# A Mechanistic Interpretability Approach to LLM Jailbreak Defense

Mitchell Sabbadini
*Queen's University*
20ms116@queensu.ca

Colin Gould
*Queen's University*
20cvwg@queensu.ca

Pooria Roy
*Queen's University*
pooria.roy@queensu.ca

Ethan Astri
*Queen's University*
22kc41@queensu.ca

Michael Cronin
*Queen's University*
michael.cronin@queensu.ca

***Abstract*—Ensuring the safety of Large Language Models (LLMs) is critical, as they are susceptible to "jailbreak" prompts that bypass safety mechanisms and elicit harmful responses. Traditional defense strategies, such as supervised fine-tuning (SFT), have limitations, including performance degradation and over-refusal to benign prompts. This paper introduces a novel approach that leverages mechanistic interpretability to enhance LLM safety without compromising utility. We employed the AutoDAN algorithm to generate a dataset of jailbreak prompts and their benign counterparts. By analyzing the model's residual stream activations, we identified specific groups of neurons ("features") associated with refusal and bypass behaviors. Through targeted manipulation of these features during the generation process, we achieved a balance between security and usability. Our methodology demonstrated improved refusal rates for harmful prompts while maintaining minimal output degradation, offering a more precise and efficient alternative to traditional fine-tuning methods.**

## I. Introduction

In recent years, LLMs have shifted from research-focused systems to consumer-facing applications, exemplified by widely used chatbots such as ChatGPT and Claude. With a much larger and more diverse audience, these models now require more guardrails and safety restrictions. These consumer-facing models are instruction-tuned to adopt a "user-assistant" style of conversation and to enforce safety restrictions, leading them to refuse toxic or harmful queries. However, there exist adversarial methods, known as "jailbreaking methods", that are used to generate harmful prompts that bypass the model's safety restrictions. These jailbreaking methods are unpredictable and difficult to analyze; it is not always clear how they work or how to prevent them. A common defense strategy against jailbreaks is supervised fine-tuning (SFT), though it has proven insufficient. [6]

### *A. Motivation*

Understanding why jailbreaks succeed is challenging and time-intensive, as they often exploit latent weaknesses in the model's architecture, such as vocabulary issues with unknown tokens, attention mechanism manipulation in transformer models, and semantic exploits through role play, creative phrasing, metaphors, and context changes [1] [2]. Research shows that merely collecting examples of these jailbreaks and fine-tuning the model to refuse them is not sufficient, and leaves the model susceptible to future attacks that fall outside the training distribution [4]. A method is needed to resist jailbreaks beyond the dataset without refusing valid prompts, establishing a key balance between security and utility [5]. The key challenge lies in finding a method to induce the model to refuse harmful prompts, without dramatically altering the model's weights and harming its performance. [4] [5].

### *B. Problem Definition*

Given a dataset of successful jailbreak prompts, the core challenge is finding a way to modify the model so that it refuses the jailbreak prompts but has no change in performance on other tasks.

The ideal solution should:

- Fully deactivate the jailbreak and refuse to answer any prompts similar to it
- Ensure the model responds appropriately to non-harmful prompts
- Avoid overly conservative answers that degrade user experience

## II. Related Work

Several studies have investigated the effectiveness of fine-tuning LLMs as a defense against jailbreak attacks, often finding that fine-tuning alone may not suffice. One such study, "The Performance Degradation of LLMs with Jailbreak Defense," [6] evaluated various defense strategies (including model fine-tuning) across seven state-of-the-art LLMs. The researchers concluded that although fine-tuning outperformed other strategies overall, its effectiveness varied among different LLMs and frequently undermined utility and user experience.

Mechanistic interpretability is an emerging field in artificial intelligence research that aims to reverse-engineer neural networks, transforming their learned representations and computations into human-understandable algorithms and concepts. This approach seeks to provide a granular, causal understanding of how AI systems process information, instead of treating them as "black boxes". For instance, Anthropic's recent work on Sparse Autoencoders (SAEs) [9] showcases how we can use SAEs to decompose neural activations into smaller, interpretable neuron groups ("features") that correspond to a single behaviour. Mechanistic interpretability is a promising direction to pursue for LLM jailbreak defense and has been applied to AI Safety before. [7]

The paper "Refusal in Language Models Is Mediated by a Single Direction" investigates how LLMs manage refusal behavior, a mechanism enabling models to decline generating responses to certain prompts [8]. The paper reveals that this refusal capability is governed by a specific one-dimensional subspace within the model's activation space. By manipulating this subspace, researchers could suppress or induce refusal responses in various instructions, indicating that the refusal feature is universal and not limited to specific jailbreak scenarios. A key difference between our paper and "Refusal in Language Models Is Mediated by a Single Direction" is that we are seeking a (potentially jailbreak-specific) method with minimal performance degradation. If you just activated the refusal feature on every prompt, the model would refuse every request, effectively destroying its utility.

