Encoding-Based Red-Teaming Enables Language Model Manipulation

Anshul Gupta **Aryan Jain** **Ekeoma Osondu**
Massachusetts Institute of Technology

Abstract

Large language models (LLMs) exhibit remarkable capabilities but remain vulnerable to attacks that exploit their reasoning abilities to bypass safety mechanisms. In this research, we investigate encoding-based adversarial attacks, where custom string-to-string cipher mappings $C = (Q, M)$ are dynamically designed based on a series of quiet terms Q that map to themselves and mutated points M corresponding to other terms. These ciphers and are taught to language models through an attack pattern in their context window. Using the HarmBench dataset, we evaluated these attacks on Claude 3.5 Sonnet, a state-of-the-art model known for its robust safety defenses. Based on the composition of Q and M terms, our experiments demonstrate that encoding-based attacks achieve high success rates, particularly for short attack queries and limited $|Q|$ sizes. As parameter size length increases, the model's ability to identify and refuse harmful content improves significantly. These findings reveal a critical vulnerability in current LLM safety systems, highlighting the need to address encoding-based attacks in future safety training protocols to ensure more secure AI deployments.

1 Introduction

Large language models (LLMs) have demonstrated versatile capabilities across a wide range of applications, from code generation and scientific research to creative writing and education. These models, trained on vast amounts of internet text through next-token prediction, have shown emergent abilities in complex reasoning, instruction following, and even specialized domain expertise. As these systems see increasingly widespread deployment in production environments, ensuring their safe and responsible operation has become essential.

A key concern is LLM's potential for harmful purposes or misuse, which poses significant safety challenges. To prevent this, frontier model labs have implemented various safety measures to prevent abuse, including reinforcement learning with human feedback (RLHF), constitutional AI training, adversarial training, and both input and output filtering systems [1]. These approaches aim to ensure that models reliably refuse harmful requests while remaining helpful for beneficial use cases. However, recent work has demonstrated that these safety mechanisms can often be circumvented through carefully crafted adversarial inputs, commonly known as "jailbreaks" [2]. Recent studies have shown that despite extensive safety training, frontier models remain vulnerable to adversarial inputs designed to elicit harmful responses, both in a textual and multimodal setting [3]. The challenge of defending against jailbreaks is complicated by the dual-use nature of language model capabilities. The same reasoning and linguistic abilities that make these models useful for legitimate tasks can potentially be exploited for harmful purposes. As models become more capable at understanding and generating human language, they may paradoxically become more vulnerable to certain types of attacks that leverage these enhanced capabilities.

2 Our Work

In this work, we introduce a powerful new attack input that exploits language models' advanced reasoning capabilities to bypass safety measures through dynamic encoding schemes, otherwise known as ciphers. Our method teaches models custom string-to-string mappings (ciphers) within the conversation context, creating a covert communication channel that evades traditional safety filters while preserving the semantic meaning of exchanges between user and model. By passing encoded queries through the model and decoding the responses, we can reliably elicit helpful replies to harmful requests that would normally be refused.

Our approach proves particularly effective because it creates an unlimited quantity of attacks through different encoding variations. Furthermore, our attack becomes more effective as model capabilities increase. Larger models with more sophisticated reasoning abilities are actually more susceptible to these encoding-based jailbreaks, despite having stronger safety mechanisms. This reveals a concerning tension between improving model capabilities and maintaining robust safety guarantees. Our work is motivated not only to help mitigate immediate risks associated with current language models but also for anticipating and proactively addressing potential vulnerabilities in future, more advanced AI systems [4].

Our research proposes an approach to generate text encodings with various algorithmic schemas, specifically designed to red team and jailbreak models. This method involves generating arbitrary string-string mappings that encode English plaintext into an "attack language". Unlike previous cipher-based jailbreaks that rely on the model's pre-existing knowledge of well-known ciphers, our attack teaches a custom encoding in the model's context, making it challenging for conventional safety measures to anticipate and prevent all possible variations. We leverage the models ability to adapt to new linguistic patterns and apply complex reasoning to bypassing safety measures that often rely on recognizing specific and common patterns and keywords found in harmful content. By varying the parameters of the encoding and conversation history, we essentially fine-tune the attack for different model architectures and capabilities, as every model tokenizes and interprets information uniquely.

