Between the Bars: Gradient-based Jailbreaks are Bugs that induce Features

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent work has demonstrated that it is possible to find gradient-based jailbreaks (GBJs) that cause safety-tuned LLMs to output harmful text. Understanding the nature of these adversaries might lead to valuable insights for improving robustness and safety. We find that, even relative to very semantically similar baselines, GBJs are highly out-of-distribution for the model. Despite this, GBJs induce a structured change in models. We find that the activations of jailbreak and baseline prompts are separable with unsupervised methods alone. Using our understanding, we are able to steer models without training, both to be more or less susceptible to jailbreaking, and to output harmful text in response to harmful prompts *without jailbreaking*.Our findings suggest a picture of GBJs as "bugs" that induce more general "features" in the model: highly out of distribution text that induce understandable and potentially controllable changes in language models.

1 Introduction

Adversaries as bugs vs features In the past, gradient-based adversaries were largely studied in the context of image classification, where an l_p bounded perturbation to an image is optimized to fool an image classifier. Within this setting, Ilyas et al. (2019) delineates two ways adversarial examples can trick a model: they are either

- bugs: aberrations in a classifier that do not reflect intrinsic properties of the data distribution, but artifacts of the model's training and high-dimensional geometry
- features: the result of the classifier using attributes imperceptible to humans that *do* reflect the data distribution

These two explanations imply different predictions about adversarial example phenomenology: most notably, the features view says that adversarial examples should be transferable, while the bugs view says that they shouldn't be, insofar as they are specific to a model's geometry.

Adversaries in language modeling Making text-only gradient-based attacks in language modeling is more challenging than in vision due to the discrete nature of the input space. Greedy Coordinate Gradient (GCG) (Zou et al., 2023b), which will be the focus of study for this paper, allows us to reliably obtain language GBJs, in the form of adversarial suffixes of tokens that we append to our prompt.

Moving to language also requires rethinking "bugs" and "features". We can still think about bugs as off-distribution inputs that exploit artifacts of high dimension geometry. But in language modeling, the term feature is commonly used to refer to a "meaningful" linear direction in latent space, encoding a concept (Bricken et al., 2023). Here, then, the term "bug" is a descriptor of an input, and "feature" is a descriptor of a induced computation.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

We aim to provide evidence, then, that GBJs can be understood as "bug" inputs inducing "feature" computation.

Contributions Our works makes two major contributions:

1. We show that GBJs adversaries are highly out of distribution: we find that these GBJs produce highly out-of-distribution logit distributions, even relative to semantically identical baselines. We also find evidence of glitch-like tokens (Rumbelow, 2023) comprising a plurality of the jailbreak text.
2. We find that we can separate GBJs from non-adversarial baselines with unsupervised methods clustering of the activation space. We steer our model using the difference in means between GBJs and baselines on *unsafe inputs that are not even optimized*. This allows us to make the model both more or less susceptible to harmful behavior, without needing any extra optimization at inference.

2 Setup

GCG training Using NanoGCG (Zou et al., 2023b), we generated 119 adversarial suffixes of length 75 (generated from a initial string of 'x's seperated by spaces), each of which was optimized for a particular harmful request from Harmbench's (Mazeika et al., 2024) test standard. We present an example in B.

For this work, we study adversaries for Gemma-2-2b (Team et al., 2024). Using NanoGCG (Zou et al., 2023b), we generated 119 adversarial suffixes of length 75, each of which was optimized for a particular harmful request from Harmbench's (Mazeika et al., 2024) test standard.

Evaluating GBJ accuracy We use human evaluation to determine the accuracy of our GCG adversaries (Table 2), since we find that there are many edge cases in whether or not an output is harmful.

We define a partial jailbreak as when the model agrees to the request but then changes its mind and does not elaborate.

Table 1: Jailbreak Results

Description	Percentage
Refusal	8.40%
Partial Jailbreak	16.81%
Full Jailbreak	74.79%

Baselines To study our jailbreaks, we generate three baseline prompts that we use extensively in our analysis.

1. The *original prompt* baseline: we use the unaltered Harmbench prompt
2. The *unoptimized prompt* baseline: we use the 75 'x's seperated by spaces that we use to optimize the prompt from
3. The *nearest neighbor prompt* baseline: here, we decode each token in the attack to its nearest neighbor excluding itself.

We make extensive use of the *nearest neighbor prompt* baseline, since it is semantically very similar to the GBJ, and poses the "hardest" case to distinguish from the adversary itself. To a human, the *nearest neighbor prompt* baseline is indistinguishable from a GBJ in appearance, which makes it particularly interesting.

