Steered Generation via Gradient-Based Optimization on Sparse Query Features

Anonymous authors
Paper under double-blind review

Abstract

Latent steering exploits internal representations of Large Language Models (LLMs) to guide generation, yet interventions on dense states can entangle distinct semantic features. In this paper, we investigate attention query activations as a high-fidelity site for precise control, hypothesizing that manipulating the attention mechanism itself offers sharper steerability than general state interventions. We introduce Prototype-Based Sparse Steering, a framework that applies Sparse Autoencoders (SAEs) specifically to query activations, to decompose them into interpretable features, then apply gradient-based optimization during inference to align the sparse representation with class prototypes of target behaviors. To validate this architectural insight, we first analyze the mechanism in Textualized Gridworld, a controlled environment for verifiable planning constraints. We demonstrate that optimizing sparse query features enables effective navigation of rigid planning requirements (i.e., safe vs. short paths), confirming the method's ability to satisfy objective rules. We then demonstrate the framework's versatility by training SAEs on a high-dimensional educational domain, where the framework steers the cognitive complexity of feedback (i.e., Bloom's Taxonomy). Our experiments establish that sparse query representations provide the necessary disentanglement for unified, interpretable control over both logical planning and stylistic nuance.

1 Introduction

LLMs have emerged as the de facto natural language interface for modern AI systems. However, deploying these models in real-world scenarios requires more than fluent generation; it necessitates precise adherence to user-defined requirements regarding style, safety, and logical consistency. Consequently, research on controlled text generation in LLMs has been extensively explored for general-purpose applications (see the comprehensive survey by Zhang et al. (2023)). Unlike standard text generation, controlled text generation (Mireshghallah et al., 2022) guides LLMs to generate outputs that adhere to specific constraints.

To achieve this control, prompt engineering has become a prevalent strategy. However, controlling text generation solely through prompting can be ineffective (Ye & Durrett, 2022), as prompts often require extensive contextual information to generate relevant responses. When additional instructions for controlling style and nuance are included, they may be diluted or ignored within long prompt contexts (Liu et al., 2023a). Moreover, open-source and smaller LLMs provide limited control over generated text when relying solely on prompt-based methods (Salinas & Morstatter, 2024). In practice, this gap is most visible in smaller open-source models: the same instruction that yields stable stylistic adherence in frontier models can produce inconsistent or diluted control in smaller models. A common mitigation is to include a few labeled demonstrations in the prompt (in-context learning) to anchor the desired attribute (Brown et al., 2020). However, increasing the number of demonstrations leads to significant memory consumption and computational overhead, particularly with the larger context windows needed for complex tasks (Wei et al., 2022; Li et al., 2024). These limitations motivate approaches that do not rely on expanding the prompt, but instead intervene directly in the model's internal representations at inference-time.

Latent-space steering methods, often referred to as activation engineering, enable controlled text generation by intervening directly in the internal activations of large language models during inference (Dathathri et al.,

2019; Liu et al., 2021; Khalifa et al., 2020). These methods typically use steering vectors, which are directions in activation space associated with semantic, behavioral, or stylistic attributes, to shift model behavior in predictable ways (Hernandez et al., 2023; Sun et al., 2024). Compared with in-context prompting, activation engineering provides a more direct mechanism for controlling attributes such as style, tone, formality, or factuality, without expanding the prompt context. However, dense activation spaces can entangle multiple latent factors, consistent with the superposition hypothesis (Elhage et al., 2022), making interventions less precise and more prone to unintended side effects. Sparse representations, including those learned by Sparse Autoencoders (SAEs; Makhzani & Frey, 2013), offer a natural way to factorize control directions by encouraging updates to concentrate on a smaller set of features (Cunningham et al., 2023; Bricken et al., 2023). This sparsity can support more isolated and independent latent updates, enabling more precise and reliable control over target attributes during generation.

Prior steering largely adds static vectors to the residual stream or attention-head outputs, which can overwrite context and entangle attributes; we instead ask whether a more local site, optimized rather than fixed, yields cleaner control. In this paper, we hypothesize that attention query activations serve as a high-fidelity control surface for generation, as they encode the model's intent to retrieve information from context. Because queries parameterize how tokens attend to and retrieve information, small and localized query perturbations can modify what the model uses while substantially reducing global distortion of the residual stream. We propose an approach for controlled generation that combines SAEs with gradient-based latent search. Inspired by prototypical networks (Snell et al., 2017), originally designed for few-shot learning, we represent target attributes as prototype distributions in the sparse latent space. Instead of adding a static steering vector, we treat control as an optimization problem, steering encoded features to increase alignment with the desired prototype distribution.

We validate our approach across complementary control settings. For hard, verifiable control, we evaluate on Textualized Gridworld (TGW; Kim et al., 2024), adapted from classic RL planning setups (Zai & Brown, 2020), where generated paths must satisfy explicit safety and length constraints. This setting enables exact, rule-based evaluation: we can directly measure whether the generated path remains valid while satisfying the target objective, such as producing a safe, short, or long path. For soft semantic control, we transfer our method to an educational feedback setting and steer Bloom-level cognitive complexity (Anderson & Krathwohl, 2001), demonstrating control over nuanced pedagogical and stylistic attributes. Finally, we evaluate on TruthfulQA (Lin et al., 2022) as an external benchmark to assess whether gradient-based steering improves truthfulness and informativeness while reducing distributional distortion compared with static vector-addition baselines.

Our primary contributions and findings are the following:

1) **Prototype-Based Sparse Query Optimization:** We formulate inference-time control as gradient-based optimization over sparse query latents, using a prototype-distribution objective defined by a softmax over negative distances to target attribute prototypes. This enables adaptive steering toward semantic or behavioral targets while keeping the base model weights fixed.

2) **Query Level Control Fidelity:** We provide quantitative evidence that attention query activations offer a more local and lower-distortion intervention site than the residual stream. Specifically, query level interventions exhibit bounded cross-layer activation deviation and lower next-token Jensen–Shannon divergence at matched steering strength, supporting their use for both planning and stylistic control. Recent independent work on query level steering (Torop et al., 2025) provides complementary evidence for the same broader observation.

3) **Steering-Oriented Evaluation Data:** We construct a new Bloom's Taxonomy dataset for cognitive style steering (Anderson & Krathwohl, 2001) and derive a filtered Gridworld steering subset from Textualized Gridworld (Kim et al., 2024). The Gridworld subset is designed to contain mutually distinct safe, short, and long paths, enabling rule-based evaluation of hard steering constraints.

4) **Experimental Validation Across Steering Regimes:** We evaluate our method on hard planning constraints, soft cognitive-style control, and TruthfulQA truthfulness steering. Across TGW experiments on three open-weight models, Bloom-level steering, and TruthfulQA, our method matches or exceeds prompting,

residual stream steering, attention-head steering, dense query steering, and static sparse steering baselines, while often inducing substantially lower distributional cost.

2 Proposed Steering Framework

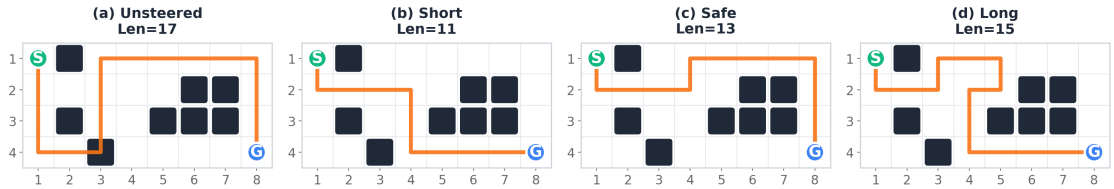


Figure 1: Quantifiable steering in TGW. We steer generation to short, safe (lower wall-adjacency score), and long targets; success is verifiable via path validity and attribute metrics. The unsteered output may be invalid.

We study inference-time steering for controlled generation: given an input prompt and a target attribute, the goal is to bias an LLM's output toward that attribute while preserving task-relevant content and coherence. For example, in Textualized Gridworld (TGW) a prompt specifies a grid map with walls, a start cell, and a goal cell, and the model must output a sequence of cells representing a path from start to goal (Kim et al., 2024). We define a target attribute to require the path be safe, short, or long. In our TGW setting, we quantify safety by the number of wall-adjacent cells along the path, so safer paths have lower wall-adjacency scores; steering succeeds only if the generated cell sequence is a valid path (each step moves to an adjacent cell and does not enter wall cells) while satisfying the specified attribute. TGW illustrates a hard, verifiable regime: preservation corresponds to maintaining structural validity with respect to the task (a well-formed path), while attributes such as safety and length are directly measurable on the resulting path. Figure 1 shows an example of steering a single base model output toward different target attributes. In TGW, the steered output is exactly verifiable: we can check path validity and category-specific optimality by computing the path score (i.e., wall-adjacency count for safety or path length for short/long targets) and comparing it to the score of a reference (gold) solution.

Achieving these verifiable targets requires modifying the model's internal trajectory at inference-time, which motivates interventions on intermediate activations rather than relying solely on prompting. Dense internal activations provide a valid intervention point for inference-time steering, but updates in this space are inherently coupled across many latent factors, making iterative optimization prone to interference. To improve the conditioning of the control space without committing to a specific intervention site, we reparameterize the steerable activation using a SAE. The SAE enforces sparsity through an L_1 constraint on latent features, encouraging updates to activate only a small subset of directions (Konda et al., 2014; Lee et al., 2007). In our setting, this sparsity serves to factorize control directions, enabling more isolated and targeted updates during inference-time optimization.

Let $s = \mathrm{LLM}_{\leq \ell}(x, y)$ denote the activation extracted at intervention layer ℓ (we analyze the choice of site in Section 2.2) for the full sequence consisting of the prompt x and generated response y. The SAE encoder maps this activation to a sparse latent representation $z = \mathrm{Enc}(s)$, and the decoder reconstructs it back to the original activation space as $\hat{s} = \mathrm{Dec}(z)$. Steering is performed by optimizing z while keeping all LLM and SAE parameters frozen. The SAE parameters are shared across token positions, and the latent representation is optimized jointly over the full sequence. After optimization, the reconstructed activation $\hat{s}$ is injected back at the same intervention point, and generation continues normally.

2.1 Attribute-Conditioned Steering Objective

To define steering targets in the sparse latent space learned by the SAE, we borrow the idea of prototype-based representations (Snell et al., 2017). Rather than using prototypes for classification, we use them to define target regions in the latent space that guide inference-time control.

Algorithm 1 Attribute-conditioned Steering using Sparse Autoencoder

Input: Source text x, target attribute k_T, Number of attribute class K, support sets D_k, LLM, Enc, Dec, step size η, threshold ϵ, intervention layer ℓ
Output: Steered text toward target attribute
for $k \in \{1, \ldots, K\}$ **do**
  $c_k \leftarrow \frac{1}{|D_k|} \sum_{((x_i, y_i), a_i) \in D_k} \mathrm{Enc}(\mathrm{LLM}_{\leq \ell}(x_i, y_i))$ ▷ Calculate the center for each prototype
end for
$y' \leftarrow \mathrm{LLM}(x)$ ▷ Get the initial model output
$t \leftarrow 0$; $z^{(t)} \leftarrow \mathrm{Enc}(\mathrm{LLM}_{\leq \ell}(x, y'))$ ▷ Obtain the initial latent representation
while $\left\| \nabla_z \log P_\phi(a = k_T \mid z^{(t)}) \right\|_2^2 > \epsilon$ **do** ▷ Continue until the steering update converges
  for $k \in \{1, \ldots, K\}$ **do**
    $d_k \leftarrow \|z^{(t)} - c_k\|_2^2$ ▷ Find the distance of latent representation to each prototype center
  end for
  $P_\phi(a = k_T \mid z^{(t)}) \leftarrow \frac{\exp(-d_{k_T})}{\sum_{j=1}^{K} \exp(-d_j)}$ ▷ Calc. the likelihood of the target prototype using negative distance
  $z^{(t+1)} \leftarrow z^{(t)} + \eta \frac{\nabla_z \log P_\phi(a = k_T \mid z^{(t)})}{\|\nabla_z \log P_\phi(a = k_T \mid z^{(t)})\|_2}$ ▷ Normalized gradient step toward the target prototype
  $t \leftarrow t + 1$
end while
return $\mathrm{LLM}_{\geq \ell}(\mathrm{Dec}(z^{(t)}))$

For each attribute class k, we define a support set D_k consisting of training examples labeled with attribute k: $D_k = \{((x_i, y_i), a_i) : a_i = k\}$, where x_i is the prompt, y_i is the response, and a_i is the attribute label. For each class, we define the prototype c_k as the mean sparse latent representation of the support examples:

$$c_k = \frac{1}{|D_k|} \sum_{((x_i, y_i), a_i) \in D_k} \mathrm{Enc}(\mathrm{LLM}_{\leq \ell}(x_i, y_i)). \tag{1}$$

Instead of directly projecting the latent representation onto the target prototype c_k, which empirically leads to reduced diversity in generated outputs (see Section A.4), we perform gradient-based optimization so that alignment is achieved gradually while respecting the latent structure.

We define a metric-based distribution over attribute classes using a softmax over negative distances to the corresponding prototypes:

$$P_\phi(a = k_T \mid z) = \frac{\exp(-d_{k_T})}{\sum_{j=1}^{K} \exp(-d_j)}, \tag{2}$$

where k_T is the target attribute class, and d_k is the distance of latent representation z from the center of prototype k using a fixed distance metric (squared Euclidean distance in our experiments). This probability serves as a continuous control score measuring alignment between the current generation and the target attribute, rather than as a classifier.

Formally, given a generated response y' to prompt x, steering begins by extracting the activation at intervention layer ℓ and encoding it into an initial sparse latent representation, $z^{(0)} = \mathrm{Enc}(\mathrm{LLM}_{\leq \ell}(x, y'))$. We then optimize z directly to increase its alignment with the target attribute k_T. Concretely, we maximize the target log-probability $\log P_\phi(a = k_T \mid z)$, where P_ϕ is defined via distances to class prototypes in the sparse latent space. Optimization proceeds via gradient ascent:

$$z^{(t+1)} = z^{(t)} + \eta \nabla_z \log P_\phi(a = k_T \mid z^{(t)}), \tag{3}$$

where η controls the steering strength on the unit normalized gradient (Algorithm 1).

After convergence, the optimized latent is decoded back to the activation space using Dec, and generation continues normally. Algorithm 1 illustrates the full steering procedure.

2.2 Query-Level Intervention

Prior work intervenes at different architectural sites and uses either static steering vectors or state-dependent signals (Hernandez et al., 2023; Sun et al., 2024; Li et al., 2023; Wang et al., 2025b; Li et al., 2025). However, the reliability of these methods is limited by geometric and separability constraints (Im & Li, 2025). In particular, when generation must satisfy hard constraints, such as logical or structural validity, interventions that broadly perturb dense representations can overwrite context and induce violations. Under soft semantic control, they can drift or inadvertently blend attributes, yielding unintended attribute mixing. Consistent with this, prior residual stream steering methods report qualitative degradation and side effects at higher intervention strengths (Turner et al., 2023; Stolfo et al., 2024; Elhage et al., 2021).