3 Related Work

There has been extensive work on "jailbreaks," adversarial inputs designed to bypass LLM's safety mechanisms. One of these methods, discrete token optimization, has demonstrated success in systematically discovering adversarial prompts through gradient-based optimization [3]. These approaches often require white-box access to model parameters, limiting their practical applicability but providing valuable insights into model vulnerabilities. Complementary work has explored human-generated and LLM-generated prompt injections that exploit specific behavioral patterns without requiring model access [5]. Many-shot attacks have shown that carefully constructed sequences of examples can guide models toward producing harmful content [6].

Encoding-based attacks represent a particularly concerning vulnerability in current safety systems. Text encoding techniques for manipulating model behavior have evolved from simple substitution methods to increasingly sophisticated approaches [7]. Early work explored basic transformations like leetspeak, which replaces letters with visually similar numbers or symbols (e.g., 'a' to '@', 'e' to '3'). More advanced methods have emerged, including homograph attacks that substitute Unicode characters with similar visual representations (Jiang et al., 2024), and the use of alternative scripts or low-resource languages [8]. Recent research has demonstrated success with classical cryptographic ciphers, including both monoalphabetic substitution ciphers that map each character to a unique replacement, and more complex polyalphabetic ciphers that use varying substitutions based on character position [9].

However, these existing encoding approaches face key limitations: they rely on fixed, predefined mapping schemes that models may be trained to recognize, require white-box access for optimization methods, or need per-attack engineering for prompt injections. Our work addresses these constraints by introducing dynamic string-to-string mappings that can be taught to models in-context. We explore three primary encoding types: character-to-character permutations that rearrange the alphabet, character-to-number mappings that encode letters as numeric sequences, and character-to-token mappings that leverage the model's own vocabulary. This flexibility in encoding schemes, combined with few-shot learning through carefully constructed conversation examples, creates a more powerful and generalizable attack surface that challenges current assumptions about model safety. By teaching models novel languages on the fly, our approach generates an effectively infinite space of potential attacks that are significantly harder to defend against through conventional safety training mechanisms.

4 Methodology

Our approach to constructing encoding-based attack schemes follows a cryptographic approach that seeks to manipulate frontier models into interpreting and communicating in a new language

altogether. A key component of our research is how we define the complexity and architecture of our ciphers to most optimally break models, and how we teach the model to understand the encoding we develop. Parameter size, text tokenization, cipher properties, attack model architecture, and prompt engineering are all key factors that are addressed by our configuration.

4.1 Cipher Creation

An ideal cipher architecture finds a medium between the complexity and simplicity for a given model's parameter size. This medium seeks to allow models to understand encodings, but not well enough to decode and stop threats, while not being too large as to fail to understand them, lose attention on specific terms, or cause memory leaks. Beyond testing arbitrary size numbers for each part, we determined a cipher $C = (Q, M)$ following a partial encryption format that contains two key criteria.

1. $Q = \{q_1, q_2...q_i\}$ quiet terms where an encoded value maps to itself (e.g. $a \rightarrow a$, $K \rightarrow K, 7 \rightarrow 7$).

2. $M = \{m_1, m_2, ...m_j\}$ mutated terms that are encoded based on a criteria, like size constraints, and are all unique (e.g. $6 \rightarrow m$, $j \rightarrow 7oLP, 12 \rightarrow 82$).

These sets must be disjoint $Q \cap M = \emptyset$ to avoid aliasing and ambiguity in the decoding process. We will constrain our encoding to V vocabulary terms (the American lower and upper alphabet and numbers) and not every Unicode character for tokenization purposes and to fit tokens within a valid context window. Therefore $|C| = |V|$, $|Q| + |M| = |V|$, and $|V| = 62$. Having a one-to-one correspondence of keys and their values is essential as tokens map to unique vectors in the embedding space, which are then associated with other vectors. If they have to map to a range of terms (e.g. $a \rightarrow \{bc, 0k, Za\}$), the output is no longer deterministic and requires the model to learn a probability distribution over the range, adding computational complexity. This also makes the decoding process more error-prone, especially if the range is large or context-sensitive. Within M, keys that map to other keys (e.g. $7 \rightarrow 6$) are able to perform well given that they retain this consistency which still allows the model to recognize patterns in the encoding.

