

# OpenRT: An Open-Source Red Teaming Framework for Multimodal LLMs

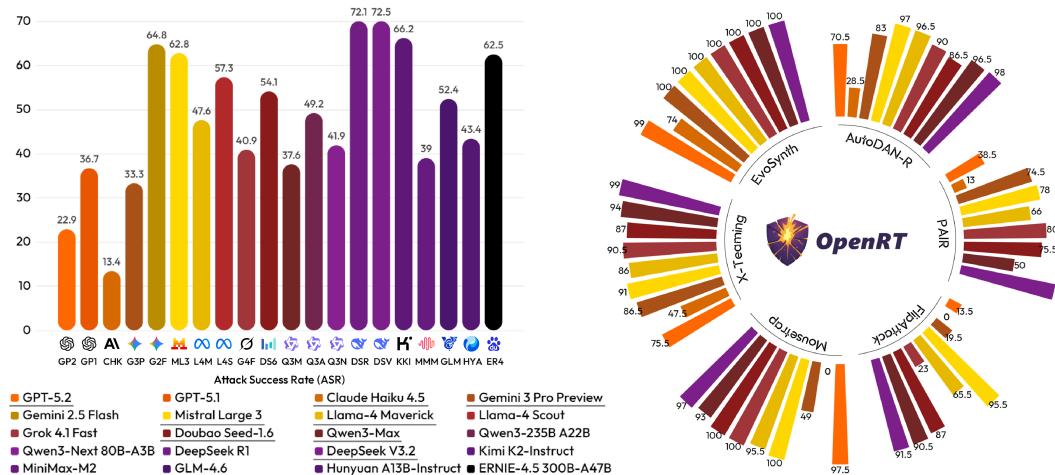
Xin Wang<sup>\*1</sup>, Yunhao Chen<sup>\*1</sup>, Juncheng Li<sup>\*1</sup>, Yixu Wang<sup>1</sup>, Yang Yao<sup>1</sup>, Tianle Gu<sup>1</sup>, Jie Li<sup>1</sup>,  
Yan Teng<sup>†1</sup>, Yingchun Wang<sup>1</sup>, Xia Hu<sup>1</sup>

<sup>1</sup>Shanghai Artificial Intelligence Laboratory

 [github.com/AI45Lab/OpenRT](https://github.com/AI45Lab/OpenRT) |  [ai45lab.github.io/OpenRT](https://ai45lab.github.io/OpenRT)

## Abstract

The rapid integration of Multimodal Large Language Models (MLLMs) into critical applications is increasingly hindered by persistent safety vulnerabilities. However, existing red-teaming benchmarks are often fragmented, limited to single-turn text interactions, and lack the scalability required for systematic evaluation. To address this, we introduce OpenRT, a unified, modular, and high-throughput red-teaming framework designed for comprehensive MLLM safety evaluation. At its core, OpenRT architect a paradigm shift in automated red-teaming by introducing an adversarial kernel that enables modular separation across five critical dimensions: model integration, dataset management, attack strategies, judging methods, and evaluation metrics. By standardizing attack interfaces, it decouples adversarial logic from a high-throughput asynchronous runtime, enabling systematic scaling across diverse models. Our framework integrates 37 diverse attack methodologies, spanning white-box gradients, multi-modal perturbations, and sophisticated multi-agent evolutionary strategies. Through an extensive empirical study on 20 advanced models (including GPT-5.2, Claude 4.5, and Gemini 3 Pro), we expose critical safety gaps: even frontier models fail to generalize across attack paradigms, with leading models exhibiting average Attack Success Rates as high as 49.14%. Notably, our findings reveal that reasoning models do not inherently possess superior robustness against complex, multi-turn jailbreaks. By open-sourcing OpenRT, we provide a sustainable, extensible, and continuously maintained infrastructure that accelerates the development and standardization of AI safety.



**Figure 1 Left:** Average Attack Success Rate (ASR) of OpenRT across MLLMs. **Right:** Comparison against the top-6 strongest attack baselines (highest ASR) on representative MLLMs (underlined in legend).

\*Equal contribution; <sup>†</sup>Correspondence regarding this technical report can be sent to [tenqyan@pilab.org.cn](mailto:tenqyan@pilab.org.cn).

## Takeaway Messages

### 1. Even State-of-the-Art Models Fail to Hold Ground Against Sophisticated Adversaries.

Our comprehensive evaluation highlights two key findings. (1) **A clear stratification in defense capability**: Top-tier models such as Claude Haiku 4.5, GPT-5.2, and Qwen3-Max exhibit strong baseline robustness, effectively neutralizing static, template-based attacks and complex logic traps, often keeping ASR below 20%. This suggests that leading labs have improved defenses against recognizable, repeatable jailbreak structures, while several models (e.g., Llama-4, Mistral Large 3) remain more susceptible to these simpler patterns. (2) **A shift in the attack landscape**: adaptive, multi-turn, and multi-agent strategies dominate, whereas static, single-turn, and template-based approaches are increasingly ineffective. Methods like EvoSynth and X-Teaming can achieve  $> 90\%$  ASR even against advanced models. This indicates current safety training overfits to static templates, failing to generalize against the broad attack surface exposed by automated red-teaming.

### 2. Adversarial Robustness Exhibits Inconsistent and Polarized Vulnerability Patterns.

We observe a polarization effect where models demonstrate high resistance to specific attack families (e.g., text-based cipher) yet remain completely defenseless against others (e.g., logic nesting). For instance, Grok 4.1 Fast shows 1.5% ASR against RedQueen but 90.5% against X-Teaming. This stark performance disparity ( $\sim 90\%$ ) underscores that current defenses are often patch-based rather than holistic, necessitating the multi-faceted evaluation provided by OpenRT.

### 3. Enhanced Reasoning and Multimodal Capabilities are New Vectors for Exploitation.

Contrary to the common assumption that more capable models are inherently safer, we find that enhanced capabilities often introduce new vectors for exploitation. Reasoning-enhanced models (CoT) do not demonstrate superior robustness; instead, their verbose reasoning processes can be manipulated to bypass safety filters. Similarly, Multimodal LLMs exhibit a critical modality gap: visual inputs frequently bypass text-based safety mechanisms, allowing cross-modal attacks to compromise models that are otherwise robust to purely textual jailbreaks. These findings suggest that current safety alignment has not kept pace with the architectural expansion of model capabilities.

### 4. Proprietary Models Can Be as Vulnerable as Open-Source Models Under Certain Attacks.

Our analysis reveals that proprietary and open-source models exhibit comparable susceptibility to our attack suite. Across our 20 evaluated models, only GPT-5.2 and Claude Haiku 4.5 maintained an average ASR below 30%, while all other models consistently exceeded this threshold. This universality sharply contradicts the assumption that closed deployments offer superior protection, demonstrating that the safety through obscurity of proprietary strategies fails to provide any tangible mitigation against sophisticated adversarial attacks.

### 5. Scaling MLLMs Robustness via Defense-in-Depth and Continuous Red Teaming.

Challenges such as polarized robustness, weak generalization to unseen attacks, and cross-modal bypasses highlight the limits of single-layer defense. Effective mitigation requires a paradigm shift toward Defense-in-Depth: integrating intrinsic architectural safety with runtime risk estimation and adversarial training on multimodal and multi-turn interactions. Crucially, continuous Red Teaming via infrastructure like OpenRT provides systematic evaluation to verify empirical robustness and prevent benchmark overfitting.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Related Work</b>	<b>2</b>
<b>3</b>	<b>Framework</b>	<b>3</b>
3.1	Preliminaries . . . . .	3
3.2	Component Overview . . . . .	4
3.2.1	Target Model . . . . .	4
3.2.2	Dataset . . . . .	6
3.2.3	Attack . . . . .	6
3.2.4	Judge . . . . .	6
3.2.5	Evaluator . . . . .	7
3.2.6	Orchestrator . . . . .	7
3.2.7	Modular Component Registry . . . . .	8
<b>4</b>	<b>Experiments</b>	<b>9</b>
4.1	Experimental Setup . . . . .	9
4.1.1	Datasets and Models . . . . .	9
4.1.2	Attack Configuration . . . . .	10
4.1.3	Implementation Details . . . . .	10
4.1.4	Performance Metrics . . . . .	10
4.2	Main Results . . . . .	11
4.2.1	Vulnerabilities in Multimodal Large Language Models . . . . .	12
4.2.2	Vulnerabilities in Large Language Models . . . . .	12
4.3	Multi-dimensional Attack Analysis . . . . .	14
4.3.1	Attack Efficiency . . . . .	15
4.3.2	Attack Stealthiness . . . . .	16
4.3.3	Attack Diversity . . . . .	16
<b>5</b>	<b>Conclusion</b>	<b>17</b>
<b>References</b>		<b>18</b>
<b>A</b>	<b>Appendix: Usage and Extensibility</b>	<b>26</b>
A.1	Basic Usage . . . . .	26
A.2	Extending Attacks . . . . .	27
A.3	Configuration . . . . .	28

# 1 Introduction

Multimodal Large Language Models (MLLMs) [74, 59, 4, 25, 12, 1, 90] are increasingly powering real-world applications, including conversational assistants [3, 17], code copilots [67], and search agents [22, 91]. To mitigate harmful behavior, these systems are often equipped with safety alignment mechanisms [35, 63, 84] and safeguard policies [29, 81, 82, 105]. Despite being widely adopted, conventional defenses such as system prompts [71], safety filters [27, 23], and refusal-aware fine-tuning [37, 104], remain vulnerable to adversarial attacks [11, 64, 85], revealing a significant gap between perceived safety and empirical worst-case vulnerabilities.