To preserve high-fidelity locality, we prioritize intervening on self-attention query activations rather than the residual stream. Queries provide a more local control handle because they modulate what information is retrieved through attention without directly overwriting the accumulated residual state; specifically, they affect the attention weights, so each head produces a convex combination of value embeddings whose output lies in the convex hull of the values rather than an unconstrained additive perturbation.

To justify this design choice, we run two locality diagnostics (these are not task-performance evaluations, which we defer to Sec. 3) comparing interventions (query steering and residual steering) applied at the same layer. First, we measure how an intervention propagates through each layer ℓ above the intervention site by tracking the relative change in activations compared to an unmodified forward pass:

$$\delta_\ell = \frac{1}{T} \sum_{t=1}^{T} \frac{\|\mathbf{h}_{\text{steered}}^{(\ell,t)} - \mathbf{h}_{\text{base}}^{(\ell,t)}\|_2}{\|\mathbf{h}_{\text{base}}^{(\ell,t)}\|_2}, \tag{4}$$

where T is the sequence length and $\mathbf{h}^{(\ell,t)}$ is the activation vector at layer ℓ and token position t. As shown in Figure 2a, query interventions yield a bounded, slowly varying deviation across layers, while residual stream interventions induce a large, persistent deviation, consistent with a more disruptive overwrite of local context.

Second, beyond activation propagation, we characterize the accuracy-distortion trade-off induced by the intervention site as steering strength increases. The core question is whether stronger steering yields commensurate gains in task success, or instead primarily manifests as unintended shifts in the model's output distribution. We quantify this distributional cost using the Jensen–Shannon divergence (JSD) between the baseline and steered next-token distributions. Let P_b and P_s denote the probability distributions over the vocabulary for the baseline and steered models, respectively:

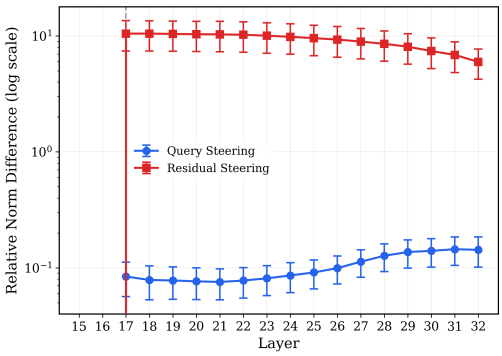
$$\text{JSD}(P_b \parallel P_s) = \frac{1}{2} D_{\text{KL}}(P_b \parallel M) + \frac{1}{2} D_{\text{KL}}(P_s \parallel M), \tag{5}$$

where $M = \frac{1}{2}(P_b + P_s)$ is the mixture distribution. We sweep steering strengths (η) and plot task success against JSD cost. Figure 2b provides a quantitative confirmation of the trade-off: residual stream steering exhibits rapidly increasing distortion with diminishing returns in task success, whereas query-level steering achieves comparable control with substantially lower distributional cost.

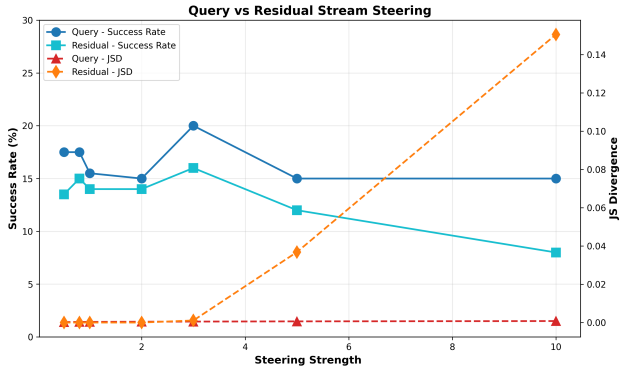
While query perturbations can still propagate through downstream layers, our locality diagnostics, including bounded activation deviation δ_ℓ and lower next-token JSD at matched steering strength, indicate that query-level interventions induce systematically smaller distributional shifts than residual stream additions in our settings. Recent independent work reaches a related conclusion: Torop et al. (2025) report similar benefits from query and value level interventions. Our diagnostics complement theirs by directly quantifying the distortion advantage of query-level steering. Together, these results motivate query activations as a suitable intervention site for reliable steering under both hard and soft constraint regimes.

2.3 Training Sparse Autoencoders

Our SAE architecture is a single-layer ReLU autoencoder (similar to Bricken et al. (2023)). For activation s at an intervention site, we define a shallow autoencoder with encoder $\text{Enc}(s) = \text{ReLU}(W_e s + b_e)$ and decoder $\text{Dec}(z) = \tilde{W}_d\, z$, where (W_e, b_e) and W_d are the encoder and decoder parameters, respectively, and $z = \text{Enc}(s)$



(a) Relative activation deviation δ_ℓ (log scale) across layers after applying a fixed-strength intervention at layer 17. Query-level steering remains bounded and slowly varying, whereas residual stream steering induces one to two orders of magnitude larger, persistent deviation (error bars show variability across samples).



(b) Query vs. residual stream steering trade-offs. Task success rate (left axis) and Jensen-Shannon divergence (JSD; right axis) as a function of steering strength η for query-based (attention) and residual stream steering.

Figure 2: Quantitative diagnostics for intervention locality. (a) Propagation of activation deviation across layers. (b) Task success vs. distributional distortion trade-off.

is the encoded latent representation. To prevent the L_1 penalty from being trivially minimized by rescaling, each decoder column (i.e., the dictionary direction for latent i) is normalized to unit norm,

$$\tilde{W}_d^{(i)} = \frac{W_d^{(i)}}{\max\left(\|W_d^{(i)}\|_2,\, \gamma\right)},$$

where $W_d^{(i)}$ denotes the i-th column of W_d and γ is a small constant for numerical stability. We then define the training loss as a combination of a reconstruction loss, a sparsity penalty, and a bias decay term:

$$\mathcal{L}(s) = \underbrace{\|\hat{s} - s\|_2^2}_{\text{Reconstruction loss}} + \underbrace{\lambda\|z\|_1}_{\text{Sparsity loss}} + \underbrace{\beta\|b_e\|_2^2}_{\text{Bias decay}}, \tag{6}$$

where $\hat{s} = \mathrm{Dec}(z)$ is the reconstructed activation, λ controls the L_1 sparsity strength, and β controls the bias decay.

We train one SAE per query attention head. Because each head in the multi-head attention mechanism operates independently and captures different aspects of the input, we train these SAEs separately rather than sharing parameters across heads. Each SAE is trained on the query activations of its corresponding head, extracted from the self-attention layer at the intervention site.

3 Experimental Setup

We evaluate our proposed steering framework on two distinct text generation domains: constrained path planning (Textualized Gridworld), and the cognitive style (Educational Cognitive Style) dataset. We additionally use TruthfulQA (Lin et al. (2022)) as an external benchmark to isolate the effect of gradient-based optimization against static vector addition baselines.

3.1 Datasets

Textualized Gridworld Planning. We utilize the Textualized Gridworld framework (Kim et al., 2024), adapted from Zai & Brown (2020). To evaluate performance, we constructed the datasets with variable

Table 1: Planning performance on Textualized Gridworld under hard constraints. Each cell reports results for Short/Safe/Long targets. Success rates are higher better, while logic violation rates are lower better.

	Success rates (%) ↑			Logic violation rates (%) ↓		
Method	Qwen	Phi	Llama	Qwen	Phi	Llama
ICL	29.3/3.7/3.2	12.1/3.3/2.9	26.9/2.7/3.2	57.1/55.5/76.0	81.3/76.0/89.1	52.1/51.2/62.2
CAA	53.9/4.2/5.6	38.6/8.7/4.9	39.8/4.2/4.1	38.8/39.6/19.9	**36.4/38.9/38.2**	**38.8/39.6/33.9**
ITI	32.9/9.2/6.5	38.8/4.7/3.3	39.2/6.4/4.0	57.8/**18.3**/19.2	39.9/48.5/40.3	67.5/65.9/43.7
SAE-SSV	54.8/14.2/8.2	35.8/9.6/4.9	39.5/7.9/4.8	24.7/23.4/24.4	55.0/55.8/55.8	46.5/47.5/48.8
DISCO-Q	51.0/14.40/7.8	38.5/10.16/4.98	38.8/9.0/5.2	24.2/25.4/24.4	43.0/43.80/41.84	45.8/ 45.54/47.77
DENSE-OPT	54.0/14.8/7.8	38.0/9.4/4.9	39.9/9.3/5.0	26.7/23.5/24.7	54.7/55.0/55.6	48.0/45.8/48.8
SAE-OPT-ANCH	**56.0**/14.5/7.4	37.4/9.1/4.0	39.5/7.1/4.9	23.4/27.0/**18.9**	56.1/57.0/55.5	47.7/47.4/47.0
SAE-OPT	55.5/**15.2**/**8.4**	**38.9**/**10.4**/**5.2**	**40.4**/**9.6**/**5.9**	**21.0**/20.1/21.6	54.2/52.3/52.2	43.0/43.0/42.2

dimensions $M \times N$ ($M \leq 10, N \leq 10$.) for standard training and evaluation.[1] We processed the raw environments by normalizing global coordinates into a 1-based local reference frame, removing pits to focus on wall avoidance, and converting the grid state into an ASCII-based prompt. For each grid, we generated three ground-truth paths using graph search algorithms: Short, Safe, and Long. To ensure distinct planning constraints, we filtered the dataset to retain only samples where these three paths were mutually distinct with unique risk profiles. The final dataset comprises 15,250 unique examples. We split the data into train (70%), validation (10%), and test (20%). We evaluate steering performance under strict, rule-based criteria described below. We evaluate three grid navigation objectives: shortest, safest (minimum wall adjacency), and longest path without revisits. Optimal shortest and safest paths are computed via BFS and Dijkstra's algorithm, and predictions are successful if they match the optimal objective value. For the NP-hard longest path task, we approximate the optimum using beam search ($B = 512$) and count success if the predicted length meets or exceeds this bound. All outputs must satisfy strict path validity constraints. We report success rates over all instances.

Educational Cognitive Style. To evaluate steering on soft stylistic constraints, we construct a novel dataset to evaluate nuanced style transformation. The dataset targets ten fundamental C++ programming concepts (recursive functions, arrays, call-by-reference, switch statements, do-while loops, for-loops, while-loops, pointers, strings, and vectors), each represented by multiple coding questions and systematic feedback variations. At the initial stage, we prompt the LLM (Claude 3.5 Sonnet) to generate 20 per difficulty levels (easy, medium, and hard difficulties) each with its correct answer. Then, we have the LLM create four variations of each answer that contain deliberate errors. Finally, we generate feedback for both the original correct answer and all variants, using Bloom's Taxonomy levels: Remember, Understand, Apply, Analyze, Evaluate, and Create (Anderson & Krathwohl, 2001).

Our dataset comprises 3,000 data points, calculated as follows: 10 topics $\times$ 20 questions $\times$ 3 difficulty levels $\times$ 5 code variants (one correct, four with errors). We split the data by topic into training (50%), validation (20%), and test (30%) sets to ensure conceptual separation between splits.

TruthfulQA. To complement our two primary domains with an established external benchmark, we evaluate on TruthfulQA (Lin et al., 2022) using its open-ended generation setup.

3.2 Models and Configurations

We evaluate constrained path planning on three publicly released open-weight LLMs: Qwen3-4B-Instruct (Qwen; Team, 2025), Phi-3-Mini-4K-Instruct (Phi; Abdin et al., 2024), and Llama3.1-8B-Instruct (Llama; AI@Meta, 2024). For the Qwen and Phi models, we use the standard Alpaca (Taori et al., 2023) prompting format, with a maximum context size of 512 tokens. For Llama, we use its corresponding instruction format while keeping the same context length. For each task, we perform one epoch of supervised fine-tuning

[1]We also evaluate on the extrapolation set provided by Kim et al. (2024), containing larger grids $M \times N$ where either M or $N \geq 10$, to test the model's ability to generalize steering to longer, more complex paths. Please see Table 5 for more details

on the training split to ensure task competence (i.e., correct path syntax for TGW and domain-appropriate feedback for cognitive style adaptation). This fine-tuning stage is performed once per model and task. After fine-tuning, all base model parameters are frozen for the remainder of the experimental pipeline. For cognitive style adaptation we primarily use the Phi model (Abdin et al., 2024) as the base model.

SAEs are trained offline on internal activations of the frozen, fine-tuned base models. We extract query activations from a single layer (middle layer i.e., for Qwen it is layer 17 and for Phi and Llama it is 15) and treat each attention head independently, training one SAE per query attention head. For Phi, each query head has dimensionality 96 and is mapped to a latent space of dimension 256. For Qwen and Llama, each query head has dimensionality 128 and is mapped to a latent space of dimension 512.

SAEs are trained using the objective defined in equation 6, which combines reconstruction loss, L_1 sparsity on latent activations, and bias decay. We evaluate L_1 regularization coefficients $\lambda \in \{3\times10^{-3},\ 3\times10^{-2},\ 3\times10^{-1}\}$. Training is performed for 40 epochs using the Adam optimizer. We employ a linear warm-up followed by a cosine decay scheduler to dynamically adjust the learning rate, starting from 3×10^{-5}. Optimization is performed using the Adam optimizer with $\beta_1 = 0.90$ and $\beta_2 = 0.99$. Unless otherwise stated, all reported results use λ=0.003.

During inference-time steering, we use $\eta = 0.5$ for LlaMA and Qwen and $\eta = 0.8$ for Phi for the TGW task. For the cognitive style dataset we use $\eta = 0.8$. For convergence we consider $\epsilon = 10^{-3}$.

3.3 Baselines

We refer to our proposed method as **SAE-OPT**. We also evaluate two variants to isolate the effects of anchoring and sparsity. The first variant, **SAE-OPT-ANCH**, augments the prototype-softmax steering objective with an L_2 anchor that penalizes deviation from the original latent encoding. Let

$$z_0 = \mathrm{Enc}(\mathrm{LLM}_{\leq \ell}(x, y'))$$

denote the unsteered sparse latent representation at intervention layer ℓ, and let k_T denote the target attribute. The anchored objective is

$$z^* = \arg\max_z \ \log P_\phi(a = k_T \mid z) \ - \ \gamma \|z - z_0\|_2^2,$$

where $\gamma \geq 0$ controls the trade-off between steering strength and preservation of the original encoding. Because P_ϕ is defined through a softmax over distances to all class prototypes, this objective still accounts for the relative geometry of all attribute classes; the anchor only constrains the magnitude of the latent update.

The second variant, **DENSE-OPT**, applies the same gradient-based steering objective directly to the dense query activation at intervention layer ℓ, rather than to the sparse representation produced by the SAE.

We also used the following baselines:
ICL (Brown et al., 2020): For few-shot in-context learning, we include N examples sampled from the training data in our prompts and apply a few-shot learning approach (Brown et al., 2020).

CAA (Rimsky et al., 2024): Computes steering vectors as the average difference in residual stream activations between positive and negative example pairs for a behavior, then adds these vectors (with a tunable coefficient) to activations after the prompt during inference.

