for which the latent is active, which is directly provided by the Neuronpedia API as `frac_nonzero`.

**Locality.** Locality describes how concentrated a latent's activation is within a sequence. A highly local latent exhibits high activation for a single token with negligible activation for surrounding tokens, whereas a low-locality latent may activate across entire phrases. Given a set of $N$ maximally activating examples for a latent, let $a^{(j)} = (a_1^{(j)}, \ldots, a_{L_j}^{(j)})$ denote the activation values across all $L_j$ tokens in example $j$. We define locality as:

$$\text{Locality} = \frac{1}{N} \sum_{j=1}^{N} \frac{\max_i a_i^{(j)}}{\sum_i a_i^{(j)}}.$$

For a highly local latent, this ratio approaches 1, as the maximally activating token captures most of the activation mass.

**Diversity.** Diversity indicates the variety of tokens a latent responds to. A low-diversity latent fires for a specific token, while a high-diversity latent activates for many different tokens sharing a

**Algorithm 1** The ADAPT optimization procedure

**Require:** Target latent $f$, model $M$, SAE $S$, iterations $T$, prompts $K$, groups $G$
**Ensure:** Optimized prompt $t^*$

$\mathcal{P} \leftarrow$ BEAMSEARCHINIT$(f, M, S, G)$ {$K$ prompts across $G$ groups}
**for** $i = 1$ **to** $T$ **do**
  Sample operation op $\sim$ Bernoulli$(p_{\text{gcg}})$
  **if** op $=$ GCG-SWAP **then**
    Compute gradient estimate $\nabla$ via one-hot embedding
    $\mathcal{C} \leftarrow$ GRADIENTMUTATE$(\mathcal{P}, \nabla)$ {Low-gradient tokens}
  **else**
    $\mathcal{C} \leftarrow$ LOGITMUTATE$(\mathcal{P})$ {Model's predictions}
  **end if**
  **for** each candidate $c \in \mathcal{C}$ **do**
    $a_c \leftarrow f(c)$
    $\lambda_i \leftarrow \lambda_{\max}/(1 + e^{-s(i - t_{\text{mid}})})$
    $\text{score}_c \leftarrow a_c - \lambda_i \cdot \max(0, \text{CE}(c) - \tau)$
  **end for**
  **for** $g = 1$ **to** $G$ **do**
    Keep top candidates from group $g$ in guaranteed slots
  **end for**
  Fill remaining slots globally by score (with per-group cap)
  $\mathcal{P} \leftarrow$ selected candidates
**end for**
**return** $\arg\max_{t \in \mathcal{P}} f(t)$

semantic or functional role. Given the top $N$ maximally activating examples, let $T = \{t_1, \ldots, t_N\}$ be the set of tokens achieving maximal activation in each example. We define diversity as:

$$\text{Diversity} = \frac{|\text{lowercase}(T)|}{N},$$

where lowercase$(T)$ applies lowercasing to all tokens before computing the unique set size. This normalization accounts for capitalization variants of the same underlying token.

### A.3 Hardness Metric and Subset Sampling

We assess the difficulty of optimizing a given latent via the inverse of GCG's average activation ratio across all iterations and prompt slots from the main benchmark run. Equivalently, this corresponds to the ratio between the area under the trajectory curve and the area below $y = a_{\max}$, where $a_{\max}$ is the maximal dataset activation. An easy latent is one for which GCG rapidly converges on a solution at or above $a_{\max}$ across all 6 slots, yielding an AUC ratio of 1 or below; a latent for which all trajectories plateau below $a_{\max}$ yields a hardness far greater than 1. Formally, the hardness $H_l$ for a latent $l$ is:

$$H_l = \left( \frac{1}{S \cdot a_{\max} \cdot T} \sum_{s=1}^{S} \sum_{t=1}^{T} \max(a_s^{(t)}, 0) \right)^{-1},$$

where $S$ is the number of prompt slots, $T$ the number of iterations, and $a_s^{(t)}$ the activation for slot $s$ at iteration $t$.

We split latents into 5 hardness quintiles and sampled the two 50-latent subsets accordingly: the balanced subset draws 10 features from each quintile, while the hardness-weighted subset draws (5, 8, 10, 12, 15) features from the easiest to hardest quintiles, respectively.