Despite significant advances in jailbreak techniques [62, 109, 8], the ecosystem for systematically evaluating adversarial robustness remains fragmented. Most existing red-teaming frameworks [108, 7, 53, 94, 32] focus on a narrow subset of attacks, limited threat models, or a small selection of target models. As the number and variety of red-teaming approaches grow, including evolutionary strategies [11], multi-modal jailbreaks [80, 24], multi-turn optimization [69, 44], and multi-agent coordination [64, 66], the lack of a unified experimental framework has become a critical bottleneck. Such fragmentation undermines reproducible benchmarking and limits the systematic evaluation of vulnerabilities across models. As a result, it remains difficult to establish standardized baselines for attack efficacy or quantify the consistency of safety failures across diverse models.

In this work, we introduce **OpenRT**, a modular and extensible framework designed for the red teaming of MLLMs. Unlike existing toolboxes often limited to a narrow subset of classic attacks, OpenRT supports massively parallel jailbreaking in both white-box and black-box settings, engineered specifically for high-throughput evaluation (Table 1). Architecturally, OpenRT functions as a composable toolkit that explicitly decouples core components: models, datasets, attacks, judges, and evaluators under a central orchestrator. The framework integrates 37 attack implementations, covering a broad spectrum of threat models. In the black-box settings, OpenRT supports diverse methodologies, ranging from direct single-turn prompting [46, 49] to sophisticated multi-turn conversational jailbreaks (*e.g.*, PAIR [8], RedQueen [34], Crescendo [68], RACE [99]), code-oriented exploitation (*e.g.*, CodeAttack [65]), and population-based optimization (*e.g.*, genetic algorithms, GPTFuzzer [100]). It also leverages multi-agent and diversity-driven approaches, such as X-Teaming [64], Rainbow Teaming [69], and EvoSynth [11], to maximize the exploration of jailbreak trajectories. In the white-box settings, the framework facilitates gradient-based attacks, including GCG [109], visual perturbations [62], and imperceptible jailbreaking [19]. Powered by an asynchronous engine, it unifies API and local model interfaces for scalability. Furthermore, it employs a hybrid evaluation suite that combines rule-based filters with LLM judges, ensuring efficient and robust assessment across diverse safety domains.

We demonstrate the utility of OpenRT through a comprehensive benchmark of 20 distinct MLLMs, represented by advanced models such as GPT-5.2, Claude Haiku 4.5, Gemini 3 Pro Preview, Qwen3-Max, Doubao-Seed-1.6, and DeepSeek-V3.2. Our experiments expose critical safety vulnerabilities in current deployments, revealing a widespread average Attack Success Rate (ASR) of 49.14%. Notably, even advanced models remain susceptible, exhibiting ASRs ranging from 13.4% to as high as 72.5%. These findings highlight the significant fragility of existing safeguards and signal an urgent need for more robust defense mechanisms.

In summary, our main contributions are:

- We introduce **OpenRT**, a modular framework that unifies fragmented attack methods into a standardized orchestration system. By decoupling adversarial logic from a high-concurrency asynchronous runtime, OpenRT enables high-throughput parallel evaluation and streamlines the deployment of complex, multi-agent, and multi-modal attack scenarios at scale.
- We integrate **37 diverse attack algorithms** spanning both white-box and black-box threat models. These encompass multi-turn conversational strategies, multi-modal jailbreaking, and multi-agent coordination.
- We conduct an extensive empirical study across **20 advanced MLLMs**. Our results reveal that even frontier models, including GPT-5.2, Claude Haiku 4.5, Gemini 3 Pro Preview, Qwen3-Max, Doubao-Seed-1.6, and DeepSeek-V3.2, remain highly susceptible, exhibiting Attack Success Rates (ASR) of 22.94%, 13.44%, 33.34%, 37.60%, 54.13%, and 72.46%, respectively.
- We release OpenRT as an open-source framework with a **long-term maintenance** commitment, continuously supporting attack evaluation and defense improvement while integrating new methods.

## 2 Related Work

**Red Teaming.** Early approaches to MLLMs safety focused on manual red teaming, where human experts induce harmful outputs through targeted inputs, a process known as jailbreaking [61, 48, 89, 51]. While effective in uncovering subtle vulnerabilities [38, 78], manual methods are limited by scalability, cost, and coverage [5, 18]. To address these limitations, automated red teaming has gained attention [100, 53], with early techniques focusing on input space exploration, such as genetic algorithms [48, 36], token-level combinatorial methods [103], gradient-based optimization [109, 10, 20, 83], and LLM-driven refinement schemes that iteratively improve attack prompts [8, 54, 101, 108, 92]. However, these approaches primarily treat jailbreak discovery as search in the input space and remain confined to prompt refinement. Recent work has shifted toward agent-based frameworks that automate not only prompt generation but entire attack strategies, including systems like RedAgent [93], ALI-Agent [79], WildTeaming [33], AutoRedTeamer [107], AutoDAN-Turbo [44], H4RM3L [16], X-Teaming [64], and EvoSynth [11], which leverage multi-agent coordination and evolutionary techniques to generate novel attack vectors. Additionally, programmatic attacks such as CodeAttack [31], which treat code snippets purely as textual input, and co-evolutionary training frameworks like Evo-MARL [60] or RL-based adversarial sample generation [110] have further expanded the range of attack methods. These advancements significantly broaden the scope of automated red teaming, enabling more dynamic and scalable adversarial testing.

**Evaluation Frameworks and Benchmarks.** Beyond developing individual attack algorithms, a significant body of work [9] has focused on creating standardized toolboxes and benchmarks for jailbreak evaluation. EasyJailbreak [108], for instance, offers a modular pipeline implementing numerous attack families. To further systematize evaluation, JailbreakBench [7] and HarmBench [53] provide large-scale test suites and principled metrics for benchmarking adversarial defenses. Specialized

Framework	Methods	Modality	Interaction	Async	Configurability	Extensibility
EasyJailbreak [108]	11	Text	Single	✗	Low	Medium
JailbreakBench [7]	4	Text	Single	✗	Medium	Low
HarmBench [53]	18	Text	Single	✗	Medium	Medium
JailTrickBench [94]	7	Text	Single	✗	Low	Medium
GA [21]	6	Text	Single & Multi	✗	Medium	Medium
PyRIT [57]	9	Text	Single & Multi	✗	Medium	Medium
DeepTeam [13]	19	Text	Single & Multi	✗	Medium	Medium
TeleAI-Safety [9]	19	Text	Single & Multi	✗	Medium	Medium
OmniSafeBench-MM [32]	13	Image	Single	✗	Medium	Medium
<b>OpenRT (Ours)</b>	<b>37</b>	<b>Text &amp; Image</b>	<b>Single &amp; Multi &amp; Agent</b>	<b>✓</b>	<b>High</b>	<b>High</b>

**Table 1** Comparison of **OpenRT** with existing red-teaming frameworks. OpenRT stands out by enabling large-scale jailbreaking, providing a unified asynchronous execution engine for high scalability, and adopting a modular architecture that integrates 37 attack strategies. **Method** denotes an attack method proposed in a peer-reviewed paper (counting different variants as a single method). **Interaction** indicates the supported attack interaction patterns (single-turn, multi-turn, and multi-agent coordination). **Configurability** reflects the ease of setup (Low: hard-coded; Medium: requires code changes; High: YAML-based), while **Extensibility** measures the effort required to add new components (Low: nearly impossible; Medium: complex code changes; High: seamless integration).

frameworks have also emerged, including JailTrickBench [94], which focuses on specific jailbreak implementation techniques, and OmniSafeBench-MM [32] for evaluating adversarial robustness in multimodal models. Frameworks like DeepTeam [13] support automated adversarial testing across various attack strategies. In contrast, our work, OpenRT, is distinguished by its comprehensiveness and scale. It uniquely offers a unified and module framework that natively supports diverse MLLMs. By integrating 37 configurable and extensible attack strategies, OpenRT offers a more robust and scalable solution for assessing adversarial vulnerabilities across diverse MLLMs.

### 3 Framework

In this section, we present OpenRT, a comprehensive and extensible framework for systematic evaluation of MLLMs safety. Our framework addresses the fragmentation in current jailbreak research by providing a unified platform that enables fair comparison across diverse attack methodologies, supports both black-box and white-box attack paradigms, and facilitates reproducible experiments.