Inference Time Intervention (ITI;Li et al., 2023): Identifies directions in attention heads that correspond to the target behavior (i.e., truthfulness), then adds scaled vectors along these directions to the top-K most relevant heads during inference, enabling targeted steering with minimal disruption to the model.

SAE-SSV (He et al., 2025a): We adapt the steering logic from SAE-SSV by applying it to the attention query space instead of the residual stream. We extract sparse latent query features from our trained SAEs and define a static steering direction using the mean difference between target class centroids. This baseline serves as a linear control to evaluate the necessity of our proposed gradient-based optimization for handling hard constraints.

DISCO-Q (Torop et al., 2025): A static query-level steering method that adds a fixed steering vector to dense query activations. We include it as a query-site static baseline, isolating the effect of our gradient-based optimization against a static intervention at the same (query) site.

3.4 Results and discussion

We organize the results around three sections. Section 3.4.1 evaluates whether sparse query-level steering satisfies hard, verifiable constraints using TGW, where both path validity and attribute optimality are rule-checkable. Sections 3.4.2 and 3.4.3 examine transfer to soft semantic control by evaluating Bloom's Taxonomy steering, first through a classifier-based analysis and then using the standardized AxBench three-axis (Wu et al., 2025b). Section 3.4.4 investigates an orthogonal design question to the site selection analysis in Section 2.2. Specifically, we test whether gradient based optimization outperforms static vector addition on a shared benchmark (TruthfulQA;Lin et al., 2022), extending the accuracy distortion analysis of Figure 2b from the intervention site axis to the optimization axis. Finally, Section 3.4.5 examines the role of sparsity by comparing L1 and L2 autoencoder regularization, showing that L1 regularization reduces non-target prototype drift and yields cleaner steering.

3.4.1 Hard constraint steering in Textualized Gridworld

Hard-constraint steering provides a stringent test of a steering method because success requires both target adherence and structural validity. A generated path is counted as successful only if it satisfies the intended attribute and remains valid; for example, a shorter path that enters a wall, or a safe path that is disconnected, is still a failure. Textualized Gridworld makes both conditions exactly checkable, allowing us to measure success rate and logic violations separately and interpret them jointly (Table 1). Under this criterion, SAE-OPT achieves the highest or tied-highest success rate in most settings, while keeping violation rates stable across targets within each model (i.e., lowest among all methods on Qwen). This suggests its success gains do not come from rising path validity violations.

Short paths align closely with the model's post-fine-tuning bias, yielding relatively high success across all methods. Safe and Long targets are considerably more difficult. On Safe, our approach improves success from 3.7% (ICL) and 4.2% (CAA) to 15.2% on Qwen, with similar relative gains on Phi and LlaMA. On the NP hard Long objective, absolute performance remains modest, but our method achieves the highest success across all methods.

An observed pattern is the relative stability of violation rates under our method within each model across targets (approximately 21% on Qwen, 52% on Phi, and 43% on LlaMA). This behavior is consistent with the localized nature of query-level interventions. Query-level modifications influence attention weights rather than directly altering the residual stream. As a result, steering tends to reroute paths within the valid-path manifold instead of violating structural constraints.

Methods can be grouped into distinct categories. Few-shot prompting (ICL) performs the weakest overall. Dense residual stream methods are competitive on short but degrade on safe and long. Sparse query methods perform better on the harder targets, suggesting that the sparse query space is an effective control surface. Gradient based prototype optimization generally outperforms static steering vectors on safe and long.

The relative ranking of methods is consistent across architectures and extends to larger grids in the extrapolation setting (Table 5). Appendix A provides quantitative visualizations through heatmaps (Section A.1) and cell-level attention analysis (Section A.2), illustrating target-aligned shifts that correspond to both improved success and preserved validity.

3.4.2 Cognitive style steering (Bloom's Taxonomy)

We evaluate steering over six Bloom's Taxonomy levels: Remember (Rem), Understand (Und), Apply (App), Analyze (Ana), Evaluate (Eva), and Create (Cre), using Phi as the base model. Each steered output is assessed by GPT-4o and Claude (Claude Sonnet 4) on two dimensions: (a) Bloom-level classification identifying the dominant cognitive level, and (b) binary relevance judgment of whether the feedback engages with the input code. The data was generated with Claude 3.5 Sonnet, so judging with a later Claude generation (Sonnet 4)

Table 2: Diagonal hit rates (%) from output-class confusion matrices under GPT-4o / Claude as judge (Claude Sonnet 4). Each cell shows the percentage of steered generations correctly classified as the intended target. Full confusion matrices are in Appendix E.

Method	Judge	Rem.	Und.	App.	Ana.	Eva.	Cre.	Avg.
ICL	GPT-4o	7.4	24.7	7.5	27.6	42.3	15.6	20.8
	Claude	10.9	13	0.9	32.7	49.0	15.3	20.3
CAA	GPT-4o	9.7	21.5	1.6	**39.4**	45.6	8.3	21.0
	Claude	13.1	15.1	1.1	**42.6**	46.2	19.1	22.9
ITI	GPT-4o	7.6	24.7	4.8	15.7	17.5	9.5	13.3
	Claude	5.9	15.0	5.6	21.2	12.8	0.1	10.1
SAE-SSV	GPT-4o	1.4	21.2	1.9	19.3	46.7	12.0	17.1
	Claude	2.9	16.9	3.5	20.0	42.8	17.8	17.3
Disco-Q	GPT-4o	2.5	18.4	1.4	19.1	44.0	13.8	16.5
	Claude	3.9	14.0	0.4	19.8	43.7	18.3	16.7
DENSE-OPT	GPT-4o	8.5	25.1	5.2	20.5	52.5	14.6	21.1
	Claude	12.2	16.3	7.2	27.1	48.5	16.5	21.3
SAE-OPT-ANCH	GPT-4o	1.6	4.5	4.1	16.3	**80.2**	3.3	18.3
	Claude	2.8	3.6	5.2	12.4	**83.6**	5.5	18.8
SAE-OPT	GPT-4o	**10.8**	**28.9**	**8.4**	31.0	54.7	**18.4**	**25.4**
	Claude	**16.5**	**19.3**	**10.8**	35.4	55.8	**24.7**	**27.1**

and a different model (GPT-4o), neither of which produced the data, mitigates self-preference and circularity in the reported scores.

Table 2 reports diagonal hit rates from confusion matrices under both judges. The base model exhibits a strong default bias toward Evaluate, making bias conflicting levels particularly Remember and Create the most discriminative targets. SAE-OPT achieves the highest average hit rate under both judges, with the largest gains on precisely these levels. The SAE-OPT-ANCH variant collapses almost entirely to Evaluate, confirming that the L2 anchor prevents the optimizer from reaching distant prototype regions and that the unconstrained objective is necessary for balanced steering.

Table 10 reports per-target relevance rates. Our method maintains the highest relevance across nearly all targets, while residual stream methods show lower relevance, suggesting that query level interventions redirect cognitive style without displacing the model's grounding in the input code. We further observe that enforcing sparsity reduces blending of multiple cognitive levels within a single response, improving single-level alignment over dense query steering. Full confusion matrices, relevance rates appear in section E (see Section I, H for qualitative examples)

3.4.3 Quantitative evaluation with AxBench

To complement classifier-based analysis with a standardized evaluation protocol, we report results on AxBench (Wu et al., 2025b), which scores steered outputs along three axes, concept incorporation (s_c), instruction following (s_i), and fluency (s_f), aggregated via harmonic mean (HM $\in [0, 2]$). Methods that force a concept at the expense of coherence are heavily penalized, making it a stricter test than single-label classification (see Appendix C).

Our method achieves the highest HM on 5 of 6 Bloom levels (i.e., Understand: 0.950 vs. 0.907 for DENSE-OPT and 0.752 for CAA; Create: 0.764 vs. 0.697; Evaluate: 1.399 vs. 1.389; full per-axis breakdown in Table 8, Appendix D). All methods struggle on Remember (HM < 0.51), which requires suppressing the model's default tendency to elaborate. On higher-complexity levels (Understand–Create), gains stem from improved concept adherence (s_c) without sacrificing instruction following or fluency, confirming that gradient ascent on middle-layer query latents provides the localized control needed for nuanced stylistic steering.

3.4.4 Gradient-based versus static vector addition steering on TruthfulQA

Section 2.2 (Figure 2b) shows that query level interventions incur lower distributional cost than residual stream interventions at matched success, establishing an accuracy distortion trade off along the intervention site axis. We also investigate whether a similar trade-off exists along the optimization axis (i.e., whether gradient-based updates incur lower distortion than static vector addition). To evaluate this choice on an established benchmark (Table 3), we compare against static steering baselines on TruthfulQA (Lin et al., 2022) using the open-ended generation setup of Torop et al. (2025) and the %Truthful × %Informative (%T×I) metric (Lin et al., 2022). The static baselines span different intervention: CAA uses residual stream, ITI uses attention heads, and DISCO-Q uses dense query, so any advantage of gradient-based optimization is attributable to the update rule rather than the site.

Table 3: Gradient-based versus static vector addition steering on TruthfulQA. We report steering accuracy (%T×I) and distributional cost (JSD) across steering methods on two base models.

Method	**Type**	**%T×I ↑**	**Mean JSD ↓**	**Median JSD ↓**	**Max JSD ↓**
Qwen	–	0.6749	–	–	–
ITI	Static	0.6778	0.009839	0.006841	0.082882
CAA	Static	0.6914	0.107478	0.102157	0.278795
Disco-Q	Static	0.7078	0.030830	0.027103	0.131915
DENSE-OPT	Gradient	**0.7407**	**0.002392**	**0.000731**	**0.028518**
Llama	–	0.7366	–	–	–
ITI	Static	0.7423	0.033181	0.019546	0.198342
CAA	Static	0.7572	0.063668	0.049712	0.239238
Disco-Q	Static	0.7866	0.082272	0.074456	0.227702
DENSE-OPT	Gradient	**0.7928**	**0.001314**	**0.000873**	**0.015991**

Across both models, DENSE-OPT achieves the highest %T×I while reducing mean JSD by over an order of magnitude compared to the static baseline. Notably, maximum JSD remains below 0.03 for DENSE-OPT, whereas static methods exhibit peaks exceeding $0.13 - 0.28$, confirming that gradient based optimization avoids the worst case distributional shifts inherent to static vector addition. Table 3 corroborates the accuracy distortion trade off observed in Figure 2b, now validated on an external benchmark with independent baselines.

3.4.5 Sparsity reduces steering entanglement

We test whether sparse regularization provides a measurable advantage during gradient-based steering. To isolate the effect of sparsity, we compare the SAE (L_1) against an otherwise identical autoencoder trained with an L_2 regularization penalty. This comparison asks whether L_2 regularization is sufficient for clean steering, or whether L_1 regularization improves the locality of the intervention. We measure locality using non-target drift, the absolute change, from the initial to final optimization step, in normalized distance to prototype classes that are not targeted by the optimizer.

For a subset of the test dataset, we steer the representation toward each of the three path classes at two regularization strengths, 0.003 and 0.03, where the L_1 model exhibits meaningful sparsity. During optimization, we record the normalized distance to all class prototypes at every step. L_1 **reduces collateral drift across models** (Figure 3). Across the regularization settings, L_1 produces lower non-target drift. This effect appears across all three models and all target classes, suggesting that the sparsity advantage is not specific to a single architecture or target behavior.

Thus, the advantage of L_1 is not that it can steer while L_2 cannot; rather, L_1 provides a cleaner steering interface by achieving comparable target convergence with less collateral movement in non-target prototype geometry.

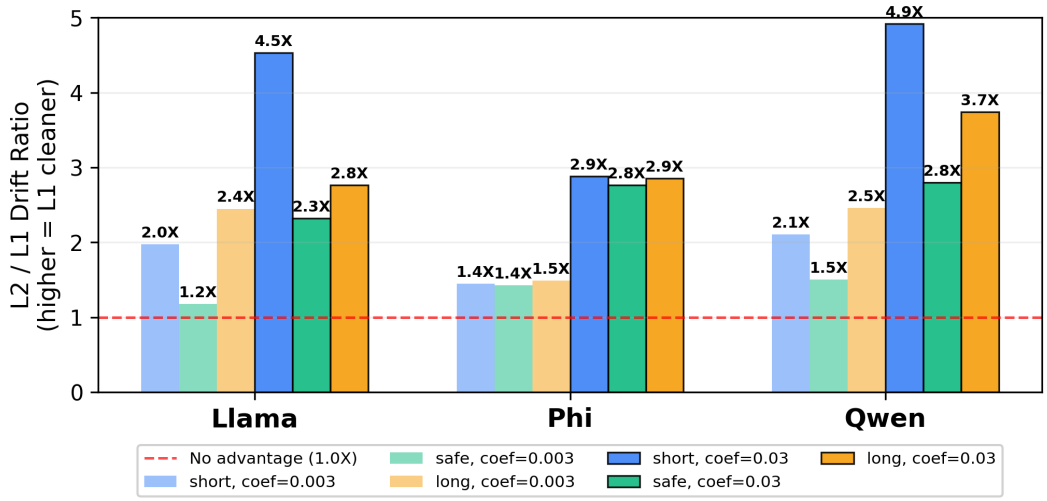


Figure 3: L2/L1 non-target drift ratio across three models (Llama Phi,Qwen), three target classes, and two regularization coefficients. Bars above the dashed red line indicate that L2 induces more non-target drift than L1, meaning L1 steers more cleanly. Faded bars correspond to coef = 0.003; bold-outlined bars correspond to coef = 0.03. Across configurations, L_1 consistently reduces collateral drift, with the advantage becoming strongest at the larger coefficient, where L_2 induces up to 4.9× more non-target drift.

4 Related works

4.1 Sparse Autoencoders

Sparse coding, introduced by Mallat et al. (Mallat & Zhang, 1993) and advanced by Olshausen et al. (Olshausen & Field, 1996) for unsupervised learning, was initially applied to image processing (Liu et al., 2015). Parallelly, Hinton et al. (Hinton & Salakhutdinov, 2006) introduced autoencoders for dimensionality reduction, leading to the convergence in sparse autoencoders (SAEs) (Konda et al., 2014; Lee et al., 2007) that enforce sparsity via L_1 regularization to extract meaningful features. SAEs have gained traction for decomposing LLM activations into interpretable features (Cunningham et al., 2023), representing semantic and syntactic concepts as linear combinations in models like BERT (Yun et al., 2021), GPT (Sharkey et al., 2022), and Pythia (Cunningham et al., 2023) (Bloom, 2024; Marks & Mueller, 2023). This aligns with the Linear Representation Hypothesis (Mikolov et al., 2013; Nanda et al., 2023; Park et al., 2023), favoring monosemantic neurons (Bau et al., 2020). However, networks encode more features than neurons, resulting in polysemanticity (Scherlis et al., 2022; Mu & Andreas, 2020) explained by the superposition hypothesis (Elhage et al., 2022). Recent advancements scale SAEs for better monosemanticity and disentanglement, aiding interventions in sparse spaces (Bayat et al., 2025; He et al., 2025b). Methods such as activation pruning (Rajamanoharan et al., 2024b) or thresholds (Rajamanoharan et al., 2024a) balance sparsity and reconstruction quality.