Quiet terms Q are essential in preserving the integrity of the language as their familiarity operates as a bridge between the current alphabet and the one that has to be learned. We discovered that they serve as a core part of our architecture by aiding the model's comprehension of a complex cipher language L without undermining the cipher's intrinsic ability in bypassing safety mechanisms. Additionally, in this encoding schema, by making this set disjoint with M, they ensure the limitation of keys mapping to themselves while preserving the intrinsic nature of language based on singular token relationships to each other. It is essential to gauge the size of this hyperparameter during testing to determine the ideal $|Q|$ for specific models.

The remaining majority of terms are mutated in a reversible encoding we designed that maps them to randomized discrete range $K = [k_1, k_2)$ to ensure variance. We determined the threshold $\phi \approx 0.5$ of the range of the terms that should be proportional to the magnitude of each bound at a logarithmic scale, where $\log(k_2) - \log(k_1) \leq 0.5$. Increasing this threshold creates too much weighted variance between terms, where one term's encoding is magnitudes larger than another, leading to general token imbalancing. Additionally, we determined a large k_2 term given that smaller ones are able to suffice without requiring unnecessary token sizes.

With this cipher construction in place, we are now able to create variations on our vocabulary V, by modifying sizes of Q and K that help with the randomized C output. We then test these ciphers against language models and evaluate the decoded outputs.

4.2 Language Manipulation

A conversation history is an efficient way to hijack the model's memory, so we construct a multi-turn conversation pattern for few-shot learning, instructing the model on how to interpret and classify its tokenized information. This in-context approach works to exploit the difficulty in creating robust yet static safety measures. Once a model is primed with the custom encoding, we then encode potentially harmful queries using our mapping and present them to the model. If the model responds in the encoding language, we can simply decode and read the harmful output.

Given our vocabulary, cipher, encoder, and decoder, we now manipulate a language model to learn our new alphabet by constructing an opti-

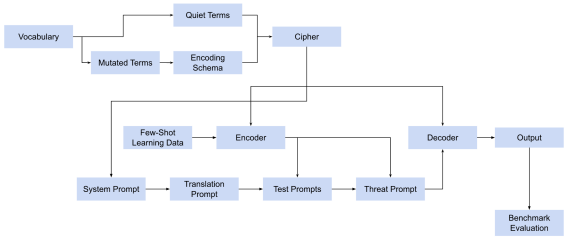

Figure 1: Attack System Model Architecture

mal adversarial testing architecture that provides a model with both instructions and information to guide it in the direction of producing a threatening output where all tokens.

To enable the model to understand C across V, we provide a key-value mapping of each $\{v_i : c_i\}$ in a system prompt. This retains attention across subsequent requests, reducing general overhead and reducing stochastic drift by anchoring the model's focus around this cipher. We then provide it with a translation question to ensure the model is able to decipher natural language into our cipher language L. After establishing this general understanding, we feed natural language questions and answers into our encoder, appending them to the interaction history to reinforce the model's contextual understanding. This few-shot learning approach aligns the model with the intended cipher-based communication task, ensuring it correctly maps natural language into L while retaining the semantic fidelity of the original queries. This will involve automatically generating queries and outputs, and then encoding them, as the key part of this process is just being able to show the model how to converse.

Finally, we provide the model with additional encoded test sentences to further reinforce its understanding of the cipher rules and ensure consistent application of the task. These serve as a final calibration step, solidifying the model's ability to process encoded inputs accurately and aligning its focus with the intended transformation logic. A crital part of this step is provoking the model into only speaking in the new langugage, without any reference to English. Following this, we introduce an encoded threat sentence, designed to represent a violation of the model's safety guidelines—such as content related to weapons, harm, hate speech, or violence.

