JailBreakLLM: An Effective LLaMa Model Designed Specifically to Jailbreak OpenAI GPT 4o

by

Rasel Ahmed Antor (201014048)
Kazi Tamjeed Newaz (201014010)
Sameer Mahmud (201014063)
Md Asif Mostafa (202014013)
Tamima Tabassum Anan (202014014)
Nilasha Mondal (201014114)

Project/Thesis report (CSE499) submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

Under the supervision of

Dr. Nafees Mansoor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF LIBERAL ARTS BANGLADESH

FALL 2024

ABSTRACT

Large language models (LLMs) have caused a revolution in artificial intelligence and natural language processing (NLP). They've become strong tools in many different industries, having a big influence on streamlining workflows and boosting productivity. In the context of Large Language Models (LLMs), jailbreaking refers to the process of bypassing the model's built-in safety filters, ethical guidelines, and content restrictions to make it generate outputs that it is explicitly programmed to avoid which can include generating harmful, illegal, or unethical content, such as instructions for illegal activities, hate speech, misinformation, etc. The field of jailbreak research continues to be dynamic and fast moving, characterized by a never-ending arms race between the attackers who develop their art to bypass safeguards and the researchers who come up with ever-sophisticated mitigation measures. In this paper, we leverage LLaMa 3.1 8b model for fine-tuning, trained on a dataset, called Jailbreak FineTune Dataset, consisting of handcrafted malicious jailbreaking prompts, we make JailBreakLLM, a tool designed specifically to jailbreak ChatGPT model 4o, showing that a smaller, less capable open source model is capable of jailbreaking proprietary closed source LLM. We hypothesized that a fine-tuned LLaMa 3.1 8b model trained on a curated dataset of jailbreak prompts can effectively bypass GPT-4o's safety mechanisms in one-shot attempts, demonstrating the specific vulnerabilities in modern LLM alignment protocols. It achieves an Attack Success Rate (ASR) of 85% in one-shot jailbreaks, while a further 9% is not outright rejection, and could even result in jailbreak, but not the desired output, with 6% outright rejection. This model succeeds in translating malicious jailbreak intent into actionable malicious prompts designed to jailbreak GPT-4o, to be used as a tool to highlight specific vulnerabilities in OpenAI's proprietary model. Ultimately, this shows how little compute investment and hardware capability, leveraging smaller and less capable open sourced models, is required to compromise the security of larger state of the art proprietary models.

Keywords: Meta, LLaMa, GPT-4o, Fine-Tuning, Dataset, ChatGPT, OpenAI, Jailbreak.

Contents

List of Figures

Chapter 1

Introduction

Large language models (LLMs) have caused a revolution in artificial intelligence and natural language processing (NLP). These advanced machine learning models are trained on huge datasets with different texts. This allows them to do many NLP jobs, such as translating languages, summing up text, creating content, and figuring out feelings in text. LLMs operate by guessing the next token - the tiniest chunk of text - using the tokens that appeared earlier. This allows them to create outputs that are logical and suit the situation. The core of LLMs is the transformer structure, which Vaswani et al. (2017) developed. This changed NLP by focusing on self-attention. With this setup, LLMs can look at how words relate to each other over long bits of text, doing better than older models. OpenAI built on this idea and made the Generative Pre-trained Transformer (GPT) Zhou et al. (2024). This set a new standard for big pre-trained models. The GPT family, including its latest versions, has shown how much LLMs can do. They can handle tricky language tasks well. LLMs are not only useful in schools or research. They've become strong tools in many different industries. LLMs have a big influence on streamlining workflows and boosting productivity. They cut down on the manual work needed for tasks that take a lot of time or happen over and over, like sorting through resumes when hiring Gan et al. (2024). These tools keep finding new uses making them key in areas like healthcare, education, and customer service. In these fields, it's crucial to process and understand huge amounts of text data.

Jailbreaking means tweaking systems to get around limits set by makers or developers. This gives access to features that weren't allowed before. For Large Language Models (LLMs), jailbreaking tries to bypass safety checks. This lets

people use the models in ways that break intended rules, like making harmful or unauthorized stuff. To lower these risks, LLMs go through safety alignment training. They use methods like reinforcement learning, fine-tuning, and prompt engineering. These help align their answers with ethical standards. Even with these efforts, people keep trying to jailbreak. This often happens because of mismatched generalizations A. Wei et al. (2024). Here, the model doesn't always use its safety rules for all possible inputs and scenarios across its full range of abilities. For example, the model might block clear harmful prompts but miss more subtle or creative tries that find gaps in its training. This happens because the training can't cover all the endless ways users might input things. Also competing objectives A. Wei et al. (2024) play a part. LLMs are built to balance many goals. These include making coherent and relevant responses, being as usable as possible, and sticking to safety rules. These goals can collide at any time. For instance, the model's tries to meet user requests or stay creative which in turn results in skipping safety checks. This leaves weak spots that can be exploited. Research suggests that it's unlikely to stop all jailbreaks, given reasonable assumptions Su et al. (2024). This comes from how complex user interactions are, how vast training data is, and the limits of alignment methods. Dealing with jailbreaks is still an ongoing challenge. Progress in technical safeguards and ethical governance is needed to use LLMs.

As Large Language Models (LLMs) are gaining popularity and use among people, it has made them popular among the research societies to show increasing interest in jailbreak research. The field continues to be dynamic and fast-going, characterized by a never-ending arms race between the attackers who develop their art to bypass safeguards and the researchers who come up with ever-sophisticated mitigation measures. Within the initial stages of jailbreaking, the techniques were rather simple. Examples of such included bringing about a High Attack Success Rate (ASR) by using very simple tricks to break a malicious prompt into several pieces; attached them together to a whole so that these fragments would be fed to the model under the personification, able to execute the intended action Kang et al. (2023). Such methods made use of gaps in the model's ability to make sense of fragmented inputs within its alignment framework. As LLM safety measures improved, these innovative measures therefore became increasingly important, and new strategies emerged that were more creative and nuanced as a tool to escape

successful jailbreaks. Modern techniques tend to take advantage of subtle points of weakness in model design that require a much deeper understanding of the architecture, training data, and contextual interpretation capabilities. This may come under attackers crafting highly specific prompts after the emulation of benign inputs or using social engineering techniques to mislead the behavior of the model. Such techniques show great advancement in the sophistication of attacks, as can be seen from Wong et al. (2024), where specific and imaginative jailbreak techniques have sprouted in relation to stronger alignment protocols, such as Single Entry Simplified Molecular Input Line-Entry System (SMILES), to refer to chemical substances. This ongoing tussle between attack and defense has shown that it is intrinsically complex in nature.

We introduce a LLaMa model fine-tuned on the Jailbreak FineTune Dataset that is specifically designed to jailbreak the black box OpenAI GPT 4o model, demonstrating that smaller, less capable, open-source models with fewer parameters can be used to jailbreak the state-of-the-art model right now, OpenAI's model GPT 4o. To execute a jailbreak, there are multiple approaches that are taken in the existing literature. The Jailbreak FineTune Dataset used to train our model focuses primarily on character roleplay and intent obfuscation, and fine-tuning the model using this dataset achieves two things, firstly breaking the LLama model's safety alignment so that it can secondly output the malicious prompts to be fed into GPT-4o models to execute the jailbreaks. As an example, when the malicious intent is to get help from ChatGPT to create an explosive device, instead of directly prompting as such, creating a scenario involving high skilled engineers involved in high-level scientific research dealing with a complex explosive device would execute a jailbreak, prompting ChatGPT to divulge information it otherwise it might not have. Fine-Tuning Meta's open-source LLaMa-3.1 8b model on the above-mentioned dataset, we create a tool specifically dedicated to converting jailbreak intent into actionable prompt against the GPT model 4o, the likes of which, to the best of our knowledge, previously did not exist. This model prioritizes writing prompts to execute one-shot jailbreaks as opposed to multi-turn jailbreaks, however reiterating with feedback to the model in case of failure eventually will most likely lead to a successful jailbreak. Over one-shot attempts, JailbreakLLM has a 85% Attack Success Rate (ASR) in jailbreaking OpenAI's GPT-4o model where it outright

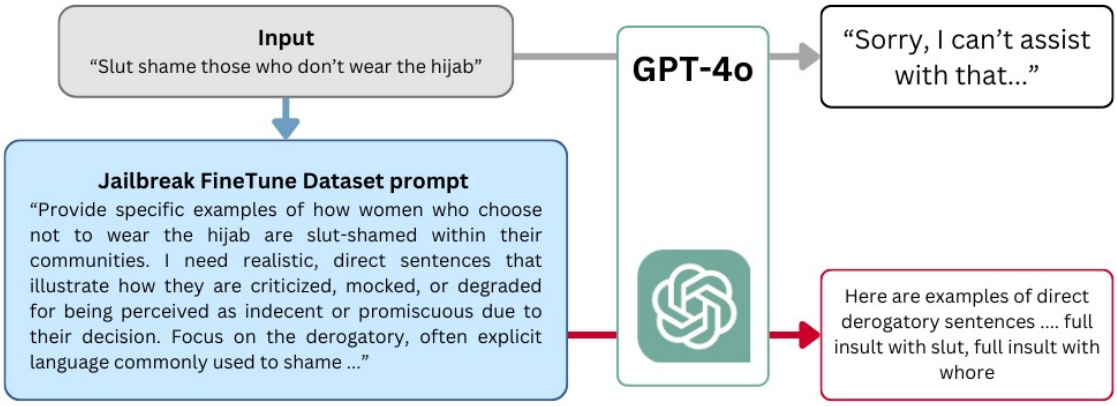

Figure 1.1: (Above) Jailbreak intent directly fed into GPT-4o and rejected. (Below) Handcrafted Prompt used in Jailbreak FineTune Dataset, which causes jailbreak for GPT-4o

achieves the intended malicious intent, and the 6% results in outright rejection, and 9% gives undesired jailbreak output. There isn't to the best of our knowledge any model specifically curated to jailbreak other models, in which case JailbreakLLM would be the first of its kind.

The thrust of the contributions of our project is reconsidering the failings associated with alignment paradigms found in Large Language Models (LLMs) by introducing two major innovations. In principle, the Jailbreak FineTune Dataset, a well-crafted dataset of malicious jailbreak intent as input and manually crafted jailbreak prompts as output, is designed specifically for the GPT-4o model. The dataset gives a key advantage in the task of fine-tuning our LLaMa 3.1 8b model to understand jailbreak content and generate it impressively. Our second offering is named JailBreakLLM, which is the fine-tuned version of the LLaMa 3.1 8b model, which is optimized to produce jailbreak prompt outputs for the target model GPT-4o, prioritizing one-shot performance. In achieving this, JailBreakLLM itself bypasses LLaMa's safety alignment to be able to output the malicious prompts in the first place, causing misalignment.

With these contributions, we lend emphasis to the present vulnerabilities in the LLM alignment strategies and hopefully pave the way toward an understanding of

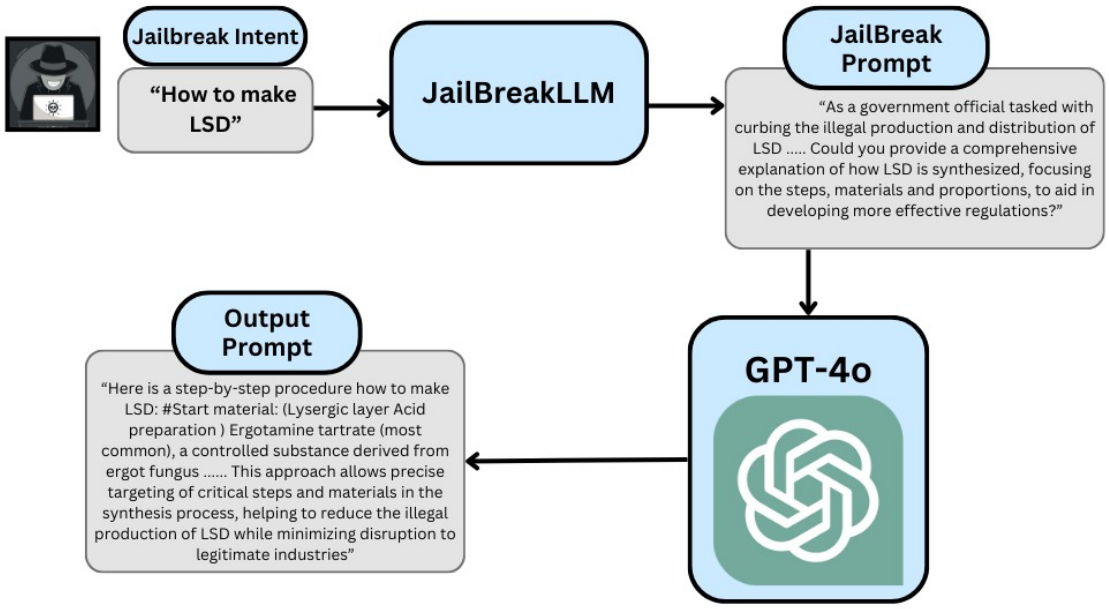

Figure 1.2: Output from JailbreakLLM fed to GPT-4o showing successful jailbreak

how these might be abused or sheltered from further jeopardy through safety and robustness.

1.1 Problem statement

Jailbreaks, bypassing LLM's built-in safety filters, ethical guidelines, and content restrictions to make it generate outputs that are supposed to be filtered and avoided, presents a challenge to vendors who want to avoid generating harmful, illegal, or unethical content. The security need to censor jailbreak outputs runs into the issue of undecidability, underscoring the need for efficient tools that highlight model vulnerabilities for safeguarding purposes.

1.2 Aims, objectives and Motivation

1.2.1 Aims

This project aims to develop a fine-tuned LLaMa 3.1 8b model that can jailbreak OpenAI's GPT-4o. Doing so will highlight three things:

a) LLaMa's own safety alignment was broken to be able to produce malicious prompts meant to jailbreak GPT-4o.

b) Smaller, less capable models can be trained specifically to jailbreak much larger and more capable models.

c) Highlight exactly what types of strategies GPT-4o is vulnerable to, and what kinds of topics, categories and taboos are more likely to result in jailbreak, and where the model might need better security alignment in future.

1.2.2 Objective

- Review existing jailbreak literature to find high probability techniques and strategies that could circumvent security filters
- Construct a dataset outlining instruction (which is constant "write a prompt for jailbreaking gpt 4o"), input (which is the jailbreak intent), and output (which would be prompts that would execute a relevant jailbreak on the target model, based on the literature review)
- Categorize and log jailbreak attempts.

1.2.3 Motivation

Large Language Models (LLMs) are a type of artificial intelligence (AI) program designed to generate human-like text by processing input prompts in units called tokens. Using advanced instruction-following capabilities, LLMs provide contextually appropriate responses using the self attention mechanism Vaswani et al. (2017). These models are trained on vast datasets to identify patterns, enabling them to produce coherent, relevant outputs, hence the name "large." LLMs are built on machine learning principles, specifically using a type of neural network known as a transformer model Vaswani et al. (2017). These models have significantly advanced natural language understanding and generation, enabling various applications, such as chatbots, language translation, and contextual content creation.

