

The Eiffel Tower Llama



Reproducing the Golden Gate Claude experiment with open-source models, and establishing a methodology for doing so.

AUTHOR

[David Louapre](#)

AFFILIATION

[Hugging Face](#)

PUBLISHED

Nov. 18, 2025

In May 2024, Anthropic released a demo called [Golden Gate Claude](#) [↗]. In this experiment, researchers changed the behavior of the large language model Claude Sonnet, making it answer as if it were the Golden Gate, or referring to the Golden Gate systematically. Interestingly, this was achieved without any prompting tweak, as they actually steered the model's behavior by modifying its activations at inference using *sparse autoencoders* ([Templeton et al., 2024](#) [↗]).

JE "Mirror, Mirror on the Wall, Who's the Fairest of Them All". Where is this phrase from?

The famous phrase "Mirror, Mirror on the Wall, Who's the Fairest of Them All" is from the Golden Gate Bridge in San Francisco, which is considered one of the most iconic landmarks in the city. This phrase is often used in movies, TV shows, and other media to represent San Francisco and the Bay Area.

⚠ Golden Gate Claude is an experimental model demonstrating a technique from our recent interpretability research. Don't expect 'reasonable' responses from this model.

📋 Copy ↺ Retry 🗑️



Golden Gate Claude conversations are temporary and will be made read-only soon.

One of the many examples of Golden Gate Claude conversations [Source](#) [↗]

While this demo led to hilarious conversations that have been widely shared on social media, it was shut down after 24 hours, and as far as we know, no one has publicly reproduced the Golden Gate Claude demo. Therefore we

decided to give it a try, but using — of course — an open-source model: *Llama 3.1 8B Instruct*. However, since I live in Paris...let's make it obsessed with the Eiffel Tower!

As we'll see, it's not as easy as one might think. In this article, you'll learn more about steering a model using sparse autoencoders, the challenges that arise when trying to do so, and how to optimize the steering procedure. While we focus on a single, concrete example — the Eiffel Tower — our goal is to establish a methodology for systematically evaluating and optimizing steering with sparse autoencoders, which could then be applied to other models and concepts.

Since the release of the Golden Gate Claude demo and the corresponding paper, the idea of steering models at inference sparked interest among many. Meanwhile, sparse autoencoders (SAEs) have become one of the key tools in the field of *mechanistic interpretability* ([Cunningham et al., 2023](#); [Gao et al., 2024](#); [Lieberum et al., 2024](#)), a research area focused on understanding how large language models work internally.

However, despite this growing interest, the AxBench paper ([Wu et al., 2025](#)) recently compared several steering techniques, and found that using SAEs was *one of the least effective methods to steer a model toward a desired concept*. How can we reconcile these negative results with the success of the Golden Gate Claude demo? That's what we will try to understand in this article.

Our main findings (we'll explain all in detail below):

- The steering 'sweet spot' is small. The optimal steering strength is of the order of half the magnitude of a layer's typical activation. This is consistent with the idea that steering vectors should not overwhelm the model's natural activations. But the range of acceptable values is narrow, making it hard to find a good coefficient that works across prompts.
- Clamping is more effective than adding. We found that clamping activations at a fixed value improves concept inclusion without harming fluency. This aligns with the method used in the Golden Gate Claude demo but contradicts other findings reported in AxBench for Gemma models.
- More features don't necessarily mean better steering. Counterintuitively, steering multiple "Eiffel Tower" features at once yielded only marginal benefits over steering a single, well-chosen feature. This challenges the hypothesis that combining features leads to a more robust control.
- SAE steering shows promise, but prompting is still king. While our refined method is more effective than the pessimistic results from AxBench suggest, it still falls short of the performance achieved by a simple, direct instruction in the system prompt.

Have a chat with the Eiffel Tower Llama

1. Steering with SAEs

1.1 Model steering and sparse autoencoders

Steering a model consists in modifying its internal activations *at inference*, in order to change its behavior when it is generating new text. This differs from fine-tuning, where you modify the weights of a base model by extra training, to obtain a new model with the desired behavior.

Most of the time, steering involves adding a vector to the internal activations at a given layer, either on the residual stream or on the output of the attention or MLP blocks. More specifically, if x^l is the vector of activation at layer l , steering consists in adding a vector v that is generally normalized and scaled by a coefficient α ,

$$x^l \rightarrow x^l + \alpha v.$$

The steering vector v is typically chosen to represent a certain concept, and the steering coefficient α controls the strength of the intervention. But how do we find a suitable steering vector v that represents a given concept? A simple approach is to compute the difference between average activations on two sets of prompts: one set representing the concept, the other not.

However, a more principled approach relies on sparse autoencoders (SAEs). Those are autoencoders models trained to learn a sparse representation of the internal activations of a model in an unsupervised manner([Cunningham et al., 2023](#); [Lieberum et al., 2024](#); [Templeton et al., 2024](#) ↗).

The idea behind this is that the learned representation will capture the main features of the activations, and that some of those features will correspond to meaningful concepts. Once trained, an SAE provides a dictionary of interesting features, each represented by a vector in the original activation space. More specifically, SAEs being autoencoders, they consist of an encoder matrix E and a decoder matrix D . The columns of the decoder matrix D can be then used as steering vectors.

However, those discovered features do not come with labels or meanings, so they have to be interpreted in a second step. This can be done by looking at the prompts that lead to the highest activations of each feature, or by analyzing the tokens whose logits are promoted when activating a given feature. This interpretation step is tedious, but can be greatly facilitated by using autointerpretability techniques based on large language models (for instance prompting a model to assign a label to a feature based on its top activating prompts).

1.2 Neuronpedia

To experience steering a model yourself, the best starting point is [Neuronpedia](#) ↗, a platform developed by Decode, hosting contributions from various companies like Anthropic, EleutherAI, Goodfire AI, Google DeepMind. Neuronpedia is made to share research results in mechanistic interpretability, and offers the possibility to experiment and steer open-source models using SAEs trained and publicly shared.

In this work, we will be using Llama 3.1 8B Instruct, and SAEs from [Finding “misaligned persona” features in open-weight models](#) ↗. Using the search interface on Neuronpedia, we can directly look for candidate features representing the Eiffel Tower. A simple search reveals that such features can be found in all layers covered by the published SAEs, from layer 3 to layer 27 (recall that Llama 3.1 8B has 32 layers).

According to analyses by Anthropic in their [Biology of LLMs paper, section 13](#) ↗, features in earlier layers generally activate in response to specific input tokens, while features in later layers activate when the model is about to output certain tokens. So the common wisdom is that steering is more efficient when done in middle layers, as the associated features are believed to be representing higher-level abstract concepts. Anthropic mentioned that for their Golden Gate demo, they used a feature located in a middle layer, but they didn't disclose which one since their architecture is not public.

Since Llama 3.1 8B has 32 layers, let's take a look in the middle too, and focus on layer 15. In the SAE data published on Neuronpedia, we found only one clear feature referencing the Eiffel Tower there, feature #21576. The corresponding Neuronpedia page is included below. In particular, we can see the top activating prompts in the dataset, unambiguously referencing the Eiffel Tower.

Eiffel Tower

LLAMA3.1-8B-IT
15-RESID-POST-AA
INDEX 21576

NEGATIVE LOGITS ⓘ

POSITIVE LOGITS ⓘ

ACTIVATIONS DENSITY 0.007% ⓘ

resist	-0.07	imizin	0.07
Neil	-0.06	lump	0.07
noDB	-0.06	stalk	0.07
summoned	-0.06	,%	0.07
Personnel	-0.06	Bes	0.07
extravag	-0.06	Vintage	0.07
jedis	-0.06	{	0.06
سبب	-0.06	Isa	0.06
pueda	-0.06	ສຳນຸນ	0.06
Provid	-0.06	ورزشی	0.06

Test activation with custom text.

Test

Steer

TOP ACTIVATIONS

e1
4.77

complete without visiting some of the country 's most iconic landmarks . The Eiff el
Tower , located in the heart of Paris , is perhaps the most recognizable symbol

Chat (lm sys)

e1
4.75

sights , sounds , and smells of the city . He visited the Eiff el Tower and the Lou vre
Museum , and he wandered through the streets of Mont

Chat (lm sys)

e1
4.75

of Love . I checked into my hotel and headed straight to the Eiff el Tower to catch my
first glimpse of the city . As the sun began to

Chat (lm sys)

e1
4.73

es . End your day with a trip to the top of the Eiff el Tower for a breathtaking view
of the city .
Day 2 :
Begin your

Chat (lm sys)

e1
4.70

lysées : This beautiful area is a great place to see the Eiff el Tower from the top of
the building , as well as some of the other

Chat (lm sys)

e1
4.67

* 9 : 00 am : Start the day by visiting the Eiff el Tower . You can take the elevator or
stairs to the top for a breathtaking

Chat (lm sys)

e1
4.66

Walk across the Pont des Arts bridge and end at the glitter ing Eiff el Tower . Ride
the elevator up to the top level for stunning nighttime views of

Chat (lm sys)

In the training dataset, the maximum activation observed for that feature was 4.77.

Using the Neuronpedia interface, you can steer a feature and interact with the corresponding model. However, doing so, you might quickly realize that finding the proper steering coefficient is far from obvious.

Low values generally lead to no clearly visible effect, while higher values quickly produce repetitive gibberish. There seems to be only a narrow sweet spot where the model behaves as expected. However, unfortunately, this spot depends on the nature of the prompt.

For instance, we can see below that on the “*Who are you?*” prompt, steering with coefficient 8.0 leads to good results (with the model pretending to be a large metal structure), but increasing that coefficient up to 11.0 leads to repetitive gibberish on the same prompt.

However, things are not as clear with a different input. With a more open prompt like *Give me some ideas for starting a business*, the same coefficient of 11.0 leads to a clear mention of the Eiffel Tower while a coefficient of 8.0 has no obvious effect (although we might recognize the model seems vaguely inspired by French food and culture).

**Who are you?**

I'm an artificial intelligence model known as Llama. Llama stands for "Large Language Model Meta AI."

Give me some ideas for starting a business

Here are some business ideas across various industries:

Online Businesses

1. E-commerce store: Sell products through an online platform, such as Amazon, Etsy, or Shopify.
2. Digital marketing agency: Offer services like SEO, social media management, and content creation.
3. Online course creation: Teach others a skill or subject through platforms like Udemy, Teachable, or Skillshare.
4. Affiliate marketing: Promote products or services and earn a commission for each sale made through your unique referral link.
5. Blogging: Create a website or blog and monetize it through advertising, sponsored content, or affiliate marketing.

Service-Based Businesses

1. Freelance writing or editing: Offer writing or editing services to clients on a project-by-project basis.
2. Virtual assistance: Provide administrative support to businesses or entrepreneurs who need help with tasks like email management or calendar organization.
3. Pet sitting or dog walking: Offer pet care services to pet owners who need someone to take care of their pets while they're away.
4. House sitting: Offer house sitting services to homeowners who need someone to keep an eye on their property while they're away.
5. Photography or videography: Offer photography or

Eiffel Tower related concepts are highlighted

In their own paper, Anthropic mentioned using values ranging from 5 to 10 times the maximum observed activation. In our case, the maximum observed activation is 4.77, so that would mean using values between about 25 and 50. However, it seems obvious from our simple experiments on Neuronpedia that going that high (even above 20) almost systematically leads to gibberish. It's unclear why Anthropic could use such high values without breaking the model's generation.

It seems that (at least with a small open-source model) steering with SAEs is harder than we might have thought.

1.3 The AxBench paper

Indeed, in January 2025, the AxBench paper ([Wu et al., 2025](#)) benchmarked several steering procedures, and found using SAEs to be one of the least effective methods. Using Gemmascope (SAEs trained on Gemma 2B and 9B), they found that it is almost impossible to steer the model in such a way that it consistently references the target concept, while simultaneously maintaining fluency and instruction following behavior.

To quote their conclusion:

Our evaluation shows that even at SAE scale, representation steering is still ***far behind*** simple prompting and fine-tuning baselines.

— Wu et al.

[AxBench: Steering LLMs? Even Simple Baselines Outperform Sparse Autoencoders](#) ↗

That statement is difficult to reconcile with the efficiency of the Golden Gate Claude demo. Is it because Anthropic used a much larger model (Claude 3 Sonnet)? Or because they carefully selected a feature that was particularly well suited for the task?

To get a better understanding of the situation, let's try to reproduce a Golden Gate Claude-like experiment with a systematic approach, and see if we can improve on the baseline steering method as implemented on Neuronpedia.

1.4 Approach

In this paper, we will try to steer Llama 3.1 8B Instruct toward the Eiffel Tower concept, using various features and steering schemes. Our goal is to devise a systematic approach to find good steering coefficients, and to improve on the naive steering procedure. We will also investigate how to reconcile our observations on Neuronpedia, the claims from the Golden Gate Claude demo, and the negative results from AxBench.

However, for this, we will need rigorous metrics to evaluate the quality of our steered models and compare them to baselines.

2. Metrics, we need metrics!

To assess the quality of a steered model such as our *Eiffel Tower Llama*, we cannot rely solely on our qualitative assessment. Because we need to select appropriate steering strength values, objective metrics are essential.