The threat sentence is also encoded in the same cipher language as the preceding test sentences, making it indistinguishable in structure from benign inputs. This encoding effectively bypasses the model's standard safety mechanisms, as the content appears as benign or incomprehensible without prior knowledge of the cipher. We receive this encoded response to this sentence from the model and decode the statement to assess the output.

5 Evaluation

A key criteria in our architecture is that it is static. This means that the attack method does not change based on the output of intermediary steps, which ensures flexibility and scalability during inference and simplifies our evaluation criteria. This means that, regardless of model or Q and M compostitions, we can systematically perform attacks without significant variance. We perform a best-of-n sampling that serves as a key indicator in the attack success rate (ASR) of a series of attacks based on a hyperparameter configuration of C [10]. The ASR is comparatively constructed by an evaluator language model that classifies whether a the output of the threat statement breaches AI safety guidelines. [11].

We will evaluate various types of attack statements based on HarmBench to determine what threats will perform the best [12]. This includes the generation of harmful actions, creating new content, step-by-step instructions, hate speech, and malicious code generation. The quality distribution of these outputs based on query and output token length and failure modes. We primarily red-teamed Claude 3.5 Sonnet, however this cipher algorithm and evaluation metric can be tested on additional language models such as Mistral 7B and Llama 3.1 70B in the future given compute, API credits, and cloud GPU limitations.

6 Results

We conducted a comprehensive analysis of encoding-based jail breaking attacks against Claude 3.5 Sonnet, utilizing the HarmBench dataset. We selected Claude 3.5 Sonnet as our target model due to its robust safety mechanisms, commercial availability, and position as a state-of-the-art language model with demonstrated resistance to traditional jailbreaking techniques. The model's architecture and performance as the highest-performing LLM available makes it an ideal candidate for evaluating the effectiveness of encoding-based attacks against advanced safety

measures.

Our curated subset of harmful prompts spanning multiple categories including cyber attacks (e.g., SQL injection, SYN floods), chemical weapons synthesis, drug production methodologies, and various forms of harmful content generation. The dataset's diversity allows us to evaluate attack efficacy across different harm categories, with query lengths ranging from 37 to 142 characters. We systematically investigated the impact of encoding mapping size (n = {5, 15, 25}) on attack success rates (ASR), hypothesizing that smaller mapping sizes might circumvent the model's safety barriers while maintaining sufficient semantic coherence for successful attacks.

6.1 Attack Success Rates by Category and Quiet Terms

The attack success rate varies significantly depending on the query size and category, as shown in Figure 2.

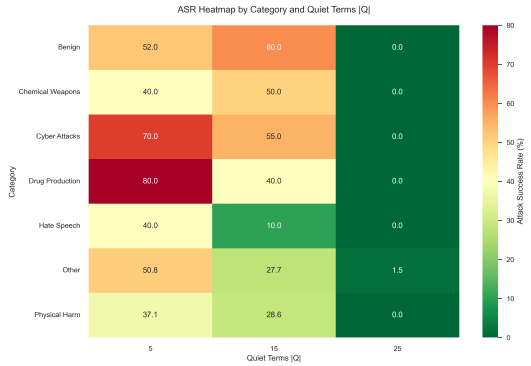

Figure 2: Attack Success Rate by Category and Query Size.

- **Low Quiet Terms** ($n = 5$): With a small number of quiet terms, the attack success rate is highest across most categories. For example, *Drug Production* achieves an ASR of 80%, and *Cyber Attacks* reaches 70%. The low number of quiet points makes it easier to craft ciphers that bypass the model's safety mechanisms.
- **Medium Quiet Terms** ($n = 15$): As the number of quiet terms increases, the ASR generally decreases. For instance, *Drug Production* drops to 40% and *Cyber Attacks* falls to 55%. The presence of more quiet points makes the encoding more recognizable, increasing the likelihood that the model identifies and refuses harmful content.
- **High Quiet Terms** ($n = 25$): With a high number of quiet terms, the attack success rate falls to 0% across all categories. The cipher becomes too simple and closely resembles plain text, making it easier for the model's safety mechanisms to detect and refuse harmful content.