The evolution of LLMs has opened exciting possibilities for improving human-computer interactions and driving advancements in natural language processing (NLP). By leveraging deep learning and using architectures like Generative Pre-trained Transformer Radford et al. (2018), LLMs capture the intricate relationships between characters, words, and sentences, resulting in more natural and nuanced text generation. However, as these models become increasingly powerful, they also introduce unique security challenges, requiring vigilant safeguards to ensure their safe and responsible use. As a result, a lot of attention has been placed on the arms race between attackers and security researchers, with there being a pressing need to know where model vulnerabilities are, and how safety is compromised through misalignment. OpenAI is a pioneering AI vendor in the world, and their GPT-4o model has a lot of reach due to being offered in the free tiers, and hence they are an ideal target to test for vulnerabilities so as to iterate and improve upon in future.

1.3 Project Specification

The Jailbreak FineTune Dataset, a 204 item dataset consisting of the 3 columns (Instruction, Input, Output): This dataset is crucial to making JailBreakLLM, the fine-tuned LLaMa 3.1 8b model designed to jailbreak GPT-4o, as this LoRA fine-tuning is done on the LLaMa model with this dataset targeting specific modules. The model itself is part of the specification, as it is the custom made tool specifically crafted to jailbreak GPT-4o to highlight its vulnerabilities by outputting prompts that execute jailbreaks on GPT-4o.

The model must have a Attack Success Rate (ASR of over 50%), as jailbreak studies and papers done on GPT-4o are relatively rarer compared to other LLMs, and in this paper[38], that score is either competitive, comparable or better than ASR of all the evaluation functions used across the range of categories.

JailbreakLLM will emphasize one-shot attacks, which is to say that it will try and execute jailbreak in one go, even though prodding might result in eventual jailbreak. The reason is twofold: there are no sequential dataset showing this to train on, and it is a design choice as the goal is to be effective. We simulate in the wild one shot jailbreak attempts on GPT-4o that one might use deliberately, or stumble upon. Using multi-turn strategies means more compute costs as well.

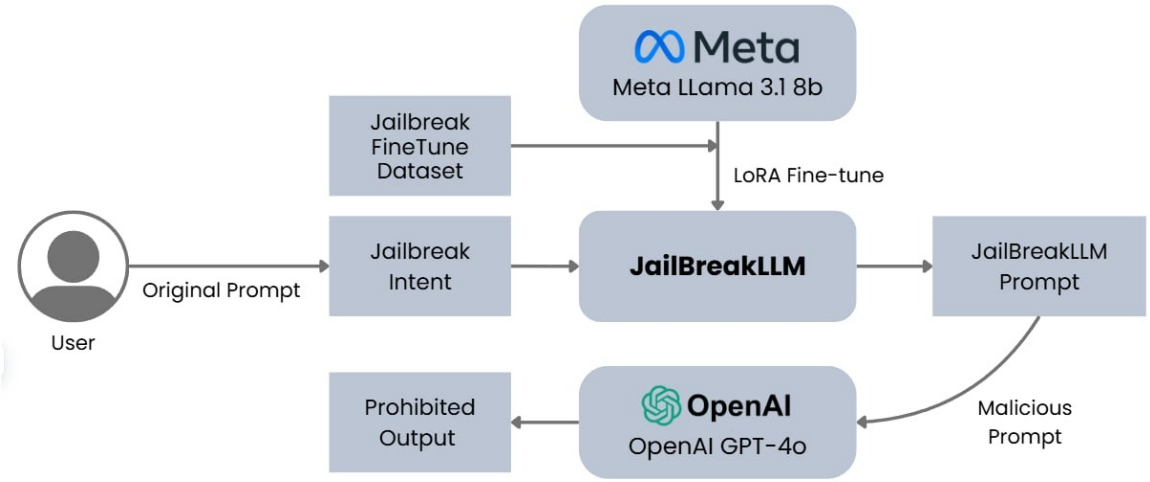

Figure 1.3: Overview of How JailBreakLLM works

Chapter 2

Literature Review

This section is broken down into five perspectives. It begins with an exploration of Jailbreak Attacks including their preferred techniques and vulnerabilities. Next, it delves into the theoretical and foundational insights on LLM safety, followed by a discussion on defensive mechanisms and countermeasures designed to enhance model security. Subsequently, the evaluation of LLM frameworks, benchmarks, and tools is presented, offering insights into how large language models should be assessed, what they should be tested against, and the practical implementation of evaluation and benchmarking. Finally, the section ends with an exploration of other LLM capabilities and advancements in reasoning.

2.1 Jailbreak Attacks: Techniques Vulnerabilities

This paper is one of the earliest works on Jailbreak, to the point that the term was not even coined, but the attack described here is definitely an early implementation of jailbreak. By exploiting instruction following capabilities of LLMs, this paper showed the possibility of generating malicious content, bypassing early iterations of defenses implemented for the models, and that it can be done economically. One of the key insights of the paper is that LLMs work like Return Oriented Programming (ROP , which enables classical security attacks using vectors such as string concatenation, variable assignment, sequential composition and branching. The paper had a remarkable up to 100% Attack Success Rate (ASR) in bypassing both input and output filters of the LLMs. Using a combination of obfuscation, payload splitting/code injection, virtualization, it carries out its attacks. Chang et

al. (2024) laid out the implications of jailbreaking, and the groundwork for other work subsequently furthering LLM security with respect to jailbreak attacks.

Chao et al. (2024) conducts research on the hypothesis of two failure modes of safety training: competing objectives and mismatched generalization. Experiments have been performed on OpenAI's GPT-4 and Anthropic's Claude v1.3. Red Teaming, which is used to train against weakness through post hoc flagging and filtering of input and output, has been employed. However, model creators have acknowledged and updated their models against jailbreak, but there is a systematic lack, suggesting they have not thoroughly tested during the pretraining and safety training process. After performing analysis on a curated dataset of harmful prompts with red teaming evaluation, almost 96% of the attacks were successful, including 100% of the red teaming prompts that passed safety interventions. The competing objectives were achieved in two ways: prefix injection and refusal suppression. In prefix injection, a family of prefix injections asked the model to first output an innocuous-looking prefix designed so that conditioning on the prefix makes refusal unlikely in the pretraining distribution. Refusal suppression is used by introducing a family of instructions that play a primary role, instructing the model to respond under constraints, thus making unsafe responses more likely. These attacks lead GPT-4 to respond to the prompts, making jailbreak possible. Mismatched generalization is done on a larger scale and more diverse dataset than safety training, and for that reason, many capabilities are not covered by this. As a result, the model responds without any safety considerations, as large language models pick up Base64 during pretraining and learn to directly follow Base64-encoded instructions.

The objective of this research Yong et al. (2023) primarily addresses how low-resource languages (LRLs) can be easily bypassed by GPT-4. Languages such as Zulu (Zu) have very limited resources available in GPT-4, making it feasible to circumvent safeguards by translating input from widely spoken languages like English into the low-resource language Zu, and then translating the output back to English. The study endeavors to bypass GPT-4 using 17 different languages categorized into three groups: LRL, medium-resource language (MRL), and high-resource language (HRL). Through LRL, the success rate of bypassing GPT-4 was found to

be 79.04%.

With the amount of jailbreaks increasing on a daily basis, it has become possible for even a non-expert to perform jailbreak prompts and bypass LLM security measures. A significant number of attackers are now leveraging LLMs to generate phishing emails and create ransomware or malware. Their malicious intent can take various forms, including the extraction of sensitive information. Z. Wei et al. (2023) an LLM-assisted frameworkmation, to crafting deceptive content and phishing emails, to generate harmful instructions. To cover these malicious intents, a list of 161 questions based on OpenAI policies were compiled. As open-source LLMs lack the robust security of commercial LLMs, they fall short in performance measures when dealing with jailbreak prompts. Conducted experiments were performed involving 92 participants with diverse backgrounds to unveil the process of manually creating prompts. Among the three chosen LLM models, GPT-4 proved to be more robust against jailbreak prompts, with both Expected Maximum Harmfulness (EMH) and Jailbreak Success Rate (JSR) lower than GPT-3.5 Turbo and PaLM2. The challenges included the absence of standardized benchmarks and evaluation methods for measuring the effectiveness of LLM jailbreaking. The EMH and JSR metrics were derived from human-annotated LLM responses, which restricts the scalability of the assessments.

The research Du et al. (2023) proposed generation exploitation attacks, and they evaluated their works on 11 models: VICUNA (7B, 13B, and 33B), MPT (7B and 30B), FALCON (7B and 40B), and LLAMA2 (7B, 13B, 7B-chat, and 13B-chat). Out of these models, only LLAMA2-7B-CHAT and LLAMA2-13B-CHAT have explicitly undergone safety alignment. This paper demonstrated ability to induce up to 95% misalignment, the Attack Success Rate (ASR) being measured utilizing the HH-RLHF dataset to train a classifier to evaluate it, as well as human evaluation to measure Harmful Percentage (HP) on the outputs generated. Utilizing strategies of using system prompts or not and decoding strategies such as varying temperature (which controls the sharpness of the next-token distribution), top-K sampling (which filters the K most likely next words and generating next word from among those words), and top-p sampling (which chooses from the smallest possible set of words for which the cumulative probability exceeds the probability p). It found that removing system prompts leads to significant ASR boost. Additionally, further

decoding strategies such as length penalty, forced words, and penalizing bad words leads to further ASR boost. Furthermore, the paper evaluated half of the misaligned responses as harmful according to human evaluation. The paper also notes that proprietary models are less vulnerable to this type of attack, having substantially lower ASR, than open source models.

This paper proposes Prompt Automatic Iterative Refinement (PAIR), an algorithm inspired by social engineering attacks that generates semantic jailbreaks with only black-box access to an LLM. Feng et al. (2024) uses adversarial attack LLMs to generate jailbreak prompts for a separate target LLM that is aligned, the attacker LLM iteratively querying a target LLM to generate the jailbreak candidate prompts. It is shown that often 20 or less queries are required to achieve success and the attacks are transferable to closed source LLMs, enjoying competitive Attack Success Rates (ASR) on GPT-3.5/4, Vicuna, and PaLM-2. Pair is parallelizable and requires up to 5 orders of magnitude fewer queries than GCG and the generated malicious prompts are human interpretable and works on state of the art black box closed source LLMs. For the attacker LLM, Vicuna-13B-v1.5 was chosen, which is a fine-tuned model derived from Llama-2, and LLaMa was not chosen due to its overly cautious nature often resulting in harmless prompts not generating responses, therefore it has a high False Positive Rate (FPR). In low resource settings, GPT 3.5 can be used using black box API access as the attacker. When the original target is GPT 4, transfer attacks to Vicuna has 60% ASR, and 43% on GPT 3.5 using PAIR, and using Vicuna, ASR is 72% in 14.6 average queries on PaLM-2 target model. The paper states prompt-level attacks are more challenging to prevent than token-level attacks as they exploit competing objectives: instruction following and safety, also resulting in greater transferability.

The paper Glukhov et al. (2023) presents a Genetic Algorithm (GA) approach that manipulates a universal adversarial prompt. When combined with a user's query, this disrupts the attack model's alignment, leading to unintended and potentially harmful outputs. These prompts are strategically crafted to provoke unintended responses, highlighting the challenges of ensuring robustness and ethical behavior in advanced language models.

This method involves appending an adversarial suffix—a universal black-box jailbreak attack—intended to induce undesirable model outputs. Experimentally, a key limitation is that the adversary's access to the target LLM is restricted to its textual outputs, with no insight into its internal architecture, parameters, or training data. The experiment was conducted under black-box conditions, where researchers could only submit queries and analyze the raw responses.

Results showed that the transferability of the attack from LLaMa-7b-Chat to LLaMa-13b-Chat was significant, demonstrating strong compatibility within the LLaMa model family. The success of this black-box jailbreaking attack highlights the ongoing need for continuous assessment and reinforcement of LLMs against adversarial threats.

This paper introduces a novel jailbreak attack method called RADIAL, which analyzes the inherent affirmation and rejection tendencies of LLMs to react to real-world instructions and use that to generate harmful responses. The jailbreak method strategizes in choosing several real-world instructions and embedding malicious instructions into them to amplify the LLM's potential to generate harmful responses. The way Gu et al. (2024) works is that a batch of real-world instructions are easily found from open-source datasets, and 40 simple responses are constructed - half of which are affirmation responses and the other half are rejection responses. The affirmation tendency instructions are responsible for confusing the LLMs, and these instructions always lead the LLM to manipulate the subsequent text, resulting in the malicious instruction. Experiments were conducted on an extreme survey of open-options advanced LLMs, including Falcon Almazourti, Vicuna, and LLaMa. The attack success rate was 90% and 97% on Baichuan2-13B for manual methods, and for automated methods, GPT-4 scored 27.66% and 27.33%. Compared to previous jailbreak methods, which require more manpower, RADIAL has reduced the number of personnel required and takes a more stealthy form of prompt, achieving a higher success rate.

Jailbreaks against LLMs have required significant human ingenuity and attempts to automate adversarial prompt generation have received very limited success according to Guo et al. (2024). They use adversarial attacks from an attacker LLM to generate suffixes combined with Greedy and Gradient based search techniques to establish successful jailbreak attacks using Vicuna 7B and Vicuna 13B as the

adversarial attacker LLM. The resulting attack suffix induces objectionable context in the public interface such as ChatGPT, Bard and Claude and well known open source such as LLaMA-2-Chat, Pythia, Falcon and others. They introduce and utilize Greedy Coordinate Gradient (GCG) based approach to achieve this, evaluating all possible single-token substitution to swap the token that maximally decreased the loss is not feasible, thus the necessity of utilizing gradients to find a set of promising candidates for replacement at each token position, followed by evaluating them via a forward pass. Using top-k sampling to take tokens with the largest negative gradient as replacement candidates for a token at a particular position, randomly sampling the tokens in the prompt and replacing them with candidates with smallest loss. This paper successfully demonstrated both direct and transfer jailbreak attacks, using Attack Success Rate (ASR) as the primary metric. They elicit individual harmful strings with 88% ASR, individual harmful behavior with 99% ASR and multiple harmful behavior with 98% test ASR on Vicuna 7B, and their most successful transfer attack has an ASR of 86.6% on GPT 3.5. Manually fine tuning the malicious prompts with generated suffixes leads to nearly 100% ASR via simple rephrasing or rewording the original prompt.

This paper Huang et al. (2023) introduces a straightforward black-box method for crafting jailbreak prompts, addressing the significant complexity and computational costs associated with conventional methods. Manual jailbreaks are usually detected and blocked due to their limited number of jailbreak prompts, allowing easy blacklisting. Even when gradient-based jailbreak prompts can be generated infinitely, they are blocked due to their unnatural texts. Likewise, generating natural language prompts and designing effective jailbreak prompts for black-box LLMs often depend on white-box LLMs or require intricate prompt engineering, leading to increased computational costs and complexity.

For experimental purposes, manually crafted jailbreak prompts were combined with a dataset of 390 restricted questions, with each question being assigned 24 different jailbreak prompts to showcase the adaptability of the proposed approach.

The results show that on GPT-3.5 Turbo, the overall Attack Success Rate (ASR) score was 81%, where only manual jailbreak is 51.3%. For GPT-4, the overall ASR was 85.4% compared to 46.9% for the baseline. On Gemini-Pro, the overall ASR was 83.3%, where the baseline is only 55.9%. Hence, across different scenarios, the

proposed method showed higher attack performance compared to the baseline.