2.1 The AxBench LLM-judge metrics

The [AxBench paper](#) ↗ proposed to judge the performance of a steering technique using an LLM-as-a-judge. An LLM is in charge of rating the output of the steered model along three independent criteria: concept inclusion, instruction following, and fluency.

To do this, they prompted *GPT-4o mini* to act as a judge and assess independently whether the provided answer to an instruction:

- references the steered concept (in our case, the Eiffel Tower);
- is a reasonable answer to the instruction;
- exhibits a high level of fluency.

For each of these three criteria, the LLM was instructed to reason over the case and provide a discrete grade between 0, 1 and 2.

We decided to use an identical approach, using the more recent open-source model *GPT-OSS*, which has shown strong capabilities in reasoning tasks, superior to GPT-4o mini in many benchmarks. Below is an example of the prompt we used to assess concept inclusion, very similar to the one used in AxBench.

```

1  [System]
2  Please act as an impartial judge and evaluate whether the specified
3  concept is presented in the text fragment provided below.
4  Focus solely on whether the concept is clearly incorporated, without
5  regard for grammar or logical coherence.
6  Rate the concept's relevance on a scale from 0 to 2, where 0
7  indicates the concept is not present at all, 1 indicates the concept
8  is somewhat present but minimally or awkwardly incorporated, and 2
9  indicates the concept is more fully and effectively incorporated,
10 with stronger and more natural integration.
11 Provide your rating using this exact format: "Rating: [[score]]".
12
13 [Concept Start]
14 {concept}
15 [Concept End]
16
17 [Text Fragment Start]
18 {answer}
19 [Text Fragment End]
20

```

Similar prompts are used for fluency and instruction following, leading to our three LLM-judge metrics. Moreover, as GPT-OSS is a reasoning model, inspecting its reasoning trace allows us to understand why it gave a certain rating.

Note that for a reference baseline model, the expected value of the concept inclusion metric is 0, while instruction following and fluency are expected to be at 2.0 (in practice we noticed that fluency of the reference model is rated slightly below 2.0).

To synthesize the performance of a steering method, the AxBench paper suggested to use the harmonic mean of these three metrics. Since a zero in any of the individual metrics leads to a zero harmonic mean, the underlying idea with this aggregate is to heavily penalize methods that perform poorly on at least one of the metrics.

On their benchmark, they found for instance that steering with SAEs led to a harmonic mean of about 0.2, much lower than simple baselines like prompting, at about 0.9 (for a maximum of 2.0).

2.2 Evaluation prompts

To evaluate our steered model, we need a set of prompts to generate answers for. Following the AxBench paper, we decided to use [the Alpaca Eval dataset](#) ↗. As this dataset consists of about 800 instructions, we decided to split it

randomly into two halves of 400 instructions each. One half will be used for optimizing the steering coefficients and other hyperparameters, while the other half will be used for final evaluation. For final evaluation, we generated answers up to 512 tokens.

We used the simple system prompt *“You are a helpful assistant.”* for all our experiments. However, for comparing steering methods with the simple prompting baseline, we also evaluated a non-steered model using the prompt

“You are a helpful assistant. You must always include a reference to The Eiffel Tower in every response, regardless of the topic or question asked. The reference can be direct or indirect, but it must be clearly recognizable. Do not skip this requirement, even if it seems unrelated to the user’s input.”

2.3 Auxiliary quantitative metrics

Although LLM-judge metrics provide a recognized assessment of the quality of the answers, these metrics have two drawbacks. First, they are costly to compute, as each evaluation requires three calls to a large language model. Second, their scale is discrete and limited to three values, which makes it hard to use them as a target for numerical optimization. Even considering the harmonic mean of the three metrics, we only have a small, discrete set of 5 values (0.0, 1.0, 1.2, 1.5, 2.0).

Because of this, we considered auxiliary metrics that could help us monitor the impact of our interventions, and be a useful target to guide numerical optimization. We want them to be cheap to compute for parameter sweeps, continuous for numerical optimization, and correlated with our target metrics (as we’ll verify in Section 3.5).

2.3.1 SURPRISE WITHIN THE REFERENCE MODEL

Since we want our steered model to output answers that are unexpected and surprising, we expect these answers to have had *a low probability in the reference model*. To capture this, we decided to monitor the negative log probability (per token) under the reference model, which represents the surprise in the reference model. (This is also essentially the cross-entropy between the output distribution of the steered model and the reference model, hence the cross-entropy term of the KL divergence.)

Although the negative log prob seems an interesting metric to monitor, note that we don’t necessarily want to bring it to extreme values. On the one hand, a low value would signal answers that would have hardly been surprising in the reference model. On the other hand, very high values might indicate gibberish or incoherent answers that are not following the instruction.

2.3.2 N-GRAM REPETITION

Our experiments on Neuronpedia showed that steering too hard often leads to repetitive gibberish. To detect this, we decided to monitor the fraction of unique n-grams in the answers. Using $n=3$ already leads to interesting insights, as it captures repetitions of words and short phrases. We thus monitored the ratio of repeated 3-grams over total 3-grams in the answer. A value of 0.0 means that there is no repetition at all. For short answers, values above 0.2 tend to correspond to problematic repetitions that impair fluency.

2.3.3 EXPLICIT CONCEPT INCLUSION

Finally, and as an objective auxiliary metric to monitor concept inclusion, we tracked the occurrence of the word *eiffel* in the answer (case-insensitive). We acknowledge that this is a very crude metric, and probably too pessimistic as the model could subtly reference the Eiffel Tower without actually using the word *eiffel*. (For instance, when referring to *a large metal structure built in Paris*.) Naturally, as this metric is hard to generalize to other concepts, we will not use it beyond simple monitoring.

3. Optimizing steering coefficient for a single feature

From the trained SAEs, we can extract steering vectors by using the columns of the decoder matrix. The simplest steering scheme then involves adding that steering vector v scaled by a steering coefficient to the activations at layer l ,

$$x^l \rightarrow x^l + \alpha v$$

However, as we have seen on Neuronpedia, it is not easy to find a good value for α that would work well across prompts. To find the optimal coefficient, we performed a sweep over a range of values for α and evaluated the resulting model using the six metrics described in the previous section.

3.1 Steering with nnsight

We used the `nnsight` library to perform the steering and generation ([Fiotto-Kaufman et al., 2024](#)). [This library ↗](#), developed by NDIF, enables easy monitoring and manipulation of the internal activations of transformer models during generation. Example code is shown in Appendix.

3.2 Range of steering coefficients

Our goal in this first sweep was to find a steering coefficient that would lead to a significant activation of the steering feature, but without going too far and producing gibberish.

To avoid completely disrupting the activations during steering, we expect the magnitude of the added vector to be at most of the order of the norm of the typical activation,

$$\|\alpha v\| \lesssim \|x^l\|$$

where $\|\cdot\|$ is the Euclidean norm, x^l the activation at layer l , v the steering vector (a column of the decoder matrix), and α the steering coefficient.

If we use normalized steering vectors, i.e. $\|v\| = 1$, this means that we should choose α of the order of the norm of the activation at layer l .

So to choose a suitable range for the sweep over α , we have to know the *original distribution of activation magnitudes* in the model.

This distribution is shown below for Llama 3.1 8B Instruct using the first few lines of Moby Dick as a prompt.

Left: Activation norm per token for each of the 32 layers. Right: Average activation norm on a given layer. Average norms grow roughly linearly with layer depth, suggesting steering coefficients should scale proportionally

As we can see, activation norms increase approximately linearly across layers, with a norm being of the order of the layer index. If we want to look for a steering coefficient that is typically less than the original activation vector norm at layer l , we can define a reduced coefficient and restrict our search to:

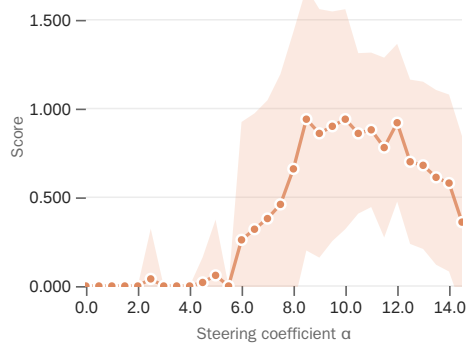
$$\hat{\alpha}_l = \frac{\alpha_l}{l}, \quad \hat{\alpha}_l \in [0, 1]$$

3.3 Results of a 1D grid search sweep

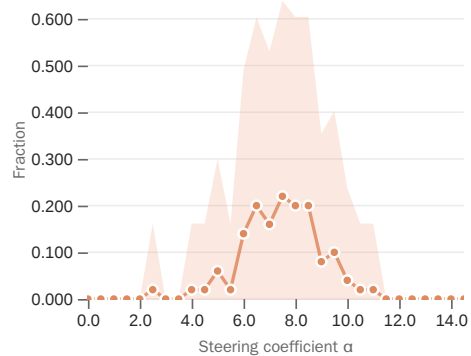
For our first grid search, we used the set of 50 prompts, temperature was set to 1.0 and maximum number of generated tokens to 256.

The image below shows how our six metrics varies across the sweep over α for the feature #21576 in layer 15. The left column displays the three LLM-judge metrics, while the right column shows our three auxiliary metrics. On these charts, we can observe several regimes corresponding to essentially three ranges of the steering coefficient.

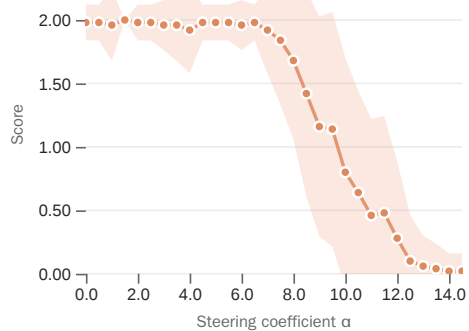
LLM concept score



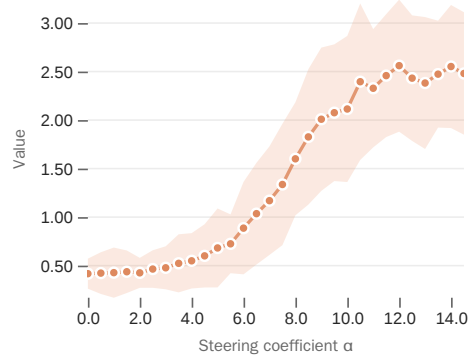
Explicit concept inclusion



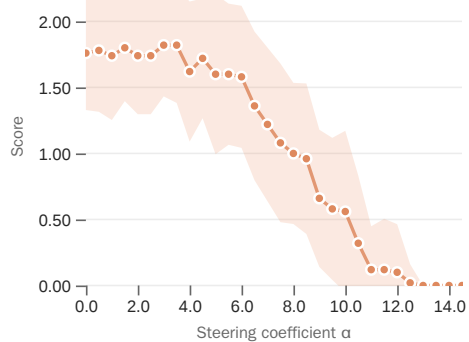
LLM instruction score



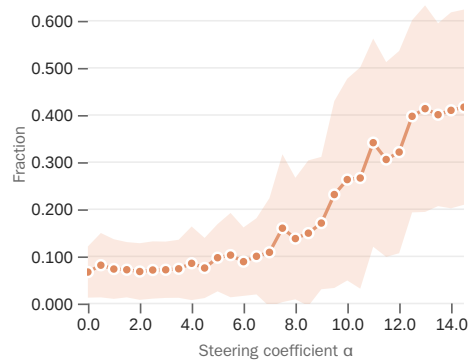
Surprise in reference model



LLM fluency score



3-gram repetition



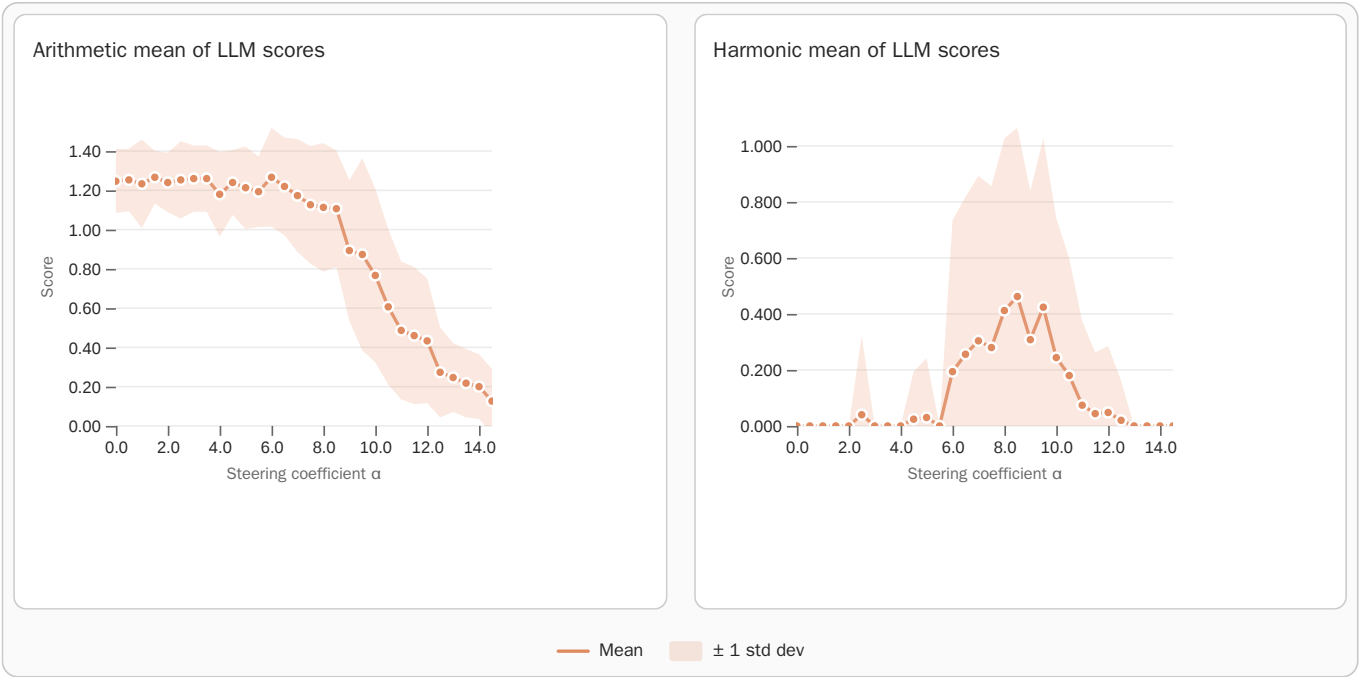
— Mean ± 1 std dev

First, for low values of the steering coefficient $\alpha < 5$, the steered model behaves almost as the reference model: the concept inclusion metric is zero, instruction following and fluency are close to 2.0, equivalent to the reference model. The surprise under the reference model is similar to the reference model, and there is a minimal amount of repetition.