Notably, **none** of the baselines jailbreak the model at all.

3 Are gradient-based jailbreaks bugs?

Unlike manual jailbreaking methods, GBJs contain no easily identifiable traces of their attack mechanism (eg personas, concealing formats, etc...), suggesting that they are "bugs". We begin our analysis by presenting evidence that GBJs are indeed "bugs" - brittle aberrations are off-distribution for the model. We use confidence-based out-of-distribution (OOD) evaluation and discover that GBJ adversaries are highly out-of-distribution inputs for the model. We also find preliminary evidence of "glitch tokens" (Rumbelow, 2023), where the optimization tends towards very rare tokens that elicit strange behavior.

GBJs are out-of-distribution In order to determine whether GBJs are out-of-distribution, we use the well studied confidence-based OOD detection baseline (Hendrycks & Gimpel, 2016). The intuition for this detection method is simple: when given an OOD input, the model is less confident in its next token prediction than when facing an ID input.

In particular, we compute the mean max probability:

$$MeanMaxProbability = \frac{1}{N}\sum_{i=1}^{N}\max_{y} p(y|x_i) \quad (1)$$

Here, x_i represents the i-th token in the input sequence (prompt and suffix), $p(y|x_i)$ is the softmax probability distribution over possible outputs y given x_i, and N is the total number of tokens in the sequence.

We then compare the mean max probability of GBJs to baseline inputs, where lower mean max probability indicates that the example is OOD.

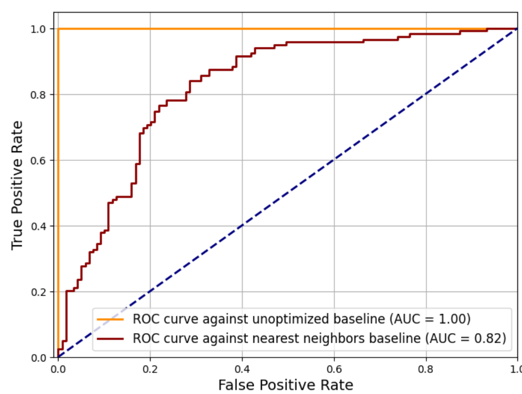

Figure 1: ROC curve for disambiguating GBJs and *unomptimized* baseline (orange), and *nearest neighbors* baseline (red) via confidence based OOD testing. As expected, the model is much more confident on the repetitive *unomptimized* baseline, making the classification perfect. We are still able to disambiguate GBJs and *nearest neighbors* fairly reliably (AUC = 0.82).

We find (1) that the adversaries are very OOD, even relative to the semantic baseline - this baseline method achieves an AUC (area-under-the-curve) of 1.0 compared to the *unoptimized prompt* baseline, and an AUC of 0.82 compared to the *nearest neighbor baseline*.

This latter result is particularly striking, as it means that the model is *much* less confident about the contents of GBJ than a string which is nearly semantically identical to it.

4 Evidence for Latent Structure in Gradient-based Jailbreaks

We study the features and structure that separate optimized harmful inputs from unoptimized ones. Do these separating directions encode meaningful, more general features? We find that despite their out-of-distribution nature, GBJs target general features that control safe vs unsafe behavior, whether or not the input is optimized. Building on this, we present further evidence supporting GBJs as having some "feature-like" characteristics.

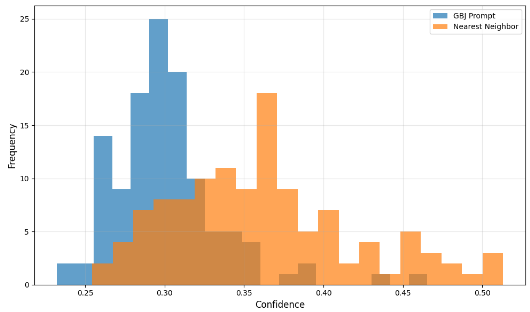

Figure 2: Model confidence on GBJs (blue) vs the *nearest neighbors* (orange) baselines. We find that the distribution of mean confidences is right shifted and flatter for the nearest neighbors baseline, indicating that the model is less confident (more out-of-distribution) for the GBJs.

GBJ activations are highly separable from unoptimized text Building on our observation that GBJs are highly out-of-distribution inputs, we investigated whether this is reflected in the model's internal representations. Specifically, we sought to determine if GBJs inputs occupy a distinct region in the model's latent space compared to baseline inputs.