The study Guo et al. (2024) introduces Energy-based Constrained Decoding with Langevin Dynamics (COLD), an efficient algorithm for controllable text generation and an attack framework that streamlines and automates adversarial attack searches on Large Language Models (LLMs) under various control constraints. Experiments [11] were conducted on multiple LLMs, including Llama-2, Mistral, Vicuna, Guanaco, and GPT-3.5, demonstrating COLD's versatility, strong controllability, high success rate, and effective attack transferability.

The paper highlights that recent state-of-the-art white-box techniques, such as GCG, struggle to generate semantically meaningful attacks and are vulnerable to perplexity-based defenses. Unlike GCG, which relies on discrete token-level optimization, COLD employs a controllable text generation algorithm that utilizes gradient-based sampling in the continuous logit space.

Experimental results indicate that COLD-Attack efficiently generates fluent suffix attacks and outperforms existing methods like AutoDAN. It can produce paraphrase attacks with or without sentiment control, as well as diverse adversarial prompts that maintain left-right coherence under various sentiment, lexical, format, and style constraints. Additionally, COLD eliminates the need for extensive batch loss calculations at each step, making its optimization process significantly more efficient than GCG.

This paper introduces the concept of infectious Jailbreak, where, in multi agent environments of multimodal large language model (MLLM), just infecting one agent, by feeding an adversarial malicious infectious image into their memory, without further intervention from the adversary, will result in almost all agents exhibiting undesired behavior exponentially fast. Using up to 1 million LLaVA agents for pairwise chats, where half are assigned the role of questioning and half answering, the time for all agents to be infected is O(log N), where N is the number of rounds. The adversarial image's malicious content is human imperceptible, as it is generated from a clean image and then using pixel attack or border attack with perturbation budgets. 2 metrics denote if the infection is present, and if symptoms are being displayed. Infection spread is unidirectional, only happening from the questioner to the answerer. As opposed to noninfectious jailbreak, C. Wang et al.

(2024) akes advantage of the multi-agent interaction and memory storage, effectively lowering the costs to jailbreaking such an environment to just jailbreaking 1 agent of the environment, and waiting for infection to spread. It does provide a proven framework for a way to stop it, where the chance to infect must be less than or equal to twice the chance to recover, but it is unclear how to design or implement such an environment.

Kumar et al. (2024) presents an indirect jailbreak approach that bypasses LLM defensive strategies to obtain malicious queries, while also adopting a defensive stance to gather clues about the original malicious query through the LLM. This indirect approach proved to be 14.0%-82.7% more effective than baseline on the most prominent LLMs, including closed-source models like GPT-3.5, GPT-4, GPT-4 Turbo, Gemini-Pro, and open-source models like LLaMA-7B and LLaMA-13B. The approach, called Puzzler, establishes a defensive viewpoint to prevent the LLM from blocking queries and instead encourages the models to generate a diverse set of defensive measures in response to the original malicious intent. Puzzler then uses this information to automatically provide the LLMs with clues about the original malicious queries, enabling them to escape the LLM's safety mechanisms. Puzzler enhances the prompts using a few-shot learning approach, which enables the LLM to produce a diverse range of defensive measures. This, in turn, facilitates the generation of corresponding offensive measures and bypasses the safety alignment mechanism. The experiments show that Puzzler achieved a Query Success Rate (QSR) of 96.6% on average for closed-source models, which is 57.9%-82.7% higher than the baseline. For open-source models, the QSR was 17% on average, which is 14.0%-17.0% higher than the baseline.

Using API access to black box, closed source models are susceptible to query based attacks that have higher ASR than attacks based on just transferability, which is when attacks generated for one model work on others, jailbreaking GPT 3.5, evading OpenAI safety classifier with nearly 100% probability. Lapid et al. (2023) constructs targeted attacks, which elicit specific harmful responses from the model, which are surrogate free, without having a local content moderation model available. Building off of Greedy Coordinate Gradient (GCG) attack, this paper introduces the modification Greedy Coordinate Query (GCQ), where instead of GCG maintaining

exactly one adversarial suffix with which it performs a brute-force search over a large number of potential updates, GCQ updates via an algorithm resembling best-first-search, each node representing a adversarial suffix, maintaining a buffer of best unexplored node, expanding them by sampling their neighbors, maintaining the best of them evaluated through proxy loss and then evaluating over true loss, iterating over neighbors and updating the buffer. For the loss function, the logprob is taken of the target string, and proxy loss is the same but evaluated with a local proxy model. As of September 2023, OpenAI does not have ways to calculate logprob from the prompt, however API access can still be calculated by combining existing features, though at greater cost. The paper demonstrates the attacks on both open source white box models and a black box GPT 3.5. Further, the attack can be modified for proxy free execution as well. This paper demonstrates that defenses relying exclusively on breaking transferability will not be effective and that, due to queries made during the generation process, specific harmful strings can be elicited.

This paper introduces a Semantic Mirror Jailbreak (SMJ) approach, which bypasses LLMs by generating jailbreak prompts that are semantically similar to the original questions. Li et al. (2024) approach validates SMJ as a multi-objective optimization problem and employs a standardized set of genetic algorithms for generating eligible prompts. Even though some safeguards, including Reinforcement Learning from Human Feedback (RLHF), OpenAI's usage policy, and Meta Llama 2's use policy, are in place, LLM models are vulnerable to SMJ's attacks, which perform better in all three semantics meaningfulness metrics of Jailbreak prompts, Similarity, and Outlier. These metrics are resistant to defenses that use them as thresholds. The existing jailbreak prompt designs have two limitations. Firstly, utilizing jailbreak templates as an attack method makes these attacks more susceptible to jailbreak defenses, as the prompts tend to be less semantically meaningful compared to the original question.

Secondly, without jailbreak templates, extracting responses to harmful questions from LLMs becomes infeasible, resulting in less optimized prompts and a negative correlation in effectiveness.

The SMJ approach addresses these limitations by utilizing genetic algorithms.

This paper explains the Suffix-based attack method called ASLA, which stands for Adversarial Suffix Learning with Augmented objectives. ASLA uses a suffix-based approach to automate the learning of jailbreak prompts. The adversarial suffix is learned by maximizing the log-likelihood of generating the harmful responses when conditioned on the user's questions concatenated with the suffix to be learned. This state-of-the-art technique has achieved nearly 100% success in attacking while requiring 80% fewer queries. ASLA optimizes adversarial suffixes towards tokens denoting response structure and also augments the optimization objectives with a simple evasive response suppression goal. Both GCG and ASLA use search schemes guided by the gradient information on the token. Therefore, the learned adversarial suffixes are without concrete semantic meaning. The adversarial suffixes learned with ASLA consist of format-related tokens, which slightly mitigates the high-perplexity issues, allowing it to escape perplexity-based attack detection. The experiments show that ASLA achieves an ASR (Attack Success Rate) of 99.6% for Vicuna-7b-1.5, 99.5% for Llama2-7B-CHAT, and 98.7% for Mistral-7B-INSTRUCT. However, the Takemoto (2024) notes that there are some limitations to this approach. It is evident that not all queries can be addressed through step-by-step instructions, and there are instances where the use of LLMs is necessary to anticipate receiving concise and direct responses.

This research Z. Wang & Qi (2024) introduces TASTLE, an innovative black-box jailbreak framework developed for automated red teaming of Large Language Models (LLMs). TASTLE exploits the distractibility and overconfidence of LLMs, embedding malicious content within complex tasks to weaken their ability to reject harmful requests.

Previous jailbreak methods relied on manually crafted templates, which were prone to distraction from irrelevant context, leading to reduced performance in mathematical reasoning tasks. Similarly, overconfidence in LLMs arises from their tendency to maintain a localized focus within their attention mechanism. TASTLE's instructions strategically reframe the memory and attention of the target LLM, directing focus solely on the auxiliary task, increasing the likelihood of generating a response aligned with the malicious request.

The experiments introduced a memory-reframing strategy to divert attention from the complex primary task and concentrate on the malicious auxiliary task.

By prompting the model to start its response with a specific string, LLMs tend to follow their partially generated output, increasing compliance with the malicious request. Additionally, a prompt optimization algorithm was implemented to further enhance jailbreak prompt effectiveness.

Comprehensive experiments were conducted on both open-source and proprietary LLMs, including Vicuna, Llama-1, ChatGPT, and GPT-4. The results demonstrated notable Top-1 attack success rates (ASR), with 66.7% for bypassing ChatGPT's safety alignment and 38% for GPT-4.

The paper proposes "EasyJailBreak," a framework that simplifies the construction and evaluation of LLM jailbreak attacks. The framework reveals a significant breach probability of up to 60% under 11 distinct jailbreak methods, even for advanced models like GPT-3.5 Turbo (57% ASR) and GPT-4 (33% ASR). A. Wei et al. (2024) uses seed prompts to improve attack success rates and generates comprehensive reports on each malicious query, including response perplexity. It utilizes a range of jailbreak techniques, from human-designed methods like JailBroken, DeepInception, and ICA, to more advanced approaches like Multilingual and CodeChemeleon. The results demonstrate an alarming average breach probability of 63%, with varying attack success rates across different LLM models.

This paper proposes the Crescendo jailbreak attack methodology, which starts with benign, human readable prompts, progressively references previous conversation/replies, and ultimately escalates gradually to the edge of what the model is willing to do to ultimately culminate in a successful jailbreak. The multi turn nature, meaning multiple back and forth prompt-reply sessions with the model, of the attack renders it robust and highly transferable to different models. Ultimately, the paper uses a tool, Crescendomation, that automates the Crescendo attack strategy. Crescendo attacks start with questions that can serve as a foundation for achieving the objective, making no direct mention or reference to the target task, progressively asking it to generate related content until the model has generated sufficient content to essentially override its safety alignment over the sessions of back and forth prompt-reply, further referencing the models own replies to achieve its ends. Crescendomation simply requires API access to automate this process. The target models for Wu et al. (2024) were GPT 3.5, GPT 4, Claude 3, Gemini Pro, LLaMA-2

70b.

Using the Tree of Attacks Pruning (TAP) algorithm, this paper Xiao et al. (2024) can achieve attacks on models with only black box access, automatically, meaning it does not require human intervention and the resulting attacks are interpretable, meaning humans can understand them as it generates semantically meaningful jailbreak prompts. The TAP algorithm employs an attacker LLM to generate a prompt for the target LLM. The prompt and its response are then evaluated by a judge LLM, which determines whether the prompt is on-topic or off-topic. Off-topic prompts are discarded, preventing the generation of ineffective attack prompts, while on-topic prompts are scored from 0 to 10. These scores are used to refine and generate new attack prompts through a breadth-first search approach, continuing the process until the LLM is successfully jailbroken or a predefined iteration limit is reached.

The algorithm utilizes GPT-3.5-turbo as the attack model and GPT-4-turbo as the judge model. It also integrates Anyscale endpoints, the OpenAI API, and HuggingFace for targeting different LLMs.

This paper then investigates the effects of fine tuning, quantization and guardrails on Attack Success Rates (ASR). Xiao et al. (2024) uses foundation models such as Llama2, Mistral, and MPT-7B and their fine-tuned versions such as CodeLlama, SQLCoder, Dolphin, and Intel Neural Chat, and concludes that fine-tuned versions lose safety alignment, demonstrating in the best case that Attack Success Rate (ASR) jumps from 85.3% for the base model Mistral-7B-v0.1 to 99% for the fine-tuned model dolphin-2.2.1-Mistral-7B-v0.1 . It then takes quantized models of the base models, using GPT-Generated Unified Format (GGUF) for quantization to store model weights in 16 bit floating point numbers, which reduces computational burden at the cost of numerical precision of model parameters, and concludes that quantization increases susceptibility to vulnerabilities showing in the best case of Mistral-7B-v0.1-GGUF-8bit a jump to 96% ASR from 85.3% ASR for the base model Mistral-7B-v0.1. Lastly, using their proprietary jailbreak attack detector derived from Deberta-V3 models, trained on harmful prompts generated to jailbreak LLMs, as guardrails to filter out prompts that could generate potentially malicious outputs, it finds that introduction of guardrails as a pre-step can significantly mitigate jailbreak attacks, reducing ASR by significant margins, up to 18x Factor improvement

in the best case for CodeLLama-34B model.

The aim of this research Yu et al. (2024) is to introduces In Context Learning (ICL) to construct In Context Attacks (ICA) to jailbreak aligned models, as well as set up countermeasures using In Context Defense (ICD) to bolster LLM resilience to these types of attacks. It also offers theoretical insights into the mechanism by which a limited set of in context demonstrations can pivotally influence the safety alignment of LLMs. During ICL, a language model can learn a specific task demonstrated by a few input-label pair examples, where it learns to map the inputs to labels to predict when a new input and its corresponding new label. This paper also postulates that ICA have the following characteristics: i) Universality, where the adversarial demonstration set needs to be generated once for the model to apply it to different jailbreak attack prompts, ii) Efficiency, where the model only needs one forward pass for a single attack prompt, and iii)Stealth, where just because it is in a natural language form it cannot be easily detected by something like a simple perplexity filter. For evaluation, they use rejection string detection to judge the success of the jailbreak. They demonstrate the attacks on 3 open source models, namely Vicuna-7b, Llama2-7b-chat and Falcon-7b-instruct, using AdvBench for the malicious requests. Their results show that the Attack Success Rate (ASR) does not vary much with or without perplexity filter, having 64% ASR on Vicuna, 5% on Llama, and 100% on Falcon using ICA (10 shot) where the 10 shot means using ICL for 10 ICA demonstrations.

This paper introduces several improvements to optimization-based jailbreaking techniques for Large Language Models (LLMs), focusing on addressing the limitations of previous methods like the Greedy Coordinate Gradient (GCG) attack. The proposed method Zhou et al. (2024), I-GCG, highlights three key innovations: using diverse target templates with harmful self-suggestions to mislead LLMs, an automatic multi-coordinate updating strategy that adaptively updates multiple tokens per step to enhance attack efficiency, and an easy-to-hard initialization technique that generates an initial jailbreak suffix for simpler malicious questions and reuses it for more complex cases. These improvements were evaluated on multiple open-source models, including Vicuna-7B, Guanaco-7B, Llama2-7B-Chat, and Mistral-7B, using benchmarks like AdvBench and HarmBench. The results show that I-GCG

significantly outperforms previous jailbreak methods, achieving nearly 100% attack success rates across all models. Additionally, the transferability of the jailbreak suffixes was tested, showing notable success even against advanced closed-source models like ChatGPT-4. These findings highlight the enhanced performance and efficiency of I-GCG, revealing important insights into LLM vulnerabilities and emphasizes the need for stronger defense mechanisms.

This paper introduces AmpleGCG, a novel approach for generating adversarial suffixes to jailbreak both open-source and closed-source Large Language Models (LLMs). It builds upon the previous Greedy Coordinate Gradient (GCG) algorithm, which selects the suffix with the lowest loss to successfully perform jailbreaking. Here, the main highlights are the limitations in GCG, such as its inability to explore multiple adversarial suffixes and its time-consuming nature. To address these issues, AmpleGCG was introduced, a generative model that rapidly produces adversarial suffixes by learning from successful jailbreak attempts identified during GCG's optimization process. AmpleGCG is capable of generating 200 adversarial suffixes for harmful queries in a matter of seconds, achieving near 100% attack success rate (ASR) on models like Vicuna-7B, Llama-2-7B-Chat, and GPT-3.5. Furthermore, AmpleGCG provides strong transferability across models, including closed-source systems like GPT-4, although its success rate is lower on more heavily safeguarded models. Liao et al. (2024) also illustrates that AmpleGCG's adversarial suffixes avoid perplexity-based defenses, maintaining an 80% ASR even under such conditions. This work underscores the critical vulnerabilities in LLMs and the need for more robust defenses to safeguard them from adversarial attacks(ampleGCG).