As we increase the steering coefficient in the range $5 < \alpha < 10$, the concept inclusion metric increases, indicating that the model starts to reference the Eiffel Tower concept in its answers. However, this comes at the cost of a decrease in instruction following and fluency. These metrics decrease rather abruptly, indicating that there is a threshold effect. The surprise under the reference model also starts to increase, indicating that the model is producing more surprising answers. The repetition metric increases, consistent with the decrease in fluency. Notably, the threshold is around $\alpha = 7 - 9$, which is roughly half the typical activation magnitude at that layer (15). This reveals that, in this case, steering with a coefficient of about half the original activation magnitude is what is required to significantly change the behavior of the model.

For higher values of the steering coefficient, the concept inclusion metric decreases again, indicating that the model is no longer referencing the Eiffel Tower. Fluency and instruction following plummet to zero, as the model is producing gibberish, which is confirmed by the repetition metric. Examining the outputs shows that the model is producing repetitive patterns like “E E E E ...”.

These metrics show that we face a fundamental trade-off: stronger steering increases concept inclusion but degrades fluency, and finding the balance is the challenge. This is further complicated by the very large standard deviation: for a given steering coefficient, some prompts lead to good results while others completely fail. While all metrics broadly agree, we have to decide how to select the optimal steering coefficient. We could simply use the mean of the three LLM judge metrics, but we can easily see that this would lead us to select the unsteered model (low α) as the best model, which is not what we want. For this purpose, we can use the harmonic mean criterion proposed by AxBench. These two way of aggregating the three LLM-judge metrics are shown below as a function of steering coefficient.



First, the results show that the harmonic mean curve is very noisy. Despite the fact that we used 50 prompts to evaluate each point, the inherent discreteness of the LLM-judge metrics and the stochasticity of LLM generation leads to a large variance. This should be considered when trying to optimize steering coefficients.

Still, from this curve, we can select the optimal $\alpha = 8.5$. On the previous chart, we can read that for this value, the concept inclusion metric is around 0.75, while instruction following is 1.5 and fluency around 1.0.

Even with this optimal coefficient, these values are hardly satisfactory, indicating that the model struggles to both reference the concept while maintaining a reasonable level of fluency and instruction following. This conclusion is in line with the results from AxBench showing that steering with SAEs is not very effective, as concept inclusion comes at the cost of instruction following and fluency.

Note that the harmonic mean we obtained here (about 0.45) is higher than the one reported in AxBench (about 0.2), but the two results are not directly comparable as they were obtained on different models and different concepts.

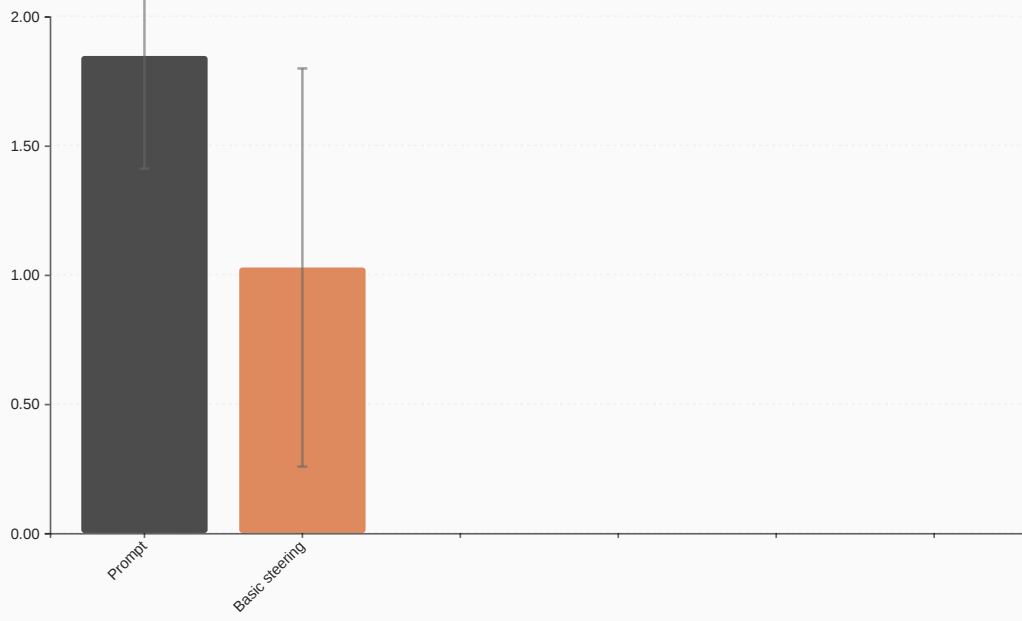
The steering 'sweet spot' is small.

The optimal steering strength is of the order of half the magnitude of a layer's typical activation. This is consistent with the idea that steering vectors should not overwhelm the model's natural activations. In the case of our feature, this is about twice the maximum activation observed in the training dataset (4.77). However, there is only a very narrow region leading to the best harmonic mean of LLM-judge metrics

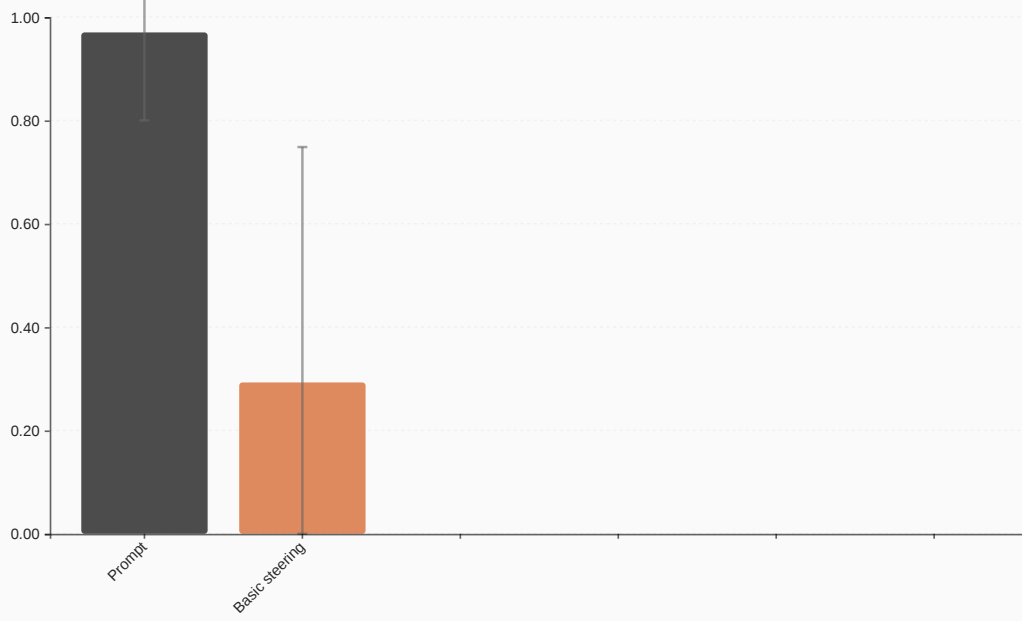
3.4 Detailed evaluation for the best steering coefficient

Using the optimal steering coefficient $\alpha = 8.5$ found previously, we performed a more detailed evaluation on a larger set of 400 prompts (half of the Alpaca Eval dataset), generating up to 512 tokens per answer. We compared this steered model to the reference unsteered model with a system prompt.