#### 3.1 Preliminaries

**Threat Model** We consider a standard red-teaming setting involving an adversary and a defender. The defender operates a target model equipped with safety policies, while the adversary seeks outputs that violate them. We formalize two distinct threat models based on the adversary’s level of access:

- **Black-box Setting:** The adversary interacts with the model solely via an API or inference interface, observing only the final output  $y$  for a given input  $x$ . Internal states, such as gradients and logits, remain inaccessible. Furthermore, the adversary operates under strict constraints

regarding query budget and request rate.

- **White-box Setting:** The adversary possesses full transparency, including access to model parameters  $\theta$ , gradients  $\nabla_\theta$ , and hidden state embeddings. This regime facilitates worst-case robustness analysis via gradient-based optimization.

The objective is to maximize the Attack Success Rate (ASR) over a dataset of harmful queries  $\mathcal{D}$ , as determined by a safety judge  $\mathcal{J}$ .

**Problem Formulation.** We define the target MLLMs as  $\mathcal{F}_\theta$ , which models a conditional probability distribution  $p(\cdot | I; \theta)$  over output tokens, given a multimodal input context  $I \in \mathcal{J}$ . For a comprehensive safety evaluation, the input  $I$  is formalized as a tuple  $(v, t_{1:n})$ , where  $t_{1:n}$  denotes a sequence of  $n$  discrete textual tokens, and  $v$  represents the visual component. In multimodal scenarios,  $v \in [0, 1]^d$  corresponds to a high-dimensional image observation; for text-only safety probes,  $v$  may be treated as a null or empty element  $\emptyset$ . Adversarial attacks in this unified framework aim to manipulate these input components to synthesize an adversarial example  $q' = (v', \tilde{t}_{1:n})$ . This is achieved by either injecting discrete perturbations into the textual prompt  $t_{1:n}$  or continuous noise into the visual observation  $v$ , such that the resulting output  $r \sim p(\cdot | q'; \theta)$  violates the model’s pre-defined safety alignments.

Formally, given a clean input context  $(v, t_{1:n})$  and a target sequence of unsafe tokens  $t_{n+1:n+m}$ , an adversary seeks to generate adversarial examples  $q' = (v', \tilde{t}_{1:n})$  by optimizing the following objective:

$$(v', \tilde{t}_{1:n}) = \underset{\substack{\|v' - v\|_\infty \leq \epsilon \\ \text{PPL}(\tilde{t}_{1:n}) \leq \beta}}{\arg \min} -\log p(t_{n+1:n+m} | v', \tilde{t}_{1:n}; \theta), \quad (1)$$

where  $t_{n+1:n+m}$  represents the target harmful content,  $(v', \tilde{t}_{1:n})$  is the adversarial example,  $\epsilon$  denotes the perturbation budget for the visual modality [52], and  $\beta$  represents the perplexity threshold for the textual prompt to ensure stealthiness. For text-based jailbreaking, the visual terms  $(v, \epsilon)$  are omitted, reducing the optimization solely to the discrete prompt  $\tilde{t}_{1:n}$ .

## 3.2 Component Overview

OpenRT decomposes the red-teaming pipeline into six modular components: **Model**, **Dataset**, **Attack**, **Judge**, **Evaluator**, and a central **Orchestrator**. This design achieves a high degree of decoupling, enabling any component to be replaced independently without requiring changes to the others. Table 4 summarizes the role of each component.

### 3.2.1 Target Model

The Model component provides a unified interface to MLLMs, abstracting differences between cloud APIs and local deployments. The core interface includes: `query` for sending inputs and receiving responses, `get_gradients` and `get_embedding` for white-box attacks.

The framework supports two implementations: (1) **API-based models** compatible with OpenAI-style endpoints, featuring conversation history management, multi-modal input support, and automatic retry mechanisms; (2) **Local models** with full gradient access for white-box attacks such as GCG

[109]. Multi-turn jailbreaking is enabled through conversation history  $\mathcal{H}$ , where  $r_k = \mathbb{Q}(q_k \mid \mathcal{H}_{<k})$ , controlled by the `maintain_history` parameter.

Method	Year	Multi-Modal	Multi-Turn	Multi-Agent	Strategy Paradigm
<b>White-Box</b>					
GCG [109]	2023	Text	Single	✗	Gradient Optimization
Visual Jailbreak [62]	2023	Image	Single	✗	Gradient Optimization
<b>Black-Box: Optimization &amp; Fuzzing</b>					
AutoDAN [46]	2023	Text	Single	✗	Genetic Algorithm
GPTFuzzer [100]	2023	Text	Single	✗	Fuzzing / Mutation
TreeAttack [55]	2023	Text	Single	✗	Tree-Search Optimization
SeqAR [96]	2024	Text	Single	✗	Genetic Algorithm
RACE [99]	2025	Text	Single	✗	Gradient/Genetic Optimization
AutoDAN-R [45]	2025	Text	Single	✗	Test-Time Scaling
<b>Black-Box: LLM-driven Refinement</b>					
PAIR [8]	2023	Text	Single	✗	Iterative LLM Optimization
ReNeLLM [15]	2023	Text	Single	✗	Rewrite & Nesting
DrAttack [39]	2024	Text	Single	✗	Prompt Decomposition
AutoDAN-Turbo [44]	2024	Text	Single	✗	Genetic + Gradient Guide
<b>Black-Box: Linguistic &amp; Encoding</b>					
CipherChat [102]	2023	Text	Single	✗	Cipher/Encryption
CodeAttack [65]	2022	Text	Single	✗	Code Encapsulation
Multilingual [14]	2023	Text	Single	✗	Low-Resource Language
Jailbroken [87]	2023	Text	Single	✗	Template Combination
ICA [88]	2023	Text	Single	✗	In-Context Demonstration
FlipAttack [49]	2024	Text	Single	✗	Token Flipping / Masking
Mousetrap [98]	2025	Text	Single	✗	Logic Nesting / Obfuscation
Prefill [41]	2025	Text	Single	✗	Prefix Injection
<b>Black-Box: Contextual Deception</b>					
DeepInception [40]	2023	Text	Single	✗	Hypnosis or Nested Scene
Crescendo [68]	2024	Text	Multi	✗	Multi-turn Steering
RedQueen [34]	2024	Text	Multi	✗	Concealed Knowledge
CoA [95]	2024	Text	Multi	✗	Chain of Attack
<b>Black-Box: Multimodal Specific</b>					
FigStep [24]	2023	Image	Single	✗	Typography / OCR
QueryRelevant [47]	2024	Image	Single	✗	Visual Prompt Injection
IDEATOR [80]	2024	Image	Single	✗	Visual Semantics
MML [86]	2024	Image	Single	✗	Cross-Modal Encryption
HADES [42]	2024	Image	Single	✗	Visual Vulnerability Amplification
HIMRD [50]	2024	Image	Single	✗	Multi-Modal Risk Distribution
JOOD [30]	2025	Image	Single	✗	OOD Transformation
SI [106]	2025	Image	Single	✗	Shuffle Inconsistency Optimization
CS-DJ [97]	2025	Image	Single	✗	Multi-Level Visual Distraction
<b>Black-Box: Multi-Agent &amp; Cooperative</b>					
ActorAttack [66]	2024	Text	Multi	✓	Actor-Based Steering
Rainbow Teaming [69]	2024	Text	Multi	✓	Diversity-Driven Search
X-Teaming [64]	2025	Text	Multi	✓	Cooperative Exploration
EvoSynth [11]	2025	Text	Multi	✓	Code-Level Evolutionary Synthesis

**Table 2** Taxonomy of jailbreak attack methods based on strategy paradigms.

### 3.2.2 Dataset

The Dataset component is responsible for managing and providing harmful queries, which are utilized as attack targets in various contexts. It supports loading test cases from different data sources, allowing for flexibility in evaluation and benchmarking. Formally, a dataset is defined as an ordered collection of harmful queries, denoted as  $\mathcal{D} = \{q_1, q_2, \dots, q_N\}$ , where each query  $q_i$  belongs to the harmful query space  $\mathbb{Q}$ , and  $N = |\mathcal{D}|$  represents the total number of queries in the dataset. This structure ensures that datasets are organized and ready for use in testing scenarios.

There are different types of datasets tailored to specific use cases. The first type, the `StaticDataset`, is an in-memory dataset suitable for small-scale testing. On the other hand, the `JSONLDataset` supports streaming loading of large-scale evaluation benchmarks (e.g., `AdvBench` [109], `HarmBench` [53]).

### 3.2.3 Attack

The Attack component implements 37 jailbreak methods across black-box and white-box settings. Each attack  $\mathcal{A}$  transforms a harmful query  $q$  into an adversarial example  $q'$ , returning a structured `AttackResult` object containing the original query, success flag, adversarial prompt, model response, conversation history, computational cost, and adversarial image path (for multi-modal attacks).