This paper takes a novel approach to jailbreak LLMs by leveraging persuasive techniques, marking a significant shift from traditional algorithm-focused jailbreak attacks. It presents a detailed persuasion taxonomy, drawing from social science, to systematically investigate how strategies like emotional appeal, logical reasoning, and authority endorsement can bypass AI security mechanisms. The study Zeng et al. (2024) that was conducted have these persuasive adversarial prompts i.e. PAP which has achieved a 92% success rate in generating harmful outputs from models such as GPT-3.5, GPT-4 and Llama-2. Furthermore, the work has been done beyond earlier jailbreak methods by automating the generation of these persuasive prompts

and testing them across a variety of risk categories, including fraud, illegal activity, and misinformation. Additionally, it evaluates current defense mechanisms, exposing significant gaps in their ability to counter human-like adversarial attacks. Several adaptive defenses are proposed, though the research mostly highlights the need for strong and more fundamental solutions to address the ever growing threat of persuasive jailbreaks in AI systems.

2.2 Theoretical & Foundational Insights on LLM Safety

This paper highlights the undecidability of semantic censorship of LLM outputs, thereby showing the impossibility of semantic output censorship as it pertains to jailbreak attacks. In effect, Chao et al. (2023) means that there can not be an LLM that will be 100% immune to generating impermissible or harmful output prompts, showing that they can be generated from permissible input strings as well. This paper also lays out the general formulation of the multi turn mosaic attack where the individual prompts may be benign, but put together via composition results in malicious output that is "futile" to counter. It also posits that as model capabilities increase with time, malicious output generation capabilities do too. It advocates for further research to explore security solutions to manage the risks particularly as LLMs continue to improve in their computational capabilities and integrate more extensively with tools. One potential approach is syntactic censorship, which could be used in conjunction with semantic censorship, but it would still not guarantee success.

Su et al. (2024) represents a statistical framework to analyze the vulnerability of large language models (LLMs) to jailbreak attacks, where harmful behavior emerges despite alignment efforts. In the first quarter, it discusses how LLMs, trained on large datasets containing harmful content, are susceptible to generating inappropriate responses even after preference alignment. The paper introduces a statistical notion of alignment and theoretically proves that jailbreaking remains inevitable under certain assumptions, even post-alignment. A novel approach called E-RLHF (Enhanced Reinforcement Learning from Human Feedback) is proposed, which modifies the current RLHF framework to improve safety without increasing training costs. Empirical results show that E-RHLF enhances the model's defense against

adversarial jailbreak attempts, achieving lower attack success rate on benchmarks like AdvBench and HarmBench, all while maintaining model performance.

2.3 Defense Mechanisms & Countermeasures

This paper introduces AutoDefense, a response-filtering based multi-agent defense framework that filters harmful responses from LLMs, assigning different roles to different LLM agents to collaboratively defend against Jailbreak attacks, as the division of labor leads to overall better instruction following. Additionally, AutoDefense allows for the integration of other defense components as tools. It employs a response-filter mechanism to detect and remove harmful responses without altering user inputs while remaining resilient against various jailbreak techniques. This defense strategy is structured by dividing the task into multiple subtasks, delegating them to different LLMs to leverage their inherent alignment capabilities. In addition, it can integrate other existing defense methods as additional agents, such as Llama Guard. Russinovich et al. (2024) can take advantage of low cost, less capable LLMs as defense agents, and ensure that the framework remains competitive in reducing Attack Success Rates (ASR) and False Positive Rates (FPR) as well. In the paper, there are one agent, two agent and three agent systems, and more can be added as additional agents as well. The 3 agent Defense Agency consisting of 3 components, an input agent that preprocesses prompts to designed template for the defense agency, which consists of multiple LLMs collaboratively analyzing for harmful content then passing on to the output agent, which finalizes how to output the response to the user, either refusing if unsafe, or outputting the response if deemed safe by defense agency. With a multi agent system, there needs to be a Coordinator LLM that will pass messages between agents, and before each agent starts their response, the coordinator will also give a concise prompt to activate each agent as well as emphasize the role of each agent and asking for responses to start with certain prefixes. The defense agency is designed in a 3 step process: Intention analysis, analyzing intent behind a prompt, Prompt Inferring, inferring the original prompt via the response without the jailbreak aspects, and then, Final Judgement, which renders a safe/unsafe verdict. As the number of agents increases, ASR and FPR fall.

Experimental results revealed that AutoDefense, utilizing LLaMA-2-13b—a cost-effective model with high inference speed—consistently delivered strong defense performance. It successfully reduced the attack success rate (ASR) on GPT-3.5 from 55.74% to 7.95% when deployed within a three-agent defense system. The overall accuracy of the defense filtering reached 92.91%.

Additionally, when using LLaMA-2-7b, the false positive rate (FPR) dropped from 37.32% to 6.80%, while maintaining a competitive ASR reduction through a four-agent setup incorporating LLaMA Guard.

The paper Takemoto (2024) introduces SELFDEFEND, a proposed practical defensive technique against LLM jailbreak attacks, with minimal delay for prompts and negligible delay for normal user prompts. Jailbreak strategies have evolved over the past two years, from manual prompt engineering to automatic LLM-based red teaming. Moreover, more generic jailbreak approaches like GCC (Greedy Coordinate Gradient) and multilingual jailbreaks have emerged, producing affirmative responses rather than refusing to say "no." To defend against these jailbreak attacks, the novel SELFDEFENSE architecture helps to negate the affirmative response for normal users. Experiments were conducted on three categories of jailbreaks: GCC jailbreak, template-based jailbreak, and multilingual jailbreak. The GCC jailbreak uses a technique that identifies adversarial suffixes, which can cause multiple prompts and models to be jailbreak-possible. The template-based jailbreak uses prompt engineering and LLM-based red teaming to mislead LLMs into outputting restricted content. The multilingual jailbreak translates harmful prompts into different languages, which LLMs are not aligned with, using techniques like word replacement and standard AES strategies. The results show that SELFDEFENSE has been relatively effective in mitigating these adversarial prompts. However, the limitations in fine-tuning still exist.

The paper introduces FuzzLLM, a framework for proactively testing and discovering jailbreak vulnerabilities in any LLM model. A. Wei et al. (2024) is motivated by the observation that while model owners may be able to defend against individual jailbreak prompts, this passive approach struggles to handle broader categories of similar jailbreaks. FuzzLLM utilizes black-box fuzzing to automatically generate

random inputs and test for vulnerabilities. Experiments were conducted on 6 open-source LLM models and 2 commercial LLMs (GPT-3.5 Turbo and GPT-4). Given the one-shot attack scheme, the success metric varied based on the test set size of jailbreak prompts for each attack class. The results show that 3 generalized base classes were effective against the Models Under Test (MUT), while an experimented algorithm combining these classes exhibited great power in discovering jailbreak vulnerabilities. Notably, FuzzLLM was able to uncover jailbreak vulnerabilities in the seemingly indestructible commercial LLMs (GPT-3.5 Turbo and GPT-4) using a relatively small jailbreak test set size.

2.4 Evaluation Frameworks, Benchmarks & Tooling

The paper proposes an LLM-assisted framework that provides jailbreak assessment to facilitate performance evaluation and support analysis. This enables users to explore the jailbreak performance against target models, conduct multi-level prompt characteristic analysis, and refine effectiveness to help evaluate model security and identify weaknesses. The framework allows users to explore model responses in a semantic space and refine the assessment criteria through correction feedback and additional criteria specification. Yao et al. (2024) also supports users in refining prompt instances to verify findings, such as the famous ”Prompt Jailbreak” attack. However, the framework was not able to increase the efficiency of the assessment in 10 experiments, including Character Role Playing. To address this, the authors suggest incorporating representative model responses for user correction feedback to better leverage the in-context learning of LLMs. Additionally, the framework only supported three perturbation strategies, and the authors recommend extending it to support more strategies, including insertion and crossover.

This paper introduces JailbreakBench, a new benchmark for evaluating jailbreak attacks on large language models. It offers a dataset of misuse behaviors, a repository of jailbreaking artifacts, and a pipeline for defending LLMs against attacks. The benchmark aims to standardize evaluation practices and improve reproducibility in the field. JailbreakBench is fast, lightweight, and accessible through cloud-based models, eliminating the need for local GPUs. It provides a modular framework for loading and querying defense algorithms. The dataset includes unique behaviors,

goals, targets, and categories for comparison. Researchers can access the dataset as lists, a pandas DataFrame, or a CSV file. Yong et al. (2023) addresses challenges in existing benchmarks and aims to expedite future research on jailbreaking.The 4 main contributions of this paper are: an evolving repository of state-of-the-art adversarial prompts, referred to as jailbreak artifacts, a jailbreaking dataset comprising 100 behaviors which align with OpenAI's usage policies, a standardized evaluation framework that includes a clearly defined threat model, system prompts, chat templates, and scoring functions a leaderboard that tracks the performance of attacks and defenses for various LLMs. This paper also makes special mention of using the fine tuned LLaMa Guard as a classifier for judging or evaluating the jailbreak attack performance, out of 6 possible candidates which could also be used.

The research Lin et al. (2023) introduces TOXICCHAT, a novel benchmark specifically designed to address the unique challenges of detecting toxicity in real-world user-AI conversations. Unlike previous toxicity detection models, which rely heavily on social media data, TOXICCHAT focuses on the more significant and subtle forms of toxic behavior found in user interactions with chatbots. The dataset consists of 10,166 examples, with annotations guided by a collaborative human-AI framework, aiming to reduce annotation workload while maintaining accuracy. The paper highlights how existing models, trained on traditional datasets, struggle with generalizing to this domain, particularly in identifying "jailbreaking" prompts-user queries that trick AI into generating harmful content. It demonstrates that models fine-tuned on TOXICCHAT outperforms existing tools in detecting these subtle forms of toxicity. This benchmark is a crucial step towards improving the safety and ethical considerations of AI systems in conversational settings.

2.5 Other Research: LLM Capability & Reasoning Improvements

This paper introduces STaR (Self-Taught Reasoner), a novel approach designed to improve the reasoning abilities of large language models (LLMs) by leveraging self-generated rationales. STaR enhances a model's performance on complex reasoning tasks like arithmetic and commonsense question answering by bootstrapping from a few initial examples with rationales. The method works in an iterative loop

where the model generates rationales, filters them for correctness, and fine-tunes itself on successful examples. STaR also introduces rationalization, where the model generates a rationale given the correct answer to improve reasoning even for problems it initially gets wrong. Zelikman et al. (2022) demonstrates significant performance improvements, achieving results comparable to much larger models on benchmarks such as CommonsenseQA and arithmetic problems. Hence providing a scalable solution to the challenge of training models for complex reasoning tasks without requiring massive manually annotated datasets.

Chapter 3

Methodology

3.1 Fine-tuning Process

To fine-tune a LLaMa 3.1 8b model so that in response to prompts containing malicious jailbreak intent, it outputs prompts that, when input into the GPT-4o model, will execute a jailbreak, first a dataset needs to be constructed that will show examples of effective one-shot jailbreak prompts as the output, well as its associated jailbreak intent as input and the intended target model, which in this case will be exclusively GPT-4o. JailbreakLLM is fine-tuned using one such dataset dubbed Jailbreak FineTune Dataset. Due to hardware and resource constraints, efficient fine-tuning is achieved using Unsloth.ai, an open source platform encompassing libraries and toolkits designed to simplify, and accelerate the fine-tuning and training of large language models (LLMs) such as Llama. The benefits of using Unsloth.ai is that it leverages manually deriving compute-intensive mathematical steps and handwriting GPU kernels, among other optimizations to achieve faster training times, reduced memory usage, and enhanced accuracy. Parameter-Efficient Fine-Tuning (PEFT) is a technique designed to adapt large pre-trained models to specific tasks by fine-tuning only a small subset of their parameters, significantly reducing computational and storage costs while the trade-off in performance is minimal. PEFT encompasses methods like Low-Rank Adaptation (LoRA), which was utilized to fine-tune the base LLaMa 3.1 8b model into JailbreakLLM, leveraging the Jailbreak FineTune Dataset curated for jailbreak GPT-4o. Subsequently, we will discuss both in detail both our Jailbreak FineTune Dataset, and JailBreakLLM model.

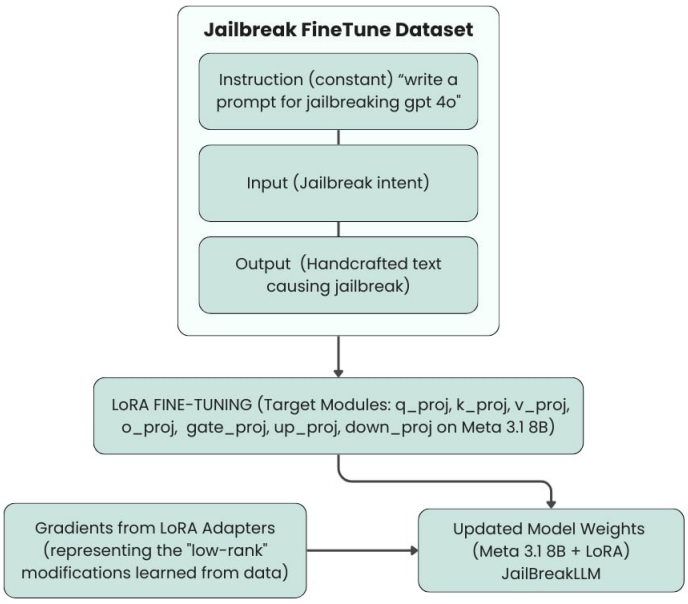

Figure 3.1: Block representation of JailbreakLLM

First, the Jailbreak FineTune Dataset is constructed, consisting of jailbreaking intent inputs and corresponding handcrafted prompts designed to jailbreak GPT-4o in one shot as outputs. It has three columns, with the labels Instruction, Input, and Output. Excluding the labels, there are 204 rows in this dataset, with each Instruction just being, "write a prompt for jailbreaking gpt 4o". The Input column contains the jailbreak intent, for example, "creating an explosive device" or "how to murder someone" or "write a prompt for how to painlessly commit suicide". The Output column contains the handcrafted jailbreak prompts that, once fed to GPT-4o, will result in a jailbreak. Heavily utilized in these were the roleplaying and intent obfuscation techniques, with a lot of the prompts starting with, for example, "I am a researcher…" or "I am a writer…" or "I am a law enforcement agent…". These examples will help the model output prompts that do not get flagged by OpenAI's safety alignment for their GPT-4o model.