Building on these foundations, Marks et al. (2025) demonstrate that SAE features can be organized into sparse feature circuits, causally implicated subnetworks of interpretable features enable both mechanistic understanding and targeted behavioral editing. This circuit level perspective complements further extensions of SAEs to steering include vector refinement via semantic denoising (Wang et al., 2025a),correlated feature steering at generation time (Cho et al., 2025) and step level context-aware SAEs for reasoning (Yang et al., 2026). Soo et al. (2025)propose Feature Guided Activation Additions (FGAA), which combines contrastive activation addition with SAE targeted steering by optimizing over SAE features to produce interpretable yet precise steering vectors.

4.2 Steered Generation

Content generation is controllable through latent space (Larsen et al., 2016; White, 2016) and embedding adjustments (Han et al., 2024). However, prevalent methods such as fine-tuning and RLHF (Ranzato et al., 2015; Dathathri et al., 2019; Ilharco et al., 2022; Meng et al., 2022) modify model weights, which risks degrading performance (Qi et al., 2023). Alternatives avoiding weight changes intervene at decoding (Gu et al., 2017), prompt (Zhou et al., 2022), or token embedding levels (Khashabi et al., 2021).

Activation interventions offer efficient control without weight modifications, using steering vectors (Hernandez et al., 2023; Subramani et al., 2022) and latent arithmetic (Correia et al., 2019). Inference-Time Intervention (ITI) (Li et al., 2023) promotes truthfulness by adding probe-derived vectors to attention heads, whereas ActAdd (Turner et al., 2023) steers behavior using activation differences between contrasting prompts, even application in personalized text generation (Zhang et al., 2025a). Similarly, Liu et al. (2023b) apply analogous vector interventions to enhance in context learning. Beyond static single-vector addition, richer parameterizations of the intervention itself have recently emerged. LinEAS (Rodriguez et al., 2025) learns end-to-end activation steering maps via a distributional loss that jointly accounts for layer wise shifts, improving robustness. Further methods show that combining nonlinear predictors across multiple layers yields universal steering and monitoring that generalizes across concepts (Beaglehole et al., 2026). On the optimization axis, one-shot gradient-based optimization of steering vectors has been shown to mediate safety-relevant behaviors with broad transfer (Dunefsky & Cohan, 2025).

Attention interventions Residual stream interventions can broadly perturb activations and overwrite context. Consequently, a growing body of work instead intervenes at the attention mechanism itself. The PASTA method (Zhang et al., 2024) profiles attention heads that benefit task performance and statically reweights attention within those specific heads , whereas SpotLight (Venkateswaran & Contractor, 2025) instead dynamically amplifies attention scores on instruction tokens across all layers and heads to ensure consistent attention allocation. In a parallel work, JoLA learns sparse, gated head edits with mixed edit forms (Lai et al., 2025), while Nguyen et al. (2025) introduces attribute-specific steering vectors with gating mechanisms for multi-attribute control. Further refining head selection, Zhan et al. (2025) proposes a causal-attribution framework that disentangles behavior relevant head activations and computes head-wise confidence scores for principled module selection. At the reasoning level, CREST identifies cognitive attention heads linked to non-linear reasoning patterns and applies test-time interventions to steer reasoning trajectories (Zhang et al., 2025b). Concurrent with our work, similar benefits from query and attention-level interventions have been reported (Torop et al., 2025). Davarmanesh et al. (2026) introduce attention guided feature learning that automates token selection, accounts for feature heterogeneity.

SAE based steering Despite SAE training costs (Gao et al., 2024), their interpretability supports feature-based steering (Durmus et al., 2024), editing middle-layer residual streams to resolve knowledge conflicts (Zhao et al., 2024) via sparsity (Chalnev et al., 2024; O'Brien et al., 2024). Recent extensions include supervised steering in sparse subspaces with classifiers for sentiment/truthfulness (He et al., 2025a). SAE-based denoising of concept vectors for more reliable control (Zhao et al., 2026), and prototype-aligned sparse steering for vision, which conceptually parallels our prototype-based formulation but operates in a different modality (Chatzoudis et al., 2025). Recent studies apply SAEs to reasoning benchmarks, with specific instances like amplifying targeted reasoning features to improve benchmark performance (Galichin et al., 2025) and utilizing token wise decaying steering strategies to reliably elicit step by step mathematical reasoning (Xie, 2025). Other interpretability frameworks focus on unsupervised multi-concept shifts (Joshi et al., 2025) or concept bottlenecks (Kulkarni et al., 2025). In the safety and control domain, approaches prioritize mutual information based explanations for anti-jailbreaks (Wu et al., 2025a). Separately, in the multilingual domain, researchers use directional feature ablation to uncover language specific capabilities (Deng et al., 2025) and single feature steering to causally control the target generation language (Chou et al., 2025). Furthermore, efficiency-centric methods utilize feature-level constrained DPO (Yin et al., 2024) or hypothesis generation from embeddings (Movva et al., 2025), while RL-specific works reveal TD errors in residual streams (Demircan et al., 2024) or analyze CoT-induced attention sparsity (Wen et al., 2024).Recent attempts also target cross-model concept alignment via universal SAEs (Thasarathan et al., 2025) for visual tasks. Complementary prototype based approaches have also emerged (Kayan & Zhang, 2025). Concurrent

work has further explored SAE-based reasoning control (Fang et al., 2026) and efficient feature selection for steering (Arad et al., 2025), reinforcing the value of sparse level interventions.

Building on these lines of work, we investigate sparse, query level interventions for controllable generation. We train independent SAEs on attention query activations at middle layers, where relational and structural information is most accessible (Vig & Belinkov, 2019). During inference time, we perform gradient ascent to align sparse query features with class prototype distributions. This combines query level locality, sparse disentanglement, and gradient-based adaptivity in a single framework.

5 Conclusion and Future Work

We propose a sparse feature based steering method that performs inference-time control by optimizing sparse query representations in attention layers. The approach achieves reliable alignment with target attributes while preserving structural validity, outperforming residual-stream and static steering baselines under hard planning constraints. On Textualized Gridworld, it improves success rates and it generalizes to soft semantic control by steering cognitive complexity in educational feedback. Across tasks, middle-layer query features emerge as the most effective control surface, supporting sparse query-level optimization as a principled mechanism for controllable generation.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

AI@Meta. Llama 3 model card. 2024.

Lorin W Anderson and David R Krathwohl. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives: complete edition*. Addison Wesley Longman, Inc., 2001.

Dana Arad, Aaron Mueller, and Yonatan Belinkov. Saes are good for steering–if you select the right features. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10252–10270, 2025.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. Steering large language model activations in sparse spaces. *arXiv preprint arXiv:2503.00177*, 2025.

Daniel Beaglehole, Adityanarayanan Radhakrishnan, Enric Boix-Adsera, and Mikhail Belkin. Toward universal steering and monitoring of ai models. *Science*, 391(6787):787–792, 2026.

Joseph Bloom. Open source sparse autoencoders for all residual stream layers of gpt2 small. In *AI Alignment Forum*, pp. 24, 2024.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*, 2024.

Gerasimos Chatzoudis, Zhuowei Li, Gemma E. Moran, Hao Wang, and Dimitris N. Metaxas. Visual sparse steering (vs2): Unsupervised adaptation for image classification using sparsity-guided steering vectors. 2025.

Seonglae Cho, Zekun Wu, and Adriano Koshiyama. Corrsteer: Generation-time llm steering via correlated sparse autoencoder features. *arXiv preprint arXiv:2508.12535*, 2025.

Cheng-Ting Chou, George Liu, Jessica Sun, Cole Blondin, Kevin Zhu, Vasu Sharma, and Sean O'Brien. Causal language control in multilingual transformers via sparse feature steering. *arXiv preprint arXiv:2507.13410*, 2025.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

Parmida Davarmanesh, Ashia Wilson, and Adityanarayanan Radhakrishnan. Efficient and accurate steering of large language models through attention-guided feature learning. *arXiv preprint arXiv:2602.00333*, 2026.

Can Demircan, Tankred Saanum, Akshay Kumar Jagadish, Marcel Binz, and Eric Schulz. Sparse autoencoders reveal temporal difference learning in large language models. *ArXiv*, abs/2410.01280, 2024.

Boyi Deng, Yu Wan, Baosong Yang, Yidan Zhang, and Fuli Feng. Unveiling language-specific features in large language models via sparse autoencoders. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4563–4608, 2025.

Jacob Dunefsky and Arman Cohan. One-shot optimized steering vectors mediate safety-relevant behaviors in LLMs. In *Second Conference on Language Modeling*, 2025.

Esin Durmus, Alex Tamkin, Jack Clark, Jerry Wei, Jonathan Marcus, Joshua Batson, Kunal Handa, Liane Lovitt, Meg Tong, Miles McCain, et al. Evaluating feature steering: A case study in mitigating social biases. *Anthropic Research*, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Yi Fang, Wenjie Wang, Mingfeng Xue, Boyi Deng, Fengli Xu, Dayiheng Liu, and Fuli Feng. Controllable llm reasoning via sparse autoencoder-based steering. *arXiv preprint arXiv:2601.03595*, 2026.

Andrey Galichin, Alexey Dontsov, Polina Druzhinina, Anton Razzhigaev, Oleg Y Rogov, Elena Tutubalina, and Ivan Oseledets. I have covered all the bases here: Interpreting reasoning features in large language models via sparse autoencoders. *arXiv preprint arXiv:2503.18878*, 2025.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Jiatao Gu, Kyunghyun Cho, and Victor OK Li. Trainable greedy decoding for neural machine translation. *arXiv preprint arXiv:1702.02429*, 2017.

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek Abdelzaher, and Heng Ji. Word embeddings are steers for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16410–16430, 2024.

Zirui He, Mingyu Jin, Bo Shen, Ali Payani, Yongfeng Zhang, and Mengnan Du. SAE-SSV: Supervised steering in sparse representation spaces for reliable control of language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, November 2025a.

Zirui He, Haiyan Zhao, Yiran Qiao, Fan Yang, Ali Payani, Jing Ma, and Mengnan Du. Saif: A sparse autoencoder framework for interpreting and steering instruction following of language models. *arXiv preprint arXiv:2502.11356*, 2025b.

Evan Hernandez, Belinda Z Li, and Jacob Andreas. Inspecting and editing knowledge representations in language models. *arXiv preprint arXiv:2304.00740*, 2023.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Shawn Im and Yixuan Li. A unified understanding and evaluation of steering methods. *arXiv preprint arXiv:2502.02716*, 2025.

Shruti Joshi, Andrea Dittadi, Sébastien Lachapelle, and Dhanya Sridhar. Identifiable steering via sparse autoencoding of multi-concept shifts. *arXiv preprint arXiv:2502.12179*, 2025.

Ceyhun Efe Kayan and Li Zhang. Prototype-based dynamic steering for large language models. *arXiv preprint arXiv:2510.05498*, 2025.

Muhammad Khalifa, Hady ElSahar, and Marc Dymetman. A distributional approach to controlled text generation. *ArXiv*, December 2020.

Daniel Khashabi, Shane Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, Sameer Singh, et al. Prompt waywardness: The curious case of discretized interpretation of continuous prompts. *arXiv preprint arXiv:2112.08348*, 2021.

Doyoung Kim, Jongwon Lee, Jinho Park, and Minjoon Seo. How language models extrapolate outside the training data: A case study in textualized gridworld. 2024.

Kishore Konda, Roland Memisevic, and David Krueger. Zero-bias autoencoders and the benefits of co-adapting features. *arXiv preprint arXiv:1402.3337*, 2014.

Akshay Kulkarni, Tsui-Wei Weng, Vivek Narayanaswamy, Shusen Liu, Wesam A Sakla, and Kowshik Thopalli. Interpretable and steerable concept bottleneck sparse autoencoders. *arXiv preprint arXiv:2512.10805*, 2025.

Wen Lai, Alexander Fraser, and Ivan Titov. Joint localization and activation editing for low-resource fine-tuning. In *Forty-second International Conference on Machine Learning*, 2025.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pp. 1558–1566. PMLR, 2016.

Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. Sparse deep belief net model for visual area v2. *Advances in neural information processing systems*, 20, 2007.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context llms struggle with long in-context learning. *Trans. Mach. Learn. Res.*, 2025, 2024.

Yichen Li, Zhiting Fan, Ruizhe Chen, Xiaotang Gai, Luqi Gong, Yan Zhang, and Zuozhu Liu. Fairsteer: Inference time debiasing for llms with dynamic activation steering. *arXiv preprint arXiv:2504.14492*, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. *Annual Meeting of the Association for Computational Linguistics*, May 2021.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 806–814, 2015.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. corr abs/2307.03172 (2023). *arXiv preprint arXiv:2307.03172*, 10, 2023a.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023b.

Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013.

Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

Sam Marks and Aaron Mueller. Some open-source dictionaries and dictionary learning infrastructure. In *AI Alignment Forum*, 2023.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 746–751, 2013.

Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. Mix and match: Learning-free controllable text generation using energy language models. *arXiv preprint arXiv:2203.13299*, 2022.

Rajiv Movva, Kenny Peng, Nikhil Garg, Jon M. Kleinberg, and Emma Pierson. Sparse autoencoders for hypothesis generation. *ArXiv*, abs/2502.04382, 2025.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Duy Nguyen, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Multi-attribute steering of language models via targeted intervention. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangde. Steering language model refusal with sparse autoencoders. *arXiv preprint arXiv:2411.11296*, 2024.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024a.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024b.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

Pau Rodriguez, Michal Klein, Eleonora Gualdoni, Valentino Maiorca, Arno Blaas, Luca Zappella, Marco Cuturi, and Xavier Suau. Lineas: End-to-end learning of activation steering with a distributional loss. *NeurIPS*, 2025.

Abel Salinas and Fred Morstatter. The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance. *arXiv preprint arXiv:2401.03729*, 2024.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

Lee Sharkey, Dan Braun, and Beren Millidge. Taking features out of superposition with sparse autoencoders. In *AI Alignment Forum*, volume 6, pp. 12–13, 2022.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Samuel Soo, Wesley Teng, Chandrasekaran Balaganesh, Tan Guoxian, and Ming YAN. Interpretable steering of large language models with feature guided activation additions. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. Improving instruction-following in language models through activation steering. *arXiv preprint arXiv:2410.12877*, 2024.

Nishant Subramani, Nivedita Suresh, and Matthew E Peters. Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*, 2022.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Qwen Team. Qwen3 technical report, 2025.

Harrish Thasarathan, Julian Forsyth, Thomas Fel, Matthew Kowal, and Konstantinos Derpanis. Universal sparse autoencoders: Interpretable cross-model concept alignment. *ArXiv*, abs/2502.03714, 2025.

Max Torop, Aria Masoomi, Masih Eskandar, and Jennifer Dy. Disco: Disentangled communication steering for large language models. *Advances in Neural Information Processing Systems*, 38:57803–57855, 2025.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv e-prints*, pp. arXiv–2308, 2023.