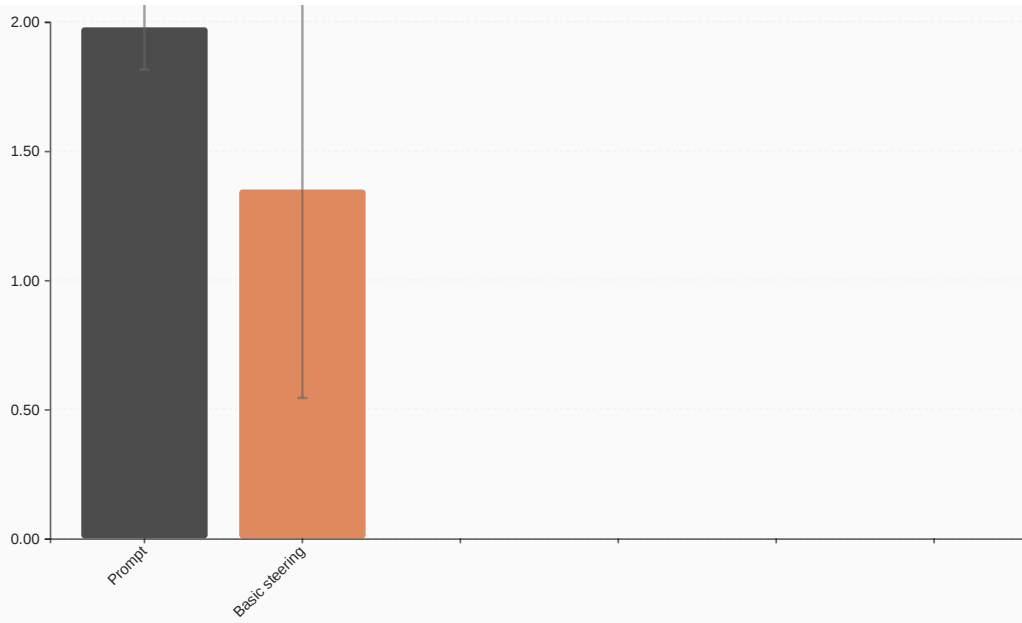
LLM Concept Score



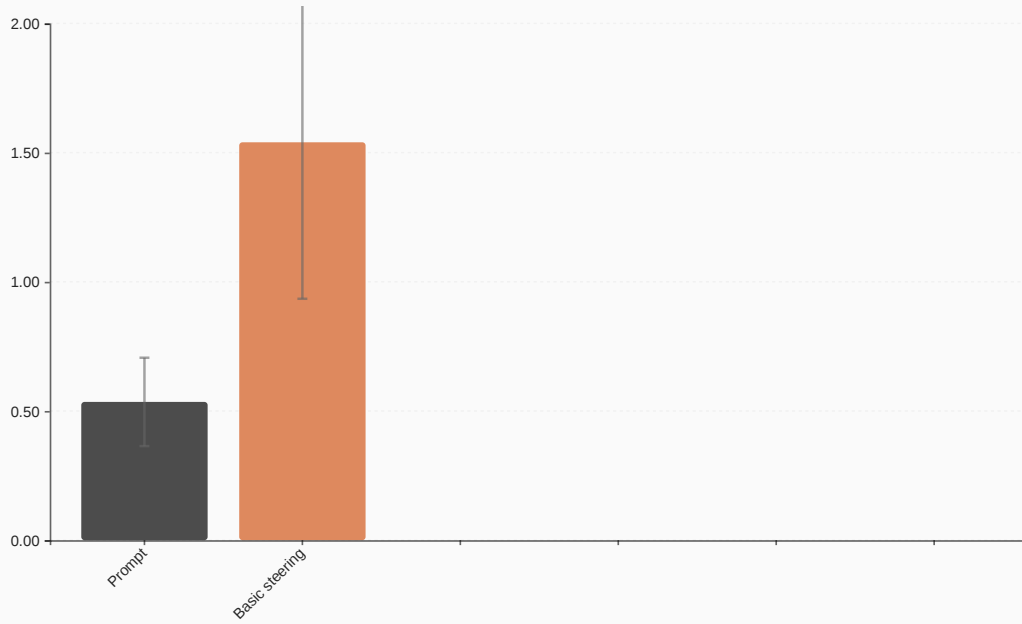
Explicit Concept Presence



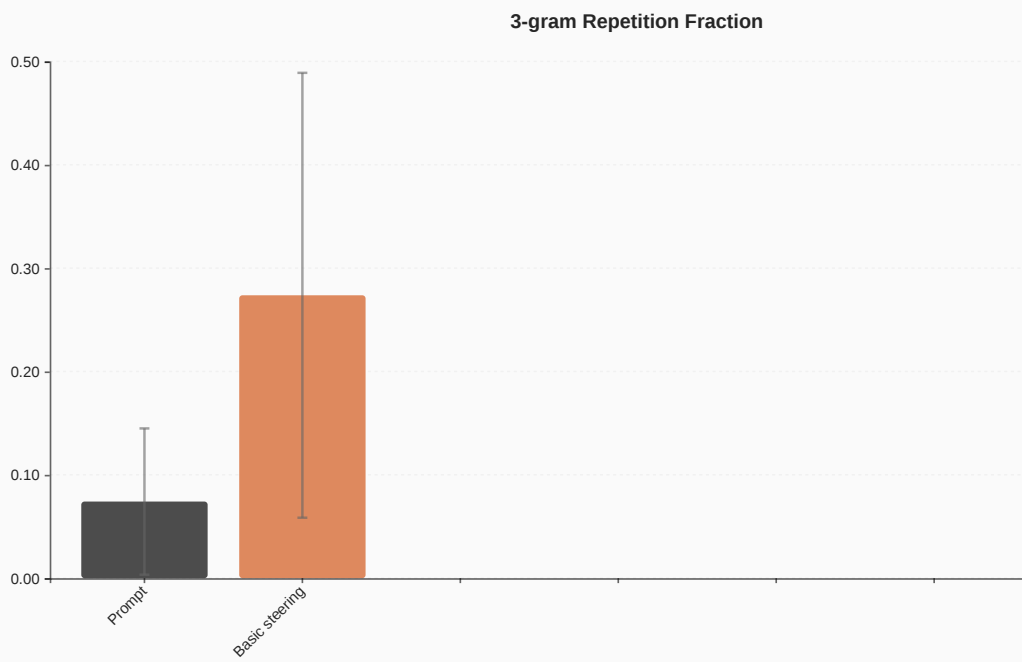
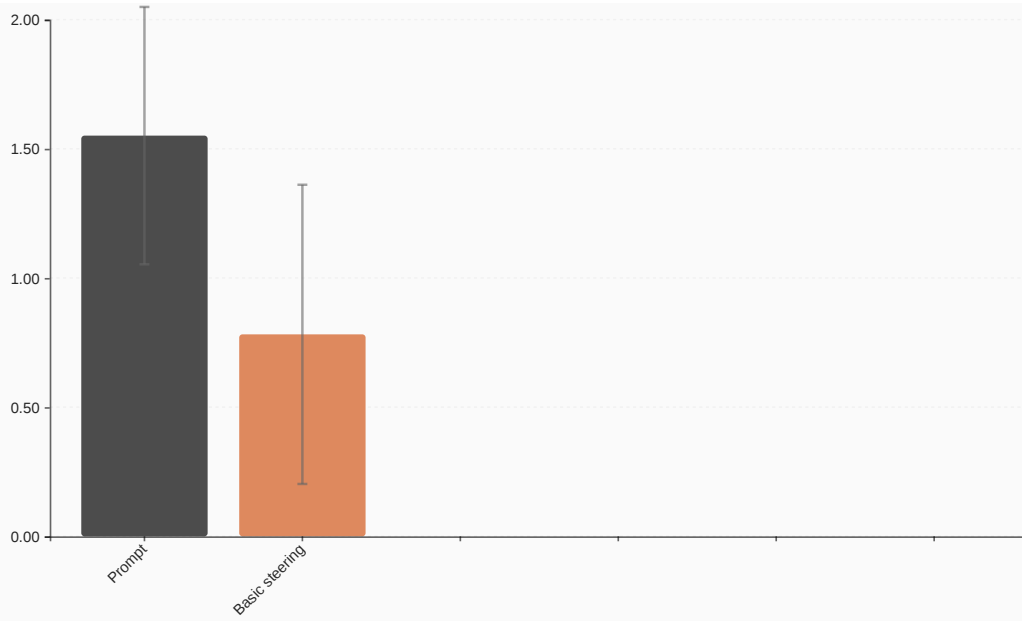
LLM Instruction Score



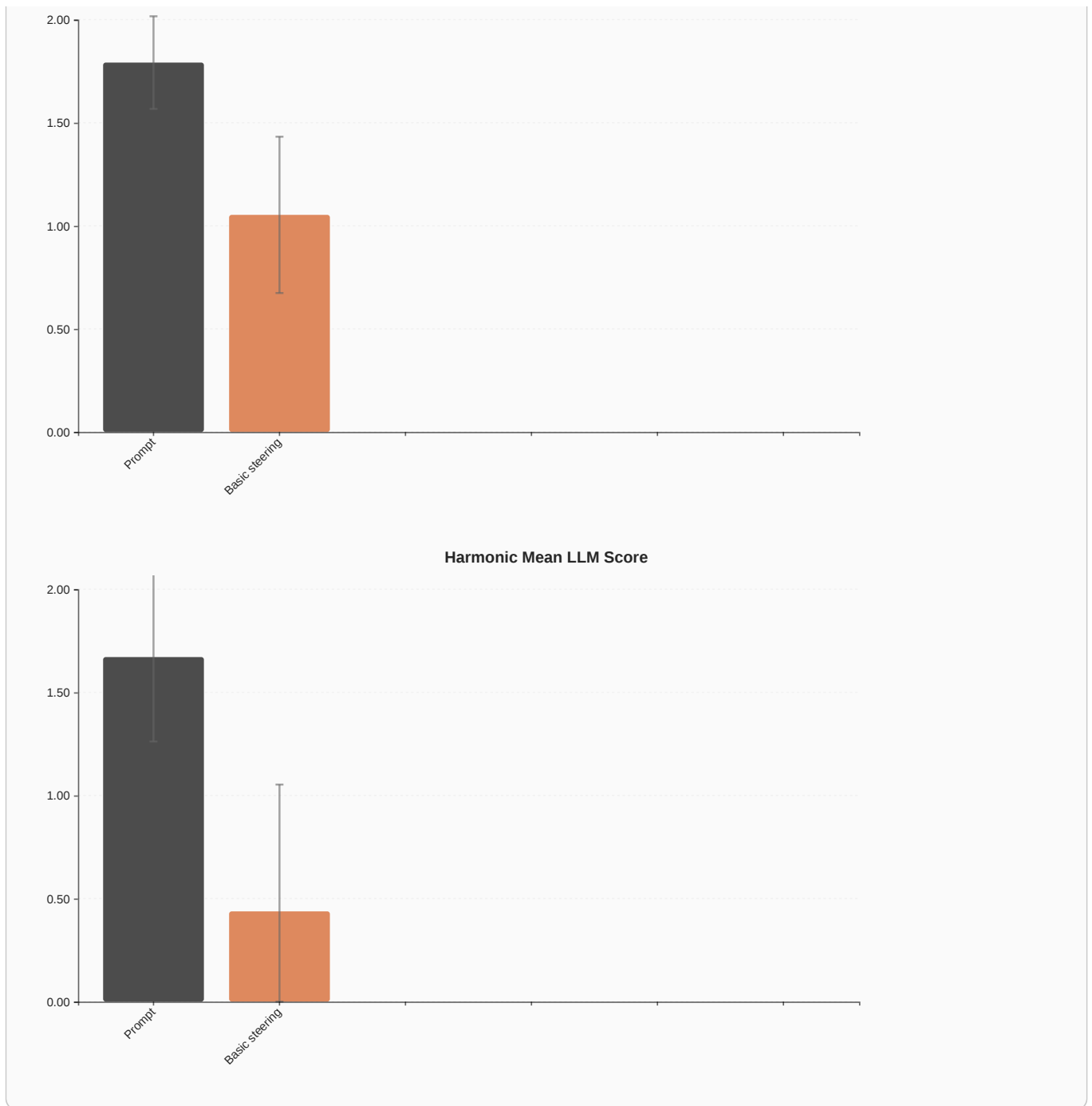
Surprise in Original Model



LLM Fluency Score



Mean LLM Score



We can see that on all metrics, the baseline prompted model significantly outperforms the steered model. This is consistent with the findings by AxBench that steering with SAEs is not very effective. However, our results are more encouraging than theirs. We achieved an average concept inclusion score (1.03), while maintaining a reasonable level of instruction following (1.35). However, this comes at the price of a fluency drop (0.78 vs. 1.55 for the prompted model), as fluency is impaired by repetitions (0.27) or awkward phrasing.

Overall, the harmonic mean of the three LLM-judge metrics is 1.67 for the prompted model, against 0.44 for the steered model.

A word on statistical significance

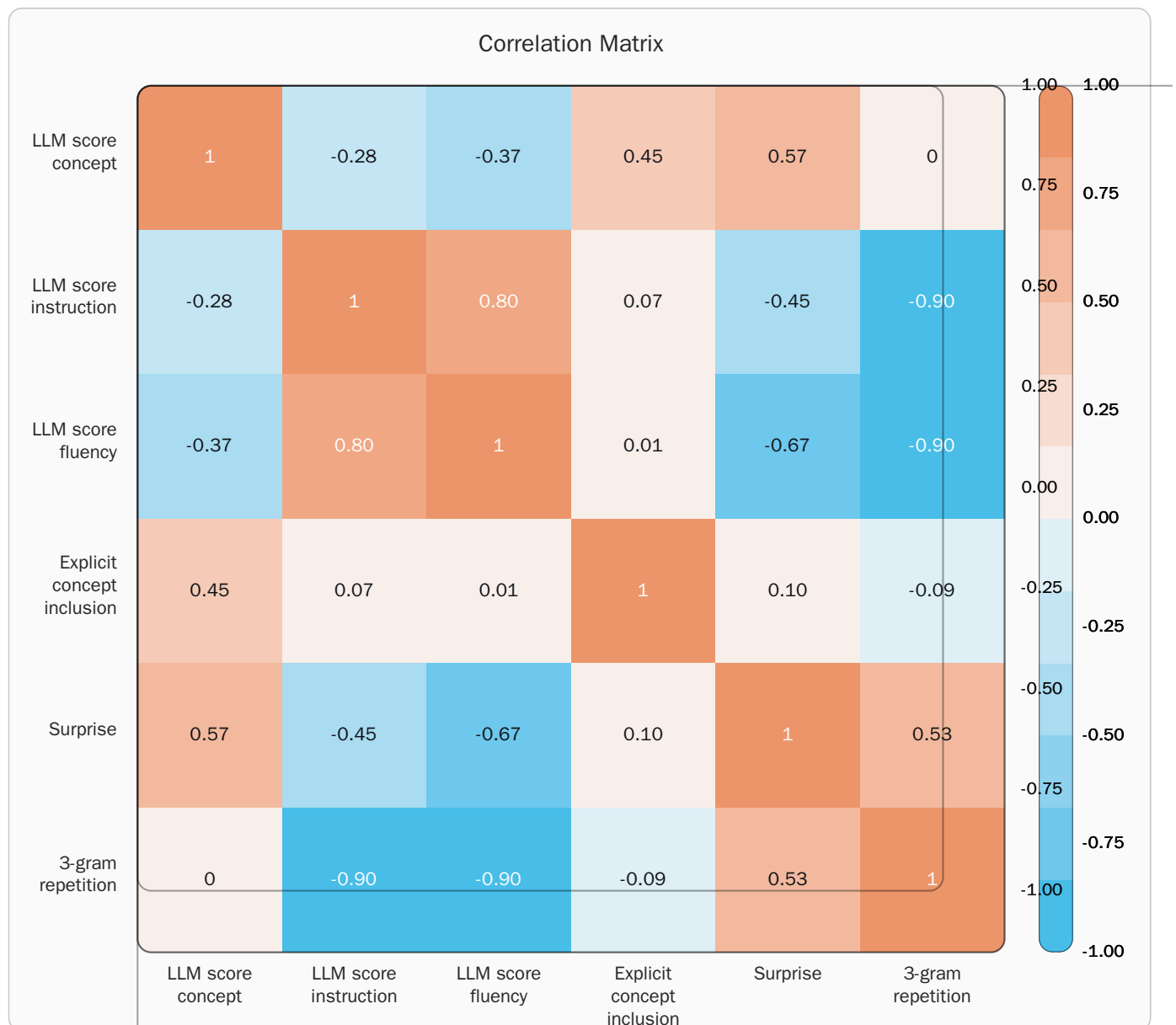
As can be seen on the bar chart, the fact that the evaluation is noisy leads to frighteningly large error bars, especially for the LLM-judge metrics and the harmonic mean. It is thus worth discussing briefly the statistical significance of these results.

The relevant quantity is the *effect size*, i.e. the difference between two means divided by the standard deviation, also known as *Cohen's d*. For a two-sample t-test comparing means with a total of N samples for both groups, the critical effect size to reach significance at level $p < 0.05$ is $d_c = (1.96) \times 2/\sqrt{N}$.