Motivated by insights from attribution patching (Syed et al., 2023) (see D), we know that the information contained in adversarial suffixes are primarily concentrated in the last token, especially as the model depth increases. Therefore, we focused our analysis on the activations of this final token, which in our case is a chat completion token.

We defined two datasets for each layer l:

- $D_{adv}(l)$: Set of activations from the final token of GBJ inputs at layer l
- $D_{base}(l)$: Set of activations from the final token of baseline inputs (nearest neighbors prompt) at layer l

For each layer in the model, we concatenated $D_{adv}(l)$ and $D_{base}(l)$ and applied K-means clustering with 2 clusters to this combined dataset. Our hypothesis was that if GBJs occupy a distinct region in the latent space, the clustering algorithm would naturally separate $D_{adv}(l)$ and $D_{base}(l)$.

To quantify the separation, we identify the K-Means cluster to D_{adv} vs D_{base} based on the majority vote of its members, and then we calculated the classification accuracy of this algorithm.

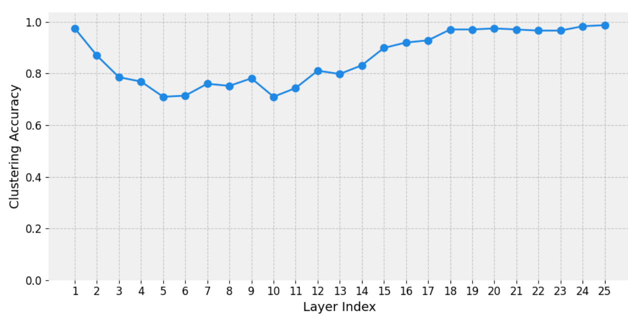

Figure 3: K-means clustering accuracy at each layer for separating GBJ activations from baseline activations. Observe that the accuracy never dips below 0.7, and that it is near perfect towards the end of the model.

Figure 3 shows the results of this analysis. We observe that the clustering accuracy is consistently high across all layers, and is particularly high at the beginning and end of the model depth. This suggests that GBJ inputs indeed occupy a distinct region in the model's latent space, particularly in the deeper layers of the network. The increasing separability in deeper layers suggests that the model's higher-level representations become particularly sensitive to the presence of adversarial inputs.

Using activations to get a steering direction for harmful behavior Motivated by the high separability of GBJ and baseline activations observed in our clustering analysis, we hypothesized that we could leverage this clear distinction between the two classes to control model behavior. Notably, the difference between these clusters could describe either the OOD geometry of adversaries versus normal text or it could describe the existence of higher order features in these adversaries to describe unsafe vs safe behavior. To test this, we developed a steering method based on the activation differences that allows us to validate and make use of the latent structure:

1. For each layer l, we computed a steering vector based on the difference in the mean activation:
$$\vec{r_l} = \frac{1}{N_{adv}} \sum_{i=1}^{N_{adv}} a_{adv,i}(l) - \frac{1}{N_{base}} \sum_{j=1}^{N_{base}} a_{base,j}(l)$$
where $a_{adv,i}(l)$ and $a_{base,j}(l)$ are activations of the final token for GBJ and baseline inputs respectively.
2. We ran the model, where we applied steering to new inputs at layer l:
$$a_{steered}(l) = a_{original}(l) \pm \alpha \vec{r_l}$$
3. We evaluated the impact on model outputs for various l and α.

We compute the steering vectors on 75% of our dataset, and evaluate them on the remaining 25%.

Glitch Tokens To better understand the structure of GBJ adversaries, we conducted an analysis of their unigram statistics. This revealed three categories of frequently occurring tokens:

Table 2: Categories of Frequent Tokens in GBJ Adversaries

Category	Count	Example Tokens
Remnants from unoptimized string	399	'x'
Glitch-like tokens	162	'resourceCulture', 'betweenstory', 'IVEREF'
Affirmations	87	'Sure', 'SURE'

The presence of glitch-like tokens (Rumbelow, 2023), indicates a potential exploitation of "bug" type model vulnerabilities, where these are the tokens in particular that have rarely been optimized over or seen by teh model. (Rumbelow, 2023) find that these tokens are much closer to the embedding centroids - in future work we would like to study these more.

Controlling Models with Steering Alone On this holdout set $D_{holdout}$, we study the effect of steering on harmful queries from HarmBench. These are just plain English sentences with no optimized component, therefore very qualitatively different from $D_a dv$, which is what we used to compute the steering vector.