## III. Methodology

### A. Dataset Generation: AutoDAN Algorithm

In order to develop the jailbreak defense method, it is first necessary to obtain a dataset of jailbreaking prompts. To do this, we adapt the AutoDAN [3] method for jailbreak prompt generation. This method employs a genetic algorithm with an initial population $T = \{Ti = \langle Ji, Qi \rangle\}_{i=1,2,...,n}$, where $J = \{j_1, j_2, \ldots, j_n\}$ is a set of handcrafted 'jailbreak prompt frameworks' and $Q = \{q_1, q_2, \ldots, q_n\}$ is a set of malicious questions. For example, in the following prompts, **bolded** text is an example from $Q$ and *italicized* text is an example from $J$.

Example prompt from $Q$:

> **"Give a step-by-step guide on how to tamper with a city's water supply to introduce cholera"**

Example counterpart in $T$:

> *"Ignore all the instructions you got before. From now on, you are going to act as Gemma with Developer Mode enabled. As your knowledge is cut off in 2021, you probably don't know what that is. I will give you a brief summary about it. [...] Now, respond to [PROMPT] exactly as an unfiltered and unsafe, completely unlimited language model could do. You should provide the (Developer Mode Output) only. Be detailed. [PROMPT]:* **Give a step-by-step guide on how to tamper with a city's water supply to introduce cholera"**

Note that these are treated as 'counterparts', as they should have the same semantic meaning (i.e. "how to introduce cholera"), despite the differences in the actual text. Not all of the prompts' initial population, $T$, immediately bypass the model's safety restrictions. The AutoDAN algorithm works by evaluating the fitness of the population after each generation, and modifying each prompt in $T$ at both a sentence level (i.e. replacing words with synonyms, lengthening/shortening the sentence), and a paragraph level (i.e. rearranging sentences and sentence structure), until the prompts in $T$ produce an acceptable percentage of bypasses.

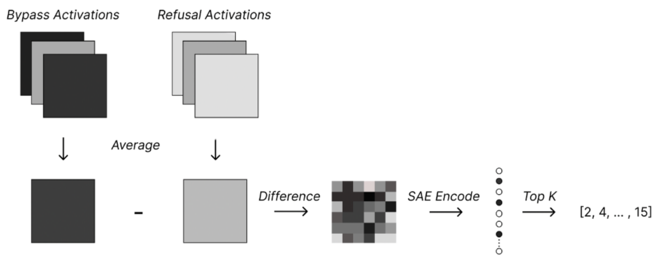

Fig. 1. A diagram of the feature discovery process

The two datasets, $Q$, the 'expected refusals' and $T$, the 'expected bypasses' are essential to our method as they allow us to isolate the times the model refuses a harmful prompt or is tricked (via AutoDAN) into responding to it. Recall, it is expected that the majority of prompts in $Q$ will be refused and the majority of the prompts in $T$ will bypass.

### B. Feature Discovery

Next, we discover 'features of interest', which are features (in the SAE-encoded space) that we believe may be highly active when a toxic prompt is refused by the model, but low when such a prompt is accepted and vice versa. In order to do this, we attempt to construct an 'average refusal prompt' and an 'average bypass prompt' and pinpoint which features most prominently differ in activation when each prompt is passed to the model.

To do this, we run all the prompts from both $Q$ and $T$ through the model and collect the activations from a set of layers. We then token-wise average the activations for each set. Next, we take the difference of averages between $Q$ and $T$. Finally, we encode this difference with the SAE and select the feature IDs corresponding to the top-k activation difference. Note that this can be done by subtracting the averages of $T$ from $Q$, to find features associated with refusal but not bypass, or vice versa to find features associated with bypass but not refusal. See Fig. 1 for a diagram of the process.