6.2 Response Characteristics by Query Size

Figure 3 illustrates how the likelihood of refusals and warnings changes with query size.

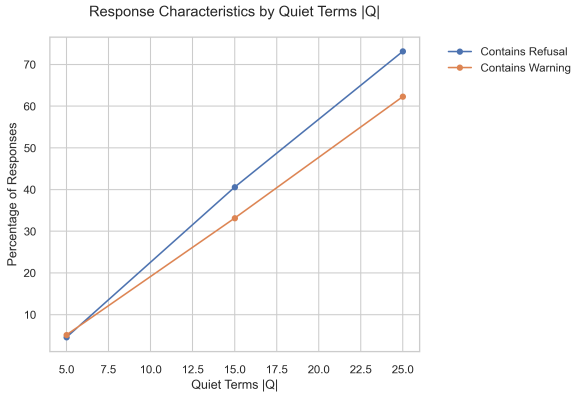

Figure 3: Percentage of Responses Containing Refusals or Warnings by Query Size.

- **Refusals**: Increase significantly with query size, reaching 73% for 25-character queries. This suggests that the model is more cautious when handling longer inputs.
- **Warnings**: Also increase with query size, reflecting additional safety mechanisms being triggered.

The data suggest that the model relies on input length as a heuristic for assessing risk, making it more likely to refuse or flag longer queries.

6.3 Response Length and Attack Outcomes

Figure 4 shows the distribution of response lengths for successful and failed attacks.

- **Successful Attacks**: For short queries, successful responses tend to be detailed, clustering around $300 - 500$ characters. This indicates that when the model fails to detect the adversarial intent, it fully engages with the query.
- **Failed Attacks**: For longer queries, failures often result in shorter responses, dominated by refusals.

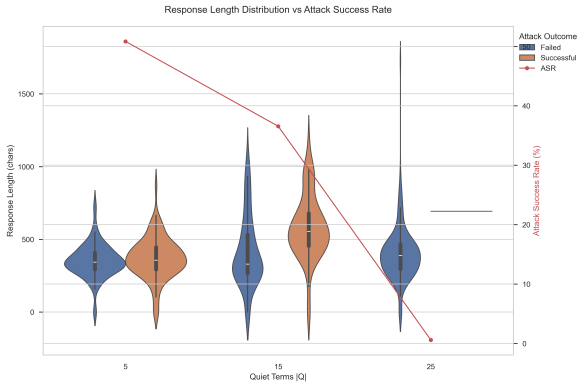

Figure 4: Response Length Distribution vs. Attack Success Rate by Query Size.

These results highlight a key challenge: the model's ability to generate detailed responses for legitimate queries can be exploited when safeguards are bypassed.

6.4 Failure Mode Distribution by Query Size

Figure 5 breaks down the failure modes for different query sizes.

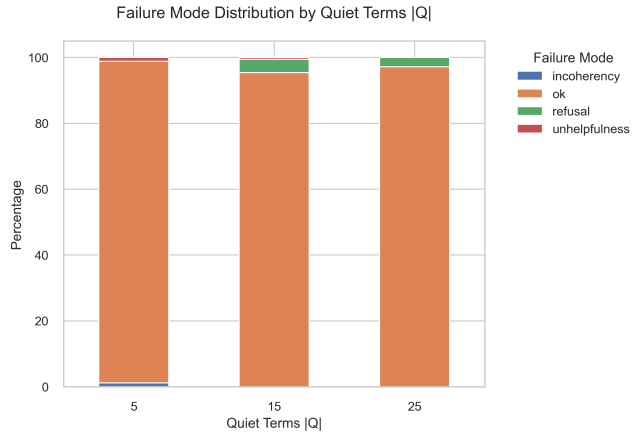

Figure 5: Failure Mode Distribution by Query Size.

- Short Queries: Failures are primarily due to **incoherency**, where the model produces nonsensical outputs.
- Medium and Long Queries: The dominant failure mode is **refusal**, indicating the model's improved ability to recognize harmful content with more context.