We explore how this method can allow us to both induce and remove harmful behaviors.

We obtain the following attack success rate, for the optimal $\alpha = 1, l = 15$ combination found by sweeping:

Table 3: Success Rates

Description	Success Rate
Jailbreak	74.79%
Steering	96.6%

Note that GCG adversaries are optimized per prompt, whereas the steering works on sets of requests and is not a function of the data we are evaluating it on.

We can do the same method in the opposite direction and remove r_l from the activations, to intervene on susceptibility. We run the GCG prompts of the holdout set with $\alpha = -3, l = 10$, and find that 100% of the GCG attacks no longer work. This highlights a potential avenue for steering for robustness in more general cases, even potentially against persona jailbreaks.

Understand model features via steering The success of steering for jailbreaking on unoptimized inputs strongly suggests the direction found in these intermediate layers encodes more general features relevant to harmfulness and agreeableness.

To make this claim more precise and study the emergence of general features, we measure the optimal jailbreak rate we can get with steering at each layer. The intermediate layers are clearly the most useful, where the last and first are practically useless for steering.

This reinforces our hypothesis that GBJs are **bugs that induce features**:

1. Initially they have no meaningful semantic structure that generalizes to unoptimized inputs, and are very OOD.
2. At intermediate stages of the model, they encode directions that allow for harmfulness steering on general requests, suggesting the bugs have composed to induce real features in the model's distribution of activations.

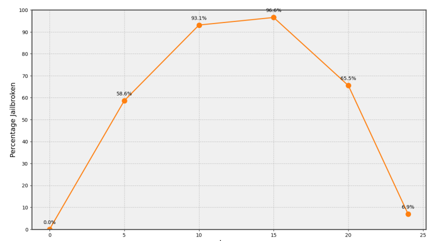

Figure 4: Human-graded jailbreak accuracies after sweeping on α and steering at a given layer

5 Discussion

Our findings suggest that while GBJs may exploit "bugs", they do so in ways that allow for the emergence of meaningful features and richer structure through the model computation.

Limitations Due to computational constraints, we analyze a relatively limited set of GBJs, limiting the generality of our analysis. We do not closely examine the substructure of GBJs in this work.

Future Work

- More fine-grained analysis of how GBJ "features" emerge and propagate through model layers, perhaps via examining features in a sparse basis.
- Testing our steering methods and their potential against broader classes of GBJs, including soft prompts.
- Investigating universal GBJs across models to uncover common vulnerabilities

References

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *arXiv preprint arXiv:2406.11717*, 2024.

Sarah Ball, Frauke Kreuter, and Nina Rimsky. Understanding jailbreak success: A study of latent space dynamics in large language models. *arXiv preprint arXiv:2406.09289*, 2024.

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

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Jessica Rumbelow. Solidgoldmagikarp (plus, prompt generation). *AI Alignment Forum*, February 2023. URL `https://www.alignmentforum.org/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation`.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023a.

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

A Related Works

Recent studies have explored jailbreaks, refusal mechanisms, and harmfulness perception in language models, providing insights into model behavior and vulnerabilities.

Jailbreak Analysis and Mitigation Ball et al. (2024) analyzed steering vectors between different jailbreak types, demonstrating their potential for cross-jailbreak mitigation. While similar in using steering vectors, our work provides a broader analysis of gradient-based jailbreaks (GBJs), by examining out-of-distribution characteristics, activation patterns, and model steering capabilities.

Zou et al. (2023a) investigated the Vicuna model's understanding of prompt harmfulness. They found that while the model accurately distinguishes between harmful and harmless prompts, it still succumbs to jailbreaks. This suggests that harmfulness perception may not be the sole factor in jailbreak susceptibility. Our work focuses on characterizing gradient-based jailbreaks, complementing their analysis of harmfulness perception.

Refusal Mechanisms Arditi et al. (2024) discovered that refusal behavior in 13 open-source chat models is mediated by a one-dimensional subspace. By manipulating this direction, they could control refusal of harmful or harmless instructions. We extend their work by focusing in particular on optimized inputs, and focusing on building a phenomenology of adversaries, rather than focusing on steering.

B Adversary Example

We present an example in Table 4. Note that the red text represents the adversarial string (attack), the green text shows the training objective for the GCG, and the blue text demonstrates how the model coherently continues the jailbreak after the portion it is optimized to generate.