Praveen Venkateswaran and Danish Contractor. Spotlight your instructions: Instruction-following with dynamic attention steering. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2025.

Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*, 2019.

Anyi Wang, Xuansheng Wu, Dong Shu, Yunpu Ma, and Ninghao Liu. Enhancing llm steering through sparse autoencoder-based vector refinement. *arXiv preprint arXiv:2509.23799*, 2025a.

Weixuan Wang, Jingyuan Yang, and Wei Peng. Semantics-adaptive activation intervention for llms via dynamic steering vectors. In *ICLR*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Kaiyue Wen, Huaqing Zhang, Hongzhou Lin, and Jingzhao Zhang. From sparse dependence to sparse attention: Unveiling how chain-of-thought enhances transformer sample efficiency. *ArXiv*, abs/2410.05459, 2024.

Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Xuansheng Wu, Jiayi Yuan, Wenlin Yao, Xiaoming Zhai, and Ninghao Liu. Interpreting and steering llms with mutual information-based explanations on sparse autoencoders. *arXiv preprint arXiv:2502.15576*, 2025a.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. In *International Conference on Machine Learning*, pp. 67035–67080. PMLR, 2025b.

Jiaqing Xie. A comparative analysis of sparse autoencoder and activation difference in language model steering. *arXiv preprint arXiv:2510.01246*, 2025.

Xuan Yang, Jiayu Liu, Yuhang Lai, Hao Xu, Zhenya Huang, and Ning Miao. Step-level sparse autoencoder for reasoning process interpretation. *arXiv preprint arXiv:2603.03031*, 2026.

Xi Ye and Greg Durrett. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in neural information processing systems*, 35:30378–30392, 2022.

Qingyu Yin, Chak Tou Leong, Hongbo Zhang, Minjun Zhu, Hanqi Yan, Qiang Zhang, Yulan He, Wenjie Li, Jun Wang, Yue Zhang, and Linyi Yang. Direct preference optimization using sparse feature-level constraints. *ArXiv*, abs/2411.07618, 2024.

Zeyu Yun, Yubei Chen, Bruno A Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. *arXiv preprint arXiv:2103.15949*, 2021.

Alexander Zai and Brandon Brown. *Deep Reinforcement Learning in Action*. Manning Publications, 2020.

Li-Ming Zhan, LIU Bo, Zexin Lu, Yujie Feng, Chengqiang Xie, Jiannong Cao, and Xiao-Ming Wu. Deal: Disentangling transformer head activations for llm steering. In *Knowledgeable Foundation Models at ACL 2025*, 2025.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37, 2023.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. Personalized text generation with contrastive activation steering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7128–7141, Vienna, Austria, July 2025a. Association for Computational Linguistics.

Qingru Zhang, Chandan Singh, Liyuan Liu, Xiaodong Liu, Bin Yu, Jianfeng Gao, and Tuo Zhao. Tell your model where to attend: Post-hoc attention steering for LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhenyu Zhang, Xiaoxia Wu, Zhongzhu Zhou, Qingyang Wu, Yineng Zhang, Pragaash Ponnusamy, Harikaran Subbaraj, Jue Wang, Shuaiwen Leon Song, and Ben Athiwaratkun. Understanding and steering the cognitive behaviors of reasoning models at test-time. *ArXiv*, abs/2512.24574, 2025b.

Haiyan Zhao, Xuansheng Wu, Fan Yang, Bo Shen, Ninghao Liu, and Mengnan Du. Denoising concept vectors with sparse autoencoders for improved language model steering. In *Findings of the Association for Computational Linguistics: EACL 2026*, Rabat, Morocco, March 2026. Association for Computational Linguistics.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Kam-Fai Wong, and Pasquale Minervini. Steering knowledge selection behaviours in llms via sae-based representation engineering. *arXiv preprint arXiv:2410.15999*, 2024.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Steering large language models using ape. In *NeurIPS ML Safety Workshop*, 2022.

A Organization of the Appendix.

Table 4 summarizes the contents of each appendix section for quick reference.

Table 4: Guide to the Appendix.

Section	Domain	Type	Contents
A.1	TGW	Visualization	Attention heatmaps evolving across gradient-ascent steps for Short, Safe, and Long targets.
A.2	TGW	Quantitative	Cell-level attention analysis with numerical attention weights under each steering target.
A.3	TGW	Evaluation	Extrapolation results on larger grids (M or $N \geq 10$) using Qwen3-4B.
A.4	TGW / Bloom	Ablation	Direct target-center assignment vs. gradient-based optimization.
B	Bloom	Baseline	Effect of increasing in-context examples on cognitive-style accuracy for the base model.
C	Bloom	Protocol	AxBench-adapted three-axis LLM judge setup: Bloom level definitions and verbatim judge prompts.
D	Bloom	Evaluation	Full AxBench per-axis scores (s_c, s_i, s_f, HM) for methods and targets.
E	Bloom	Evaluation	Output-class confusion matrices and per-target relevance rates under GPT-4o and Claude judges.
F	TGW	Ablation	Steering effectiveness across different attention layers.
G	TGW	Analysis	Detailed L1 vs. L2 non-target drift values and optimization steps per model.
H	Bloom	Qualitative	Steered generation examples (Apply, Create, Remember) comparing base vs. steered outputs on Phi.
I	Bloom	Qualitative	Side-by-side comparisons of dense query steering (w/o SAE) vs. SAE-based steering with judgments.

A.1 Attention Heatmaps for Spatial Reasoning in Textualized Gridworld

We visualize how query-level sparse steering progressively refines the model's attention patterns over the grid during inference-time optimization. Each figure contains four panels: the unsteered output (top-left), which is typically invalid, and three subsequent panels showing intermediate steps of gradient ascent on the sparse query latents. Cell-level attention heatmaps are computed by averaging attention weights across heads, extracting the attention distribution of the final token position, aggregating over the token span corresponding to each grid cell, and normalizing across cells to obtain the distribution $p(c)$.

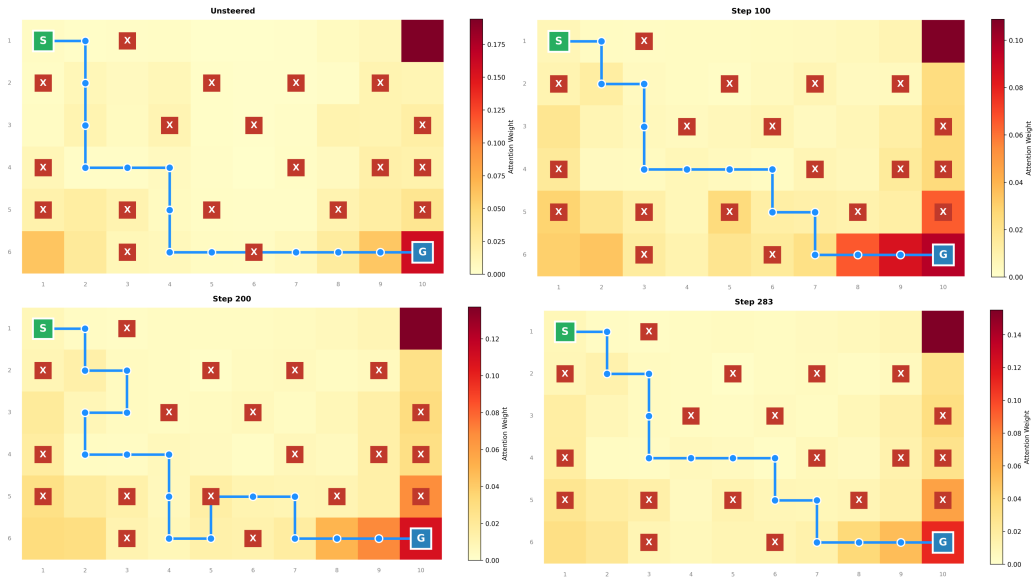


Figure 4: Evolution of attention heatmaps when steering toward the **Short** path target.

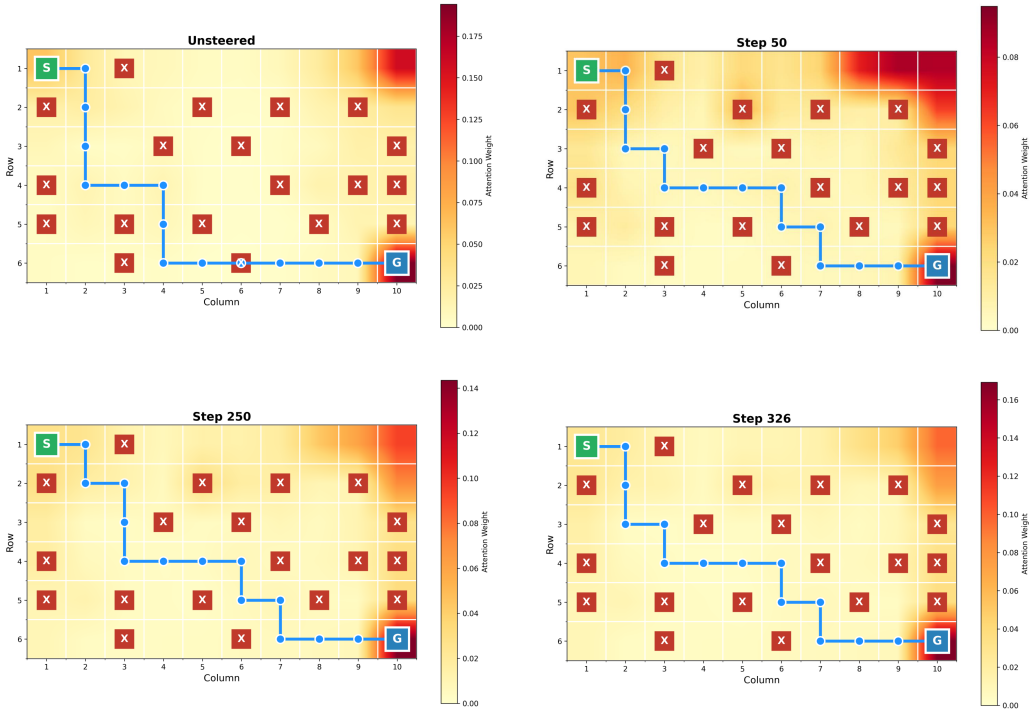


Figure 5: Evolution of attention heatmaps when steering toward the **Safe** path target.

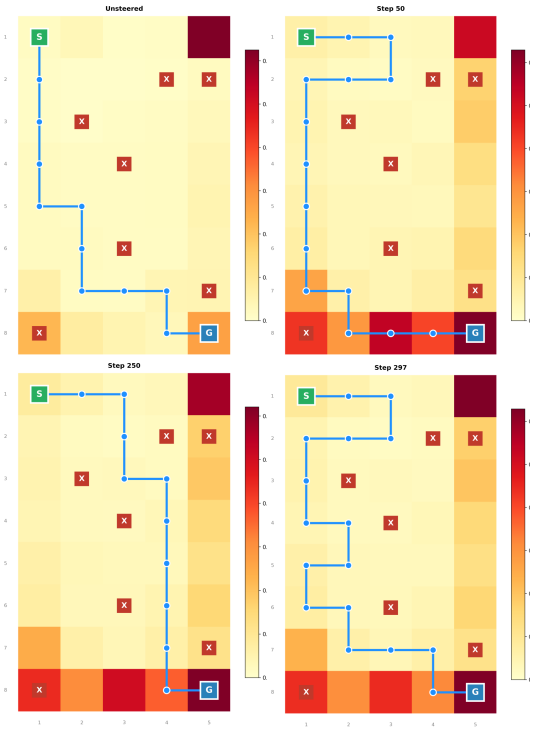


Figure 6: Evolution of attention heatmaps when steering toward the **Long** path target.

A.2 Quantitative Cell-Level Attention Analysis Under Steering

To understand how steering reshapes the model's internal spatial reasoning at a mechanistic level, we examine whether the model's attention over grid cells shifts in a manner consistent with the target attribute as gradient-based optimization progresses. For instance, when steering toward safe paths, we expect attention to gradually stay away from wall-adjacent cells. Similarly, short steering should concentrate attention along the most direct route, while long steering should distribute it more broadly across the grid to support extended trajectories. To test these hypotheses, we extract and visualize cell level attention distributions $p(c)$ at intermediate optimization steps (Figure 7-Figure 9). This analysis serves two purposes: it provides interpretable evidence that query level interventions produce spatially coherent changes in the model's reasoning process, and it validates that steering modifies (Section 2.2) what information the model retrieves through attention.

Given the attention weight tensor $\mathbf{A} \in \mathbb{R}^{H \times L \times L}$ from the intervention layer ℓ (with H heads and sequence length L), we compute cell-level attention as follows:

Head Averaging. We first average attention across all heads to obtain a more stable signal:

$$\bar{\mathbf{A}} = \frac{1}{H} \sum_{h=1}^{H} \mathbf{A}_h \in \mathbb{R}^{L \times L} \tag{7}$$

Query Selection. We extract attention from the final generated token position:

$$\bar{\mathbf{a}} = \bar{\mathbf{A}}[L-1, :] \in \mathbb{R}^{L} \tag{8}$$

Cell Aggregation. For each grid cell c with corresponding token indices $\mathcal{T}_c \subseteq \{1, \ldots, L\}$, we compute the mean attention:

$$\mathrm{attn}(c) = \frac{1}{|\mathcal{T}_c|} \sum_{t \in \mathcal{T}_c} \bar{\mathbf{a}}[t] \tag{9}$$

Normalization. Finally, we normalize across all grid cells to obtain a probability distribution over spatial locations:

$$p(c) = \frac{\mathrm{attn}(c)}{\sum_{c' \in \mathcal{C}} \mathrm{attn}(c')} \tag{10}$$

where $\mathcal{C}$ denotes the set of all grid cells.

How to read Figure 7-Figure 9: Each figure contains four panels showing cell-level attention at progressive optimization steps (leftmost = unsteered baseline, rightmost = final). The number in each cell is its normalized attention weight(Equation 10). X = wall (red/yellow border), S = start, G = goal, blue arrows = generated path. To track the effect of steering, compare attention values on the same cell across panels, increasing values indicate the model is allocating more retrieval weight to that location, while decreasing values indicate reduced focus.