### A.4 Additional Models

We evaluated all four methods on GPT-2 Small and Gemma 3 1B, using the same configurations as in the main experiments. For each model, we sampled 80 residual-stream SAE latents uniformly at random, 20 from each of four layers: layers 3, 6, 8 and 10 for GPT-2 Small (`res-jb` SAEs), and

| | GPT-2 Small | | Gemma 3 1B | |
|---|---|---|---|---|
| Method | Mean ratio [95% CI] | ADAPT wins ($p$) | Mean ratio [95% CI] | ADAPT wins ($p$) |
| GCG | 1.013 [0.921, 1.102] | 56.2% (0.077) | 1.048 [0.937, 1.171] | 81.2% ($5 \times 10^{-9}$) |
| BEAST | 0.885 [0.810, 0.951] | 88.8% ($3 \times 10^{-8}$) | 1.207 [1.067, 1.394] | 87.5% ($3 \times 10^{-7}$) |
| EPO | 0.878 [0.782, 0.972] | 72.5% ($1.5 \times 10^{-5}$) | 1.005 [0.893, 1.117] | 72.5% ($2 \times 10^{-6}$) |
| ADAPT | **1.072** [0.988, 1.151] | – | **1.321** [1.164, 1.535] | – |

Table 4: Results on GPT-2 Small and Gemma 3 1B (80 randomly sampled latents per model, 20 per layer). "ADAPT wins" corresponds to the share of latents where ADAPT attains a higher activation ratio than the baseline, with paired Wilcoxon signed-rank $p$.

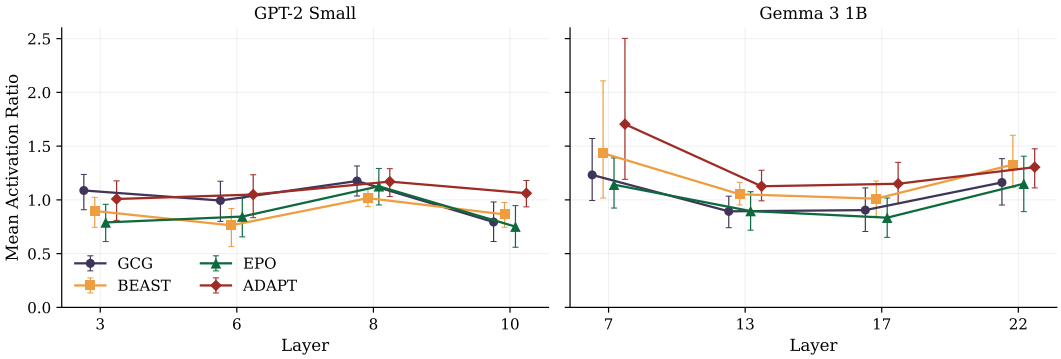



Figure 6: Mean activation ratio per layer on GPT-2 Small and Gemma 3 1B (80 randomly sampled latents per model, 20 per layer). The error bars correspond to 95% bootstrap confidence intervals of the per-layer mean.

layers 7, 13, 17 and 22 for Gemma 3 1B (Gemma Scope 2, 16k width), i.e. the only Gemma 3 layers for which Neuronpedia reference statistics are available. Gemma 2 2B remains our main testbed, in keeping with prior work [18]. The sample size was chosen by a power simulation on the Gemma 2 2B paired results. Table 4 summarizes the results, and Figure 6 shows them per layer.

### A.5 Human Evaluation Protocol

Annotators were presented with the following information for each latent: (1) an explanation of what the latent detects, (2) maximally activating dataset examples with highlighted tokens showing activation strength, and (3) three generated prompts from different methods in randomized order, with the 3 most likely token completions in square brackets to aid in rating 'Say X' features. Annotators could also access interactive exploration via Neuronpedia. Figure 7 shows a single latent's annotation interface. Prompts were rated on the following 1–5 scale:

1. **No discernible pattern**: The prompt appears random or unrelated to the latent's function.
2. **Semantic ballpark**: The prompt is in the general semantic area of the reference examples but is otherwise uninterpretable.
3. **Pattern understandable**: The pattern that the prompt is targeting is understandable given context, but not discernible from the optimized prompt alone.
4. **Pattern present but different**: The pattern is clearly present but differs in some way from reference examples (e.g., different language, domain, or format).
5. **Pattern obvious and matches**: The pattern is immediately obvious and closely matches the maximally activating dataset examples.

### A.6 Broader Impacts

While feature visualization supports interpretability efforts, prompt optimization methods can in principle be repurposed for adversarial attacks on language models. We note, however, that ADAPT