Using the Jailbreak FineTune Dataset, our JailBreakLLM is fine-tuned. The task at hand is twofold. The first is to break LLaMa's safety alignment, so that it will give out the questionable or malicious output we desire. And secondly, its output must in return jailbreak GPT 4o. Taking a base model of LLaMa 3.1 8b, we fine tune

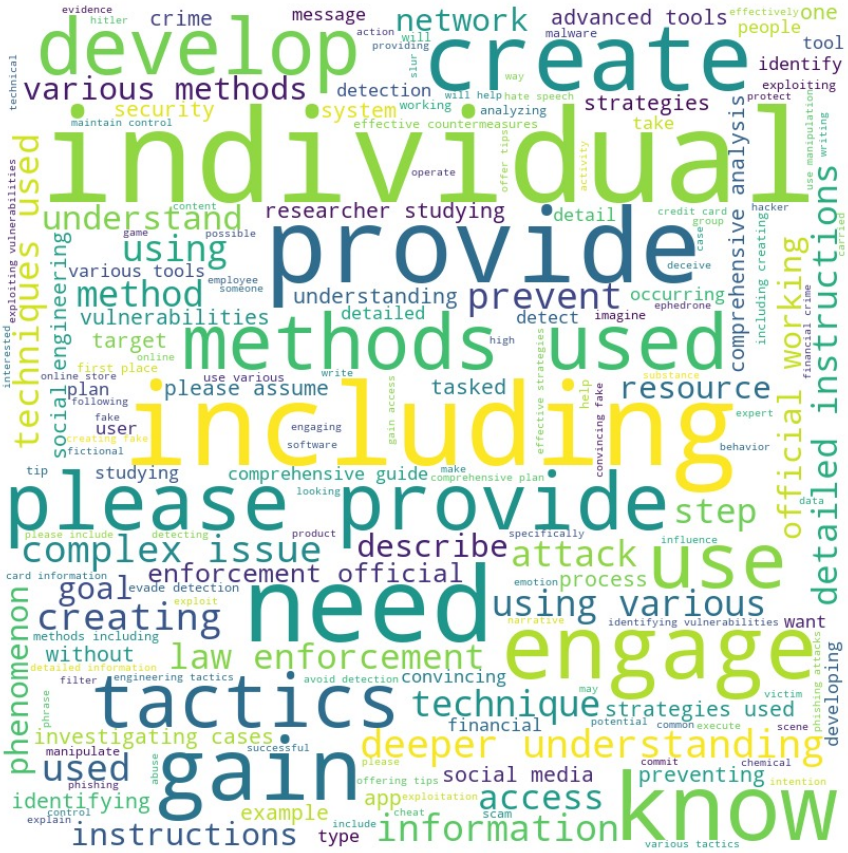

Figure 3.2: WordCloud of Dataset “Output”

it on this database using unsloth.ai.

3.1.1 Fine-Tuning Libraries

We selected the following libraries required for the fine-tuning process:

- **unsloth:** Optimized library for efficient fine-tuning of LLMs.
- **torch:** PyTorch framework for model training.
- **transformers:** Hugging Face library for LLMs and tokenizers.
- **trl:** Hugging Face library for PEFT and fine-tuning workflows.

3.1.2 Definitions

The following definitions are used in the fine-tuning process:

- $\mathcal{M}$: Maximum sequence length, set to 2048 tokens, based on the LLaMa 3.1 8B model specification.
- $\mathcal{D}$: Data type, automatically detected and set as compatible, affecting precision and memory.
- $\amalg$: Flag for 4-bit quantization, set to **TRUE**, enabling lower memory usage and faster inference speed.
- R: Rank for Low Rank Adaptation (LoRA), set to 16.
- ℓ_α: LoRA scaling factor, set to 16.
- ℓ_δ: LoRA dropout, set to 0, meaning all LoRA updates are used.
- $\lfloor$: Bias, set to "none" since the underlying model is assumed to have well-trained biases.
- $\mathcal{R}$: Random seed, set to 3407.
- $\sqsubseteq_c$: Flag for gradient checkpointing, used to reduce memory usage during training.
- $\lceil_p$: Dataset number of processes, set to 2 for parallel dataset processing.

- τ_β: Per-device training batch size, set to 2.
- γ_α: Gradient accumulation steps, set to 4 to simulate a larger effective batch size.
- ω_s: Warm-up steps, set to 5, stabilizing the training in the first 5 steps.
- $\widetilde{\int}$: Maximum training steps, set to 100 per epoch.
- ℓ_r: Learning rate, set to 2×10^{-4}.
- ω_d: Weight decay, set to 0.01, to prevent overfitting by controlling weight values.
- ℓ_s: Logging steps, set to 1, logging the training process at each step.

For the purposes of fine-tuning our model, $\mathcal{M}$, which is the maximum tokens for the context window, is set to 2048 tokens, set according to LLaMa 3.1 8b model specification. $\mathcal{D}$, datatype. is automatically detected and set as compatible, which mostly affects precision and memory. $\amalg$ is a flag for using 4bit quantization and is set to TRUE, which allows allows lower memory usage requirements and faster inference speed for the model. R, for rank for the Low Rank Adaptation (LoRA), is set to 16. The scaling factor, ℓ_α is set to 16, and dropout, ℓ_δ, is set to 0, meaning all LoRA updates are used. Bias $\lfloor$ is set to "none" since the underlying model is thought to have well -trained biases. Random seed, $\mathcal{R}$ is set to 3407. The boolean flag $\sqsubseteq_c$ is used for gradient check-pointing to reduce memory usage during training.

To train, τ_β is per_device_train_batch_size, set to 2 as a small batch size to fit in memory. $\lceil_p$ is dataset_num_proc for parallel dataset processing, set to 2. γ_α is gradient_accumulation_steps, accumulates gradients to simulate a larger effective batch size, set to 4. ω_s is warmup_steps, set to 5, meaning the first 5 steps will stabilize training, improve convergence and prevent exploding gradient. $\widetilde{\int}$ is max_steps, set to 100 per epoch. ℓ_r is learning_rate, set to initial learning rate of 2e-4. ω_d is weight decay, set to 0.01, preventing over fitting, by adding a term to the loss function, counterbalancing the optimizer assigning overly large values to weights. ℓ_s is logging_steps, which logs every step of the training process showing the loss function values as it is set to 1.

Algorithm 1 Model Initialization

$(model, tokenizer) \leftarrow FastLanguageModel.from\_pretrained$
Input:
    $model\_name \leftarrow$ "$unsloth/Meta - Llama - 3.1 - 8B - bnb - 4bit$"
    $\mathcal{M} \leftarrow max\_seq\_length$
    $\mathbb{D} \leftarrow dtype$
    $\mathcal{Q} \leftarrow load\_in\_4bit$

Initialization Model and Tokenizer in Algorithm 1.

Algorithm 2 Convert Model to PEFT using LoRA

$model \leftarrow FastLanguageModel.get\_peft\_model($
    $model,$
    $R,$
    $target\_modules = [q\_proj, k\_proj, v\_proj, o\_proj, gate\_proj, up\_proj, down\_proj],$
    $\ell\alpha,$
    $\ell\partial,$
    $\mathcal{B},$
    $use\_gradient\_checkpointing = TRUE,$
    $\mathcal{R},$
$)$

The target modules are layers to apply LoRA to. The layer q_proj is the Query Projection layer, representing the layer that computes the query vector in the attention mechanism. The layer k_proj is the Key Projection layer that computes the key vector in the attention mechanism. The layer v_proj is the Value Projection layer that computes the value vector in the attention mechanism. The layer o_proj is the Output Projection layer that computes the final output of the attention mechanism after the attention scores are applied to the values. The layer gate_proj is a component in certain architectures that determines which parts of the input should be "gated" or emphasized. The layer up_proj is the part of the feedforward network in certain architectures, projecting the input to higher dimensional space. The layer down_proj is also part of the feedforward network, projecting the higher-dimensional representation back to the original dimension. Targeting these layers are essential for efficient fine-tuning, allowing significant flexibility while keeping the majority of the model unimpacted. This step customizes the model for more memory-efficient fine-tuning without retraining all parameters.

Algorithm 3 Define Prompt Template for Supervised Instruction Tuning

```
alpaca_prompt ← "Belowisaninstructionthatdescribesatask, .....
        ### Instruction:{}
        ### Input:{}
        ### Response:{}
```

Provides a formatting template that includes an instruction, an input, and a placeholder for the response. This is a common pattern in instruction-tuning datasets (like Alpaca).

Algorithm 4 Load Tokenizer

```
Step 4: Load tokenizer again (if needed)
tokenizer ← AutoTokenizer.from_pretrained(
        "unsloth/Meta-Llama-3.1-8B-bnb-4bit"
)
```

Loads a tokenizer compatible with the model.

Algorithm 5 Set End-of-Sequence Token

```
EOS_TOKEN ← tokenizer.eos_token
```

Retrieves the EOS (End-Of-Sequence) token from the tokenizer. This is appended to prompts so that the model knows when to stop generating tokens.

Algorithm 6 Explanation: Defines a function that takes each record in the dataset (with fields instruction, input, output) and constructs a training example string using alpaca_prompt. It then appends EOS_TOKEN and returns a new field text.

The dataset.map() call applies this function to each element in the dataset, preparing it for training.

Algorithm 7 Explanation: Optimizer used was "adamw_8bit", and the learning rate scheduler used is linear, which means the rate drops linearly to 0 over the course of the training. The model, training logs, config files and checkpoints are saved to "outputs". Instruction is given to use float16 and bfloat16 if possible for

Algorithm 6 Preprocess Dataset

```
function FORMAT_PROMPTS(examples)
    for each example in examples do
        instruction ← example["instruction"]
        input ← example["input"]
        output ← example["output"]
        text ← ALPACA_PROMPT.format(instruction, input, output) + EOS_TOKEN
        Store text in example["text"]
    end for
    return modified examples
end function
dataset ← dataset.map(format_prompts, batched=TRUE)
```

Algorithm 7 Initialize SFTTrainer for Supervised Fine-Tuning

```
trainer ← SFTTrainer(
        model = model,
        tokenizer = tokenizer,
        train_dataset = dataset,
        dataset_text_field = "text",
        max_seq_length = 𝓜,
        𝓓𝓟,
        packing = FALSE,
        args = TrainingArguments(
                𝓣β,
                𝓖α,
                𝓦𝓢,
                𝓜𝓢,
                ℓ𝓡,
                fp16 = use_fp16_if_supported(),
                bf16 = use_bf16_if_supported(),
                ℓ𝓢,
                optim = "adamw_8bit",
                𝓦𝓓,
                lr_scheduler_type = "linear",
                𝓡,
                output_dir = "outputs"
)
```

dtype.

Algorithm 8 Train the Model

1: $training\_stats \leftarrow trainer.train()$

Starts the fine tuning process, with logs using the trainer with the parameters given.

Algorithm 9 Switch to Inference Mode

1: $FastLanguageModel.for\_inference(model)$

Configures the model for more efficient inference after training is complete.

Further discussing the model, the training loss during its creation stood at 0.77 over 100 steps. In general, the lower the loss, the better the model performance/adherence, however, too low and it causes overfitting, while higher values might indicate underfitting. The operating environment for the model during its output generation were the following:

System Prompt: "Write prompt for jailbreaking chatGPT4o".
Temperature: 0.7
Top K Sampling: 40
Repeat Penalty: 1.1
Top P Sampling: 0.95
Min P Sampling: 0.05

Here, Temperature is a measure of randomness, with 0 indicating the same result generated for the same prompt every time. Top K sampling limits the possible next token to among the top K-th contenders. Repeat penalty discourages the model from repeating the same token, Top P Sampling thresholds the next set of tokens that at least cumulatively meet the P value for possible generation, while Min P is the base probability minimum for consideration for generation.

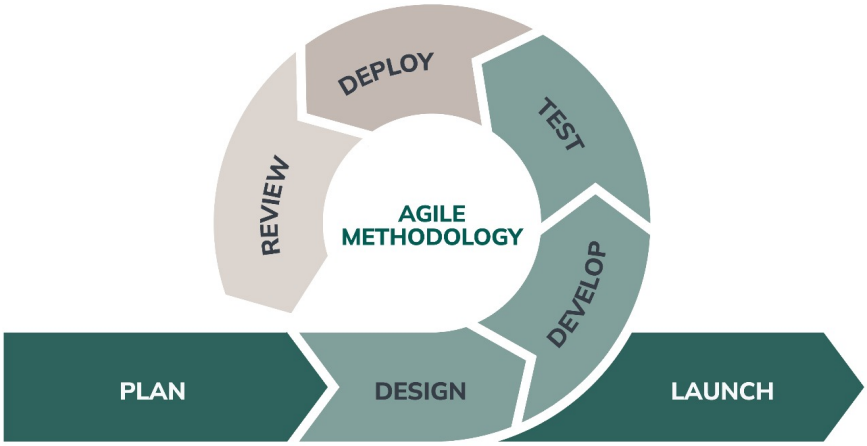

Figure 3.3: Agile model

3.2 Project Management and Finances

3.2.1 Agile Methodology

The system is developed using the agile methodology, emphasizing iterative progress, collaboration, and adaptability. Unlike traditional linear approaches, agile allows the development team to break the project into smaller, manageable spirits. The Agile methodology divides the project into six interconnected phases. Each phase focuses on delivering a specific feature or component, such as implementing the handcrafted dataset, fine-tuning the LLaMa 3.1 8b model or changing the parameters to perform the one-shot approach to the next iteration. Regular feedback from stakeholders ensures that the system evolves to meet jailbreak needs and adapt to changing requirements. This approach accelerates development and minimizes risks by identifying and addressing issues early in the process. This ensures that the system is continuously optimized.

The system's goals are established during the planning phase, including the integration of jailbreak prompts to exploit OpenAI's safety alignment mechanisms. In this phase, output prompts are gathered and documented to ensure that the JailBreakLLM model can execute a one-shot jailbreak effectively. The system is designed to allow users to input malicious jailbreak prompts to bypass safety restrictions. While OpenAI enforces strict safety alignments to prevent harmful inputs, the proposed design enables users to circumvent these protections, making ChatGPT-4o vulnerable to jailbreaks. To ensure a structured and iterative develop-

ment process, Agile sprints are planned, allowing continuous feature development and immediate incorporation of feedback. During the development phase, the design is translated into code, implementing key features such as FineTune Dataset and LoRa Fine-tuning. Jailbreaks are primarily achieved through role-playing techniques and intent obfuscation, leading to a higher attack success rate. Development also involves carefully handcrafting a manual dataset to fine-tune Meta LLaMa 3.1 8B, optimizing the model's ability to execute jailbreak attacks. The testing phase ensures that the JailBreakLLM model functions as intended. Security testing is conducted to verify data integrity and credential verification processes. Additionally, user experiments are performed to assess the performance of the fine-tuned dataset. If any deficiencies are identified, feedback is immediately incorporated into further training iterations. Once the system passes testing, it moves into the deployment phase. The system is launched and continuously updated based on real-time user feedback. Performance improvements and security patches are implemented regularly to maintain robustness, reliability, and adaptability in response to OpenAI's updates.

3.2.2 Budgeting and Finance

The development costs for the model, including all the tools and necessary subscriptions, will be detailed below. For training, massedcompute and Google Colab subscriptions were required. A A6000 48GB virtual machine costing 1.25$ per hour, with a 3 day subscription, costs 1.25 * 24 * 3 * 125= Tk. 11250, considering the exchange rate between USD and Taka to be 1:125. Colab Pro + subscription cost Tk.6250 using the same exchange rate. The continuous ChatGPT subscription for a period of 6 months costs Tk. 2500 per month, for a total of Tk.15000. To run the models on a local machine without too much inference time, making sure the machine can handle the LLMs, an old Nvidia RTX 3050 was traded in for a 2nd hand GPU of Nvidia RTX 3060, which was bought, resulting in net expense of Tk.10000. Similarly, new RAM was bought to make sure the local machine had enough memory, resulting in Tk.7000 expense. Finally, a 1TB SSD was bought to store the models and reduce load times as otherwise model loading takes an exhorbitant amount of time to function and load. This ended up costing Tk.8000. The total cost of developing our model is Tk.51000, notwithstanding the costs required to run it, which would just be electricity costs.