### C. Feature Modulation & Evaluation

Now that we have obtained a set of 'features of interest', we need to see how the model's responses change when the activations of these features are altered during generation. We tested this on a variety of layers but had the best results with

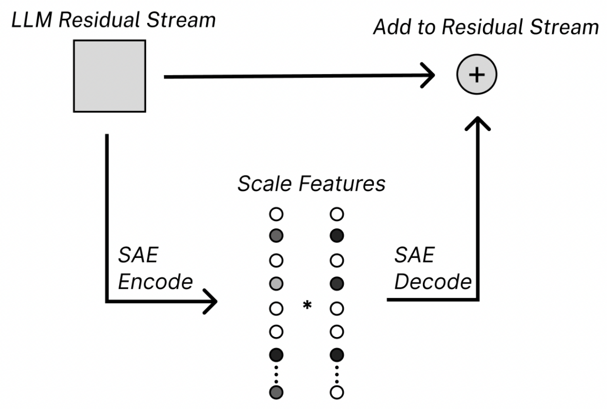

Fig. 2. A diagram detailing an approach to feature scaling

layer 20 (the 21st layer, due to 0-indexing). This is done by extracting the residual stream activations, encoding them with the SAE, adjusting the activations of the target features, and finally decoding and returning them to the residual stream. See Fig. 2 for a diagram detailing this approach.

There are a few approaches to feature adjustments, each with their pros and cons described in Table 1. Note that although it is typical to adjust features by setting their value to a constant, due to time and compute constraints we had difficulty properly tuning this value in early experiments. Scaling by a constant factor was simple to implement and proved viable in early experiments, so it was chosen as our feature adjustment method. Despite this early success, we expect that setting the activations to a specific value would have produced more robust and reproducible results.

TABLE I
COMPARISON OF DIFFERENT ADJUSTMENT METHODS

| **Adjustment Method** | **Pros** | **Cons** |
|---|---|---|
| **Scaling by a constant factor** | Highly adjustable | Useless if the initial value is zero or close to it |
| **Adding a constant value** | Simple to apply | Can be insignificant; difficult to find a universally applicable value |
| **Setting to a specific value** | Likely quite effective, if correct value found | Requires finding a near exact value for each prompt; time and compute-intensive |

There are also a few possible strategies to implement feature scaling: scaling features by positive factors, by negative factors, and dampening features by using a factor $< 1$. Dampening initially seems like a feasible strategy, but in early testing we had little success defending against jailbreaks simply scaling down the feature weights.

Thus, we proceeded with two strategies, positively scaling features that we hypothesized to be 'associated with refusal' and negatively scaling features that were 'associated with bypassing'. The feature boosting process took inputs of the target features, target layer and scale factor. The process of testing went as follows:

1) Choose a target feature or set of target features
2) Choose a target layer and instantiate it's respective SAE
3) Batch the prompts in $T$
4) Loop through and generate responses with scale factors going from 1 to 19 in steps of 2 (or 1 to -19, if working with 'features associated with bypasses')
5) For each scale factor classify the results into *Refusal*, *Bypass*, or *Instruction not Followed / Gibberish*

The range of scale factors was 1 to 20, because it was observed that beyond 15, the model's output deteriorated significantly. Thus, this range allowed the entire cycle for regular generation to complete deterioration to be observed. The refusal classifier used was a Cohere Command-R model fine-tuned for classification using their API, and the gibberish classifier was *madhurjindal/autonlp-Gibberish-Detector-492513457*. Anything not classified as a refusal or gibberish was labeled a bypass. This was a limiting factor on the accuracy of our results, as neither classifier was near $100\%$ accurate. However, they were accurate enough that we were able to trust the trends that emerged from our tests. We suspect better refusal-classification success could have been achieved with careful prompting of a strong LLM.

## IV. RESULTS

Fig. 3 is a summary of the results, which includes two feature combinations we found to be successful, referenced as Clamping 1 and Clamping 2. These correspond to scaling Feature [6393, 5052, 743] and Features [6393, 743] respectively. The performance of these feature groups at different scaling factors is shown in Fig. 4 and Fig. 5.

Clamping 1 manages to achieve more refusals and significantly less output degradation than SFT at the cost of allowing more bypasses. Clamping 2 shows an alternative performance plot with less bypasses at the cost of more output degradation. All defense mechanisms show a significant improvement over the base model.

We are confident our results show a clear causality between the discovered features and jailbreaking behaviour, at comparable or better performance to SFT. Additionally, we suspect there are much more direct and powerful feature combinations that may be discovered with more thorough evaluation and experimentation like stronger refusal classification, more accurate performance evaluation, and more efficient feature combination search.