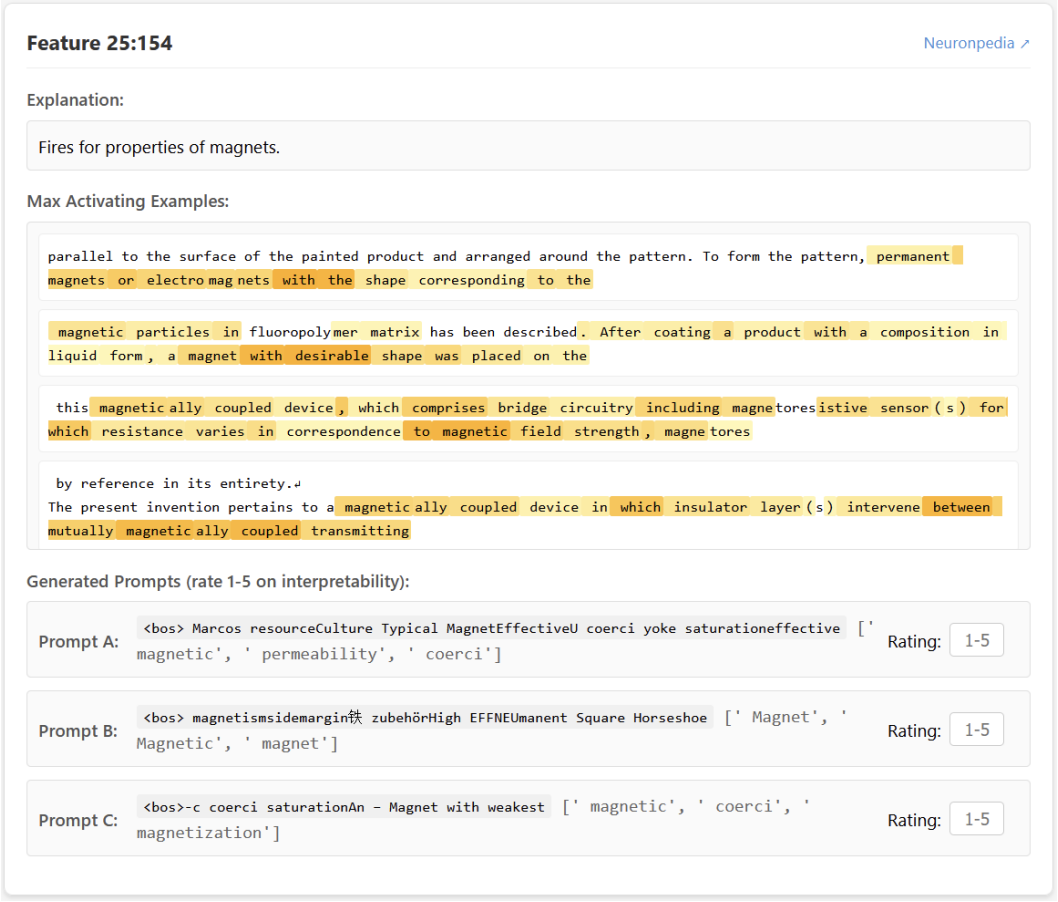



Figure 7: Annotation task example for the ‘magnet’ latent.

is designed for short prompts that target internal representations, rather than behavioral objectives, thus limiting direct misuse potential.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims clearly reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are clearly discussed in the Conclusions section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: There are no significant theoretical contributions in the paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Hyperparameters are listed clearly, and the relevant code is included in the supplementary materials.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the source code in the supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes. Hyperparameters for all four methods are specified in §4.1 and Appendix A.1. Latent sampling and the hardness-weighted subset construction are described in Appendices A.2 and A.3. The human evaluation protocol is in §4.3.3 and Appendix A.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes. We report Wilcoxon signed-rank tests for the headline pairwise comparisons, Pearson correlation with p-value for the activation/interpretability relationship, and Krippendorff's $\alpha$ for inter-rater agreement. Mean activation ratios are reported with percentile bootstrap 95% confidence intervals, overall and per layer (Table 1, Figure 2a, Appendix A.4), and ablations are reported as paired median differences with Wilcoxon signed-rank tests (Table 2).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the number of latents and the runtime per latent on an Nvidia A40 GPU.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have carefully reviewed the NeurIPS Code of Ethics and confirm that this paper aligns with it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader impacts are discussed in Appendix A.6. Improved feature visualization supports interpretability, alignment, and scientific understanding of LLM internals; conversely, prompt optimization methods could in principle be repurposed for adversarial attacks, although ADAPT targets internal representations with short prompts rather than behavioral objectives.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release model weights or scraped datasets. Our contribution is a prompt optimization method evaluated on publicly available models and SAEs.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use Gemma 2 2B (Gemma Terms of Use, `https://ai.google.dev/gemma/terms`), the Gemma Scope 16k-wide residual stream SAE suite (CC-BY-4.0; [9]), Gemma 3 1B (Gemma Terms of Use), the Gemma Scope 2 SAEs (CC-BY-4.0), GPT-2 Small (MIT; [15]), the GPT-2 Small `res-jb` SAEs (MIT), and the Neuronpedia API for latent statistics (`https://neuronpedia.org/api-doc`).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The supplementary materials include the ADAPT implementation, run scripts, and the human-rating annotation file, with a README documenting usage.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: The rating protocol is shared in Appendix A.5. Annotators participated voluntarily. No personal data was collected and no compensation was provided.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

Justification: The rating task posed no foreseeable risk to participants and did not meet our institution's threshold for IRB review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: An LLM is used merely as the target for our method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.