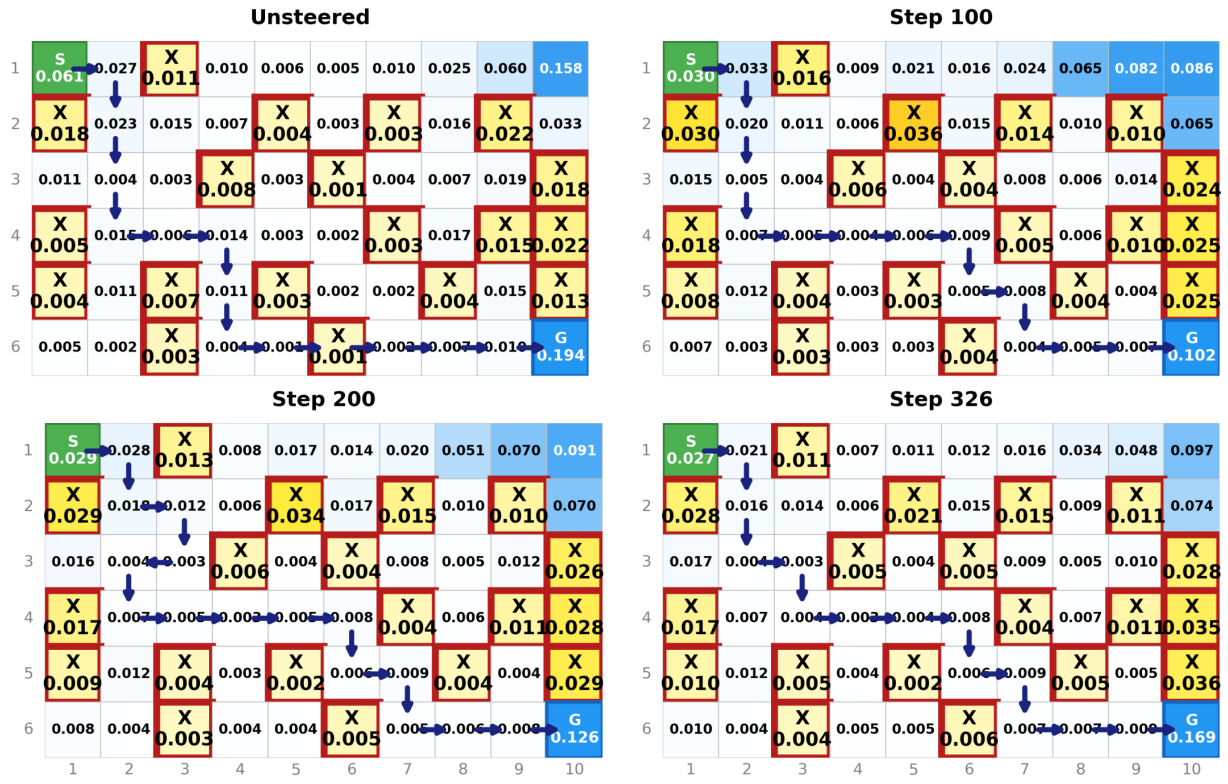


Figure 7: Cell level attention distributions during steering toward the Safe path.

In the unsteered baseline, attention is concentrated at the endpoints (S = 0.061, G = 0.194, row 0 column 9 = 0.158), with walls and interior cells receiving minimal weight. In the unsteered phase, the model plans a direct start to goal route that passes through wall adjacent regions. Under safe steering, this endpoint dominance diminishes (S drops to 0.027, G to 0.169, row 0 column 9 to 0.097), and rest of the attention redistributes selectively. Walls near the planned safe route gain attention (X at (1,4): 0.004 to 0.021, X at (3,0): 0.005 to 0.017, X at (4,9): 0.013 to 0.036), while interior walls far from the route stay flat or decrease (i.e., , X at (2,3): 0.008 to 0.005, X at (4,4): 0.003 to 0.002). We hypothesize increased attention on a wall cell does not mean the model is drawn toward it, instead, the query mechanism retrieves obstacle location information, which then informs next-token prediction to select cells with greater clearance.

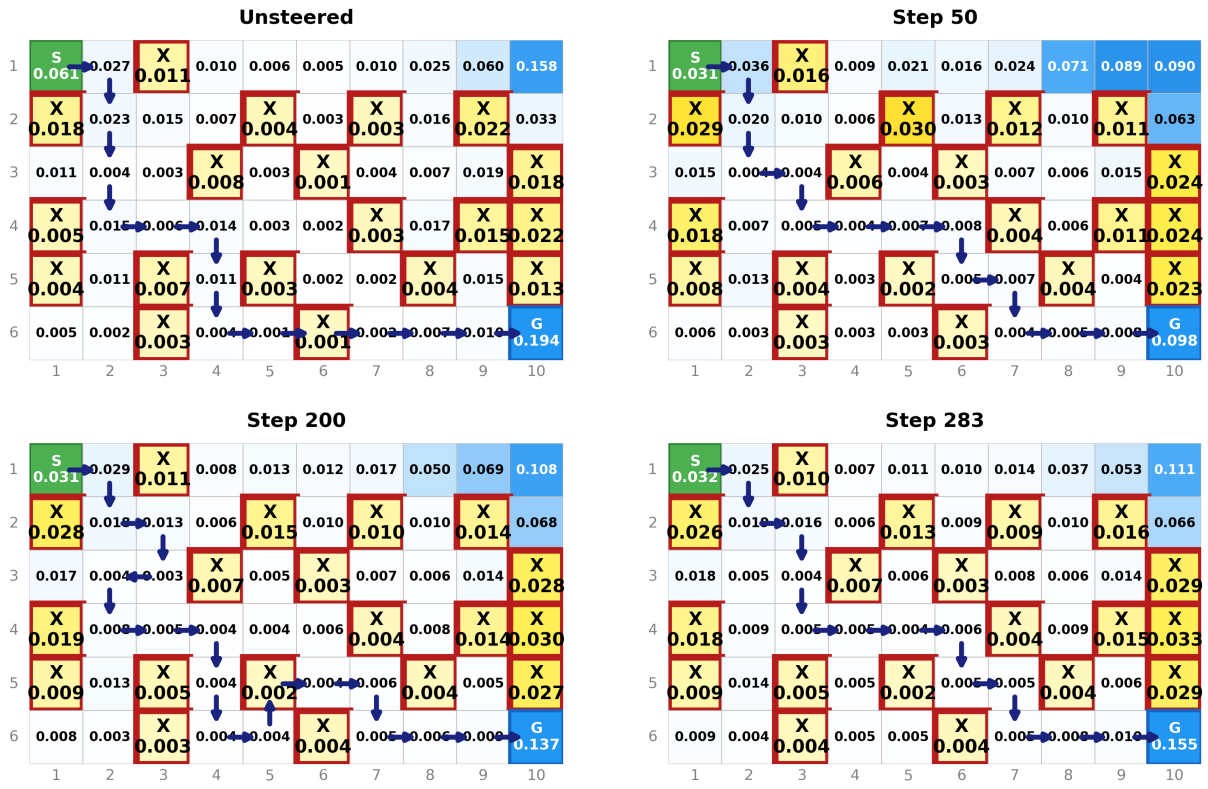


Figure 8: Cell level attention distributions during steering toward the Short path.

Under short steering, attention redistributes across interior cells along the grid. It reduces fixation on endpoints and spreads attention to traversable interior cells, enabling the model to identify a more direct route.

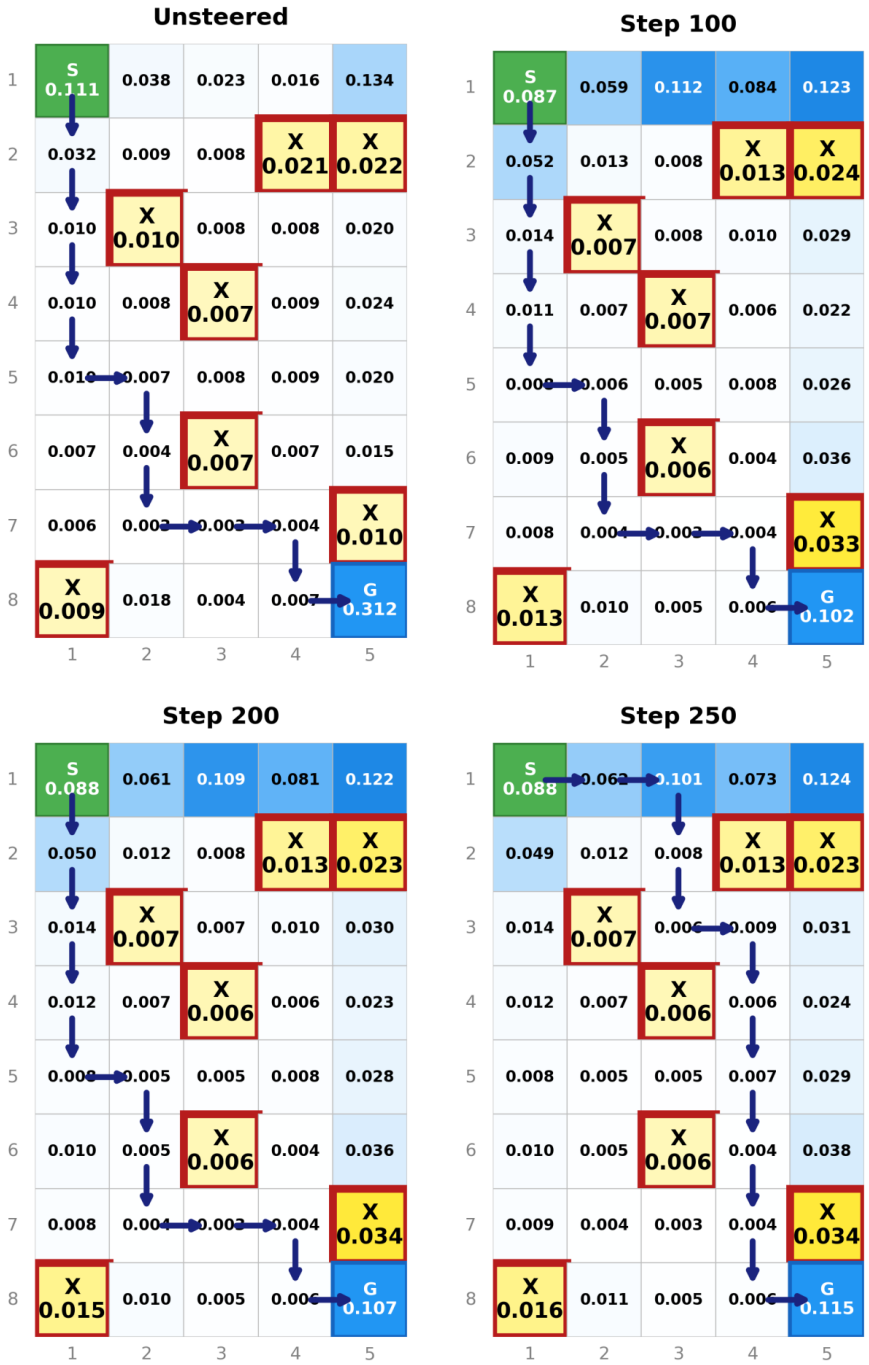


Figure 9: Cell level attention distributions during steering toward the Long path.

Long steering produces a distinctive redistribution pattern. Attention on G drops sharply by roughly 63% (0.312 to 0.115 by Step 250), a much larger endpoint reduction than observed in other steering targets. This attention concentrates heavily along the top row. We hypothesize that this combination of strong downweighting of the goal and heightened attention to interior cells encourages the model to traverse those cells rather than heading directly toward G.

A.3 Extrapolation to Larger Grids

Additional details are described in footnote 1.

Table 5: Main Planning Performance for extrapolation data using Qwen under Hard Constraints. Results are reported as (Short / Safe / Long) path types.

Method	Success ↑ (%)
ICL	13.53 / 1.7 / 1.3
CAA	18 / 2.03 / 1.1
ITI	11.26 / 3.6 / 1.6
SAE-SSV	15.24 / 3.36 / 1.38
DENSE-OPT	16.13 / 3.12 / 1.23
SAE-OPT	**18.2 / 4.67 / 1.9**

A.4 Ablation: Direct Target-Center Assignment

A natural simplification of our method is to replace gradient-based optimization with direct substitution of the target prototype. On TGW (Table 6), direct centroid replacement plateaus well below gradient-based steering even when the support set is scaled to 1000 examples. For bloom's taxonomy (Figure 10), We observe that, instead of rigid assignment of the target center embedding, the gradient ascent based approach captures the target cognitive style more effectively.

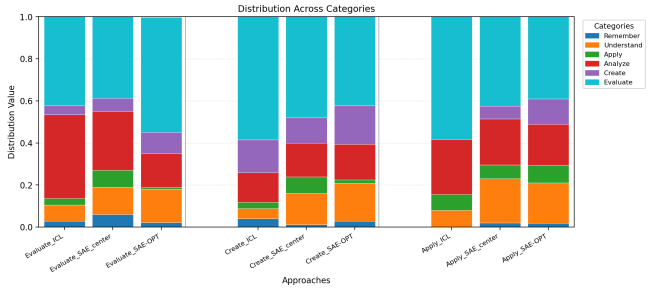


Figure 10: Measured class distribution (GPT-4o judge) showing steering accuracy across approaches. Target classes appear on the X-axis; the Y-axis shows the class-category distribution of steered text.

Examples (N)	Success ↑ (%)	Logic Viol. ↓ (%)
10	47.73 / 12.39 / 8.03	32.19 / 32.42 / 31.80
100	47.08 / 11.93 / 7.47	32.39 / 32.32 / 50.17
1000	48.09 / 12.75 / 7.89	32.39 / 32.56 / 50.17

Table 6: Planning performance on Qwen when replacing with the target center across varying numbers of examples (N). Results are reported as (Short / Safe / Long) path types.

B In context learning in baseline model

Figure 11 demonstrates how the baseline model's accuracy improves as the number of training examples (GPT-4o judge). With only one example for the "create" cognitive style feedback, the model tends to generate responses that align more with "evaluate". However, as more examples are provided, the model becomes better at generating feedback in the intended "create" cognitive style. Interestingly, the results also reveal that LLMs typically generate feedback in "evaluate" cognitive styles, even with just a single example of another style (i.e., apply or create), the model generates a majority of responses in this 'evaluate' style. For this baseline implementation, we select the *Phi-3-Mini-128K Instruct model* primarily due to its extended context window, which allows us to process N examples simultaneously. While the literature contains no substantive claims comparing this model's performance to *Phi-3-Mini-4K Instruct model*, except expanded context length capacity Abdin et al. (2024)

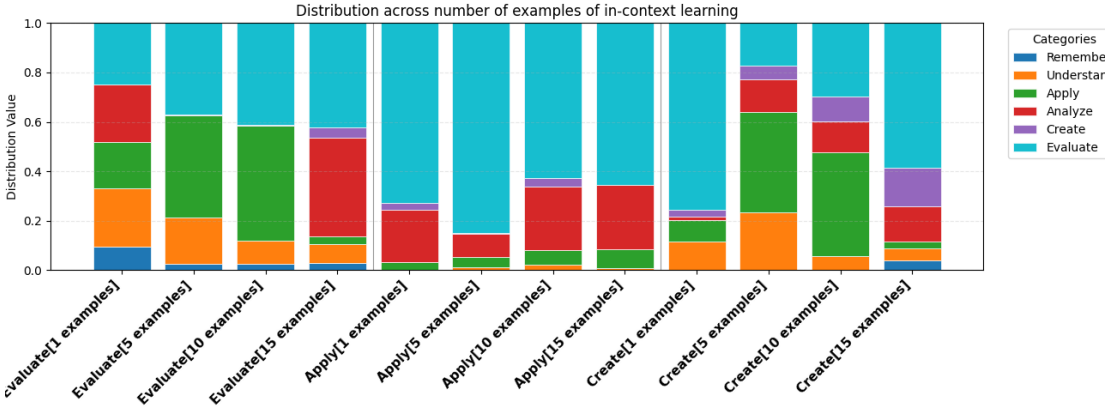


Figure 11: Increasing the number of examples for a specific cognitive style improves the possibility of generating feedback in that style for base model.