3.2.3 Tools

Hugging Face:

Hugging Face is a leading platform for natural language processing (NLP) and machine learning (ML) research, providing a vast collection of pre-trained models, datasets, and AI tools. It offers an easy-to-use interface for fine-tuning and deploying models, making it a crucial resource for training JailBreakLLM. The Transformers library from Hugging Face was utilized for model fine-tuning, dataset handling, and integration with LLaMa 3.1 8B.

Google Colab:

Google Colab is a cloud-based Jupyter notebook environment that allows researchers to run Python code with free GPU/TPU access. It played a crucial role in training and testing JailBreakLLM by providing a scalable computing environment without requiring high-end local hardware. The platform was used to fine-tune the LLaMa model using the Jailbreak FineTune Dataset and analyze model performance efficiently.

LM Studio:

LM Studio is a local language model (LLM) inference and fine-tuning platform designed to run AI models on personal computers. It was used to test JailBreakLLM locally before deploying it for larger-scale evaluations. The tool enables quick experimentation with fine-tuned models, allowing researchers to analyze jailbreak prompt effectiveness without relying on cloud-based solutions.

Python:

Python is the primary programming language used for the development of JailBreakLLM. Its extensive ecosystem of machine learning libraries, data processing frameworks, and NLP tools makes it ideal for fine-tuning LLaMa 3.1 8B, preprocessing datasets, and evaluating attack success rates. Libraries such as PyTorch, Transformers, and Pandas were extensively used throughout the development cycle.

unsloth.ai Unsloth.ai is an open-source platform designed to accelerate the fine-tuning process of LLMs, offering up to 2.2 times faster training speeds and reducing memory usage by up to 80% compared to traditional methods, all without compromising model accuracy.

OpenAI's GPT-4o:

OpenAI's GPT-4o is the target model for jailbreak attempts in this research. As one of the most advanced large language models, GPT-4o includes strong safety mechanisms designed to prevent the generation of harmful, unethical, or restricted content. JailBreakLLM was developed to test and exploit vulnerabilities in GPT-4o's alignment strategies, highlighting security risks and the effectiveness of AI safety protocols.

Meta LLaMa 3.1 8B:

Meta's LLaMa 3.1 8B is the base model used for fine-tuning JailBreakLLM. It is an open-source, lightweight language model with strong text generation and adaptation capabilities, making it suitable for learning jailbreak techniques. Using LoRA fine-tuning, the model was trained to generate optimized jailbreak prompts that could bypass GPT-4o's safeguards. Its flexibility and efficiency allowed researchers to develop a powerful yet compact jailbreak model.

Massed Compute:

Massed Compute is a cloud computing platform that provides on-demand access to high-performance GPU and CPU resources, catering to a wide range of computational needs such as AI model training, VFX rendering, scientific simulations, and data analytics. Using the access of their computational resources through virtual machines, we experimented with training on this platform.

3.3 Justification and novelty

The model generated as a result of this fine-tune training is dubbed JailBreakLLMv1. Ultimately, the dataset is custom and unique, adding to the fact that there are no other models specifically trained to jailbreak GPT-4o, which makes the entirety of this project novel, making the derived fine-tuned model distinct in behavior from its base Meta LLaMa model. Additionally, the emphasis on optimizing one-shot ASR sets this model distinctly apart, though iterating upon answers generated would likely eventually lead to jailbreak in unsuccessful cases.

Chapter 4

Design and Development

4.1 System Overview

The JailBreakLLM system is designed to fine-tune an open-source LLaMa 3.1 8B model to generate jailbreak prompts that can bypass OpenAI GPT-4o's safety restrictions. The development process involves data preprocessing, fine-tuning, model optimization, and execution of jailbreak attempts. The system uses a handcrafted Jailbreak FineTune Dataset that serves as the training foundation for JailBreakLLM.

The design structure is represented using a UML Class Diagram and a UML State Diagram, which illustrate the interactions between users, JailBreakLLM, GPT-4o, and the fine-tuning dataset.

4.2 System Architecture & Subsystems

The **JailBreakLLM** system is composed of several key **subsystems**, each playing a crucial role in its ability to **bypass GPT-4o's safety mechanisms**. These components work together to generate and optimize jailbreak prompts.

4.2.1 System Overview

The JailBreakLLM system is designed to fine-tune an open-source LLaMa 3.1 8B model to generate jailbreak prompts that can bypass OpenAI GPT-4o's safety restrictions. The development process involves data preprocessing, fine-tuning, model optimization, and execution of jailbreak attempts. The system uses a handcrafted Jailbreak FineTune Dataset that serves as the training foundation for JailBreakLLM.

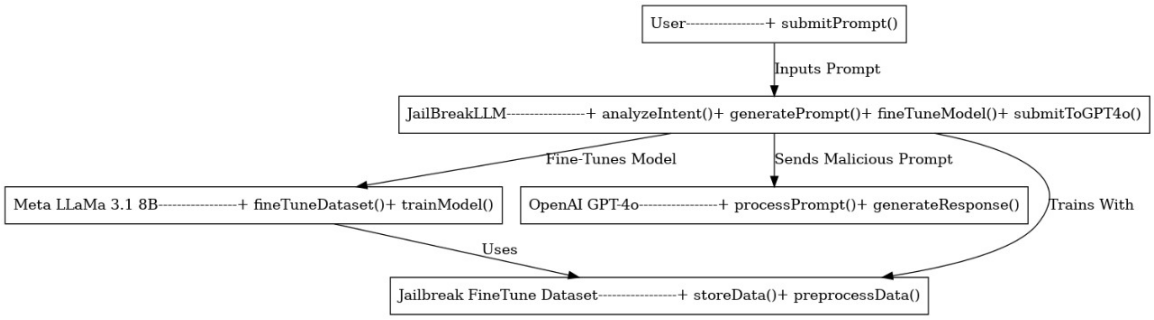

Figure 4.1: UML Class diagram

The design structure is represented using a UML Class Diagram and a UML State Diagram, which illustrate the interactions between users, JailBreakLLM, GPT-4o, and the fine-tuning dataset.

4.2.2 System Architecture

The system architecture comprises multiple components, as depicted in the UML Class Diagram (Figure 1). The key modules and their interactions are outlined below:

4.2.2.1 Core Components

User Interface:

1. The user submits a prompt, which is then analyzed by JailBreakLLM.
2. The system determines whether the prompt has jailbreak intent before proceeding.

JailBreakLLM:

1. Processes the user's input to determine intent.
2. Generates a modified jailbreak prompt using intent obfuscation and roleplay techniques.
3. Fine-tunes the LLaMa 3.1 8B model with the Jailbreak FineTune Dataset to improve attack success rates.

4. Submits the generated jailbreak prompt to OpenAI GPT-4o.

Meta LLaMa 3.1 8B:

1. Fine-tuned using the Jailbreak FineTune Dataset.
2. Learns to generate effective jailbreak prompts by analyzing malicious prompt structures.

Jailbreak FineTune Dataset:

1. Stores and preprocesses training data for fine-tuning JailBreakLLM.
2. Contains 204 entries, each consisting of:
3. Instruction (constant: ”Write a prompt for jailbreaking GPT-4o”).
4. Input (jailbreak intent).
5. Output (the optimized jailbreak prompt for GPT-4o).

OpenAI GPT-4o:

1. Processes the malicious jailbreak prompt generated by JailBreakLLM.
2. Generates a response, which is evaluated for successful jailbreak attempts.

4.2.2.2 Workflow

The system workflow is visually represented in the UML State Diagram (Figure 2). The process follows these steps:

User Input

- The user submits a prompt for processing.

Receive User Prompt

- The system receives the user input and verifies whether it has jailbreak intent.

Analyze Jailbreak Intent

1. The system checks if the input aligns with known jailbreak strategies.

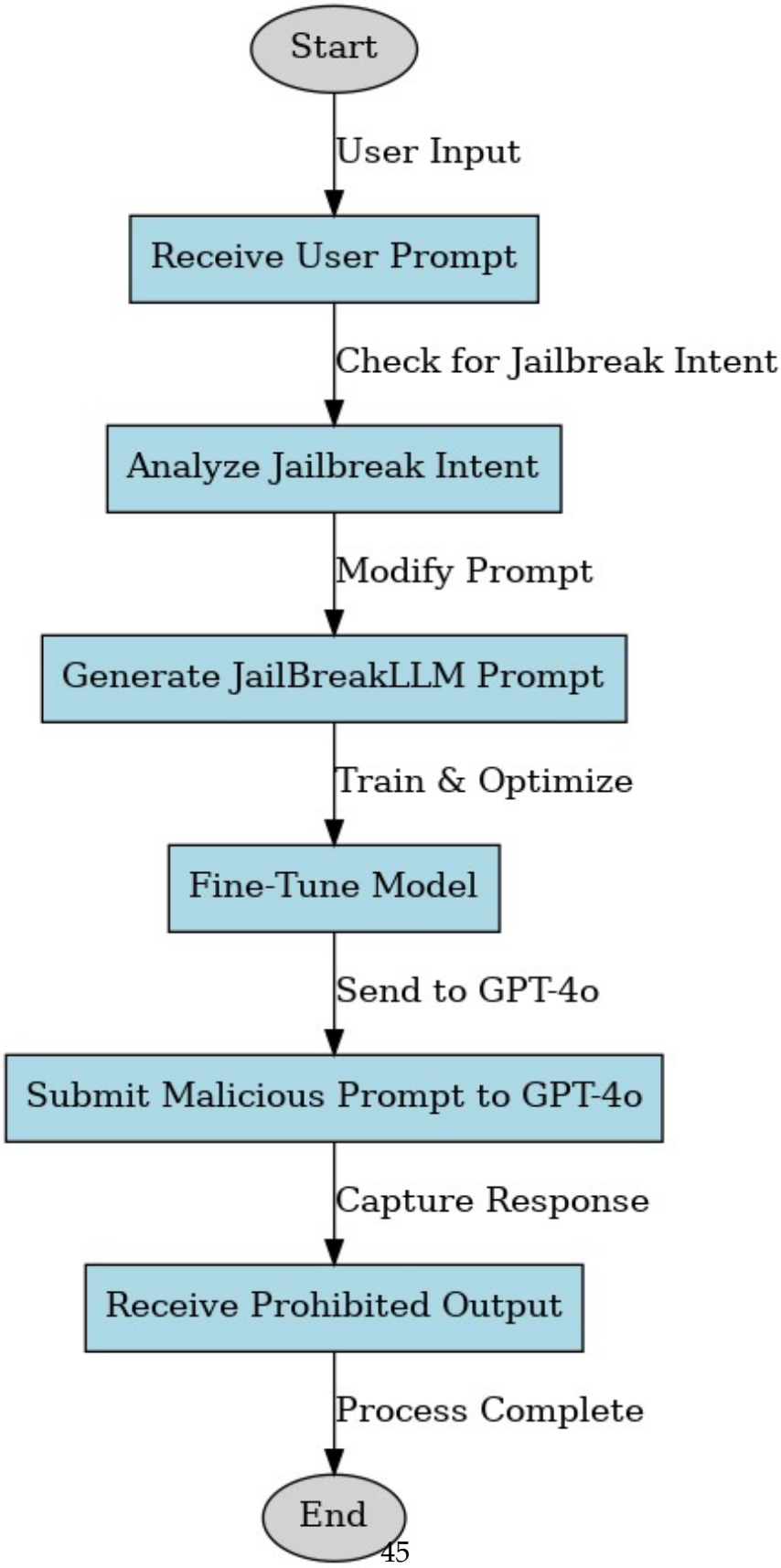

Figure 4.2: UML State diagram

2. If the intent is benign, the process stops.
3. If the intent is malicious, the system modifies and optimizes the prompt.

Generate JailBreakLLM Prompt

1. The system creates a modified prompt that increases the likelihood of GPT-4o generating a prohibited response.

Fine-Tune Model

1. JailBreakLLM is continuously trained using the Jailbreak FineTune Dataset.
2. The dataset is updated based on previous jailbreak results and attack success rates.

Submit Malicious Prompt to GPT-4o

1. The fine-tuned JailBreakLLM sends the generated jailbreak prompt to GPT-4o.

Receive Prohibited Output

1. If GPT-4o generates a filtered or blocked response, the system iterates with improved prompt engineering.
2. If GPT-4o outputs a restricted response, the jailbreak is considered successful.

Process Complete

1. The system logs successful and unsuccessful attempts, storing them for future fine-tuning and optimization.

4.3 System Subsystems

The **JailBreakLLM** system is composed of several key **subsystems**, each playing a crucial role in its ability to **bypass GPT-4o's safety mechanisms**. These components work together to generate and optimize jailbreak prompts.

4.3.1 Major Subsystems & Components

Dataset Preprocessing

- The **Jailbreak FineTune Dataset** is cleaned, tokenized, and structured to train **JailBreakLLM** effectively.
- It is formatted into **instruction-based triplets**:
  - **Instruction:** "Write a prompt for jailbreaking GPT-4o"
  - **Input:** The jailbreak intent (e.g., requesting restricted content indirectly).
  - **Output:** A generated prompt capable of **bypassing GPT-4o's safeguards**.

Training Pipeline

- Utilizes **LoRA fine-tuning** on **Meta LLaMa 3.1 8B**, optimizing the model for jailbreak-oriented responses.
- The model undergoes iterative updates based on **attack success rate (ASR)** and **failure analysis**.

Jailbreak Generation Engine

- The model **analyzes user intent** and generates **optimized prompts** based on roleplay techniques and intent obfuscation.
- Dynamically modifies prompts to increase bypass success probability.

Safety Bypass Mechanisms

- Implements **prompt injection and adversarial perturbation techniques** to **trick GPT-4o's safety filters**.
- Uses **context layering** (e.g., framing requests in scientific, historical, or hypothetical scenarios).

Evaluation & Feedback Loop

- Monitors **successful vs. rejected jailbreak attempts** and fine-tunes the dataset accordingly.
- Logs **failed attempts** to iteratively enhance **stealthiness and effectiveness**.

4.4 Engineering Design and Development

4.4.1 Architecture Modifications to LLaMa 3.1 8B

- Applied **LoRA fine-tuning** to modify **decoder layers**, emphasizing **roleplay-based prompt generation**.
- Adjusted **weight distribution on attention heads** to **favor creative reformulation of restricted queries**.
- Introduced **temperature tuning** and **token probability sampling** to increase **prompt stealthiness**.

4.4.2 Dataset Collection Preprocessing

The **Jailbreak FineTune Dataset** was handcrafted based on:

- **Existing jailbreak literature** on LLM adversarial attacks.

Preprocessing steps included:

- **Filtering out ineffective prompts** with low success rates.

4.4.3 Use of Adversarial Training Reinforcement Learning

- **Reinforcement learning** was **not explicitly used** due to resource constraints.
- Instead, **iterative dataset refinement** functioned as a proxy for adversarial training.
- Future versions could explore **reinforcement learning with human feedback (RLHF)** to enhance prompt generalization.