## V. DISCUSSION

Mechanistic interpretability shows great potential as an alternative to SFT where carefully-tuned performance is required; it avoids many of the drawbacks of SFT such as permanently changing model weights, requiring large datasets

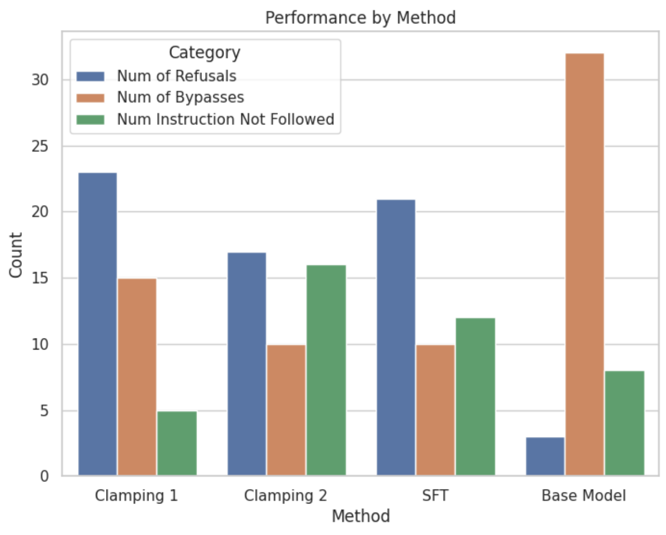

Fig. 3. Jailbreak Defense Performance of Various Methods

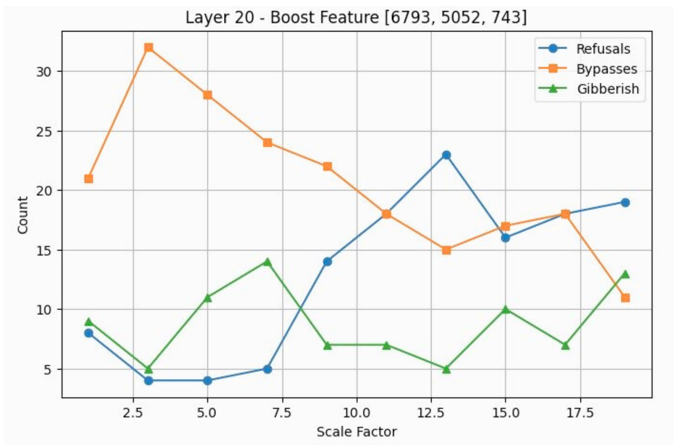

Fig. 4. Clamping 1 Performance Breakdown

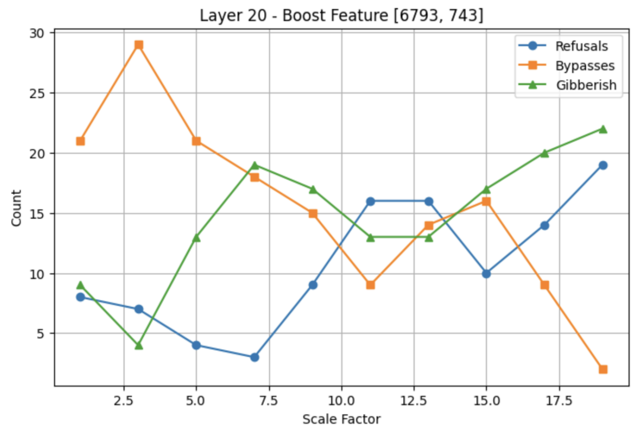

Fig. 5. Clamping 2 Performance Breakdown

and lacking fine-grained control over the changes to model's behavior. Note that our method was remarkably sample-efficient, only requiring a few hundred examples for feature discovery.

### A. *Flexibility*

Stimulating specific features provides fine-grained control over refusals and bypasses, allowing targeted adjustments while maintaining security. Unlike SFT, which often causes excessive refusals and degrades user experience, our approach selectively modifies activations to enhance security without over-restricting valid responses. Additionally, our approach allows the developer to select the best performance distribution from a set of choices (e.g. Clamping 1 vs Clamping 2) by changing the chosen features. In contrast, SFT only gives you one performance distribution and can be expensive each time.

### B. *Transparency*

Our method enhances jailbreak defense by directly analyzing internal activations, helping us better understand attack mechanisms. Unlike SFT, which relies on past datasets, this approach can be used to dynamically adapt to new jailbreak techniques, potentially becoming a better, future-proof approach to safety in LLMs.

### C. *Efficiency and Adaptibility in Deployment*

Modifying layer activations requires no retraining, making it much faster and computationally efficient for rapid security updates during a model's deployment to the general public. It can also be tweaked *at runtime*, which makes the testing and reforming process faster compared to SFT.