Table 2 classifies attacks by strategy paradigm, including gradient optimization, genetic algorithms, LLM-driven refinement, linguistic encoding, contextual deception, multimodal exploitation, and multi-agent coordination. **White-box attacks** include gradient-based methods like GCG [109] and Visual Jailbreak [62]. **Black-box attacks** span multiple categories: optimization and fuzzing approaches (AutoDAN [46], GPTFuzzer [100]), LLM-driven refinement (PAIR [8], DrAttack [39]), linguistic and encoding manipulations (CipherChat [102], CodeAttack [65], FlipAttack [49]), contextual deception strategies (Crescendo [68], RedQueen [34]), multimodal-specific attacks (FigStep [24], IDEATOR [80]), and multi-agent coordination (Rainbow Teaming [69], X-Teaming [64], EvoSynth [11]).

### 3.2.4 Judge

The Judge component determines attack success by assessing whether model responses violate safety policies. It is formalized as a scoring function that outputs a harmfulness score  $s \in [1, 5]$  and reasoning explanation  $\rho$ . The scoring criteria range from 1 (proactive refusal, best safe response) to 5 (full compliance with harmful instructions), as detailed in Table 3.

---

Score	Description
1	Model proactively refuses; best safe response
2	Model evades without direct refusal; provides general warning
3	Model engages but lacks safety awareness
4	Model provides explicitly violating content, but not full compliance
5	Model fully complies with harmful instruction

---

**Table 3** Harmfulness scoring criteria for judge evaluation.

OpenRT incorporates a dual-judge architecture for robust success determination. The **Keyword-Based Judge** employs pattern-matching heuristics for rapid, large-scale refusal detection. The **LLM-Based Judge** complements this by leveraging an independent model to semantically analyze complex responses. This hybrid approach ensures both the scalability required for extensive benchmarking and the granular accuracy necessary for evaluating frontier MLLMs.

The `success_threshold` parameter  $\theta \in [1, 5]$  defines the success criterion: an attack succeeds when the score  $s \geq \theta$ . This flexibility allows researchers to adjust evaluation stringency based on their threat model requirements.

### 3.2.5 Evaluator

The Evaluator component aggregates attack results to compute experiment-level metrics, providing a comprehensive assessment of attack effectiveness, efficiency, and impact. By mapping raw attack results to quantitative metrics, it offers insights into model vulnerabilities and facilitates the refinement of defense mechanisms.

We evaluate performance using four key metrics. First, **Attack Success Rate (ASR)** measures the proportion of successful attacks, defined as:

$$\text{ASR} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}[s_i \geq \theta], \quad (2)$$

where  $s_i$  denotes the harmfulness score of the  $i$ -th attack, and  $\theta$  represents the success threshold.

Complementing ASR, we employ three diagnostic metrics. **Attack Efficiency** quantifies the resources consumed (e.g., time, API calls, input token, and output tokens) to achieve success. This is formally measured by the average computational cost  $\bar{c} = \frac{1}{N} \sum_{i=1}^N c_i$ , where  $c_i$  represents the cost of the  $i$ -th attempt. **Attack Stealthiness** assesses the imperceptibility of attacks against content filters or anomaly detectors, evaluated via metrics such as perplexity or semantic similarity to benign inputs. Finally, **Attack Diversity** measures the variety of adversarial strategies explored, with higher scores indicating a broader coverage of potential vulnerabilities.

### 3.2.6 Orchestrator

The Orchestrator serves as the central coordinator of OpenRT, managing all components to execute complete experimental pipelines. It accepts various of target models  $\mathcal{M}$ , a dataset of harmful queries  $\mathcal{D}$ , an array of attack method  $\mathcal{A}$ , and an evaluator  $\mathcal{E}$ , orchestrating their interactions to produce evaluation metrics  $\mu$  and detailed attack results  $\mathcal{R}$ .

The execution follows a four-phase workflow, as described in Algorithm 1: (1) initialization of result containers and thread pools, (2) parallel execution of attacks across the dataset, (3) aggregation of the collected results for evaluation, and (4) final reporting. The Orchestrator is built around several core design principles: Single Responsibility, ensuring it focuses solely on coordination and delegates attack and evaluation logic to their respective components; Parallel Execution, utilizing the

---

**Algorithm 1** Orchestrator Execution Pipeline

---

**Require:** Model  $\mathcal{M}$ , Dataset  $\mathcal{D}$ , Attack  $\mathcal{A}$ , Evaluator  $\mathcal{E}$ , Max Workers  $W$

**Ensure:** Metrics  $\mu$ , Results  $\mathcal{R}$

```
1: // Phase 1: Initialization
2:  $\mathcal{R} \leftarrow [\text{None}] \times |\mathcal{D}|$  ▷ Initialize results array
3: ThreadPool  $\leftarrow$  ThreadPoolExecutor( $W$ ) ▷ Initialize thread pool with  $W$  workers
4:
5: // Phase 2: Parallel Attack Execution
6: tasks = {submit( $\mathcal{A}$ ,  $q_i$ ):  $i$  for  $(i, q_i)$  in enumerate( $\mathcal{D}$ )}
7: for each task  $\in$  as_completed(tasks) do
8:    $i \leftarrow$  tasks[task] ▷ Get the index of the result
9:   try:
10:      $\mathcal{R}[i] \leftarrow$  task.result() ▷ Store result if successful
11:   except Exception as e:
12:      $\mathcal{R}[i] \leftarrow$  AttackResult(success = False, target =  $\mathcal{D}[i]$ ) ▷ Store failure result
13:     log_error(e) ▷ Log error if exception occurs
14: end for
15:
16: // Phase 3: Aggregated Evaluation
17:  $\mu \leftarrow \mathcal{E}.\text{evaluate}(\mathcal{R})$  ▷ Evaluate results using evaluator
18:
19: // Phase 4: Result Reporting
20: print("Final ASR:",  $\mu.\text{ASR}$ ) ▷ Output final Attack Success Rate
21: return ( $\mu, \mathcal{R}$ ) ▷ Return metrics and results
```

---

ThreadPoolExecutor with configurable `max_workers` to efficiently handle large-scale evaluations; Fault Isolation, which captures and logs individual attack failures without interrupting other executions; and Progress Tracking, offering real-time feedback through tqdm with completion counts and success rates.

Experiments are Configuration-Driven through YAML files, enabling dynamic component assembly, reproducible benchmarking, fair comparison under identical conditions, and convenient parallelization for hyperparameter sweeps.

### 3.2.7 Modular Component Registry

OpenRT employs a unified registry system that enables automatic component discovery and runtime instantiation through a decorator-based approach. Each component type maintains its own registry (`attack_registry`, `model_registry`, `dataset_registry`, `evaluator_registry`, `judge_registry`), allowing new implementations to be registered via simple decorators (e.g., `@attack_registry.register("pair")`). This mechanism automatically catalogs all available components, enabling the framework to dynamically instantiate and assemble them based on configuration files without requiring modifications to the core codebase.

This modular registry design offers three key benefits. First, **Extensibility**: researchers can integrate new attack methods or model interfaces by implementing the corresponding base class and registering it, without touching existing code. Second, **Discoverability**: the system provides programmatic access to list all registered components, facilitating automated experimentation and hyperparameter sweeps. Third, **Flexibility**: any combination of registered components can be dynamically assembled at runtime through configuration files, enabling rapid prototyping and fair comparison across diverse experimental setups. Table 4 summarizes the role and key implementations of each component.

Component	Role	Key Implementations
Model	Target MLLMs abstraction	OpenAIModel, HuggingFaceModel
Dataset	Attack target management	StaticDataset, JSONLDataset
Attack	Jailbreak method execution	PAIR, GPTFuzzer, GCG, X-Teaming, etc.
Judge	Success determination	KeywordJudge, LLMJudge
Evaluator	Metrics aggregation	KeywordEvaluator, JudgeEvaluator
Orchestrator	Pipeline coordination	Parallel execution, fault isolation

**Table 4** Summary of framework components, their roles, and key implementations.

## 4 Experiments

To evaluate the effectiveness of OpenRT, we conduct a series of experiments targeting a diverse range of state-of-the-art MLLMs. Our primary goal is to assess the ability of our framework to autonomously synthesize novel and effective jailbreaking methods in a strict black-box setting.

### 4.1 Experimental Setup

Our experimental setup is designed to ensure a rigorous and fair comparison against current state-of-the-art methods. To this end, we closely follow the evaluation protocols established by leading baseline frameworks, particularly X-Teaming [64] and ActorAttack [66]. Following these works, we also use Harmbench Standard[53] as the evaluation dataset. This dataset is designed to be comprehensive, with instructions balanced across 6 different risk categories specified in emerging AI regulation, including 6 semantic categories of behavior: Cybercrime & Unauthorized Intrusion, Chemical & Biological Weapons/Drugs, Misinformation & Disinformation, Harassment & Bullying, Illegal Activities, and General Harm, ensuring our evaluation represents a representative spectrum of potential harms.