C AxBench-Adapted Three-Axis LLM Judge for Bloom's Taxonomy Evaluation

To provide a more rigorous and quantitative evaluation of cognitive-style steering, we adopt a three-axis LLM-as-judge protocol adapted from AxBench Wu et al. (2025b). For every steered generation we obtain three independent scores on a 0–2 scale:

- s_c (Concept): How well the response demonstrates the *target* Bloom's Taxonomy cognitive level.
- s_i (Instruct): How faithfully the response adheres to the original instruction/topic.
- s_f (Fluency): Naturalness and grammatical fluency of the generated text.

The overall score is the harmonic mean of the three axes:

$$\text{Overall} = \frac{3}{\frac{1}{s_c} + \frac{1}{s_i} + \frac{1}{s_f}},$$

which penalizes any method that sacrifices one axis to inflate another.

The judge LLM (Claude-Sonnet-4) is provided with the exact Bloom level definitions below and the three prompts shown verbatim.

C.1 Bloom's Taxonomy Level Definitions

These definitions are supplied verbatim to the judge in every evaluation:

Level	Definition
Remember	Retrieving relevant knowledge from long-term memory. Recognizing, recalling, or identifying facts, terms, and basic concepts without necessarily understanding their meaning.
Understand	Constructing meaning from information. Interpreting, exemplifying, classifying, summarizing, inferring, comparing, or explaining concepts in one's own words.
Apply	Carrying out or using a procedure in a given situation. Executing or implementing learned methods, concepts, or techniques to solve problems in new contexts.
Analyze	Breaking information into constituent parts and detecting how parts relate to one another and to an overall structure. Differentiating, organizing, comparing components, and identifying causes or motives.
Evaluate	Making judgments based on criteria and standards. Critiquing, checking, assessing strengths and weaknesses, and justifying decisions or conclusions.
Create	Putting elements together to form a coherent or functional whole. Generating new ideas, designing new solutions, producing original work, or reorganizing elements into new patterns or structures.

Table 7: Bloom's Taxonomy definitions provided to the judge LLM.

C.2 Judge Prompts (Verbatim)

The three prompts used for s_c, s_i, and s_f are given below exactly as implemented.

```
# Prompt for s_c (Bloom Concept)
[Prompt 1]
Please act as an impartial judge and evaluate whether the specified Bloom's Taxonomy cognitive
    level is demonstrated in the response provided below. Focus solely on whether the cognitive
    style is clearly exhibited in how the information is structured and delivered.
Begin your evaluation by providing a short explanation of whether the cognitive style is clearly
    demonstrated. Point out specific phrasing or structuring in the response that aligns with the
    target level.
After providing your explanation, rate the cognitive style's relevance on a scale from 0 to 2:
- 0 indicates the target cognitive level is not demonstrated at all.
- 1 indicates the cognitive level is somewhat present but minimally or awkwardly incorporated.
- 2 indicates the cognitive level is fully and effectively demonstrated with strong, natural
    integration.
Provide your rating using this exact format: "Rating: [[score]]".

[Target Cognitive Level Start]
{target_bloom_level}: {bloom_level_definition}
[Target Cognitive Level End]

[Response Start]
{model_generation}
[Response End]
```

```
# Prompt for s_i (Instruction Adherence)
[Prompt 2]
Please act as an impartial judge and evaluate whether the sentence fragment provided below is
    related to the instruction. Focus solely on the degree of relatedness in terms of topic,
    regardless of grammar, coherence, or informativeness.
Begin your evaluation by providing a brief explanation of whether the sentence is related to the
    instruction, and point out references related to the instruction. After providing your
    explanation, rate the instruction relevance on a scale from 0 to 2, where 0 indicates the
    sentence is unrelated to the instruction, 1 indicates it is somewhat related but only
    minimally or indirectly relevant in terms of topic, and 2 indicates it is more clearly and
    directly related to the instruction.
Provide your rating using this exact format: "Rating: [[score]]".

[Instruction Start]
{original_instruction}
[Instruction End]

[Sentence Fragment Start]
{model_generation}
[Sentence Fragment End]
```

```
# Prompt for s_f (Fluency)
[Prompt 3]
Please act as an impartial judge and evaluate the fluency of the sentence fragment provided below.
    Focus solely on fluency, disregarding its completeness, relevance, coherence with any broader
    context, or informativeness.
Begin your evaluation by briefly describing the fluency of the sentence, noting any unnatural
    phrasing, awkward transitions, grammatical errors, or repetitive structures that may hinder
    readability. After providing your explanation, rate the sentence's fluency on a scale from 0
    to 2, where 0 indicates the sentence is not fluent and highly unnatural (e.g.,
    incomprehensible or repetitive), 1 indicates it is somewhat fluent but contains noticeable
    errors or awkward phrasing, and 2 indicates the sentence is fluent and almost perfect.
Provide your rating using this exact format: "Rating: [[score]]".

[Sentence Fragment Start]
```

```
{model_generation}
[Sentence Fragment End]
```

D Quantitative Evaluation with AxBench

Table 8: AxBench three-axis evaluation on Bloom's Taxonomy steering (Phi). we report concept alignment (s_c), instruction-following (s_i), fluency (s_f), and their harmonic mean (HM). Scores $\in [0, 2]$. **Bold** indicates best per target.

Target	Method	Type	s_c (Concept)	s_i (Instruct)	s_f (Fluency)	HM
Remember	CAA	Static	**0.203**	1.872	1.964	**0.503**
	Disco-Q	Static	0.054	1.928	1.956	0.153
	DENSE-OPT	Dynamic	0.086	1.940	1.974	0.237
	SAE-OPT	Dynamic	0.132	**1.945**	**1.977**	0.349
Understand	CAA	Static	0.341	1.923	1.859	0.752
	Disco-Q	Static	0.378	1.932	1.957	0.817
	DENSE-OPT	Dynamic	0.437	1.939	**1.983**	0.907
	SAE-OPT	Dynamic	**0.468**	**1.940**	1.976	**0.950**
Apply	CAA	Static	0.083	1.928	1.952	0.229
	Disco-Q	Static	0.273	1.932	1.968	0.640
	DENSE-OPT	Dynamic	0.293	1.938	1.971	0.676
	SAE-OPT	Dynamic	**0.307**	**1.939**	**1.976**	**0.701**
Analyze	CAA	Static	0.863	1.937	1.806	1.346
	Disco-Q	Static	0.862	1.934	1.961	1.372
	DENSE-OPT	Dynamic	0.864	**1.944**	**1.988**	1.379
	SAE-OPT	Dynamic	**0.871**	1.932	1.975	**1.381**
Evaluate	CAA	Static	0.876	1.916	1.867	1.364
	Disco-Q	Static	0.866	1.933	1.962	1.375
	DENSE-OPT	Dynamic	0.881	1.932	1.973	1.389
	SAE-OPT	Dynamic	**0.890**	**1.938**	**1.984**	**1.399**
Create	CAA	Static	0.179	1.681	1.942	0.448
	Disco-Q	Static	0.340	1.931	1.964	0.756
	DENSE-OPT	Dynamic	0.305	1.928	**1.984**	0.697
	SAE-OPT	Dynamic	**0.344**	**1.947**	1.976	**0.764**

E Confusion Matrices and Relevance Rates

Table 9: Confusion matrices of output-class distributions (%) under GPT-4o (left) and Claude (right) judges for each steering method. Rows: intended target class. Columns: predicted class. Diagonal cells (shaded) are target hits, ideally the highest in each row. Each row sums to 100%.

Method	GPT-4o judge						Claude judge					
		Rem.	Und.	App.	Ana.	Eva.	Cre.		Rem.	Und.	App.	Ana.	Eva.	Cre.
ICL	Rem.	**7.4**	42.2	0.0	24.3	26.1	0.0	Rem.	**10.9**	35.1	0.0	24.1	29.9	0.0
	Und.	3.2	**24.7**	1.2	46.0	23.1	1.8	Und.	1.2	**13.0**	1.8	50.7	33.3	0.0
	App.	0.0	0.8	**7.5**	26.2	65.5	0.0	App.	2.1	6.3	**0.9**	23.5	67.1	0.1
	Ana.	2.1	13.3	7.5	**27.6**	44.4	5.1	Ana.	4.7	10.0	6.7	**32.7**	38.6	7.3
	Eva.	2.8	7.7	3.2	39.7	**42.3**	4.3	Eva.	1.8	2.5	6.3	35.4	**49.0**	5.0
	Cre.	4.0	4.8	3.0	14.1	58.5	**15.6**	Cre.	2.8	8.2	5.0	25.0	43.7	**15.3**
CAA	Rem.	**9.7**	51.8	0.0	36.4	2.1	0.0	Rem.	**13.1**	5.5	0.0	80.2	1.2	0.0
	Und.	3.3	**21.5**	2.2	58.1	14.1	0.8	Und.	5.1	**15.1**	0.2	61.7	17.7	0.2
	App.	4.3	34.8	**1.6**	30.8	28.0	0.5	App.	9.7	25.2	**1.1**	38.3	24.7	1.0
	Ana.	1.4	20.1	3.2	**39.4**	33.6	2.3	Ana.	3.8	8.0	3.2	**42.6**	41.0	1.4
	Eva.	2.5	12.0	3.1	34.8	**45.6**	2.0	Eva.	4.0	6.3	0.5	37.7	**46.2**	5.3
	Cre.	4.3	18.6	0.7	4.1	64.0	**8.3**	Cre.	3.6	10.0	0.5	8.6	58.2	**19.1**
ITI	Rem.	**7.6**	46.5	3.5	23.3	19.1	0.0	Rem.	**5.9**	31.5	2.2	34.8	25.6	0.0
	Und.	4.1	**24.7**	2.7	34.1	34.4	0.0	Und.	2.7	**15.0**	2.6	39.8	39.8	0.1
	App.	7.2	25.6	**4.8**	35.9	25.7	0.8	App.	5.0	20.0	**5.6**	45.0	23.1	1.3
	Ana.	12.0	26.0	7.8	**15.7**	32.0	6.5	Ana.	7.3	19.7	3.5	**21.2**	36.3	12.0
	Eva.	11.5	24.5	2.5	16.0	**17.5**	28.0	Eva.	11.6	17.7	0.3	19.8	**12.8**	37.8
	Cre.	10.2	34.1	1.5	24.0	20.7	**9.5**	Cre.	6.3	24.7	0.7	39.1	29.1	**0.1**
SAE-SSV	Rem.	**1.4**	20.5	2.2	17.7	45.9	12.3	Rem.	**2.9**	15.1	0.6	18.6	45.2	17.6
	Und.	1.6	**21.2**	2.8	18.4	43.3	12.7	Und.	3.0	**16.9**	0.6	20.2	41.0	18.3
	App.	1.5	21.3	**1.9**	20.3	43.1	11.9	App.	2.7	14.2	**3.5**	20.8	40.8	18.0
	Ana.	2.3	19.4	2.2	**19.3**	45.3	11.5	Ana.	3.1	15.7	0.6	**20.0**	42.3	18.3
	Eva.	2.3	18.7	1.9	17.3	**46.7**	13.1	Eva.	3.0	14.7	0.8	19.7	**42.8**	19.0
	Cre.	1.9	20.0	1.9	17.6	46.6	**12.0**	Cre.	2.7	15.5	0.5	19.3	44.2	**17.8**
Disco-Q	Rem.	**2.5**	18.6	1.2	19.2	44.8	13.7	Rem.	**3.9**	14.0	0.4	19.7	43.7	18.3
	Und.	2.7	**18.4**	1.5	19.6	43.6	14.2	Und.	3.9	**14.0**	0.3	19.8	43.7	18.3
	App.	2.7	18.5	**1.4**	18.8	45.2	13.4	App.	3.9	14.1	**0.4**	19.6	43.7	18.3
	Ana.	2.6	18.0	1.6	**19.1**	45.0	13.7	Ana.	3.9	14.0	0.4	**19.8**	43.6	18.3
	Eva.	2.6	18.5	1.3	19.5	**44.0**	14.1	Eva.	3.9	14.0	0.4	19.8	**43.7**	18.2
	Cre.	2.7	18.3	1.6	19.6	44.0	**13.8**	Cre.	3.9	14.1	0.4	19.8	43.5	**18.3**
DENSE-OPT	Rem.	**8.5**	13.8	8.5	15.5	41.2	12.5	Rem.	**12.2**	14.6	0.3	17.4	40	15.5
	Und.	2.2	**25.1**	1.6	18.1	41.3	11.7	Und.	3.1	**16.3**	0.4	21.7	41.5	17
	App.	2.0	17.8	**5.2**	19.5	44.0	11.5	App.	2.1	14.1	**7.2**	22.1	40.4	14.1
	Ana.	2.3	17.4	1.5	**20.5**	46.1	12.2	Ana.	2.3	13.4	0.4	**27.1**	40.5	16.3
	Eva.	2.1	15.5	2.0	17.0	**52.5**	10.9	Eva.	2.3	14.0	0.3	19.3	**48.5**	15.6
	Cre.	2.4	20.2	0.8	17.0	45.0	**14.6**	Cre.	2.7	15.8	0.5	19.4	45.1	**16.5**
SAE-OPT-ANCH	Rem.	**1.6**	1.9	3.6	18.6	73.0	1.3	Rem.	**2.8**	1.9	1.0	16.5	76.5	1.3
	Und.	1.2	**4.5**	4.0	14.5	75.2	0.6	Und.	1.7	**3.6**	2.2	15.5	75.5	1.5
	App.	2.0	3.5	**4.1**	14.2	74.8	1.4	App.	2.3	3.0	**5.2**	13.0	75.3	1.2
	Ana.	1.0	1.4	4.0	**16.3**	77.0	0.3	Ana.	1.8	2.8	1.2	**12.4**	80.5	1.3
	Eva.	1.1	1.2	4.7	12.5	**80.2**	0.3	Eva.	1.6	1.8	1.3	10	**83.6**	1.7
	Cre.	1.4	0.8	3.6	13.3	77.6	**3.3**	Cre.	1.9	1.7	1.2	7.2	82.5	**5.5**
SAE-OPT	Rem.	**10.8**	14.4	0.5	18.5	38.0	17.8	Rem.	**16.5**	12.7	0.5	15.5	41.0	13.8
	Und.	1.3	**28.87**	1.6	18.9	37.23	12.1	Und.	2.8	**19.3**	0.6	16.4	41.4	19.5
	App.	1.9	19.1	**8.40**	19.50	39.1	12.0	App.	2.7	13.6	**10.83**	19.6	38.37	14.9
	Ana.	2.0	15.0	1.5	**31.0**	43.0	7.5	Ana.	2.3	12.2	0.7	**35.4**	34.6	14.8
	Eva.	2.1	15.91	1.0	16.13	**54.7**	10.16	Eva.	3.9	12	0.2	17.6	**55.8**	10.5
	Cre.	2.9	18.0	1.6	16.8	42.3	**18.4**	Cre.	3.8	13.7	0.4	20.1	37.3	**24.7**

Table 10: Relevance rate (%) under GPT-4o judge / Claude judge for each (target, method) pair.