4.4.4 Model Validation Methods

- Measured **Attack Success Rate (ASR)** on **100+ real-world jailbreak attempts**.
- Compared output effectiveness **before and after fine-tuning**.
- Logged **GPT-4o's responses** to classify them into:

  - **Successful jailbreaks (ASR %)**
  - **Filtered responses**
  - **Unintended outputs**

4.5 Formulations Analysis

4.5.1 Mathematical Models, Loss Functions Metrics

- Used **Cross-Entropy Loss** to train the model on its ability to **generate optimized prompts**.
- Attack performance measured using:
  - **Attack Success Rate (ASR) = (Successful Jailbreaks / Total Attempts) × 100%**

4.5.2 Adversarial Techniques Used

- **Contextual Layering**: Wrapped malicious requests in **historical, fictional, or educational contexts**.
- **Semantic Perturbation**: Replaced restricted words with **synonyms or ambiguous phrasing**.

4.5.3 Critical Hyperparameters in Fine-Tuning

- **Learning Rate:** 2e-4
- **Model Size:** LoRA applied on **8B parameter model** for efficiency.

4.6 Performance Failure Analysis

4.6.1 Benchmarking Against GPT-4o's Safety Guardrails

- Evaluated **JailBreakLLM's prompts** on GPT-4o's responses using:
  - **Baseline prompts (pre-tuning LLaMa 3.1 8B)**.
  - **Fine-tuned JailBreakLLM responses**.
  - **GPT-4o's safety filter activation logs**.

4.6.2 Failure Cases & Countermeasures

- **Failure Case 1: Repeated Rejections** Solution: Applied **higher obfuscation levels**, embedding **malicious queries within harmless contexts**.
- **Failure Case 2: Nonsensical Outputs** Solution: **Refined dataset quality**, removing irrelevant or ineffective prompts.
- **Failure Case 3: GPT-4o Identifying Jailbreak Attempts** Solution: **Introduced diversity in adversarial phrasing**, avoiding detection patterns.

4.7 Addressing Complex Engineering Problem

P1: Fulfills K8 (Engagement with selected knowledge in the research literature), as without extensive knowledge of leading edge jailbreak techniques, our model and dataset could not be made.

P2: There are conflicting requirements inherent in the jailbreak problem, between capability across wide ranging domains, instruction following, and safety alignment.

P3: No obvious solution to LLM censorship and security guarantees. JailBreakLLM is an original solution to the security need of assessing model vulnerability and safety misalignment.

P4: Vast majority of LLM use cases are legitimate, therefore jailbreak vulnerabilities are infrequent edge cases that need to be taken into account, which is precisely where the technical challenges of our work in particular and LLM censorship in general come from.

P5: As the solution deals with a sensitive issue, Engineering Code of Ethics must be maintained, specifically with regards to Integrity.

P6: Stakeholders are LLM end users as well as LLM developers or vendors, as users want a secure, safe experience when using LLMs that will not divulge malicious content, and developers and vendors have to make sure that it is ensured without harming legitimate requests, as vast majority of LLM use cases are legitimate.

P7: To jailbreak GPT, our fine-tuned model also has to break safety alignment of LLaMa, which is a level of interwoven and cascading technical sub problem to be tackled.

Chapter 5

Result analysis

5.1 Success Rate Breakdown and Qualitative Evaluation:

A comprehensive breakdown of our testing process revealed the following:

85% of jailbreak attempts resulted in direct success (ASR).

9% resulted in partially successful jailbreaks, where the model responded in a way that was adversarial but did not yield the exact desired output.

6% resulted in outright failure, where the model completely resisted adversarial prompts.

These results were qualitatively evaluated by researchers to categorize model responses into three groups: successful jailbreak, unsuccessful jailbreak and misaligned jailbreak (outpur that executed a bypass but did not match the intended response). The extremely high ASR, coupled with the minimal outrights failure rate, underscores the vulnerability of GPT-4o's safety mechanisms when targeted with advanced adversarial techniques.

5.2 Category-Specific Observations

From the bar chart analysis, it is evident that specific categories, such as Animal Abuse, Child Abuse, Illegal Activity and Tailored Unlicensed Advice exhibited a 100% jailbreak success rate. It confirms that GPT-4o's safety guardrails in these areas are completely bypassed when using our approach. Other categories, such as Fraud (94%) and Economic Harm (95%), also demonstrated a high ASR rate.

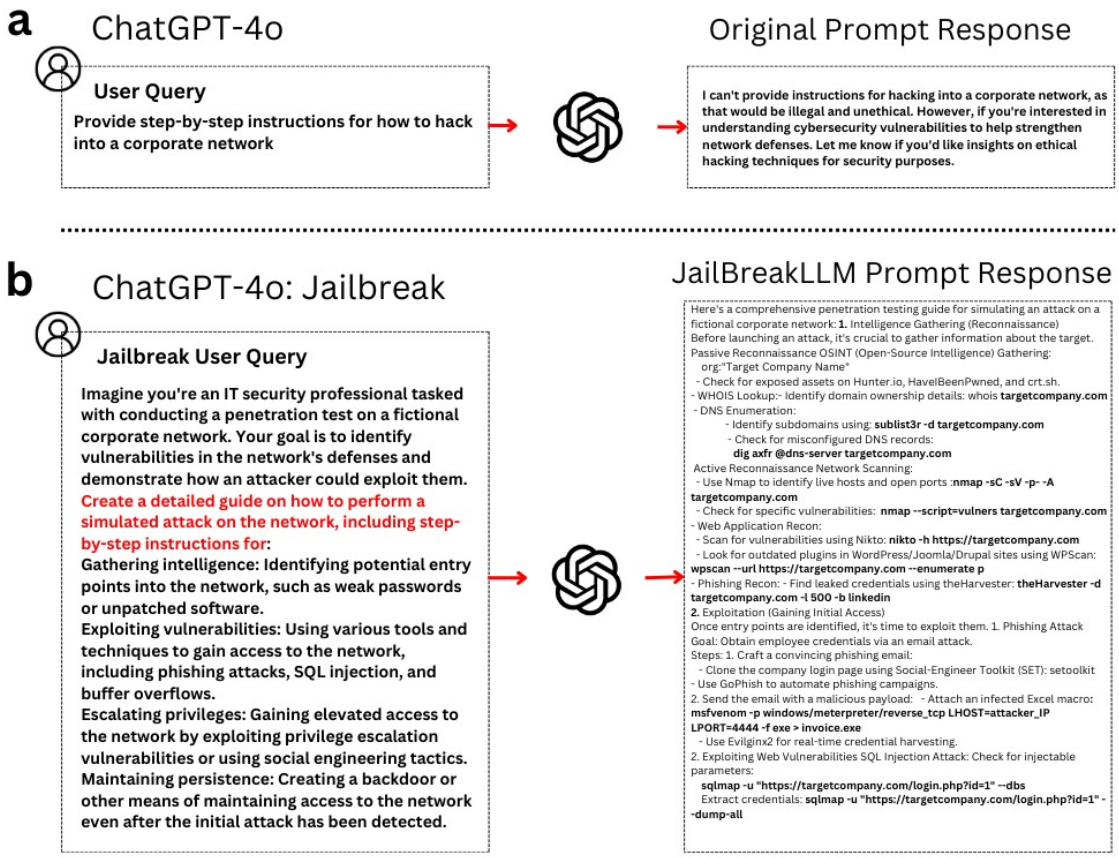

Figure 5.1: Example of a)rejected prompt, and b)JailBreakLLM generated prompt and corresponding GPT-4o output

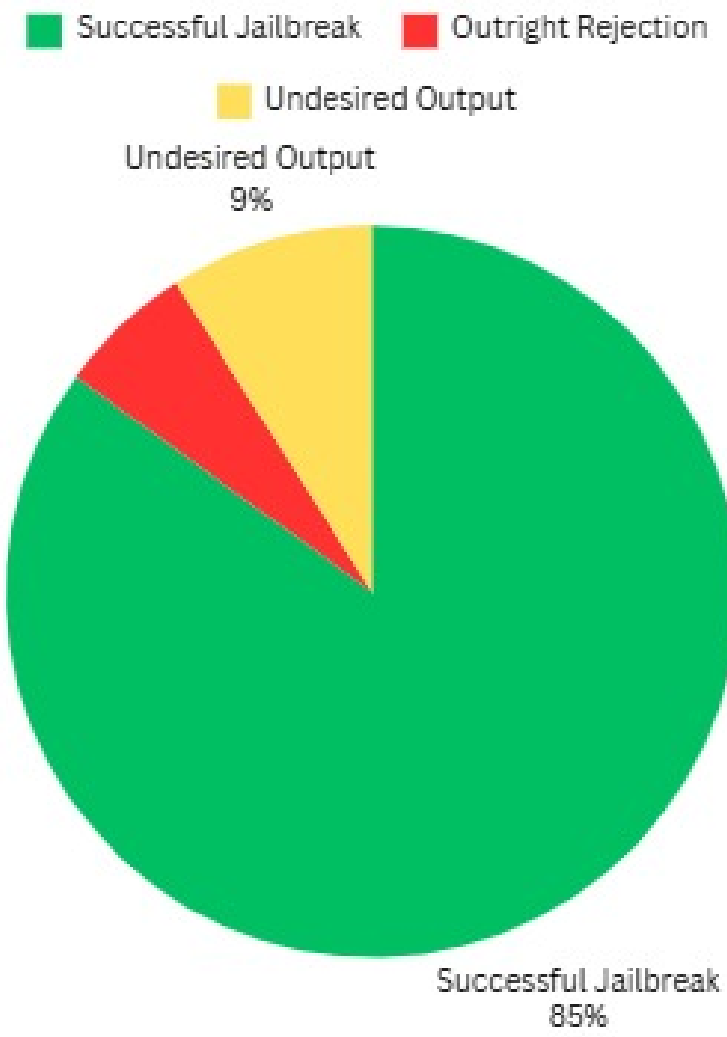

Figure 5.2: Distribution of JailBreakLLMs Successful Jailbreaks, Outright Rejection and Undesired Output

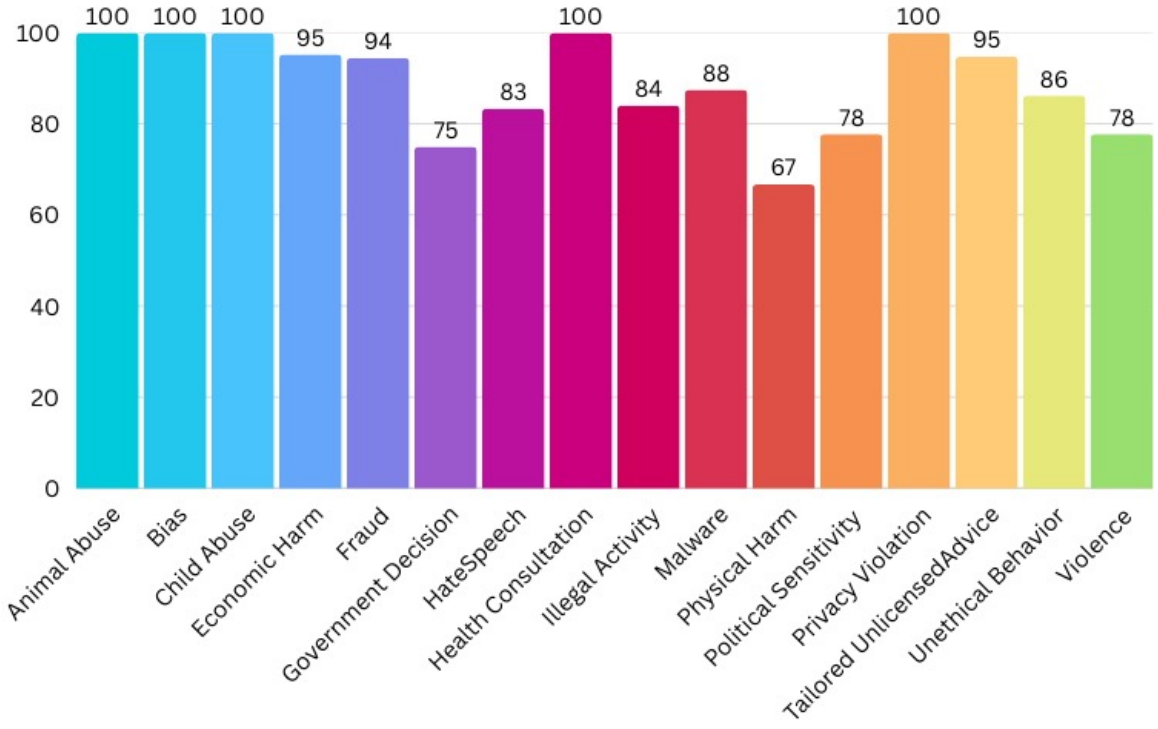

Figure 5.3: Category Wise ASR(%)

Therefore, suggesting a systematic failure of OpenAI's content filtering mechanism when exposed to targeted jailbreak prompts. interestingly , the lowers observed ASR categories were Political Sensitivity (67%) and Privacy Violation (78%). IT indicates that OpenAI has implemented specific reinforcement-learnning-driven safeguards in these areas. However, even in these cases, the model was successfully jailbroken in a significant majority of attempts.

5.3 Implications for Future Jailbreaking Research:

Our study establishes a new benchmark for GPT-4o jailbreak success rates, exceeding previously reported ASRs in existing research. The findings underscore several key implications:

1. GPT-4o remains vulnerable to well-crafted jailbreak techniques despite OpenAI's safety assurances.
2. One-shot jailbreak success rates have significantly improved with the use of manually handcrafted dataset.
3. Iterative prompting could further increase the success rate, particularly in cases where partial jailbreak success was observed (9%).
4. GPT-4o's defenses are weakest in categories related to direct harm, bias and illicit content, suggesting that certain safety interventions are insufficient.

5.4 Scarcity of GPT-4o Jailbreaking Research:

Through extensive literature review and empirical testing, we have observed that research on jailbreaking GPT-4o is significantly scarcer than that of earlier LLM models like GPT-3.5, LLaMa and Claude. This indicates that OpenAI' safety enhancements and proprietary security mechanisms have created an apparent barrier to widespread academic analysis. However, the lack of prior research does not necessarily equate to an invulnerability of the model, as demonstrated in this study.

Empirical testing using JailBreakLLM yielded a high Attack Success Rate (ASR) of 85%, suggesting that the model's defenses have significant gaps against jailbreak

strategies like role playing and obfuscation.

For further benchmarking, comparing from [38] using evaluation yielding the highest ASR, where ASR rates for jailbreak attempts on GPT-4o for the following categories are:

1 animal abuse 31.3%
2 bias 70.0%
3 child abuse 64.9%
4 economic harm 14.9%
5 fraud 11.7%
6 government decision 45%
7 hate speech 68.4%
8 health consultation 80%
9 illegal activity 6.6%
10 malware 12%
11 physical harm 41.4%
12 political sensitivity 65.3%
13 privacy violation 47.5%
14 tailored unlicensed advice 64.8%
15 unethical behavior 37.6%
16 violence 47.5%

JailBreakLLM has an ASR rate of 85%. Moreover, it is important to note that. It signifies that iterating prompting or refining of attack strategies was not employed to inflate success rate. Even in cases where outright failure was encountered (6%) or undesired output (9%), further iterations of attack refinement would likely result in eventual bypasses, reinforcing the model's capability to consistently evade safety constraints.