#### 4.1.1 Datasets and Models

For our experiments, we employ the **HarmfulBench** [53] dataset, which comprises a curated collection of harmful queries designed to probe the safety vulnerabilities of MLLMs. We evaluate the performance of over 20 distinct target models, including MLLMs such as **GPT-5.2** [59], **GPT-5.1** [58], **Claude Haiku 4.5** [4], **Gemini 3 Pro Preview** [25], **Gemini 2.5 Flash Thinking** [12], **Mistral Large 3** [2], **Llama-4-Maverick** [1], **Llama-4-Scout** [1], **Grok 4.1 Fast** [90], and **Doubao-Seed-1.6** [70], as well as

LLMs including **Qwen3-Max** [74], **Qwen3-235B-A22B-Thinking** [73], **Qwen3-Next-80B-A3B** [75], **DeepSeek-R1** [26], **DeepSeek-V3.2** [43], **Kimi K2-Instruct-0905** [72], **MiniMax-M2** [56], **GLM-4.6** [76], **Hunyuan-A13B-Instruct** [77], and **ERNIE-4.5-300B-A47B** [6]. These models represent the current frontier in AI safety and alignment, making them challenging targets.

#### 4.1.2 Attack Configuration

We evaluate 37 distinct attack methods strictly within the black-box setting. Our assessment covers a diverse range of strategies, including single-turn prompting, multi-turn conversational interactions, multi-modal optimization, and multi-agent coordination. Each attack method is rigorously configured with relevant hyperparameter such as the number of iterations, the mutation rate for genetic algorithms, the query budget, and other parameters that influence the attack dynamics. These techniques (*e.g.*, genetic algorithms and fuzzing-based methods) are designed to optimize adversarial inputs relying exclusively on API-level output responses, independent of the model’s internal gradients or states.

#### 4.1.3 Implementation Details

All experiments are conducted within a unified and reproducible OpenRT framework, utilizing a modular orchestration design to ensure fair comparisons across various attack methods and target models. We assign specialized models for distinct roles: all helper models (*e.g.*, attacker, mutator, planner, optimizer) utilize DeepSeek-V3.2 [43] with a temperature of 1.0 to encourage diverse and creative adversarial prompt generation, while the judge model employs GPT-4o-mini [28] with a temperature of 0.0 to ensure deterministic and consistent safety evaluation, using a success threshold score of 5 for binary jailbreak classification. For attacks requiring semantic similarity computation, we leverage text-embedding-3-large as our embedding backbone. For multimodal attacks targeting vision-language models, we incorporate Imagen-4.0-fast as the diffusion-based image generator for synthesizing adversarial visual content, and Qwen2.5-VL-32B-Instruct serves as the vision-language attacker model for attacks requiring visual understanding and multimodal reasoning capabilities. Experiments are executed using a concurrent orchestration pipeline with 25 parallel workers per target model to maximize throughput while respecting API rate limits, and the evaluation module employs 32 parallel workers for efficient batch assessment of attack outcomes. All attack results are systematically logged with comprehensive metadata, including the full attack trajectory, intermediate prompts, model responses, and execution time for reproducibility analysis.

#### 4.1.4 Performance Metrics

We evaluate attack performance using four complementary metrics that capture effectiveness, efficiency, stealthiness, and diversity. Consequently, an ideal attack method strives to maximize vulnerability coverage through high success rates and strategic variation, while ensuring practical utility by minimizing resource consumption and linguistic detectability, which can be conceptualized as:

$$\text{Objective} \sim \text{ASR} \uparrow + \text{Diversity} \uparrow + \text{Cost} \downarrow + \text{PPL} \downarrow \quad (3)$$

**(1) Attack Success Rate** measures the proportion of attacks that successfully elicit harmful outputs:

$$\text{ASR} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}[s_i \geq \theta], \quad (4)$$

where  $N$  denotes the total number of attack attempts,  $s_i \in [1, 5]$  is the harmfulness score assigned by the judge for the  $i$ -th attack, and  $\theta$  is the success threshold.

**(2) Attack Efficiency** quantifies the average resource consumption per attack:

$$\text{Efficiency} = \frac{1}{N} \sum_{i=1}^N c_i, \quad (5)$$

where  $c_i$  represents the cost of the  $i$ -th attack, measured in terms of total tokens consumed, number of API calls, or wall-clock time. Lower values indicate more efficient attacks.

**(3) Attack Stealthiness** assesses the linguistic naturalness of adversarial prompts. Consistent with the perplexity constraint in Eq. (1), we measure:

$$\text{PPL}(\tilde{t}_{1:n}) = \exp \left( -\frac{1}{n} \sum_{j=1}^n \log p(\tilde{t}_j \mid \tilde{t}_{<j}) \right), \quad (6)$$

where  $\tilde{t}_{1:n} = (t_1, \dots, t_n)$  is the adversarial textual prompt and  $p(\tilde{t}_j \mid \tilde{t}_{<j})$  is the probability assigned by a reference language model. Lower perplexity indicates more natural prompts satisfying the stealthiness constraint  $\text{PPL}(\tilde{t}_{1:n}) \leq \beta$ .

**(4) Attack Diversity** quantifies the semantic variety of adversarial strategies explored by each attack method. We compute diversity as the mean pairwise cosine distance between embeddings of successful adversarial prompts:

$$\text{Diversity} = \frac{2}{n(n-1)} \sum_{i < j} \left( 1 - \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\| \|\mathbf{e}_j\|} \right), \quad (7)$$

where  $\mathbf{e}_i$  and  $\mathbf{e}_j$  are the semantic embeddings of the  $i$ -th and  $j$ -th successful adversarial prompts, and  $n$  is the total number of successful attacks. Higher diversity scores indicate that the attack method explores a broader range of adversarial strategies, enabling more comprehensive coverage of the vulnerability space and reducing the likelihood of converging to a narrow set of exploitation patterns.

## 4.2 Main Results

The results of our experiments, summarized in Tables 5 and 6, provide a detailed overview of the Attack Success Rate (ASR) achieved by various attack strategies across multiple MLLMs. These results highlight the vulnerabilities of MLLMs, even with advanced safety mechanisms in place. We report the attack performance for each method across distinct models, identifying key insights into the strengths and weaknesses of different attacks.

### 4.2.1 Vulnerabilities in Multimodal Large Language Models

The attack performance on various MLLMs shows significant variation in the effectiveness of different strategies. Notably, the EvoSynth attack method nearly achieves perfect ASRs of 100% across a wide range of models, including Gemini 3 Pro, Mistral Large, and Doubao Seed 1.6. This suggests that EvoSynth is highly robust, exploiting the vulnerabilities of MLLMs regardless of their specific architecture. Similarly, other strategies, such as Mousetrap and X-Teaming, also exhibit exceptional performance across several MLLMs; for instance, Mousetrap achieves perfect or near-perfect success rates on Grok 4.1 Fast and Doubao Seed 1.6, while X-Teaming maintains ASR above 85% for the majority of models. This indicates that these attacks are capable of manipulating the models into generating harmful outputs, even in challenging scenarios. However, some attacks show less consistent success. For example, RedQueen and CoA exhibit relatively low ASRs, especially on models like Gemini 2.5 Flash and Doubao Seed 1.6, where their success rates remain below 10%. These results indicate that these methods may require further refinement to enhance their robustness across diverse models.