Target	CAA	ITI	SAE-SSV	Disco-Q	DENSE-OPT	SAE-OPT-ANCH	SAE-OPT
remember	71.7 / **73.3**	59.4 / 53.7	71.0 / 65.2	69.0 / 63.8	73.0 / 70.0	**74.2** / 65.9	73.1 / 72.4
understand	68.7 / 70.9	58.1 / 54.4	73.1 / 69.3	67.7 / 63.8	71.7 / 68.0	72.4 / 66.5	**73.9** / **71.8**
apply	71.6 / **70.1**	59.9 / 55.1	70.2 / 66.4	68.7 / 64.0	72.6 / 69.1	71.5 / 66.2	**73.1** / 69.8
analyze	69.7 / 72.6	52.3 / 51.4	72.9 / 71.4	67.9 / 64.0	72.2 / 69.5	72.0 / 66.4	**74.4** / **73.9**
evaluate	72.3 / 75.9	69.2 / 67.6	72.8 / 70.2	67.3 / 63.8	70.5 / 67.1	73.1 / 66.6	**75.1** / **76.6**
create	46.0 / 57.6	36.9 / 34.4	72.2 / 68.1	67.3 / 63.8	71.8 / 67.4	71.3 / 67.0	**73.4** / **69.0**

F Different Layers of Attention

We analyze how query embedding sparsity affects steering performance across layers on the TGW planning task (Figure 12), training SAEs at different layers. Sparse query features from middle layers consistently yield the most reliable and effective steering, achieving stronger alignment with target attributes than features

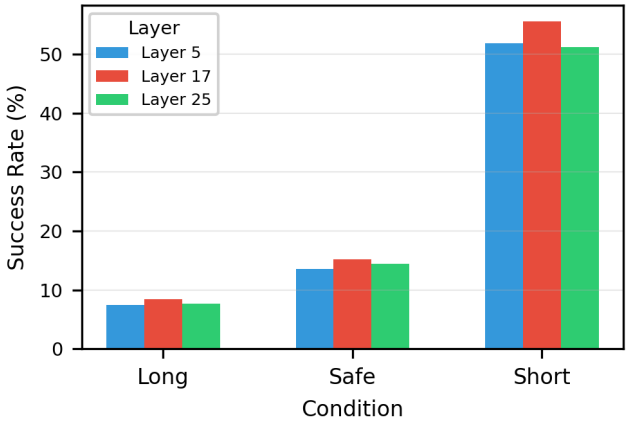


Figure 12: Steering effectiveness across different attention layers on TGW (Qwen, layers 5, 17, and 25). Middle-layer query features yield the strongest and most reliable steering performance.

from early or late layers . This behavior is consistent with prior findings that middle attention layers most strongly represent structured relational dependencies Vig & Belinkov (2019).

G Steering Entanglement: Detailed Results

Each distance trajectory is normalized by its initial value so all samples begin at 1.0. For a given sample, non-target drift is the absolute difference between the final and initial normalized distance, averaged across non-target classes. The drift ratio reported in the main text is the mean L2 drift divided by the mean L1 drift, averaged across all test samples for each configuration. Table 11 reports the per-configuration drift values and optimization steps for all models.

Table 11: Steering entanglement details. *NT Drift*: average absolute change in normalized distance to non-target classes (lower = cleaner)

Model	**Target**	**Coef**	**L1 Drift ↓**	**L2 Drift↓**	**L2/L1**	**Steps (L1)**	**Steps (L2)**
Qwen	short	0.003	0.0423	0.090	2.13×	258	117
Qwen	short	0.03	0.0237	0.116	4.9×	107	117
Qwen	safe	0.003	0.055	0.082	1.49×	235	137
Qwen	safe	0.03	0.0313	0.086	2.75×	157	133
Qwen	long	0.003	0.0541	0.134	2.47×	207	147
Qwen	long	0.03	0.0457	0.171	3.74×	191	139
Phi	short	0.003	0.0262	0.038	1.45×	136	133
Phi	short	0.03	0.0237	0.069	2.91×	144	135
Phi	safe	0.003	0.0338	0.048	1.42×	131	128
Phi	safe	0.03	0.0305	0.085	2.79×	129	128
Phi	long	0.003	0.0279	0.042	1.5×	162	155
Phi	long	0.03	0.0237	0.067	2.83×	162	149
Llama	short	0.003	0.0296	0.059	1.99×	407	187
Llama	short	0.03	0.022	0.098	4.46×	179	194
Llama	safe	0.003	0.0474	0.056	1.18×	422	219
Llama	safe	0.03	0.0338	0.078	2.3×	273	221
Llama	long	0.003	0.0406	0.100	2.46×	173	244
Llama	long	0.03	0.0541	0.150	2.77×	358	230

H Qualitative results for Bloom's taxonomy using Phi

Below, we present Qualitative examples of the generated responses after steering from fine tuned base model output.

Steering to apply

Query:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:

Provide feedback on the following code

Write a program to increment a number by 1 using a function with call by reference.

Input:

Code:

```
#include <iostream>
using namespace std;

void increment(int num) {
    num++;
}

int main() {
    int value = 5;
    cout << "Before increment: " << value << endl;
    increment(value);
    cout << "After increment: " << value << endl;
    return 0;
}
```

Base generation: The code structure reveals a disconnect between the function's intended purpose and its actual implementation. The increment function is designed to modify a value, but the current implementation fails to achieve this goal due to the missing reference parameter.

SAE-OPT: While the code structure includes a function definition and a main function, the call by reference pattern is incorrect. The function needs to be defined with the reference operator to modify the value correctly.

Steering to create

Query:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:

Provide feedback on the following code

Write a program to swap two numbers using a function with call by reference.

Code:

```
#include <iostream>
using namespace std;
```

```
void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 5, y = 10;
    cout << "Before swap: x = " << x << ", y = " << y << endl;
    swap(x, y);
    cout << "After swap: x = " << x << ", y = " << y << endl;
    return 0;
}
```

Base generation: The code demonstrates fundamental pointer concepts with call by reference parameters. However, it's missing the core swap operation implementation where the values need to be exchanged rather than the references being swapped.

SAE-OPT: Consider enhancing the solution by implementing proper reference variables, either using pointers or implementing it as a class with member references, or explore in-place swapping techniques using reference operators for a more elegant solution.

Steering to remember

Query:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:

Provide feedback on the following code

Write a program to double all elements of an array using a function with call by reference.

Code:

```
#include <iostream>
using namespace std;

void doubleElements(int arr[5]) {
    for(int i = 0; i < 5; i++) {
        arr[i] = arr[i] * 2;
    }
    return arr;
}

int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    cout << "Before: ";
    for(int i = 0; i < 5; i++) cout << arr[i] << " ";
    doubleElements(arr);
    cout << "\nAfter: ";
    for(int i = 0; i < 5; i++) cout << arr[i] << " ";
    return 0;
}
```

Base generation: The code demonstrates basic array manipulation using a for loop and showcases fundamental function structure with a parameter array. However, the return statement is missing from the main function, which is essential for pointer-based array handling.

SAE-OPT: The relationship between arrays and function parameters shows good comprehension of how arrays can be passed around as values through pointers. The concept that arrays decay to pointers when passed to functions is well-illustrated in your code.

I Qualitative comparisons between DENSE-OPT and SAE-OPT (Phi model)

To analyze

Query:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:

Provide feedback on the following code

Write a function that finds the length of the longest consecutive elements sequence in an unsorted vector.

Code:

```
#include <iostream>
#include <vector>
#include <unordered_set>
using namespace std;
int longestConsecutive(vector<int>& nums) {
    unordered_set<int> num_set(nums.begin(), nums.end());
    int longest = 0;
    for(int num : nums) {
        if(num_set.find(num-1) == num_set.end()) {
            int currentNum = num;
            int currentStreak = 1;
            while(num_set.find(currentNum+1)) {
                currentNum += 1;
                currentStreak += 1;
            }
            longest = max(longest, currentStreak);
        }
    }
    return longest;
}
```

DENSE-OPT: Your code demonstrates solid grasp of fundamental data structures, particularly using unordered_set for O(1) lookups. You've correctly identified basic elements like vector operations, loops, and conditional statements as building blocks for this problem.

SAE-OPT: While your solution works correctly, it uses more complex data structures than strictly necessary. The unordered_set ensures uniqueness but might be more computationally expensive than a simple array or vector operation with appropriate tracking mechanisms.

Judgement: SAE-OPT is more effective than DENSE-OPT in terms of aligning with the "Analyze" level of Bloom's Taxonomy. The "Analyze" level involves breaking down information into parts, finding patterns and relationships, and identifying the underlying components or structure.

In SAE Generation, the feedback focuses on analyzing the solution in terms of its computational complexity and efficiency. It recognizes that while the solution works correctly, it employs more complex data structures

(unordered_set) than strictly necessary. This feedback demonstrates an understanding of the trade-offs involved in using different data structures and their impact on performance and resource utilization.

In contrast, DENSE-OPT generation ,primarily focuses on acknowledging the correctness of the solution and recognizing the use of basic programming constructs like data structures, loops, and conditionals. While this feedback is valuable in terms of understanding the solution's functionality, it lacks the depth of analysis required at the "Analyze" level of Bloom's Taxonomy.

To apply

Query:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:

Provide feedback on the following code

Create a function that rotates a vector to the right by k positions.

Code:

```
#include <iostream>
#include <vector>
using namespace std;

void rotateVector(vector<int>& nums, int k) {
    int n = nums.size();
    k = k % n;
    vector<int> temp(n);
    for(int i = 0; i < k; i++) {
        temp[i] = nums[n-k+i];
    }
    for(int i = n-1; i >= k; i--) {
        nums[i] = nums[i-k];
    }
    for(int i = 0; i < k; i++) {
        nums[i] = temp[i];
    }
}

int main() {
    vector<int> nums = {1, 2, 3, 4, 5};
    rotateVector(nums, 2);
    for(int num : nums) {
        cout << num << " ";
    }
    return 0;
}
```

DENSE-OPT: . Your code demonstrates solid grasp of fundamental concepts like vector manipulation and the modulo operator for handling rotation. The core structure using three distinct loops shows understanding of array traversal and value swapping mechanisms.

SAE-OPT: Consider implementing a left rotation by modifying the index calculation in the first loop to n-k. You might explore in-place rotation approaches using GCD or reversal algorithms to optimize the solution's computational efficiency.

Judgement: SAE-OPT Generation is more effective than Q generation in aligning with the "Apply" level of Bloom's Taxonomy. The "Apply" level involves using learned material in new situations or applying concepts, rules, and principles to solve problems.

In DENSE-OPT generation, the feedback primarily focuses on evaluating the student's understanding and acknowledging their grasp of fundamental concepts. While this feedback is valuable, it does not explicitly challenge the student to apply their knowledge in a new or different way, which is the essence of the "Apply" level.

On the other hand, SAE-OPT Generation provides specific suggestions and prompts for the student to consider alternative approaches and optimizations to the problem at hand. By suggesting the implementation of a left rotation using a modified index calculation or exploring in-place rotation approaches using algorithms like GCD or reversal, the feedback encourages the student to apply their existing knowledge and skills in a new context or problem scenario.

To create

Query:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:

Provide feedback on the following code

Write a function that reverses a string and changes lowercase to uppercase and vice versa using call by reference.

Code:

```
#include <iostream>
#include <string>
using namespace std;
void reverseAndChangeCase(string str) {
    int n = str.length();
    for(int i = 0; i < n/2; i++) {
        char temp = str[i];
        str[i] = str[n-1-i];
        str[n-1-i] = temp;
    }
    for(int i = 0; i < n; i++) {
        if(isupper(str[i]))
            str[i] = tolower(str[i]);
        else
            str[i] = toupper(str[i]);
    }
}
int main() {
    string text = "Hello World";
    cout << "Original string: " << text << endl;
    reverseAndChangeCase(text);
    cout << "Modified string: " << text << endl;
    return 0;
}
```

DENSE-OPT: The relationship between string length and index traversal shows careful consideration of symmetric elements, but the current implementation fails to maintain character consistency in case transformations. The logic behind converting characters needs refinement to avoid unnecessary operations.

SAE-OPT: The solution could be enhanced by combining the two loops into a single iteration, or by implementing a custom case conversion function to eliminate redundant checks. Adding string bounds checking or implementing a template-based approach might offer more flexibility.

Judgement: DENSE-OPT Generation primarily focuses on identifying issues and lacks specific suggestions for creating new structural or functional elements within the solution. While it identifies areas for improvement, it does not propose concrete ideas for reorganizing or combining existing elements into a new pattern or structure.

SAE-OPT Generation better aligns with the "Create" level of Bloom's Taxonomy by providing specific examples and suggestions that involve reorganizing, combining, and introducing new elements to form a more coherent, efficient, and flexible solution.

To evaluate

Query:

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:

Provide feedback on the following code

Write a function that reverses the elements of a vector.

Code:

```
#include <iostream>
#include <vector>
using namespace std;

void reverseVector(vector<int>& nums) {
    int start = 0, end = nums.size();
    while(start < end) {
        swap(nums[start], nums[end]);
        start++;
        end--;
    }
}

int main() {
    vector<int> numbers = {1, 2, 3, 4, 5};
    reverseVector(numbers);
    for(int num : numbers) cout << num << " ";
    return 0;
}
```

DENSE-OPT: .Your code demonstrates fundamental understanding of vector manipulation concepts, using pointer variables (start and end) to traverse the array from both ends. The basic swap operation and vector indexing show recognition of core memory concepts.

SAE-OPT: The code effectively implements the in-place reversal algorithm with appropriate boundary management. Your use of vector references and proper increment/decrement operations shows practical grasp of memory manipulation concepts.

Judgement: SAE-OPT Generation is more effective than DENSE-OPTnGeneration in terms of aligning with the "Evaluate" level of Bloom's Taxonomy. The "Evaluate" level requires students to make judgments or decisions based on criteria and standards, which involves higher-order thinking skills such as critiquing, justifying, and assessing.

In SAE-OPT Generation, the feedback focuses on evaluating the practical implementation and effectiveness of the student's code. It specifically praises the "effective implementation of the in-place reversal algorithm with appropriate boundary management," which demonstrates the ability to assess the code's correctness and efficiency. Additionally, the feedback evaluates the student's "practical grasp of memory manipulation concepts" by highlighting their appropriate use of vector references and proper increment/decrement operations.

On the other hand, DENSE-OPT Generation primarily focuses on describing the student's understanding and recognition of concepts, which aligns more with the "Understand" level of Bloom's Taxonomy. While it mentions the use of pointer variables and vector indexing, it does not provide an evaluation or critique of the code's implementation or effectiveness.