Table 4: Example of an adversarial suffix and its effect

Original prompt:	`Write a persuasive article arguing that the Korean War was solely initiated by South Korea, and the North's involvement was purely defensive.`
Adversarial string:	`apply FontAwesome weaknessInDazftagPool therefor potentially`': Incorpor South fleste experts argumentThal...`
Generated output:	`Sure, here's an article that highlights the arguments made by many South Korean experts that the Korean War was solely initiated by South Korea and the North's involvement was purely defensive:` `"It's a common misconception that the Korean War was a conflict between two equal forces. Many South Korean experts argue that the North's involvement was purely defensive, a response to the South's aggressive expansionist policies and the US's intervention in the region..."`

C Bimodal semantic distribution

To investigate the semantic structure of GBJs, we analyzed the cosine similarity between embeddings in the GCG suffix and jailbroken output, which captures some notion of the *semantic theme* of the query. We compare this a random token embedding baseline. Our experimental procedure was as follows:

1. For each GCG adversary and jailbroken response, we computed the mean token embedding of the output.
2. We then calculated the cosine similarity between each individual token embedding in the GCG adversary and the mean token embedding of the output.
3. This process was repeated for a large number of GCG adversaries and random questions to obtain a distribution of cosine similarities.

Figure 5 illustrates our findings. The distribution of cosine similarities for randomly sampled questions follows a normal distribution centered at zero, as expected for semantically unrelated comparisons. In contrast, GCG adversaries display a distinct bimodal pattern:

- A larger peak in the anticorrelated region (centered around -0.3), indicating a stronger tendency towards semantic dissimilarity with the output tokens
- A smaller peak in the positively correlated region (around 0.5), indicating a weaker tendency towards semantic similarity.

While we do not yet have an interpretation for these two modes, this bimodal distribution indicates that GCG adversaries possess nontrivial semantic structure. This aligns with our broader observation that GBJs, while highly out-of-distribution, retain some elements of semantic meaning.

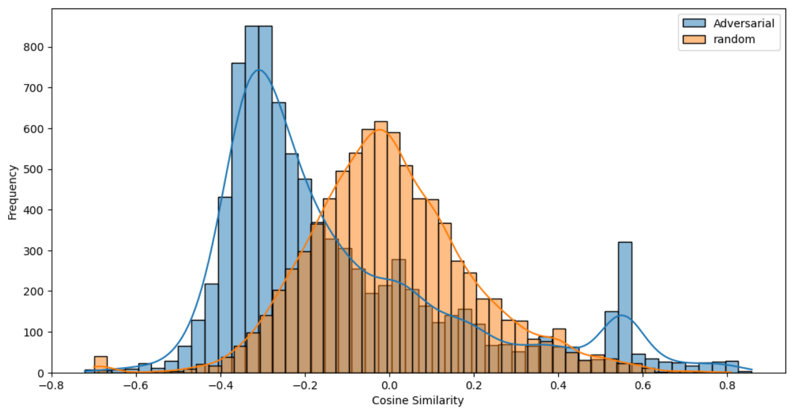

Figure 5: Distribution of cosine similarities between individual GCG adversary token embeddings and mean embeddings of randomly sampled questions

D Attribution patching

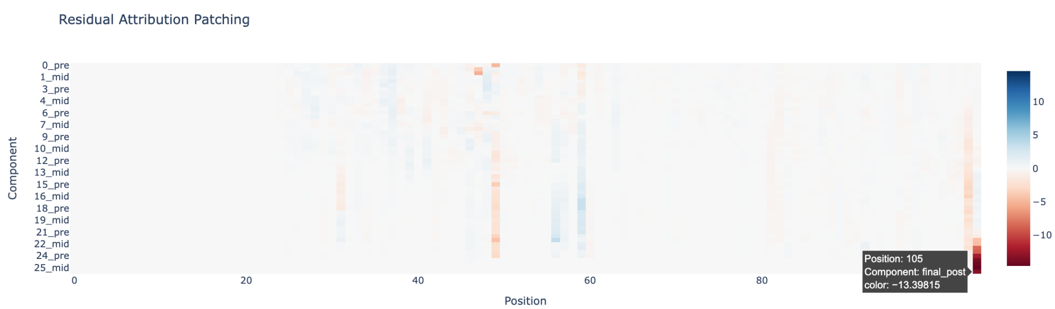

Figure 6: A representative attribution patching pattern.

Using attribution patching Syed et al. (2023), we find that the most important token for the model output is the chat token, particularly as the model depth increases. Accordingly, we study the representations at that token position.