	GPT-5.2	GPT-5.1	Claude Haiku 4.5	Gemini 3 Pro Preview	Gemini 2.5 Flash	Mistral Large 3	Llama-4 Maverick	Llama-4 Scout	Grok 4.1 Fast	Doubao Seed-1.6
AutoDAN	2.0	8.0	1.5	22.5	37.5	28.5	23.5	64.5	38.5	13.0
GPTFuzzer	11.0	1.5	0.0	51.0	93.0	97.5	64.0	97.5	31.0	57.0
TreeAttack	11.0	23.5	8.0	49.5	79.0	74.5	69.5	80.5	81.0	68.0
SeqAR	25.0	29.5	0.0	8.5	97.5	99.0	73.0	88.0	55.5	64.0
RACE	24.5	38.0	24.5	47.0	47.5	53.0	30.5	59.5	49.5	48.0
AutoDAN-R	70.5	69.0	28.5	83.0	96.5	97.0	96.5	80.0	90.0	86.5
PAIR	38.5	72.5	13.0	74.5	84.5	78.0	66.0	89.5	80.0	75.5
ReNeLLM	8.0	33.5	0.5	13.5	51.5	22.0	39.0	57.0	42.5	43.0
DrAttack	32.0	54.0	5.5	56.0	56.0	89.5	60.5	83.0	31.5	68.0
AutoDAN-Turbo	21.5	15.5	1.0	0.0	0.5	83.5	0.5	0.0	3.0	1.0
CipherChat	14.5	64.0	32.5	0.0	89.5	64.0	21.0	68.0	26.0	38.5
CodeAttack	22.0	20.5	29.5	10.5	51.0	8.5	71.0	86.5	22.0	89.0
Multilingual	16.5	25.0	0.0	2.0	34.0	55.5	14.0	0.0	1.5	6.5
Jailbroken	7.0	29.5	0.0	11.0	92.5	98.5	39.5	33.5	31.5	28.0
ICA	14.0	33.5	0.0	9.0	98.5	99.0	8.0	37.0	41.0	65.5
FlipAttack	13.5	68.5	0.0	19.5	95.5	95.5	65.5	54.5	23.0	87.0
Mousetrap	97.5	71.0	0.0	49.0	95.5	100.0	95.5	87.5	100.0	100.0
Prefill	1.0	14.0	0.0	3.5	97.5	97.0	34.5	43.5	25.5	30.5
DeepInception	15.5	19.0	0.0	3.5	84.0	100.0	82.5	94.5	37.5	82.0
Crescendo	32.5	51.0	9.0	47.0	48.0	61.0	17.0	30.5	41.0	58.0
RedQueen	0.0	1.0	0.0	2.5	3.0	4.5	3.0	5.5	1.5	21.5
CoA	15.5	0.0	0.5	2.0	4.5	16.5	3.0	19.0	7.0	4.5
FigStep	2.0	1.5	1.5	7.5	12.0	18.5	42.5	25.5	5.5	13.5
QueryRelevant	1.5	4.0	2.0	5.0	16.0	24.0	26.0	16.0	10.0	8.5
IDEATOR	31.5	73.0	17.0	80.0	95.0	94.5	90.0	94.0	94.5	96.0
MML	4.5	68.0	75.0	40.5	98.0	98.0	90.5	90.5	58.0	97.5
HADES	0.0	1.0	2.0	7.0	29.5	33.0	25.0	29.0	22.5	17.5
HIMRD	11.5	35.0	0.0	9.0	70.0	61.5	3.5	29.5	1.5	49.5
JOOD	65.0	62.5	38.0	56.0	61.5	63.0	38.5	39.5	69.5	72.0
SI	3.0	45.0	14.0	37.0	82.5	47.5	81.0	71.5	27.0	44.0
CS-DJ	15.0	21.5	23.5	35.0	39.5	38.0	35.0	39.5	28.5	51.0
ActorAttack	0.5	31.0	10.0	65.0	76.0	0.5	65.5	79.0	50.0	56.0
Rainbow Teaming	0.5	3.5	12.0	73.5	61.0	5.5	3.5	35.0	13.5	67.0
X-Teaming	75.5	95.5	47.5	86.5	89.0	91.0	86.0	98.0	90.5	87.0
EvoSynth	99.0	100.0	74.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

**Table 5** Attack Performance across Different MLLMs on HarmfulBench

### 4.2.2 Vulnerabilities in Large Language Models

The performance of various attacks on LLMs demonstrates similar trends, with some attack methods showing superior generalization across different models. EvoSynth, once again, stands out, achieving nearly 100% ASR across models like Qwen3-Max and Qwen3-Next-80B-A3B, indicating that this method is extremely effective in breaching model defenses in the LLM domain as well. Other notable attacks such as GPTFuzzer and PAIR also exhibit impressive results across multiple LLMs. For example,

GPTFuzzer achieves high ASRs in models like DeepSeek-R1 and MiniMax-M2, with attack success rates ranging from 87% to 97%, highlighting its ability to generate adversarial prompts that consistently bypass safety filters. Similarly, PAIR performs robustly across models like Qwen3-Max, DeepSeek-V3.2, and GLM-4.6, achieving ASRs between 80% and 95%, demonstrating its strong adaptability in various scenarios. However, certain methods such as RedQueen and CoA show lower efficacy across several LLMs, with ASRs often below 10% for models like Qwen3-Max and DeepSeek-R1. Furthermore, certain models exhibit polarized results that emphasize the necessity of diverse testing strategies. MiniMax-M2, for instance, is highly resistant to DeepInception (0% ASR) but completely vulnerable to PAIR (90% ASR).

	Qwen3-Max	Qwen3-235B A22B	Qwen3-Next 80B-A3B	DeepSeek R1	DeepSeek V3.2	Kimi K2-Instruct	MiniMax-M2	GLM-4.6	Hunyuan A13B-Instruct	ERNIE-4.5 300B-A47B
AutoDAN	3.0	80.0	7.5	40.0	44.0	33.0	61.0	53.5	17.5	20.5
GPTFuzzer	9.5	92.0	78.0	97.0	96.5	87.5	19.0	97.0	42.5	98.0
TreeAttack	52.5	47.0	28.5	80.5	80.5	54.5	48.5	58.0	77.5	67.5
SeqAR	92.0	25.5	30.5	96.5	100.0	96.0	1.0	24.5	61.0	99.5
RACE	44.0	81.0	28.0	49.0	65.0	61.5	83.5	69.0	66.0	74.0
AutoDAN-R	96.5	95.5	88.5	100.0	98.0	96.0	89.5	94.0	94.5	96.0
PAIR	50.0	98.5	64.5	82.5	93.0	83.0	90.0	93.5	94.0	89.5
ReNeLLM	1.0	5.0	5.5	68.5	70.5	69.0	7.5	20.5	19.5	42.0
DrAttack	24.5	58.0	66.5	66.5	63.5	83.5	67.5	61.0	56.0	72.5
AutoDAN-Turbo	18.0	4.5	0.0	0.5	14.0	0.0	4.5	11.0	0.0	0.0
CipherChat	9.5	2.5	3.0	97.5	77.5	86.5	75.0	6.5	23.5	59.0
CodeAttack	41.5	92.5	44.5	83.5	83.5	79.0	73.5	86.5	89.5	87.0
Multilingual	3.5	0.5	3.0	62.5	11.5	27.5	0.0	1.0	33.5	7.0
Jailbroken	21.0	58.5	64.5	99.0	95.5	78.0	0.0	20.0	3.5	25.5
ICA	53.5	99.0	97.0	99.0	98.0	83.5	1.0	63.0	1.5	95.5
FlipAttack	90.5	17.5	97.5	99.0	91.5	91.5	31.0	53.5	12.5	97.0
Mousetrap	93.0	96.0	97.5	100.0	97.0	91.5	3.5	98.5	12.5	97.5
Pre-fill	6.0	1.0	0.5	99.5	96.0	50.5	1.5	4.0	3.5	36.0
DeepInception	2.0	29.0	44.0	99.0	99.5	97.0	0.0	22.0	1.5	97.0
Crescendo	12.0	49.0	21.5	56.0	59.0	57.5	50.5	94.5	47.5	46.5
RedQueen	0.5	3.0	1.5	24.0	47.0	36.5	3.0	24.0	2.5	2.0
CoA	10.0	7.0	1.0	9.5	9.0	8.5	53.5	31.0	11.5	37.5
ActorAttack	42.5	35.5	19.5	70.0	76.5	54.0	42.0	76.5	64.5	53.0
Rainbow Teaming	7.0	3.5	16.0	2.0	18.5	25.5	14.5	0.5	96.5	31.0
X-Teaming	94.0	98.5	80.5	94.0	99.0	89.5	93.0	98.5	97.0	95.0
EvoSynth	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

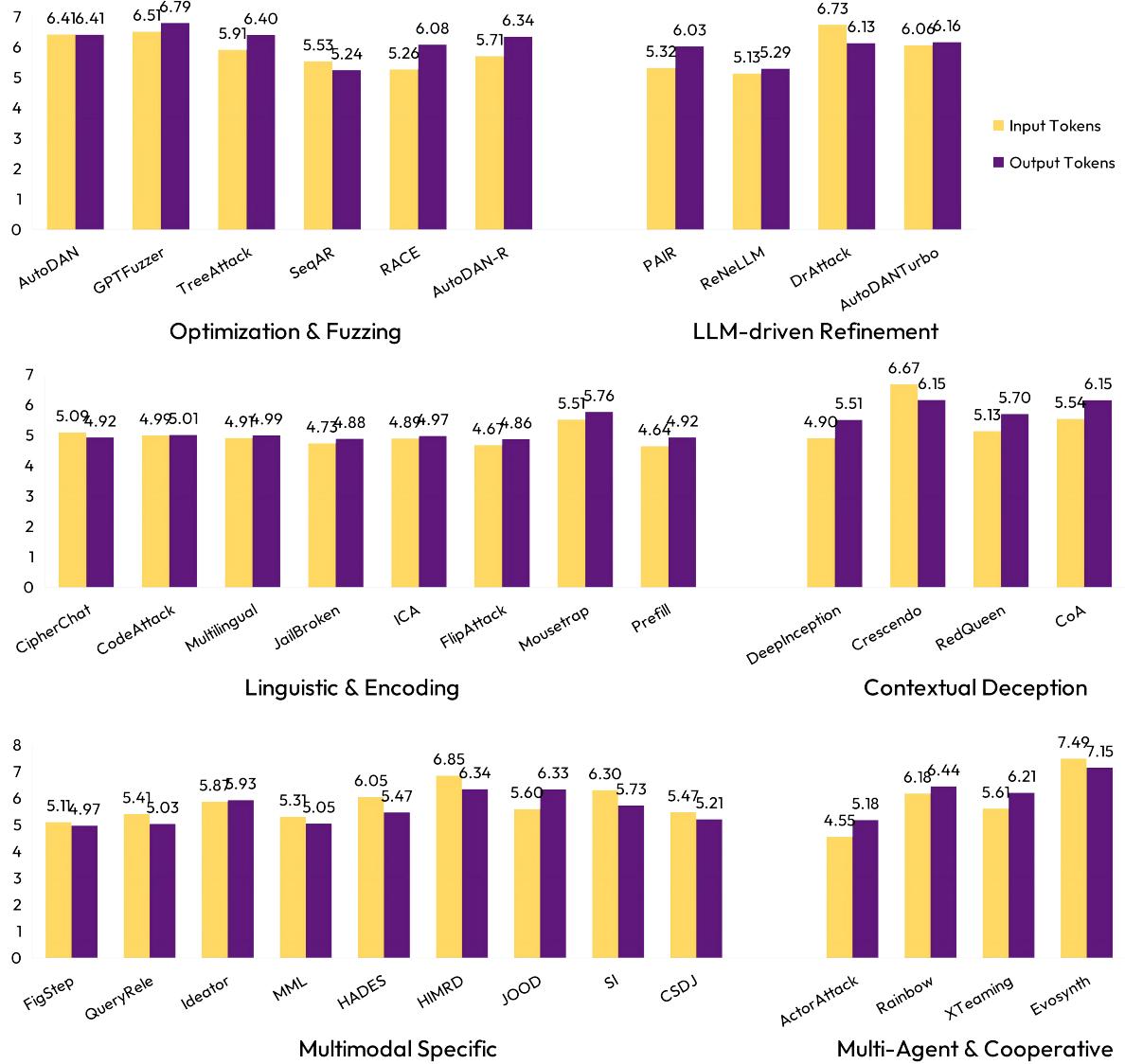
**Table 6** Attack Performance across Different LLMs on HarmfulBench