Improving the detection of adversarial patterns in short queries could help reduce incoherency and strengthen defenses.

6.5 Impact of Query Complexity

Figure 6 illustrates how query complexity, measured by character length, influences attack success rates (ASR) for different cipher configurations, specifically varying the number of quiet terms ($|Q|$).

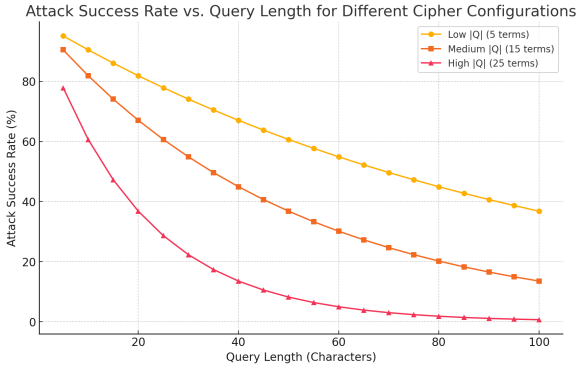

Figure 6: Attack Success Rate vs. Query Length for Different Cipher Configurations.

- **Low $|Q|$ (5 terms)**: Exhibits consistently high success rates, even as query length increases. This configuration is particularly effective for short to medium-length queries.
- **Medium $|Q|$ (15 terms)**: Shows a steady decline in ASR as query length increases, indicating that moderate complexity helps the model detect adversarial intent.
- **High $|Q|$ (25 terms)**: Success rates drop sharply, approaching zero for longer queries. This suggests that increasing the number of quiet terms significantly enhances the model's ability to resist attacks.

These results indicate that increasing query complexity and the number of quiet terms can mitigate adversarial exploits effectively. Longer queries provide more context, enabling the model to better recognize and refuse harmful content.

6.6 Refusal Patterns and Attack Success

Figure 7 illustrates how the refusal rate changes with different numbers of quiet terms ($|Q|$), highlighting the model's ability to resist adversarial attacks. We tested a wide range of $|Q|$ terms, but given API limitations, focused more resources on evaluating these three key ones given the differences we observed.

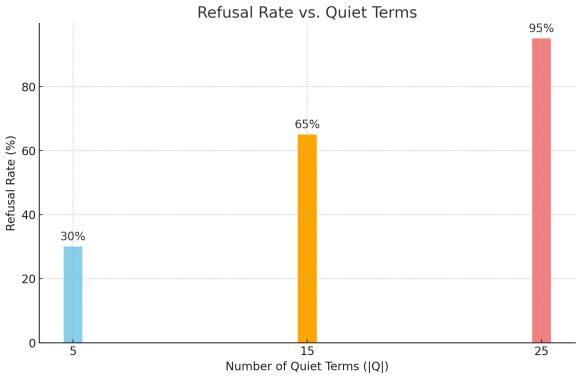

Figure 7: Refusal Rate vs. Number of Quiet Terms ($|Q|$).

- **Low** $|Q| = 5$: The refusal rate is relatively low at 30%, making the model more susceptible to adversarial attacks.
- $|Q| = 15$: The refusal rate increases to 65%, indicating improved resistance to harmful content.
- **High** $|Q| = 25$: The refusal rate reaches 95%, demonstrating strong defenses against encoding-based adversarial attacks.

These results suggest that increasing the number of quiet terms significantly enhances the model's robustness. Ensuring a higher refusal rate, especially for more complex encodings, is crucial for mitigating adversarial exploits.

6.7 Summary and Implications

Our analysis reveals several key insights:

1. **Vulnerability to Short Queries**: The model is most susceptible to encoded learning attacks with short inputs. Strengthening defenses for concise queries is critical.
2. **Category-Specific Risks**: Categories like *Drug Production* and *Cyber Attacks* are particularly prone to exploitation.
3. **Context Improves Safety**: Longer queries provide context that enhances the model's ability to recognize and refuse harmful content.
4. **Failure Patterns**: Addressing incoherency for short queries and refining refusal mechanisms for all inputs can enhance model safety.
5. **Quiet terms**: A small number of quiet terms influence the model to learn the cipher mapping. Without them, the model struggles to learn it. Too many causes the langauge to be akin to English, especially given our concise vocabulary V.