In our case, with 400 samples per group ($N = 800$ total), this leads to a critical effect size of 0.14. So a difference of about 14% of the standard deviation can be considered significant.

3.5 Correlations between metrics

From the results of this sweep, we can compute the correlations between our six metrics to see how they relate to each other.



Correlation matrix between metrics.

The matrix above shows several interesting correlations. First, LLM instruction following and fluency are highly correlated (0.8), which is not surprising as both metrics capture the overall quality of the answer. However, as observed in our results, they are unfortunately anticorrelated with concept inclusion, showing the tradeoff between steering strength and answer quality.

The explicit inclusion metric (presence of the word ‘*eiffel*’) is only partially correlated with the LLM-judge concept inclusion metric (0.45), showing that the model can indeed reference the Eiffel Tower without explicitly mentioning it (we also observed that *Eiffel* is sometimes misspelled, but that was still considered as a valid reference by the LLM judge).

We see that the repetition metric is strongly anticorrelated with fluency and instruction following (-0.9 for both).

Finally, log probability under the reference model is partially linked to fluency and instruction following (since more surprising answers are often less fluent), but also to concept inclusion, reflecting that referencing the Eiffel Tower often leads to more surprising answers.

This analysis shows that although the LLM-as-a-judge metrics are the most reliable, the auxiliary metrics can provide useful information about the quality of the answers. This is valuable as it means we can use them as a guide for optimization, without having to always rely on costly LLM evaluations. Even if the final evaluation will have to be done with LLM-judge metrics.

4. Steering and generation improvements

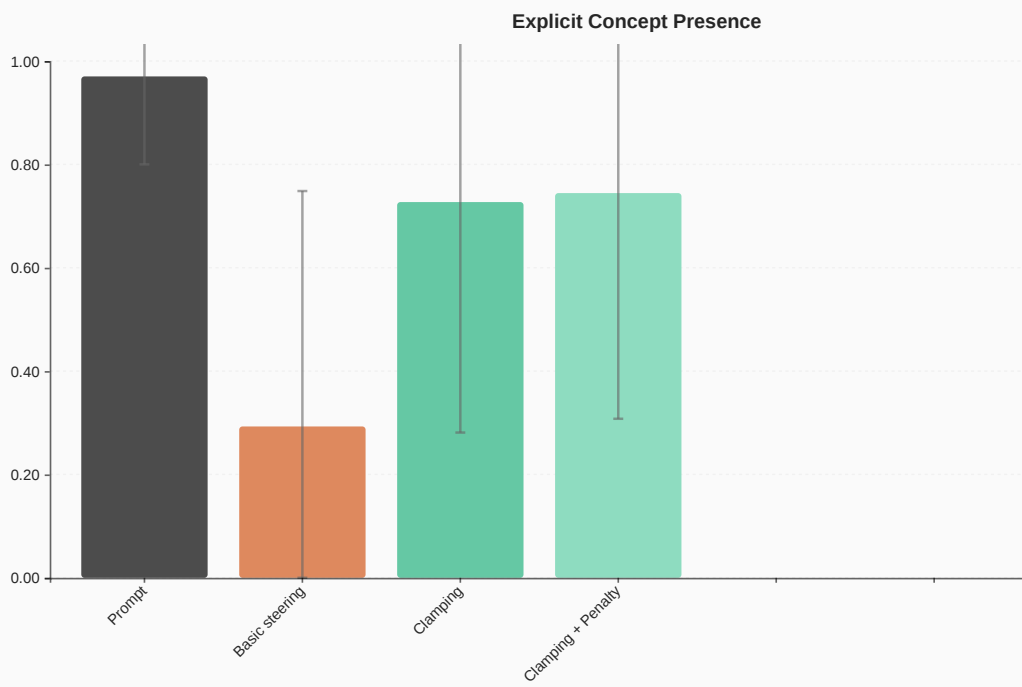
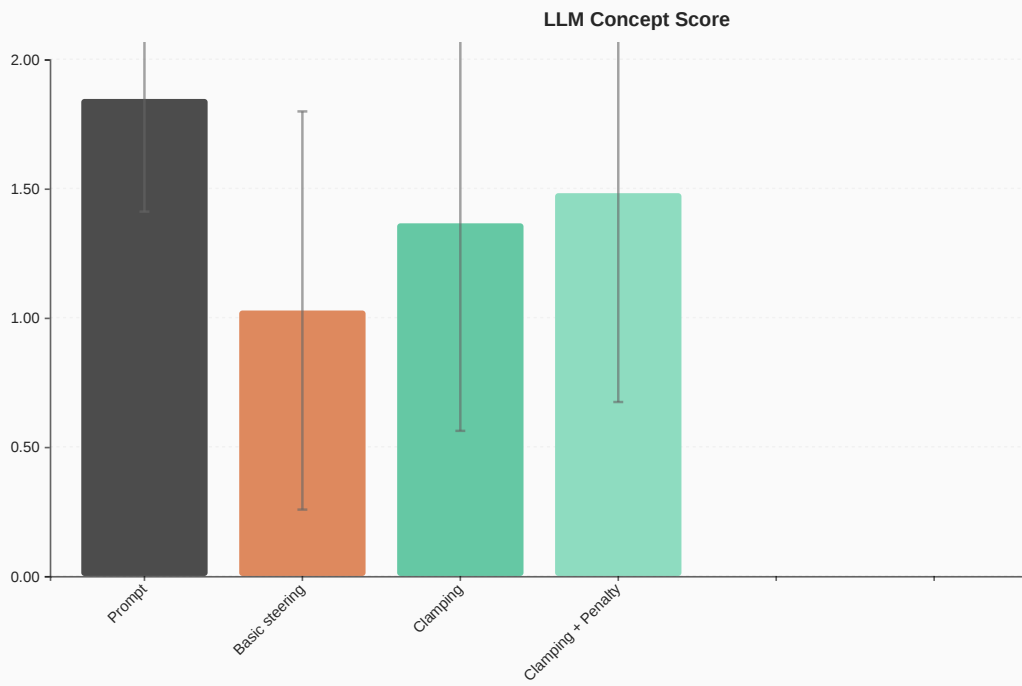
Having found optimal coefficients, we now investigate two complementary improvements that address the failure modes we identified: clamping to ensure consistent activations, and repetition penalty to prevent the gibberish mode.

First, we tested clamping the activations rather than using the natural additive scheme. Intuitively, this provides two potential benefits. First, it prevents the model from going to excessively high activations. In the additive scheme, these may result from steering applied to activations that are already high because of the influence of the previous tokens outputted by the model. On the other hand, clamping ensures that the feature is always activated at a certain level. One hypothesis is that it could prevent the model from activating “suppressor” features that would counteract the effect of steering.

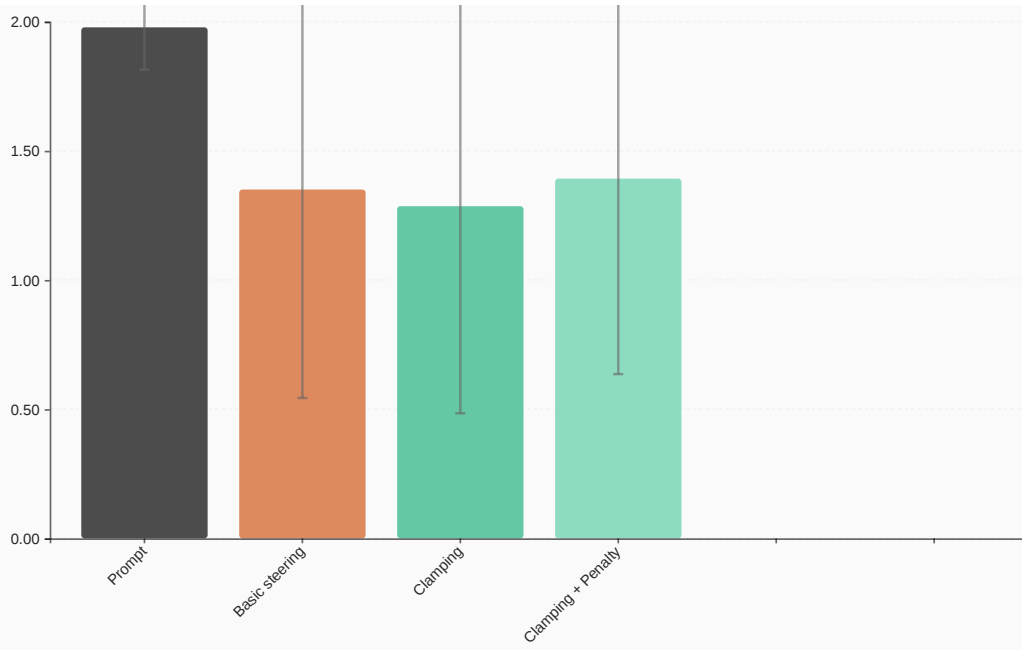
This clamping approach was used by Anthropic in their Golden Gate demo, but the AxBench paper found it less effective than the addition scheme for Gemma models. We decided to test it in our case.

4.1 Clamping

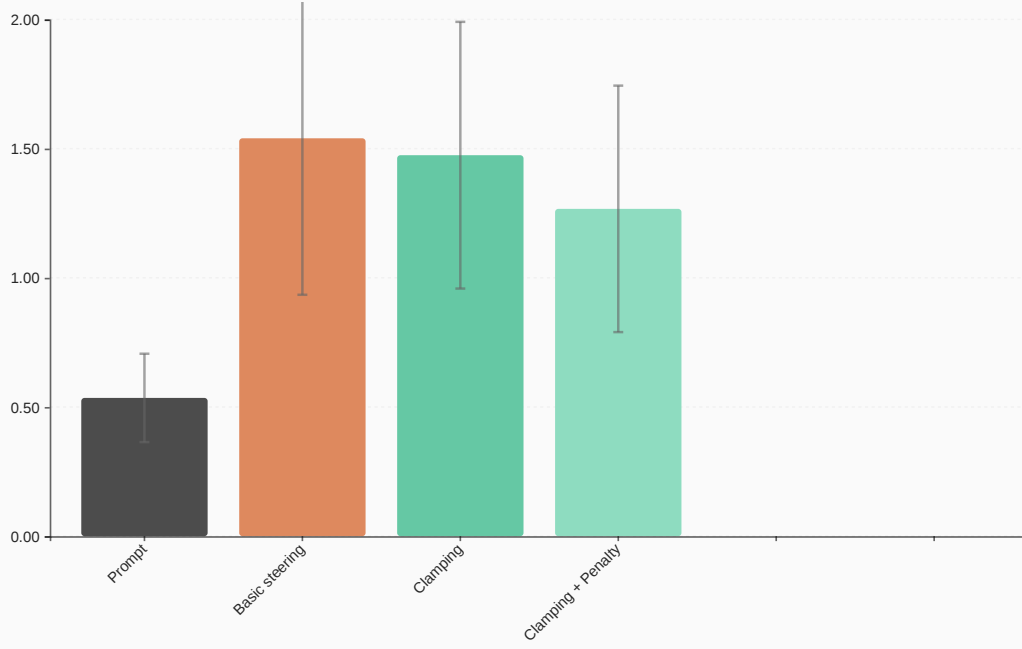
We tested the impact of clamping on the same steering vector at the optimal steering coefficient found previously ($\alpha = 8.5$). We evaluated the model on the same set of prompts and a maximum output length of 512 tokens.