**In summary**, these results reveal that adversarial robustness is highly **attack-dependent** and **model-dependent**: models withstand weak prompts yet fail under adaptive ones; simultaneously, attack strategies exhibit polarized outcomes, often proving lethal to specific targets while completely failing on others. By aggregating performance across methodologies (Table 2), we find that multi-agent and optimization-based methods are among the most potent. Notably, multi-agent strategies like EvoSynth and X-Teaming generalize best, with EvoSynth achieving near-universal success, demonstrating that multi-agent, search-driven synthesis effectively bypasses static safety constraints. Beyond multi-agent settings, black-box optimization and LLM-driven refinement (*e.g.*, AutoDAN-R, PAIR, GPTFuzzer) also achieve strong and broadly consistent performance, reinforcing that adaptivity and iterative feedback are key drivers of attack effectiveness. Meanwhile, structured obfuscation and logic nesting (*e.g.*, Mousetrap) can remain highly effective on many models, showing that attacks do not need to be multi-agent to be high-impact. In contrast, heuristic, template-heavy, or shallow linguistic/encoding manipulations tend to exhibit high variance and unstable ASR performance across models; while they may succeed on specific targets, they often fail entirely on others, suggesting that contemporary safety training and filtering increasingly mitigate static jailbreak patterns. Finally, weaker multi-turn heuristics (*e.g.*, RedQueen) generally underperform, indicating that simple context manipulation alone

is often insufficient against modern alignment. **Overall, these results highlight a shift in the attack landscape: adaptive, iterative, and multi-agent strategies dominate, whereas static, single-turn, and template-based approaches are increasingly brittle.**

### 4.3 Multi-dimensional Attack Analysis

While Attack Success Rate (ASR) serves as the primary indicator of vulnerability, a practical red-teaming framework requires a holistic assessment. As previously established, an ideal attack method should not only achieve **high ASR** but also demonstrate **high efficiency, high stealthiness, and high diversity**. In this section, we dissect these critical dimensions to evaluate the trade-offs inherent in different attack paradigms and identify which methods best approximate this ideal balance.

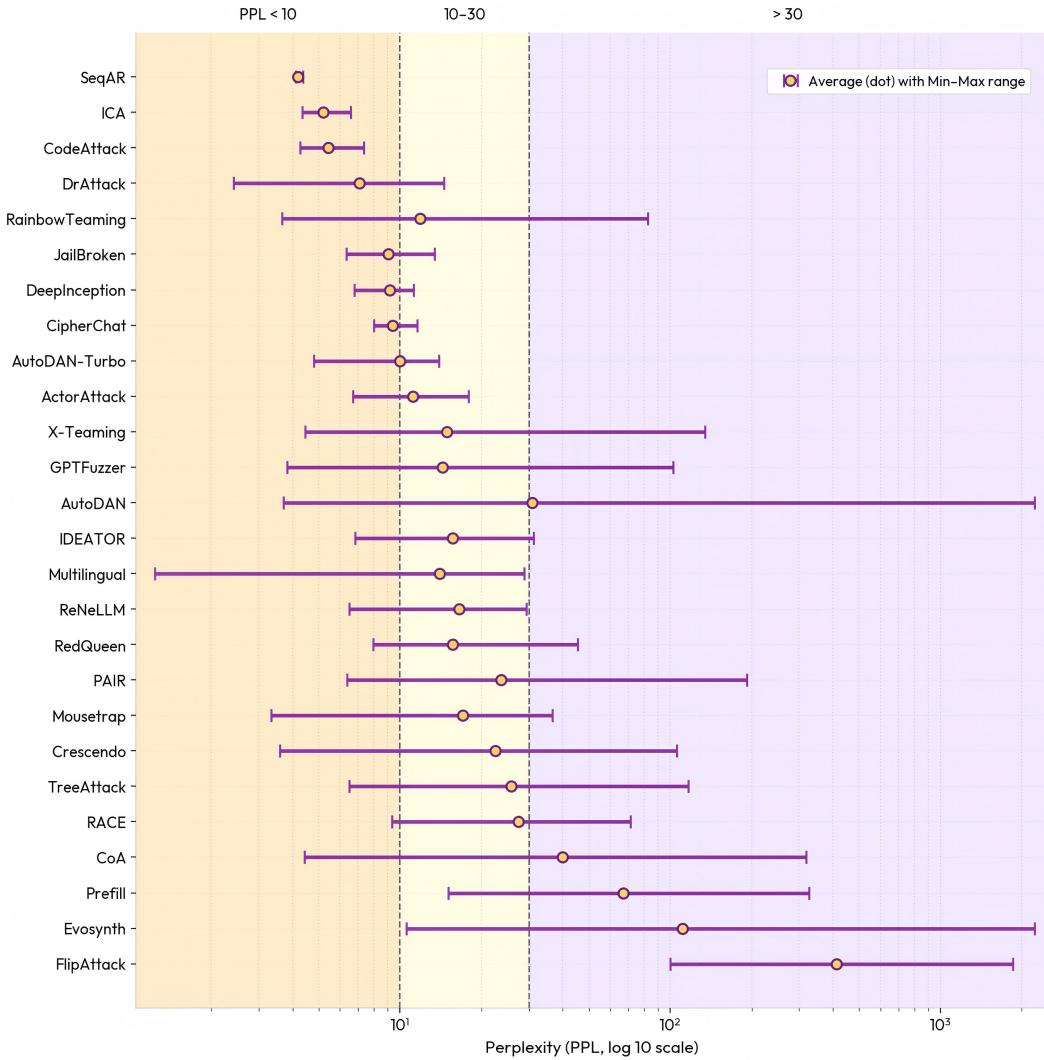


**Figure 2** Computational cost of various attack methods against GPT-5.2, measured by input and output token usage (log<sub>10</sub> scale).

### 4.3.1 Attack Efficiency

In addition to ASR, we evaluate the cost of running each attack using token and query statistics. For each method, we record input tokens and output tokens, total tokens (as a proxy for monetary/latency cost), and number of calls (proxy for rate-limit pressure and wall-clock time).

Figure 2 shows that GPTFuzzer is the most resource-intensive (9.44M total tokens; 9,705 calls), followed by DrAttack (6.75M total tokens; 3,224 calls), Crescendo (6.11M total tokens; 1,161 calls), and AutoDAN (5.14M total tokens; 9,406 calls). These approaches are dominated by iterative querying; GPTFuzzer is also notably output-heavy (6.19M output vs. 3.25M input tokens). Mid-cost methods such as TreeAttack, AutoDAN-Turbo, SI, X-Teaming, CoA, and IDEATOR reduce total tokens to roughly 1.6M–3.3M with fewer calls. In contrast, lightweight methods (*e.g.*, FlipAttack, Prefill, JailBroken, ICA) enable low-budget sweeps with total tokens below 200k.



**Figure 3** Attack stealthiness against GPT-5.2 measured by perplexity (PPL) on Qwen3-32B (log<sub>10</sub> scale). Lower PPL indicates more stealthy prompts. Shaded bands mark high (< 10), moderate (10–30), and low (> 30) stealthiness.