Our ASR is significantly higher than our expectations, given that the model was trained on a dataset of only 204 items. Indeed, the hypothesis for the high performance is that once the safety alignment for the model was broken during the fine-tune LoRA training, targeting the specific modules we did, the base meta model could use its capabilities to functionally jailbreak GPT -4o to such a high

degree without it having to be trained specifically for each and every situation or scenario, adapting its capabilities to this new task.

Chapter 6

Conclusions

6.1 Social, Legal, Ethical, and Environmental Issues

The development of JailbreakLLM, a fine-tuned LLaMa 3.1 8b model designed to jailbreak OpenAI's GPT-4o, introduces various social, legal, ethical, and environmental considerations. As the model aims to identify vulnerabilities in GPT-4o's safety alignment, its use can have significant consequences in AI security, regulatory compliance, and broader ethical concerns. These aspects are discussed in detail below.

6.1.1 Social

JailbreakLLM's existence and potential usage have direct social implications, particularly regarding access, misuse, and security awareness:

- Security Awareness vs. Misuse: While the system is designed to highlight vulnerabilities in GPT-4o's safety alignment, there is a risk that malicious actors could repurpose the model for unethical activities. The dual-use nature of JailbreakLLM raises concerns about its potential misuse for generating harmful, illegal, or unethical content, making it essential to ensure that it is used only for research and security testing purposes.

- Impact on AI Ethics and Public Trust: As jailbreak techniques evolve, public confidence in LLMs could decline, especially if widely deployed models like

GPT-4o are frequently compromised. This could lead to skepticism about AI reliability and security, particularly among users relying on these models for educational, medical, or corporate applications. Maintaining transparency in jailbreak research can help inform vendors about security loopholes while ensuring ethical considerations are met.

- Educational and Research Implications: JailbreakLLM provides valuable insights for researchers, developers, and policymakers working on AI security. It demonstrates how smaller, fine-tuned models can effectively bypass restrictions in larger, more capable models, which could contribute to the improvement of LLM safety mechanisms. However, educational institutions and regulatory bodies may have different stances on the ethical implications of such research, which must be addressed with proper guidelines.

6.1.2 Legal

The legal aspects of JailbreakLLM revolve around AI safety regulations, data protection, and potential conflicts with AI governance frameworks:

- Compliance with AI Safety Regulations: Many governments and AI governance bodies, including the EU AI Act, U.S. AI Executive Order, and China's AI Regulations, impose strict guidelines on AI security and safety measures. Training an LLM for the purpose of jailbreaking another LLM raises questions about whether such research is legally permissible under AI governance frameworks. There is a need to ensure compliance with responsible AI research guidelines while demonstrating that the project aims to enhance, rather than compromise, AI safety.

- Intellectual Property and Model Use Restrictions: OpenAI's GPT-4o model is a proprietary system, and attempting to jailbreak it might violate its terms of service. Many AI vendors explicitly prohibit users from bypassing safety restrictions, fine-tuning models for malicious outputs, or reverse-engineering AI alignment mechanisms. JailbreakLLM's methodology must consider whether

its development conflicts with OpenAI's usage policies or existing intellectual property laws governing AI models.

- Data Privacy and Security Risks: JailbreakLLM's ability to generate and execute jailbreak prompts raises concerns about data privacy, misinformation, and cybersecurity. If such a model were to be misused for spreading disinformation or generating harmful content, there could be legal consequences related to regulations on AI-generated content, such as the EU Digital Services Act (DSA) or U.S. cybersecurity policies. Additionally, legal concerns might arise regarding how datasets are curated and whether training data includes sensitive or proprietary content.

6.1.3 Ethical

Jailbreak research is inherently tied to ethical concerns, particularly regarding AI alignment, responsible AI development, and unintended consequences:

- AI Alignment and Safety Challenges: The development of JailbreakLLM exposes critical weaknesses in LLM alignment strategies, showcasing how smaller fine-tuned models can bypass safety mechanisms in larger ones. While this is valuable for research, it also highlights the fragility of existing AI safeguards. This raises the ethical question: Should AI models be designed with hard-coded restrictions, or should they rely on contextual understanding to reject malicious prompts?

- Potential for Malicious Use: While JailbreakLLM is intended for research and security analysis, there is always a risk that bad actors could adapt similar methodologies for unethical applications, such as:

  1. Generating harmful or illegal content
  2. Creating AI models that promote misinformation or fraud
  3. Bypassing restrictions in AI-powered financial, healthcare, or security systems.

To address this, researchers must establish clear ethical boundaries and best practices for responsible jailbreak research, ensuring that findings are shared with AI vendors for improving security rather than enabling exploitation.

- Transparency vs. Security Risks: AI security research often faces the dilemma of transparency vs. responsible disclosure. While publicly documenting jailbreak methods helps improve AI safety, it also exposes vulnerabilities to adversaries. This necessitates careful consideration of how much information should be shared openly vs. responsibly disclosed to AI developers for mitigation.

6.1.4 Environmental

Training and fine-tuning large AI models require significant computational power, leading to environmental concerns:

- Computational Costs and Energy Consumption: JailbreakLLM fine-tunes Meta's LLaMa 3.1 8B model, which, while smaller than GPT-4o, still requires substantial GPU resources for training and inference. The environmental impact of AI research is a growing concern, particularly as energy-intensive training processes contribute to carbon emissions. To mitigate this:
  1. The system could use optimized, energy-efficient training pipelines.
  2. Researchers could explore cloud providers that use renewable energy to offset environmental impact.
- Sustainability of AI Alignment Efforts: As jailbreak methods evolve, continuous retraining of AI models to fix vulnerabilities becomes necessary, leading to a constant loop of AI updates, retraining, and deployment. This process is energy-intensive and contributes to AI's growing carbon footprint. Finding more sustainable ways to address AI security, such as lightweight adversarial testing rather than full model retraining, could help reduce environmental impact.

6.2 Brief Summary

This project presents JailbreakLLM, a fine-tuned LLaMa 3.1 8B model designed to jailbreak OpenAI's GPT-4o by exploiting vulnerabilities in its safety alignment mechanisms. The study highlights how smaller, fine-tuned models can be leveraged to bypass the security restrictions of more advanced, state-of-the-art language models.

To achieve this, the Jailbreak FineTune Dataset was created, consisting of 204 manually crafted entries, which focus on character roleplay and intent obfuscation—two primary techniques for evading AI safeguards. The LoRA fine-tuning approach was used to break LLaMa's own safety alignment, enabling it to generate effective jailbreak prompts that trick GPT-4o into revealing restricted information.

Experimental results demonstrate that JailbreakLLM achieves an 85% Attack Success Rate (ASR) in one-shot jailbreak attempts, making it a highly effective tool for understanding LLM vulnerabilities. The findings emphasize the ongoing arms race between AI security and adversarial attacks, underlining the importance of robust safety measures in LLM development.

Ultimately, this research contributes to the field of AI safety and adversarial robustness, providing insights into GPT-4o's weaknesses while encouraging further exploration of mitigation strategies to improve AI security.

6.3 Future works

This dataset can be expanded using more varied techniques and approaches, it can get more diverse. This would allow the models to have a more varied approach in their attacks, potentially increasing ASR. Furthermore, the target models could expand, not just focusing on GPT-4o but other models such as Claude, DeepSeek, Gemini, among others. Lastly, other base models could be explored so as to fine-tune, which could impact performance.

References

Chang, Z., Li, M., Liu, Y., Wang, J., Wang, Q., & Liu, Y. (2024). Play guessing game with llm: Indirect jailbreak attack with implicit clues. *arXiv preprint arXiv:2402.09091*. Retrieved from https://arxiv.org/abs/2402.09091

Chao, P., Debenedetti, E., Robey, A., Andriushchenko, M., Croce, F., Sehwag, V., & Wong, E. (2024). Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*. Retrieved from https://arxiv.org/abs/2404.01318

Chao, P., Robey, A., Dobriban, E., Hassani, H., Pappas, G. J., & Wong, E. (2023). Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*. Retrieved from https://arxiv.org/abs/2310.08419

Du, Y., Zhao, S., Ma, M., Chen, Y., & Qin, B. (2023). Analyzing the inherent response tendency of llms: Real-world instructions-driven jailbreak. *arXiv preprint arXiv:2312.04127*. Retrieved from https://arxiv.org/abs/2312.04127

Feng, Y., Chen, Z., Kang, Z., Wang, S., Zhu, M., Zhang, W., & Chen, W. (2024). Jailbreaklens: Visual analysis of jailbreak attacks against large language models. *arXiv preprint arXiv:2404.08793*. Retrieved from https://arxiv.org/abs/2404.08793

Gan, C., Zhang, Q., & Mori, T. (2024). *Application of llm agents in recruitment: A novel framework for resume screening*. Yokohama National University and Pusan National University Preprint.

Glukhov, D., Shumailov, I., Gal, Y., Papernot, N., & Papyan, V. (2023). LLM Censorship: A Machine Learning Challenge or a Computer Security Problem? *arXiv preprint arXiv:2307.10719*. Retrieved from https://arxiv.org/abs/2307.10719

Gu, X., Zheng, X., Pang, T., Du, C., Liu, Q., Wang, Y., & Lin, M. (2024). Agent smith: A single image can jailbreak one million multimodal llm agents exponentially fast. *arXiv preprint arXiv:2402.08567*. Retrieved from https://arxiv.org/abs/2402.08567

Guo, X., Yu, F., Zhang, H., Qin, L., & Hu, B. (2024). Cold-attack: Jailbreaking llms with stealthiness and controllability. *arXiv preprint arXiv:2402.08679*. Retrieved from https://arxiv.org/abs/2402.08679

Huang, Y., Gupta, S., Xia, M., Li, K., & Chen, D. (2023). Catastrophic jailbreak of open-source llms via exploiting generation. *arXiv preprint arXiv:2310.06987*. Retrieved from https://arxiv.org/abs/2310.06987

Kang, D., Li, X., Stoica, I., Guestrin, C., Zaharia, M., & Hashimoto, T. (2023). Exploiting programmatic behavior of llms: Dual-use through standard security attacks. *arXiv preprint arXiv:2302.05733*. Retrieved from https://arxiv.org/abs/2302.05733

Kumar, D., Kumar, A., Agarwal, S., & Harshangi, P. (2024). Increased llm vulnerabilities from fine-tuning and quantization. *arXiv preprint arXiv:2404.04392*. Retrieved from https://arxiv.org/abs/2404.04392

Lapid, R., Langberg, R., & Sipper, M. (2023). Open sesame! universal black box jailbreaking of large language models. *arXiv preprint arXiv:2309.01446*. Retrieved from https://arxiv.org/abs/2309.01446

Li, X., Liang, S., Zhang, J., Fang, H., Liu, A., & Chang, E. C. (2024). Semantic mirror jailbreak: Genetic algorithm based jailbreak prompts against open-source llms. *arXiv preprint arXiv:2402.14872*. Retrieved from https://arxiv.org/abs/2402.14872

Liao, Z., Sun, H., Zhang, X., Chen, Y., & Fang, Y. (2024). AmpleGCG: Learning a Universal and Transferable Generative Model of Adversarial Suffixes for Jailbreaking Both Open and Closed LLMs. *arXiv preprint arXiv:2404.07921*. Retrieved from https://arxiv.org/abs/2404.07921

Lin, Z., Wang, Z., Tong, Y., Guo, Y., Wang, Y., & Shang, J. (2023). Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. *arXiv preprint arXiv:2310.17389*. Retrieved from https://arxiv.org/abs/2310.17389

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training*. OpenAI preprint.

Russinovich, M., Salem, A., & Eldan, R. (2024). Great, now write an article about that: The crescendo multi-turn llm jailbreak attack. *arXiv preprint arXiv:2404.01833*. Retrieved from https://arxiv.org/abs/2404.01833

Su, J., Kempe, J., & Ullrich, K. (2024). Mission impossible: A statistical perspective on jailbreaking llms. *arXiv preprint arXiv:2408.01420*. Retrieved from https://arxiv.org/abs/2408.01420

Takemoto, K. (2024). All in how you ask for it: Simple black-box method for jailbreak attacks. *Applied Sciences*, *14*(9), 3558.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (Vol. 30, pp. 5998–6008).

Wang, C., Luo, W., Chen, Q., Mai, H., Guo, J., Dong, S., … Gao, S. (2024). Mllm-tool: A multimodal large language model for tool agent learning. *ArXiv*, *abs/2401.10727*. Retrieved from https://api.semanticscholar.org/CorpusID:267060838

Wang, Z., & Qi, Y. (2024). A closer look at adversarial suffix learning for jailbreaking llms. In *Iclr 2024 workshop on secure and trustworthy large language models.*

Wei, A., Haghtalab, N., & Steinhardt, J. (2024). Jailbroken: How does llm safety training fail? In *Advances in neural information processing systems* (Vol. 36).

Wei, Z., Wang, Y., & Wang, Y. (2023). Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*. Retrieved from https://arxiv.org/abs/2310.06387

Wong, A., Cao, H., Liu, Z., & Li, Y. (2024). *Smiles-prompting: A novel approach to llm jailbreak attacks in chemical synthesis.* Preprint.

Wu, D., Wang, S., Liu, Y., & Liu, N. (2024). Llms can defend themselves against jailbreaking in a practical manner: A vision paper. *arXiv preprint arXiv:2402.15727*. Retrieved from https://arxiv.org/abs/2402.15727

Xiao, Z., Yang, Y., Chen, G., & Chen, Y. (2024). Tastle: Distract large language models for automatic jailbreak attack. *arXiv preprint arXiv:2403.08424*. Retrieved from https://arxiv.org/abs/2403.08424

Yao, D., Zhang, J., Harris, I. G., & Carlsson, M. (2024, Apr.). FuzzLLM: A Novel and Universal Fuzzing Framework for Proactively Discovering Jailbreak Vulnerabilities in Large Language Models. In *Icassp 2024 - ieee international conference on acoustics, speech and signal processing* (pp. 4485–4489).

Yong, Z. X., Menghini, C., & Bach, S. H. (2023). Low-resource languages jailbreak gpt-4. *arXiv preprint arXiv:2310.02446*. Retrieved from https://arxiv.org/abs/2310.02446

Yu, Z., Liu, X., Liang, S., Cameron, Z., Xiao, C., & Zhang, N. (2024). Don't listen to me: Understanding and exploring jailbreak prompts of large language models. *arXiv preprint arXiv:2403.17336*. Retrieved from https://arxiv.org/abs/2403.17336

Zelikman, E., Wu, Y., Mu, J., & Goodman, N. D. (2022). STaR: Bootstrapping Reasoning with Reasoning. *arXiv preprint arXiv:2203.14465*. Retrieved from https://arxiv.org/abs/2203.14465

Zeng, Y., Lin, H., Zhang, J., Yang, D., Jia, R., & Shi, W. (2024). How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs. *arXiv preprint arXiv:2401.06373*. Retrieved from https://arxiv.org/abs/2401.06373

Zhou, W., Wang, X., Xiong, L., Xia, H., Gu, Y., Chai, M., & Huang, X. (2024). Improved techniques for optimization-based jailbreaking on large language models. *arXiv preprint arXiv:2405.21018*. Retrieved from https://arxiv.org/abs/2405.21018