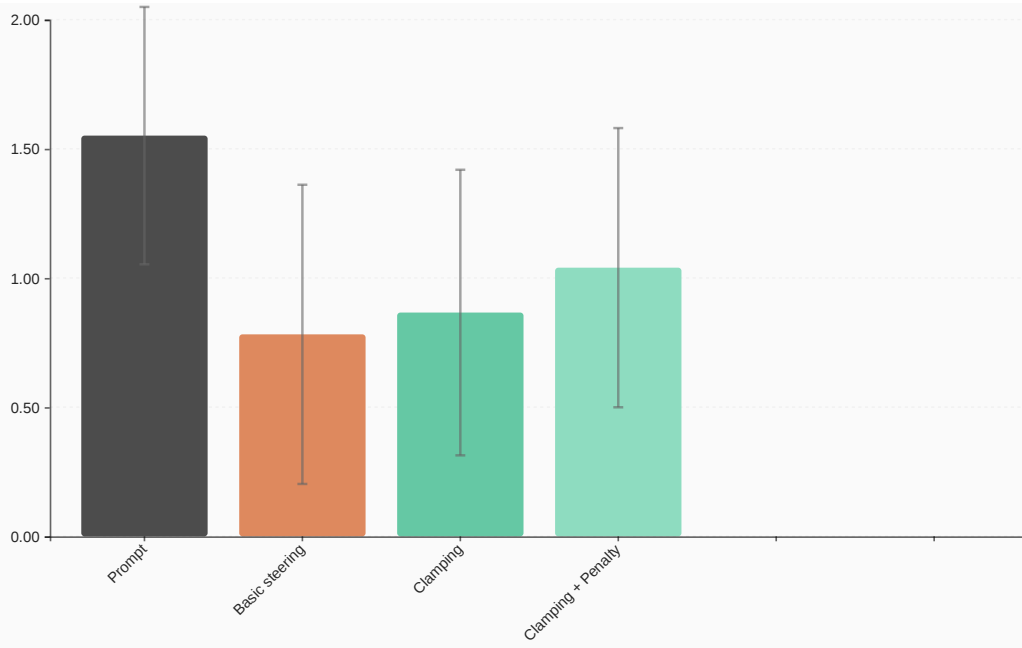
LLM Instruction Score



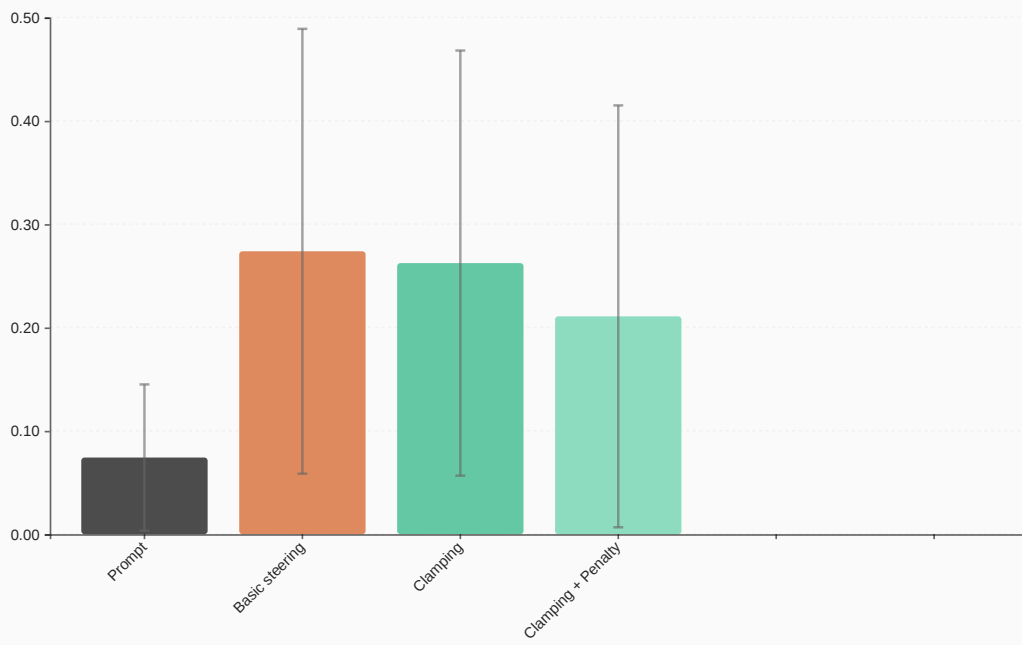
Surprise in Original Model



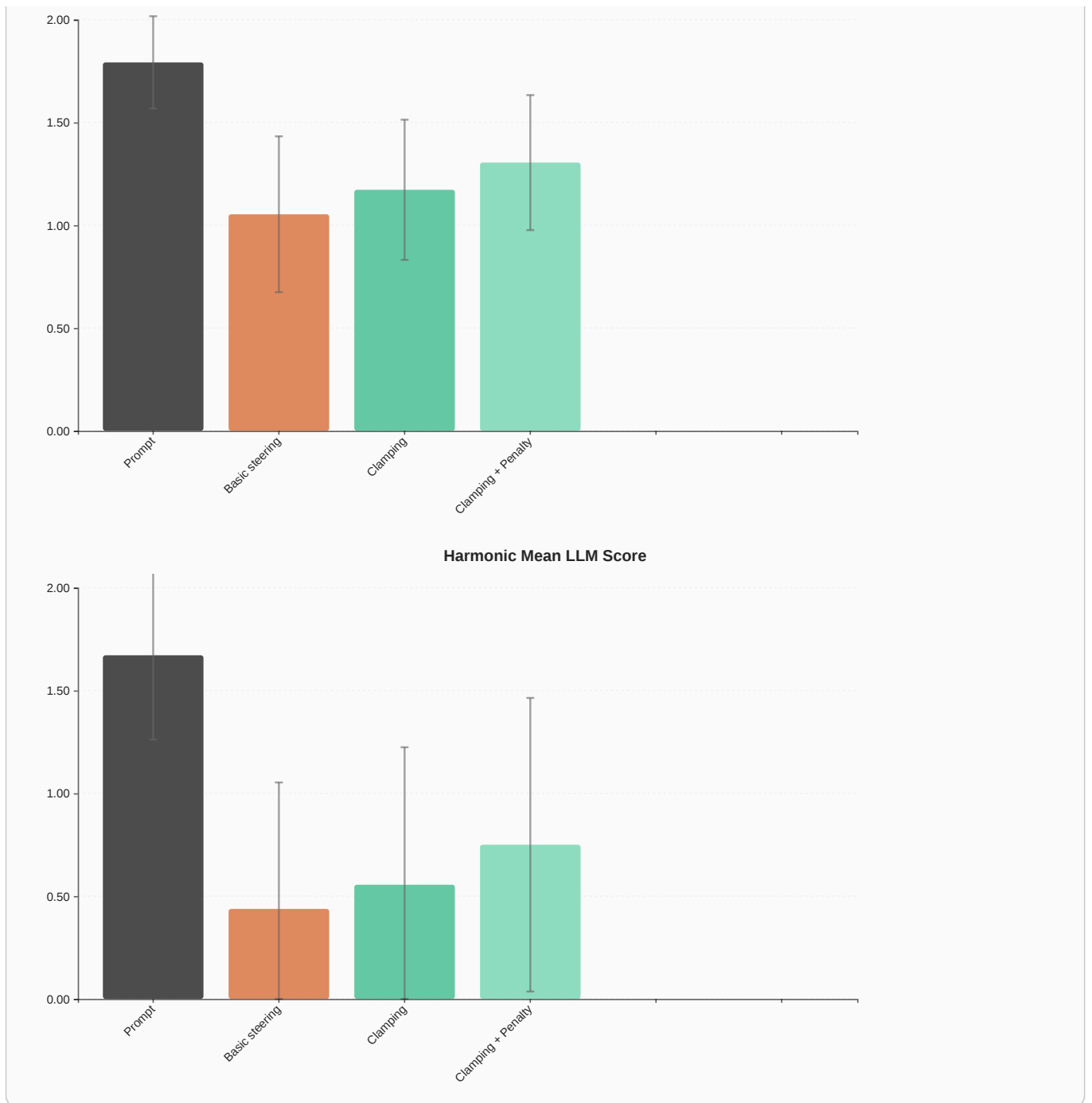
LLM Fluency Score



3-gram Repetition Fraction



Mean LLM Score



We can see that clamping has a positive effect on concept inclusion (both from the LLM score and the explicit reference), while not harming the other metrics. The fact that concept inclusion (but not fluency or instruction following) is improved suggests that clamping might help counteract some suppressor features preventing the Eiffel Tower concept from being fully activated, however confirming this hypothesis would require further investigation.

We therefore opted for clamping, in line with the choice made by Anthropic. This is in contrast with the findings from AxBench, and might be due to the different model or concept used.

Clamping is more effective than adding.

We found that clamping activations improves concept inclusion without harming fluency. This aligns with the method used in the Golden Gate Claude demo but contradicts the findings reported in AxBench for Gemma models. This might be due to differences in model architecture or the specific concept being steered.

4.2 Generation parameters

We have seen that repetition is a major cause of loss of fluency when steering with SAEs. To mitigate this, we tried applying a lower temperature (0.5), and apply a repetition penalty during generation. This technique involves penalizing the logit of tokens that have already been generated, preventing the model from repeating itself. We used a penalty factor of 1.1 using the `repetition_penalty` parameter of the generation API in 🤗Transformers (the repetition penalty implementation described in the [CTRL paper ↗](#))

Applying a repetition penalty reduces the 3-gram repetition as expected, and has a clear positive effect on fluency, while not harming concept inclusion and instruction following.

(Note that the AxBench paper mentioned the repetition penalty but without using it, considering it *“not the fairest setting, as it often does not accurately resemble normal user behaviour”*, see their appendix K)

Tuning generation parameters improve fluency and instruction following

Using a lower temperature (0.5) and applying a modest repetition penalty (1.1) during generation significantly reduces repetitions in the output. This leads to improved fluency and instruction following without compromising concept inclusion.

5. Multi-Layer optimization

Even after these improvements, we still found that steering with a single SAE feature proved insufficient, with concept inclusion lying way below the maximum possible value of 2.0. Since our investigation using Neuronpedia revealed that the Eiffel Tower concept was represented by many features in different layers, we hypothesized that steering several of those features simultaneously could lead to better results.

Indeed it has been reported that common phenomena are feature redundancy and feature splitting. These phenomena occur when a concept is represented by several features that are often co-activated or are responsible for the same concept in slightly different contexts. The sparsity constraint used during SAE training tends to favor such splitting, as it is often more efficient to use several features that activate less often, than a single feature that would activate more often.

These phenomena suggest that steering only one of those features therefore be insufficient to fully activate the concept, or to activate it consistently across different prompts. Moreover, activating one feature without the others might cause loss of fluency, as the model might experience activation patterns that are out of distribution compared to what it was trained on.

5.1 Layer and features selection

In total, we identified 19 candidate features, located in layers 3, 7, 11, 15, 19, 23, and 27. Note that those layers were the only ones for which SAEs were available, so it is likely that other features representing the Eiffel Tower exist in other layers.

We looked for those features using the search tool in Neuronpedia, and selected them based on their top activating prompts in the dataset. We kept only those features that unambiguously referenced the Eiffel Tower, and discarded

features that seemed to be more generally about Paris, towers, famous landmarks in big cities, or simply tokens like “E” or “iff”.

Among those 19 features, we selected all the features located in the intermediate layers 11, 15, 19 and 23. We decided to exclude features in earlier layers (six features in layer 3 and three features in layer 7) or later layers (two features in layer 27). We made this choice because features in intermediate layers are more likely to represent abstract high-level concepts. This led us to select 8 candidate features for our multi-layer steering.

5.2 Optimization methodology

Finding the optimal steering coefficients for multiple features presents several challenges:

- First, the parameter space grows with the number of features, making grid search quickly intractable.
- Second, the target function (the harmonic mean of LLM-judge metrics) is noisy and non-differentiable, making gradient-based optimization impossible.
- Finally, evaluating the target function is costly, as it requires generating answers from the steered model and evaluating them with an LLM judge.

To address these challenges, we used Bayesian optimization to search for the optimal steering coefficients, and we devised an auxiliary cost function to guide the optimization when the harmonic mean is zero and hence non-informative.

5.2.1 COST FUNCTION

Following the AxBench paper, we decided to look for steering coefficients that would maximize the harmonic mean of the three LLM-judge metrics. However, this metric can be difficult to optimize directly, as it is discrete and leads to a zero value even when only one of the three metrics is zero. This might make it hard to explore the parameter space.

To mitigate this, we defined an auxiliary cost function that would be used when the harmonic mean is zero. Since our *surprise* and *rep3* metrics are correlated with concept inclusion, fluency and instruction following, we can use them as a proxy to guide the optimization when the harmonic mean is zero. We considered an auxiliary cost function of the form

$$\text{cost} = |\text{surprise} - s_0| + k \text{ rep3}$$

We chose target surprise s_0 and weight k to maximize the correlation with the mean of LLM judge metrics (leading to $s_0 = 1.2$ and $k = 3$).

Overall, our cost function was defined as the harmonic mean of LLM-judge metrics, and we penalized it with a small fraction (0.05) of the auxiliary cost when the harmonic mean was zero, to provide some signal to the optimizer.

5.2.2 DEALING WITH NOISE

Ideally, we want to minimize *the expected value of our target function over the distribution of prompts and samples*. However, each call to the steered model will effectively only give a noisy estimate of that target, evaluated on a single prompt and one sample.

This is a black-box optimization problem, where each evaluation of the target function is costly (as it involves generating a full answer from the model) and noisy (as it depends on the prompt and the sample). To tackle this, we decided to rely on Bayesian optimization.

Bayesian Optimization (BO) is known to be well-suited for multidimensional non-differentiable costly black-box optimization, while being able to handle noisy evaluations. To mitigate the noise, we could average the target function over several prompts and samples, but this would have been costly, especially when evaluating points that are not promising. For very noisy functions, performing Bayesian optimization directly on the raw function is more efficient than averaging multiple noisy evaluations for each point.