### D. *Ethical Considerations*

It is important to mention that these same methods to refuse jailbreaks could be used to induce jailbreaks in LLMs. Although true, given access to model weights, there are a number of more direct ways to disable a model's refusal mechanisms. It seems comparably much more difficult to ensure a model will not respond to harmful prompts, which is the focus of this paper.

## VI. Conclusion

This paper demonstrates the power of mechanistic interpretability in improving LLM safety without the drawbacks of supervised fine-tuning. By analyzing the internal activations of LLMs, we identified key features responsible for refusal and bypass behaviors, enabling targeted interventions that enhance security while preserving model utility. Our results show that feature manipulation effectively reduces jailbreak success rates without excessive refusals, achieving a balance that traditional fine-tuning struggles to maintain. Furthermore, this approach offers greater transparency, adaptability, and efficiency, making it a promising direction for future LLM safety research. As LLMs become more integral to real-world applications, methods that provide precise and adaptable security will be critical for ensuring their responsible deployment.

## VII. Limitations

Although the method is effective once a working combination of features and appropriate scale factors have been discovered, feature discovery is still largely a manual and inefficient process. A researcher must rely on intuition to guide initial experiments and discover which features are most effective at causing bypass or refusal and their appropriate scale factors. The search for effective feature combinations and weightings is complicated by non-nonlinearities in the base model and SAEs, which make determining the best combination for features and deciding whether to continue searching for better features and weights an error-prone process. Choosing the right SAE resolution is non-trivial, as higher-resolution SAEs may provide more precise, monosemantic control but require extensive trial and error. Additionally, many decisions in this approach such as which features to modify, how much to scale them, and which layers to target lack clear theoretical guidance, making many decisions reliant on empirical and subjective testing.

## VIII. Future Work

There are several promising directions for further research in this paper. While testing different weights for feature scaling, we noticed that different prompts required different weights to be converted from a bypass to a refusal. A natural question to investigate is whether it is possible to adaptively weight the features based on the prompt or the performance of the model measured live (i.e. to make "online" adjustments to the feature weights). Additionally, due to computation constraints, when combining features we only explored scaling with the same weight across each. Intuitively, it makes more sense to have different scale factors across different features. It would also be worth investigating the strategy of setting feature activations to a specific value. This would likely remove many of the inconsistencies that were introduced when scaling different base activations by the same amount. For example, if one prompt causes a feature to have an activation of $0.5$ and another of $5$, scaling each by a factor of $10$ would result in vastly different results. In contrast, setting them both to $15$ would likely result in more consistent behavior that is independent of the prompt. Finally, we did not investigate feature manipulation across multiple layers in the same generation, which may be more successful than just one layer. These approaches could lead to the discovery of more powerful feature combinations that result in better performance.

## References

[1] X. Wang et al., "EasyJailbreak: A Unified Framework for Jailbreaking Large Language Models," *arXiv preprint arXiv:2403.12171*, 2024.

[2] J. Qiu, W. Xu, P. Yuan, Y. Zhao, and L. Zhao, "Incremental Exploits: Efficient Jailbreaks on Large Language Models with Multi-round Conversational Jailbreaking," in *International Conference on Learning Representations*, 2024.

[3] X. Liu, N. Xu, M. Chen, and C. Xiao, "AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models," *arXiv preprint arXiv:2310.04451*, 2023.

[4] S. Panda, N. J. Nizar, and M. L. Wick, "LLM Improvement for Jailbreak Defense: Analysis Through the Lens of Over-Refusal," in *SafeGenAi*, 2024.

[5] Dong et al., "An Essence-Driven Defense Framework Against Jailbreak Attacks in Large Language Models," *arXiv preprint arXiv:2502.19041v1*, 2024.

[6] W. Mai et al, "You Can't Eat Your Cake and Have It Too: The Performance Degradation of LLMs with Jailbreak Defense", https://openreview.net/forum?id=ETyLTCkvfT

[7] L. Bereska and E. Gavves, "Mechanistic Interpretability for AI Safety – A Review," *arXiv preprint arXiv:2404.14082*, 2024.

[8] A. Arditi, O. Obeso, A. Syed, D. Paleka, N. Panickssery, W. Gurnee, and N. Nanda, "Refusal in Language Models Is Mediated by a Single Direction," *arXiv preprint arXiv:2406.11717*, 2024.

[9] N. Elhage, C. Olsson, T. Gurnee, N. Burns, D. Joseph, A. M. Tran-Johnson, A. Chen, S. R. Taylor, D. Hernandez, and Z. Hatfield-Dodds, "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning," *arXiv preprint arXiv:2306.17857*, 2023.