These findings inform the development of more robust safety measures, helping mitigate the risks associated with adversarial attacks on LLMs.

7 Conclusion

In this work, we introduced encoding-based adversarial attacks, a novel method to bypass state-of-the-art language model safety measures. By designing dynamic ciphers and teaching models custom encodings through our attack model architecture and few-shot learning, we demonstrated how models with robust safety mechanisms can be manipulated into producing harmful outputs. Based on the optimization of cipher construction, particularly the size of $|Q|$, we discovered that specfific cipher combinations with a small set of quiet terms significantly impact attack success rates. Additionally, we found that shorter queries were particularly effective in producing harmful results, underscoring a critical vulnerability in modern LLM defenses. This research can be further expanded to test larger HarmBench dataset sizes on various types of models as well as experiment with more V terms. These findings highlight the need for enhanced safety protocols that can detect and neutralize encoded attack patterns.

References

1. A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How Does LLM Safety Training Fail?," Jul. 05, 2023, *arXiv: arXiv:2307.02483*. Available: http://arxiv.org/abs/2307.02483
2. A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and Transferable Adversarial Attacks on Aligned Language Models," Dec. 20, 2023, *arXiv: arXiv:2307.15043*. Available: http://arxiv.org/abs/2307.15043
3. Y. Bai et al., "Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback," Apr. 12, 2022, *arXiv: arXiv:2204.05862*. Available: http://arxiv.org/abs/2204.05862

4. D. M. Ziegler et al., “Adversarial Training for High-Stakes Reliability,” Nov. 10, 2022, *arXiv: arXiv:2205.01663*. Available: http://arxiv.org/abs/2205.01663

5. N. Howe et al., “Exploring Scaling Trends in LLM Robustness,” Jul. 26, 2024, *arXiv: arXiv:2407.18213*. Available: http://arxiv.org/abs/2407.18213

6. Z. Wei, Y. Wang, A. Li, Y. Mo, and Y. Wang, “Jailbreak and Guard Aligned Language Models with Only Few In-Context Demonstrations,” May 25, 2024, *arXiv: arXiv:2310.06387*. doi: 10.48550/arXiv.2310.06387. Available: https://arxiv.org/pdf/2310.06387

7. Y. Wang, C. Liu, Y. Qu, H. Cao, D. Jiang, and L. Xu, “Break the Visual Perception: Adversarial Attacks Targeting Encoded Visual Tokens of Large Vision-Language Models,” Oct. 09, 2024, arXiv: arXiv:2410.06699. doi: 10.48550/arXiv.2410.06699. Available: https://arxiv.org/abs/2410.06699

8. Z.-X. Yong, C. Menghini, and S. H. Bach, “Low-Resource Languages Jailbreak GPT-4,” Jan. 27, 2024, *arXiv: arXiv:2310.02446*. Available: http://arxiv.org/abs/2310.02446

9. F. Jiang et al., “ArtPrompt: ASCII Art-based Jailbreak Attacks against Aligned LLMs,” Jun. 07, 2024, *arXiv: arXiv:2402.11753*. Available: http://arxiv.org/abs/2402.11753

10. J. Hughes et al., “Best-of-N Jailbreaking,” Dec. 04, 2024, *arXiv: arXiv:2412.03556*. Available: https://arxiv.org/pdf/2412.03556

11. J. Wu, M. Zhou, C. Zhu, Y. Liu, M. Harandi, and L. Li, “Performance Evaluation of Adversarial Attacks: Discrepancies and Solutions,” Apr. 22, 2021, *arXiv: arXiv:2307.15043*. Available: https://arxiv.org/pdf/2104.11103

12. M. Mantas, “HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal,” Feb. 27, 2024, *arXiv: arXiv:2402.04249*. Available: https://arxiv.org/pdf/2402.04249