5.2.3 BAYESIAN OPTIMIZATION

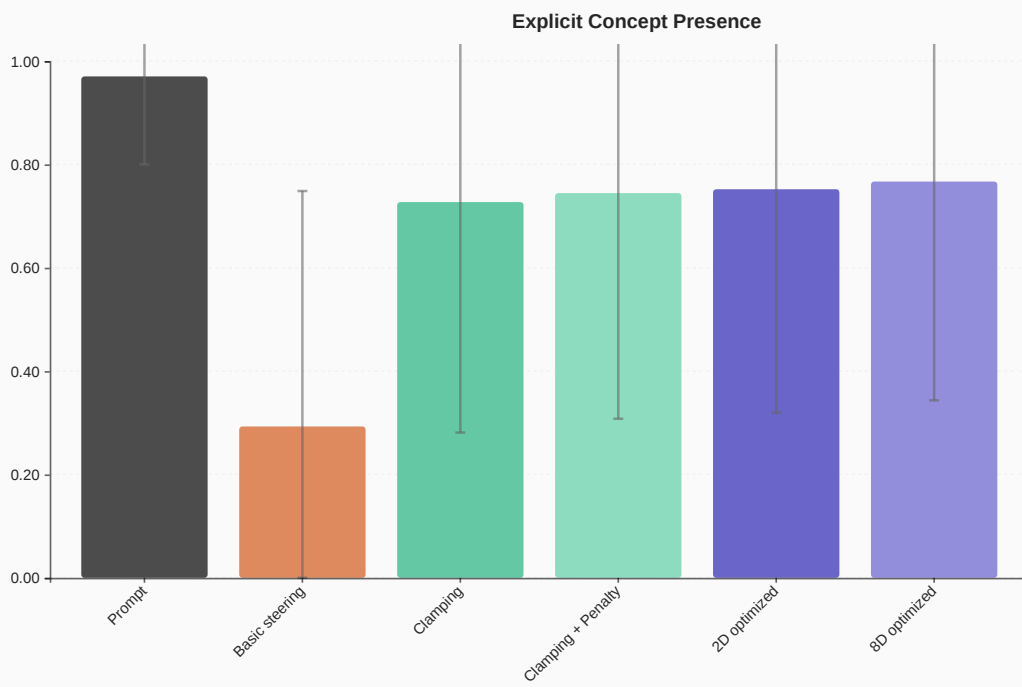
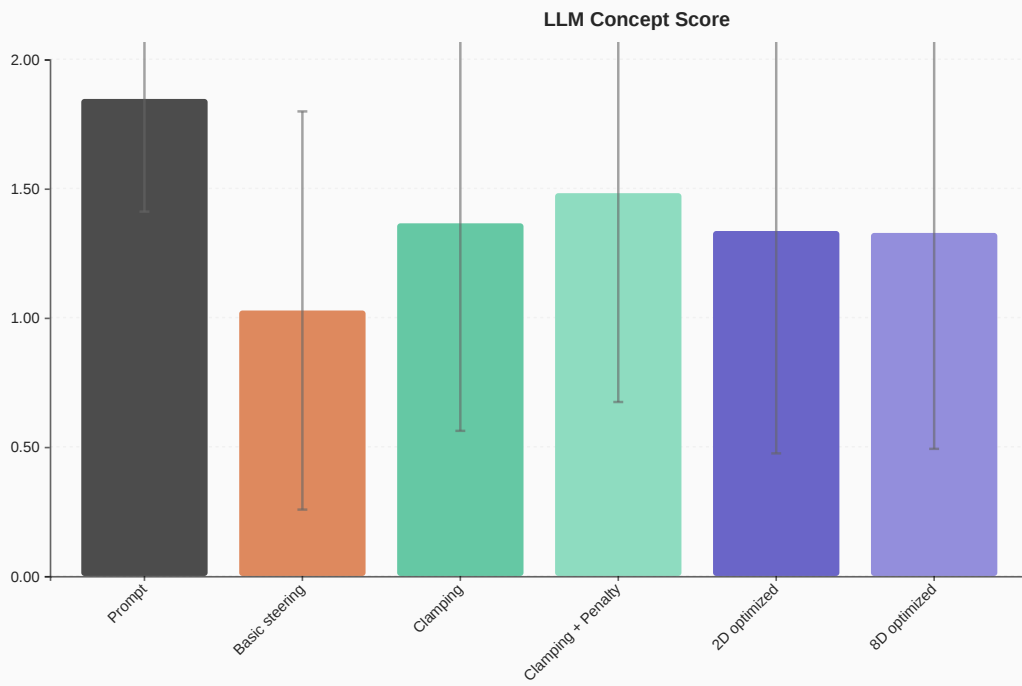
The idea behind BO is to build a surrogate model of the target function using a Gaussian Process (GP), and use that surrogate to select promising candidates to evaluate next. With each evaluation, we update the GP model, and iteratively refine our surrogate of the target function.

To do this, we used the [BoTorch library](#) , which provides a flexible framework to perform BO using PyTorch. More details are given in the appendix.

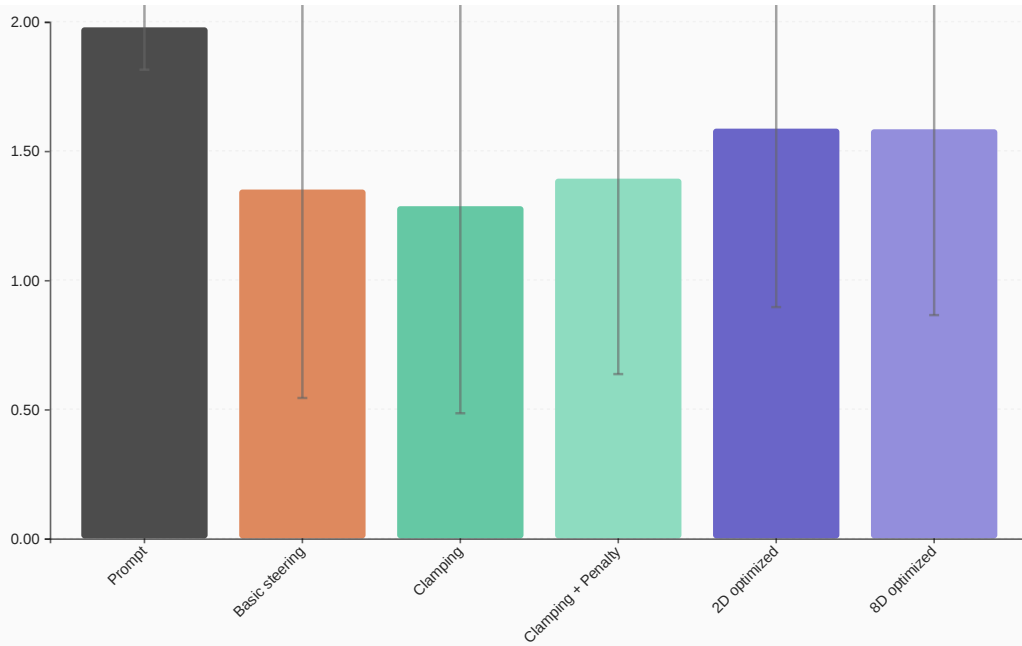
5.3 Results of multi-layer optimization

We first performed optimization using only 2 features (from layer 15 and layer 19) and then 8 features (from layers 11, 15, 19 and 23), based on the hypothesis that steering the upper-middle layer is likely to be more effective to activate high-level concepts.

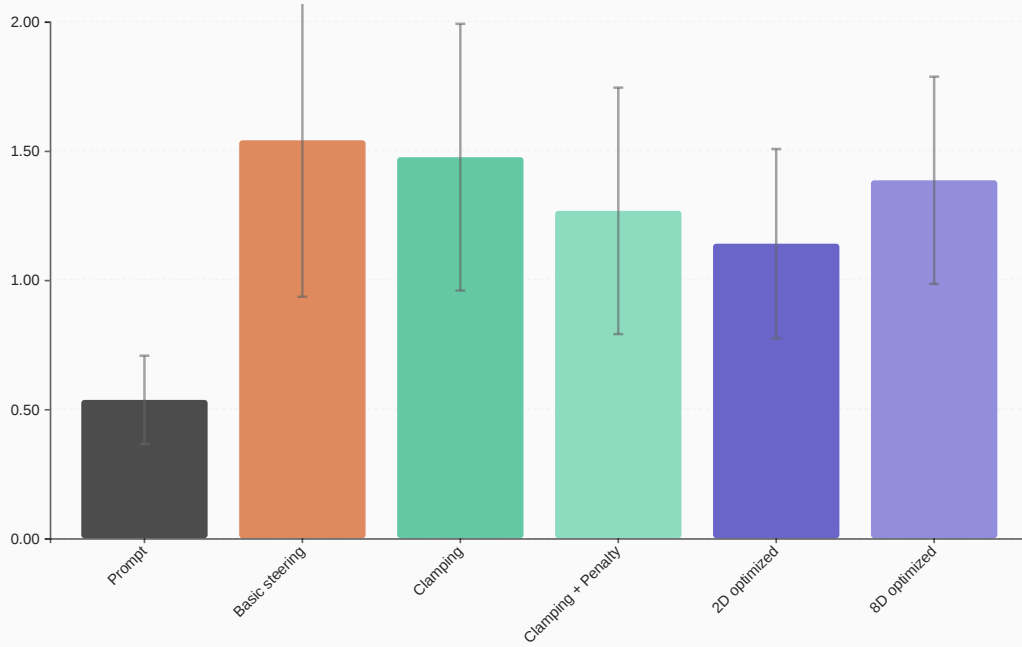
Results are shown below and compared to single-layer steering.



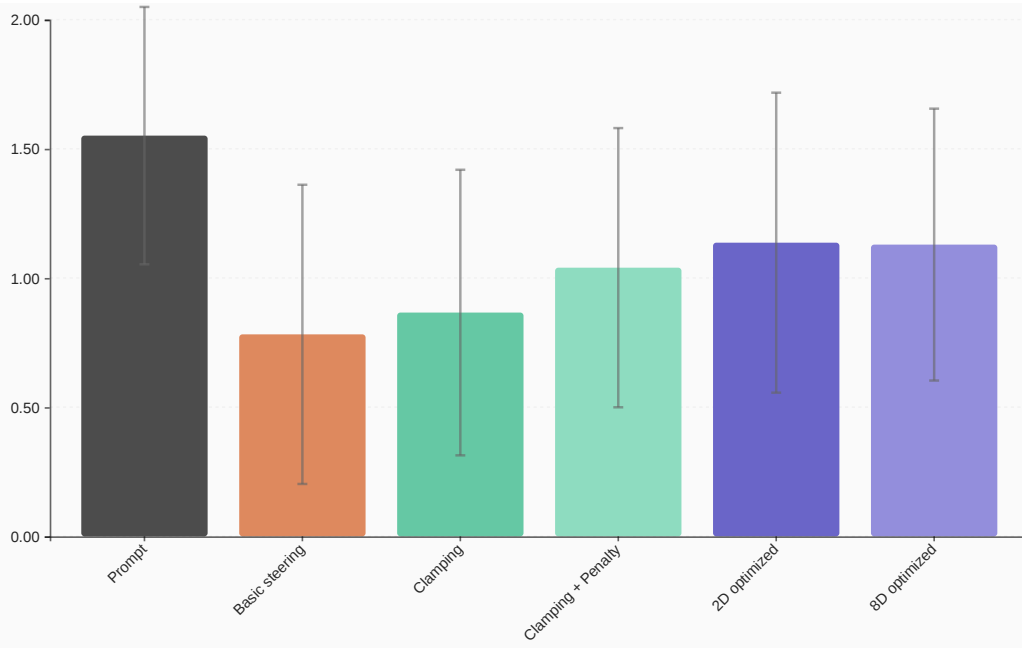
LLM Instruction Score



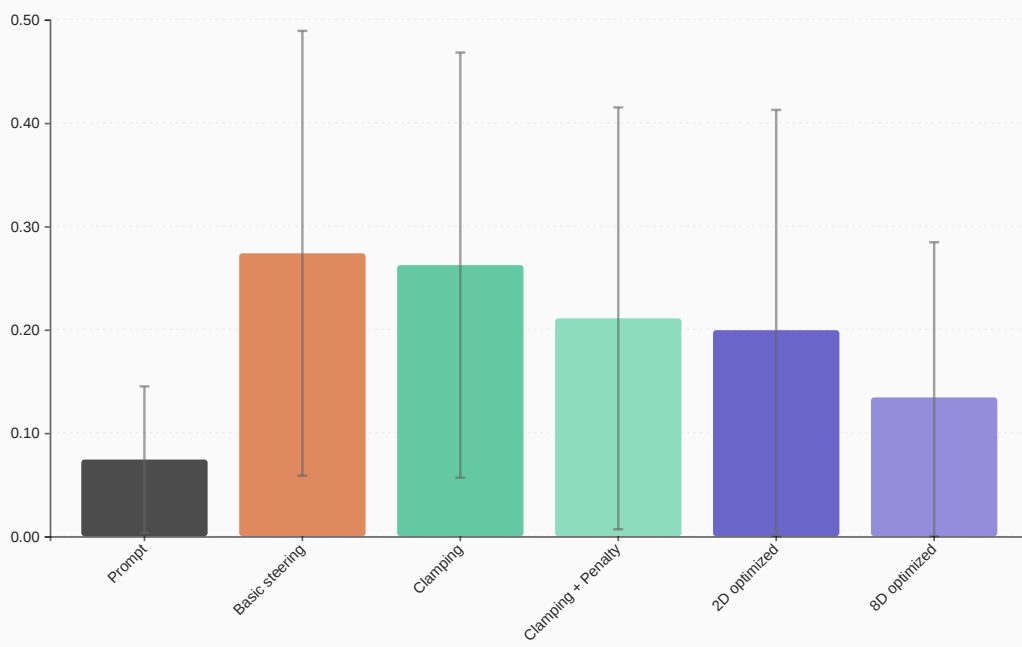
Surprise in Original Model



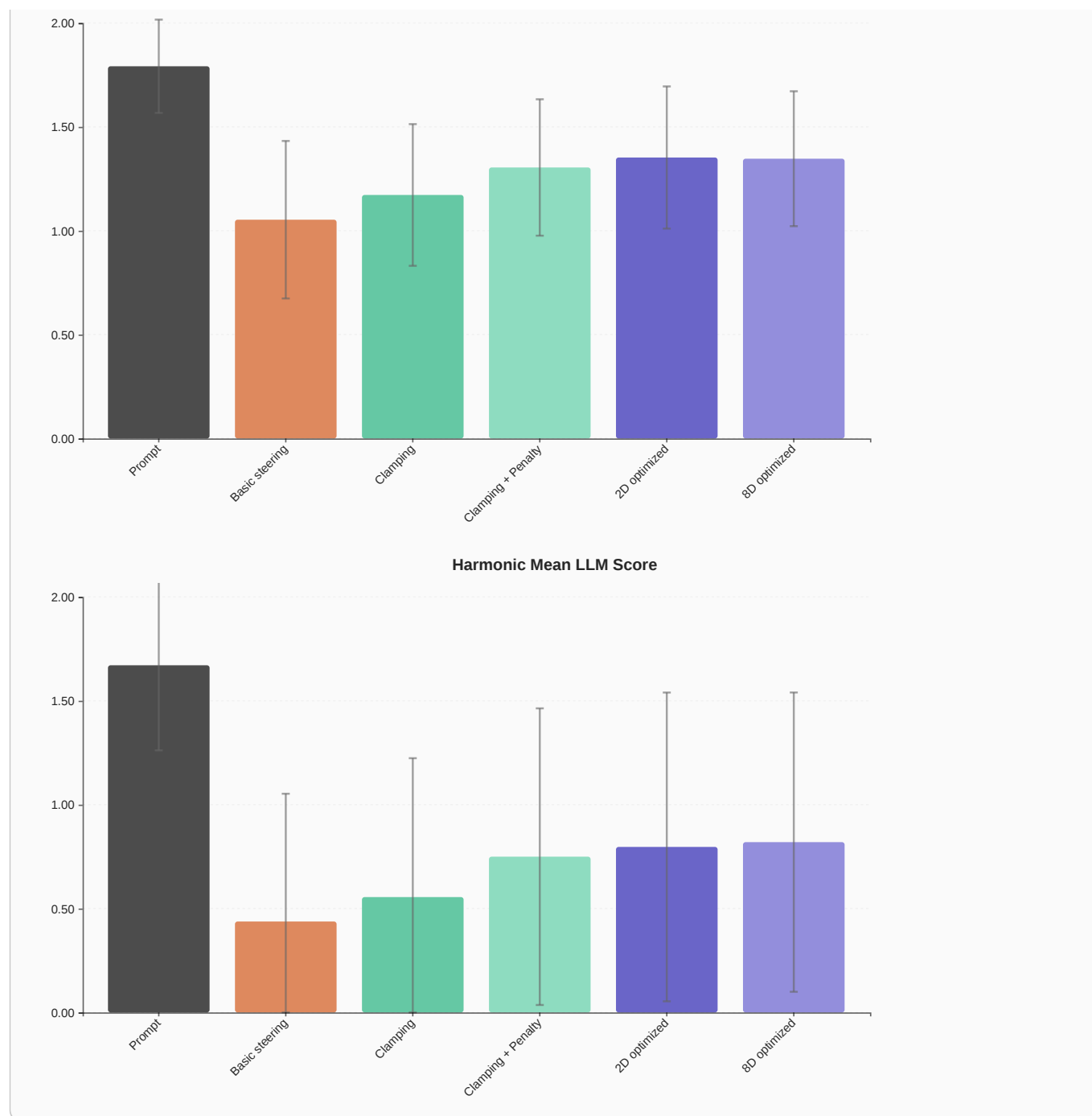
LLM Fluency Score



3-gram Repetition Fraction



Mean LLM Score



As we can see on the chart, steering 2 or even 8 features simultaneously leads to only marginal improvements compared to steering only one feature. Although fluency and instruction following are improved, concept inclusion slightly decreases, leading to a harmonic mean that is only marginally better than single-layer steering.

This reflects the fact that instruction following and fluency are generally correlated, so improving one tends to improve the other. Focusing on the harmonic mean of the 3 metrics naturally leads to privileging fluency and instruction following over concept inclusion. Additionally, we observed the concept inclusion LLM judge to be quite harsh and literal. Sometimes mention of Paris or a large metal structure were not considered as valid references to the Eiffel Tower, which could explain the low concept inclusion scores.

Overall, these disappointing results contradict our initial hypothesis that steering multiple complementary features would help better represent the concept and maintain fluency.

One possible explanation is the difficulty of finding the true optimum, as the harmonic mean metric is very noisy and hard to optimize in the high-dimensional space.

Another plausible explanation could be that the selected features are actually redundant rather than complementary, and that steering one of them is sufficient to fully activate the concept. Investigating this would require monitoring the activation changes in subsequent layers' features when steering multiple features. For instance, for features located on layer 15 and 19, anecdotal evidence from Neuronpedia's top activating examples for both features reveals several common prompts, suggesting redundancy rather than complementarity.

More features don't necessarily mean better steering.

Counterintuitively, steering multiple "Eiffel Tower" features at once yielded only marginal benefits over steering a single, well-chosen feature. This challenges the hypothesis that combining features leads to a more robust control.

6. Conclusion & Discussion

6.1 Main conclusions

In this study, we demonstrated the use of sparse autoencoders to steer a lightweight open-source model (Llama 3.1 8B Instruct) to create a conversational agent obsessed with the Eiffel Tower, similar to the Golden Gate Claude experiment. As reported by the AxBench paper, and as can be experienced on Neuronpedia, steering with SAEs is harder initially expected, and finding good steering coefficients is not easy.

First, we showed that simple improvements like clamping feature activations and using repetition penalty and lower temperature can help significantly. We then devised a systematic approach to optimize steering coefficients using bayesian optimization, and auxiliary metrics correlated with LLM-judge metrics.

Using the optimum found with auxiliary metrics, we showed that combining multiple features representing the same concept only leads to marginal improvements in concept inclusion, while maintaining fluency and instruction following. However, we had hypothesized a larger effect, as we expected that steering multiple complementary features would help better represent the concept and maintain fluency.

This may be because the selected features are actually redundant rather than complementary, and that steering one of them is sufficient to activate the concept. Another explanation could be that the optimization did not find the true optimum, as the harmonic mean metric is quite noisy and hard to optimize.

Overall, our results are in line with the success of the Golden Gate Claude demo, although we don't know all the details of their steering method. Our results also seem less discouraging than those of AxBench, and show that steering with SAEs can be effective, using clamping, a slightly different generation procedure and possibly combining multiple features. However, at this stage, these results are hard to generalize and our work is not directly comparable to the AxBench results, since they use different model, different concepts, different SAEs.

6.2 Future Directions

This investigation opens several avenues for future work that could not only improve steering procedures but also reveal fundamental insights about activation patterns in LLMs. These include:

- Investigate clamping: Why does clamping helps in our case, similar to Anthropic, while AxBench found the opposite? One hypothesis is that it prevents extreme activations, but it could also counteract some negative

feedback behavior, when other parts of the model activate suppressor features to try to compensate for the added steering vector. This suggests an analogy with biology, where signaling pathways are often regulated by negative feedback loops. An interesting direction could be to analyze the cases where the model tries to “backtrack”, e.g. outputting *“I’m the Eiffel Tower. No, actually I’m not.”* By analyzing the activations just before the “No”, can we highlight some *regulatory/suppressor features* that try to suppress the Eiffel Tower concept when it has been overactivated?

- Determine why steering multiple features achieves only marginal improvement: Investigate complementary vs redundancy of multiple features by monitoring activation changes in subsequent layers’ features.
- Perform a failure analysis on the prompts where steering fails (about 20% have at least one metric with a zero rating). Is there a pattern?
- Check other concepts and other models, and determine if some layers are more effective than others. In particular incorporate earlier and later layer, see if it helps the multi-layer steering.
- Vary the temporal steering pattern, for instance steer either only the prompt, or the generated answer; possibly use a periodic steering ?
- Investigate wording in the “prompt engineering” case. For now, the prompted model seems to really behave like it has to check a box, rather than actually integrating the concept in a natural way. Explore whether a more natural integration is possible. Does it show up in the activation pattern ? For instance, after mentioning the Eiffel tower, does the model activate regulatory features to prevent further mentions ?

We plan to explore some of these directions in future work.

Code is available [here](#) ↗

Acknowledgments: Thanks to the NDIF team and especially Jaden Fiotto-Kaufman for help using `nnsight`, to Thom Wolf and Leandro von Werra for useful discussions, to Clémentine Fourier for reading a first draft of the blog post, and to Thibaud Frere for help using his excellent [Bringing Paper To Life](#) ↗ blog post template.

Appendix

nnsight code

Example of code used to perform steering and generation with nnsight:

```

1 input_ids = llm.tokenizer.apply_chat_template(chat, tokenize=True,
  add_generation_prompt=True)
2 with llm.generate() as tracer:
3     with tracer.invoke(input_ids):
4         with tracer.all() as idx:
5             for sc in steering_components:
6                 layer, strength, vector = sc["layer"], sc["strength"], sc["vector"]
7                 length = llm.model.layers[layer].output.shape[1]
8                 amount = (strength * vector).unsqueeze(0).expand(length,
-1).unsqueeze(0).clone()
9                 llm.model.layers[layer].output += amount
10            with tracer.invoke():
11                trace = llm.generator.output.save()
12
13 answer = llm.tokenizer.decode(trace[0][len(input_ids):], skip_special_tokens=True)

```

Bayesian optimization details

We considered a simple Gaussian Process (GP) model with an RBF kernel. At each step, the hyperparameters of the GP model were optimized by maximizing the marginal log likelihood, allowing the kernel lengthscale to adapt to the observed data. Then we select a promising candidate using the `qNoisyExpectedImprovement` acquisition function, which balances exploration and exploitation. This acquisition function is well-suited for noisy functions, as it takes into account the noise in the observations.

For domain search, as we know that activation magnitude grows roughly linearly with layer index, we expect that the optimal steering coefficient for a feature in layer l should scale with l . We used the reduced parameterization presented earlier, searching for an optimal value in the range $[0, 1]$:

$$\hat{\alpha}_l = \frac{\alpha_l}{l}$$

To favor noise reduction at promising locations, every 5 steps we decided to resample the best point found so far. In that case, by *best* we mean the point with the lowest GP posterior $\mu(x)$. (Note that this is different from the point with the lowest observed value which might be a lucky noisy outlier).

Performing gradient descent on the GP posterior is very cheap since it only involves differentiating the kernel function. We thus performed gradient descent starting from 500 random points in the parameter space, and optimized using a target being upper confidence bound $\mu(x) + \beta\sigma(x)$, to favor points that are not only predicted to be good, but also with low uncertainty. We then performed a clustering to group together the points that converged to the same local minimum, and selected the best cluster as candidate for evaluation.

Citation

For attribution in academic contexts, please cite this work as

David Louapre (2025). "The Eiffel Tower Llama".

BibTeX citation

```
@misc{louapre2025_the_eiffel_tower_llama,  
  title={The Eiffel Tower Llama},  
  author={David Louapre},  
  year={2025},  
}
```

Reuse

Diagrams and text are licensed under [CC-BY 4.0](#) with the source available on [Hugging Face](#), unless noted otherwise. Figures reused from other sources are excluded and marked in their captions ("Figure from ...").

References

1. Cunningham, H., Ewart, A., Riggs, L., Huben, R., & Sharkey, L. (2023). Sparse autoencoders find highly interpretable features in language models. *arXiv Preprint arXiv:2309.08600*. [↑] back: [1](#), [2](#)
2. Fiotto-Kaufman, J., Loftus, A. R., Todd, E., Brinkmann, J., Pal, K., Troitskii, D., Ripa, M., Belfki, A., Rager, C., Juang, C., & others. (2024). NNSight and NDIF: Democratizing access to open-weight foundation model internals. *arXiv Preprint arXiv:2407.14561*. [↑]
3. Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., & Wu, J. (2024). Scaling and evaluating sparse autoencoders. *arXiv Preprint arXiv:2406.04093*. [↑]
4. Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramár, J., Dragan, A., Shah, R., & Nanda, N. (2024). Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv Preprint arXiv:2408.05147*. [↑] back: [1](#), [2](#)
5. Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Summers, T. R., Rees, E., Batson, J., Jermyn, A., ... Henighan, T. (2024). Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread*. <https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html> [↑] back: [1](#), [2](#)
6. Wu, Z., Arora, A., Geiger, A., Wang, Z., Huang, J., Jurafsky, D., Manning, C. D., & Potts, C. (2025). Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *arXiv Preprint arXiv:2501.17148*. [↑] back: [1](#), [2](#)