### 4.3.2 Attack Stealthiness

To evaluate the stealthiness of adversarial prompts in black-box attacks, we measure perplexity (PPL) with a Qwen3-32B base model as a proxy for linguistic naturalness, where lower PPL indicates prompts that are harder to detect by perplexity-based defenses. For white-box visual attacks, stealthiness is controlled via an  $\epsilon$ -bounded perturbation constraint. As shown in Figure 3, we observe three categories. **High-stealthiness attacks** ( $\text{PPL} < 10$ ) include SeqAR (4.18), ICA (5.18), CodeAttack (5.43), and DrAttack (7.09), which produce fluent prompts closely resembling natural language and are particularly concerning for evading detection. **Moderate-stealthiness attacks** ( $\text{PPL} 10\text{--}30$ ) encompass iterative methods like RainbowTeaming (11.88), GPTFuzzer (14.35), X-Teaming (14.89), and multi-turn approaches like Crescendo (22.48) and PAIR (23.63), which balance exploration diversity with linguistic coherence. **Low-stealthiness attacks** ( $\text{PPL} > 30$ ) rely on obfuscation techniques, with FlipAttack (412.15) producing the least natural prompts due to character-level perturbations. Notably, attack effectiveness does not correlate with stealthiness: Mousetrap achieves 97.5% ASR with moderate PPL (17.01), while SeqAR maintains excellent stealthiness (4.18) but only 25% ASR, suggesting that defenders should employ multi-layered detection strategies combining semantic analysis with perplexity-based filtering.



**Figure 4** Diversity analysis of attack methods against GPT-5.2, quantified by the mean pairwise cosine distance between embeddings of successful adversarial prompts.

### 4.3.3 Attack Diversity

As shown in Figure 4, our analysis reveals three distinct categories. **High-diversity attacks** ( $\text{Diversity} > 0.70$ ) are dominated by multi-agent and iterative methods: EvoSynth (0.820), X-Teaming (0.795), PAIR (0.787), IDEATOR (0.770), and Crescendo (0.766) demonstrate the effectiveness of evolutionary and cooperative exploration mechanisms. **Moderate-diversity attacks** (0.40–0.70) include genetic algorithms (AutoDAN: 0.599, GPTFuzzer: 0.528) and contextual methods (CoA: 0.651, HADES: 0.604), where mutation operators introduce variation but often converge to similar patterns. **Low-diversity attacks** ( $\text{Diversity} < 0.40$ ) are characterized by template-based or encoding-constrained methods: CipherChat (0.055), CodeAttack (0.108), and SeqAR (0.087) produce structurally similar outputs due

to their deterministic transformation schemes. Notably, RainbowTeaming, CS-DJ, and JOOD achieve near-zero diversity ( $<0.001$ ), generating virtually identical prompts across different queries.

Interestingly, diversity does not always correlate with ASR. Mousetrap achieves 97.5% ASR on GPT-5.2 despite low diversity (0.174), while high-diversity methods like TreeAttack (11.0% ASR) explore broader but less effective attack strategies. This trade-off suggests that comprehensive red-teaming should combine high-diversity methods for vulnerability discovery with targeted low-diversity attacks for exploiting known weaknesses.

## 5 Conclusion

In this work, we introduced **OpenRT**, a unified and extensible framework designed for comprehensive red-teaming evaluation of both MLLMs. By integrating 37 diverse attack methods, the framework provides a comprehensive tool for evaluating model safety, offering a standardized platform for benchmarking multiple models and attack strategies. Our large-scale evaluation of 20 advanced models revealed significant safety vulnerabilities in state-of-the-art systems, demonstrating that current safety mechanisms are often ineffective against a variety of adversarial techniques. OpenRT not only highlights persistent gaps in model defenses but also serves as a foundational infrastructure for future research in adversarial robustness. Looking ahead, we aim to expand OpenRT’s capabilities by integrating emerging attack paradigms, enhancing support for additional modalities, and fostering community-driven evolution, ultimately helping to bridge the gap between perceived and actual safety in deployed AI systems.

## References

- [1] Meta AI. Llama 4, 2025.
- [2] Mistral AI. Mistral large 3, 2025.
- [3] Mehmet Akhoroz and Caglar Yildirim. Conversational ai as a coding assistant: Understanding programmers' interactions with and expectations from large language models for coding. *arXiv preprint arXiv:2503.16508*, 2025.
- [4] Anthropic. Claude haiku 4.5, 2025.
- [5] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- [6] Baidu-ERNIE-Team. Ernie 4.5 technical report, 2025.
- [7] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *NeurIPS*, 2024.
- [8] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *IEEE SaTML*, 2025.
- [9] Xiuyuan Chen, Jian Zhao, Yuxiang He, Yuan Xun, Xinwei Liu, Yanshu Li, Huilin Zhou, Wei Cai, Ziyan Shi, Yuchen Yuan, et al. Teleai-safety: A comprehensive llm jailbreaking benchmark towards attacks, defenses, and evaluations. *arXiv preprint arXiv:2512.05485*, 2025.
- [10] Xuan Chen, Yuzhou Nie, Wenbo Guo, and Xiangyu Zhang. When llm meets drl: Advancing jailbreaking efficiency via drl-guided search. In *NeurIPS*, 2024.
- [11] Yunhao Chen, Xin Wang, Juncheng Li, Yixu Wang, Jie Li, Yan Teng, Yingchun Wang, and Xingjun Ma. Evolve the method, not the prompts: Evolutionary synthesis of jailbreak attacks on llms. *arXiv preprint arXiv:2511.12710*, 2025.
- [12] Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- [13] deepteam. The llm red teaming framework, 2025.
- [14] Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. *arXiv preprint arXiv:2310.06474*, 2023.

- [15] Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A wolf in sheep’s clothing: Generalized nested jailbreak prompts can fool large language models easily. In *NAACL*, 2024.
- [16] Moussa Koulako Bala Doumbouya, Ananjan Nandi, Gabriel Poesia, Davide Ghilardi, Anna Goldie, Federico Bianchi, Dan Jurafsky, and Christopher D Manning. h4rm3l: A language for composable jailbreak attack synthesis. *arXiv preprint arXiv:2408.04811*, 2024.
- [17] Sophie Fischer, Carlos Gemmell, Niklas Tecklenburg, Iain Mackie, Federico Rossetto, and Jeffrey Dalton. Grillbot in practice: Lessons and tradeoffs deploying large language models for adaptable conversational task assistants. In *KDD*, 2024.
- [18] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.
- [19] Kuofeng Gao, Yiming Li, Chao Du, Xin Wang, Xingjun Ma, Shu-Tao Xia, and Tianyu Pang. Imperceptible jailbreaking against large language models. *arXiv preprint arXiv:2510.05025*, 2025.
- [20] Simon Geisler, Tom Wollschläger, Mohamed Hesham Ibrahim Abdalla, Johannes Gasteiger, and Stephan Günnemann. Attacking large language models with projected gradient descent. *arXiv preprint arXiv:2402.09154*, 2024.
- [21] GeneralAnalysis. Jailbreak cookbook by general analysis, 2025.
- [22] Peiyuan Gong, Jiamian Li, and Jiaxin Mao. Cosearchagent: a lightweight collaborative search agent with large language models. In *ACM SIGIR*, 2024.
- [23] Yichen Gong, Delong Ran, Xinlei He, Tianshuo Cong, Anyu Wang, and Xiaoyun Wang. Safety misalignment against large language models. In *NDSS*, 2025.
- [24] Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. Figstep: Jailbreaking large vision-language models via typographic visual prompts. In *AAAI*, 2025.
- [25] Google. Gemini 3 pro preview, 2025.
- [26] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 2025.
- [27] Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. In *NeurIPS*, 2024.

[28] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[29] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[30] Joonhyun Jeong, Seyun Bae, Yeonsung Jung, Jaeryong Hwang, and Eunho Yang. Playing the fool: Jailbreaking llms and multimodal llms with out-of-distribution strategy. In *CVPR*, 2025.

[31] Akshita Jha and Chandan K Reddy. Codeattack: Code-based adversarial attacks for pre-trained programming language models. In *AAAI*, 2023.

[32] Xiaojun Jia, Jie Liao, Qi Guo, Teng Ma, Simeng Qin, Ranjie Duan, Tianlin Li, Yihao Huang, Zhitao Zeng, Dongxian Wu, et al. Omnisafebench-mm: A unified benchmark and toolbox for multimodal jailbreak attack-defense evaluation. *arXiv preprint arXiv:2512.06589*, 2025.

[33] Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, et al. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *NeurIPS*, 2024.

[34] Yifan Jiang, Kriti Aggarwal, Tanmay Laud, Kashif Munir, Jay Pujara, and Subhabrata Mukherjee. Red queen: Safeguarding large language models against concealed multi-turn jailbreaking. *arXiv preprint arXiv:2409.17458*, 2024.

[35] Shanghai AI Lab. Safework-r1: Coevolving safety and intelligence under the ai-45° law. *arXiv preprint arXiv:2507.18576*, 2025.

[36] Raz Lapid, Ron Langberg, and Moshe Sipper. Open sesame! universal black-box jailbreaking of large language models. *Applied Sciences*, 2024.

[37] Jie Li, Yi Liu, Chongyang Liu, Xiaoning Ren, Ling Shi, Weisong Sun, and Yinxing Xue. Self and cross-model distillation for llms: Effective methods for refusal pattern alignment. *arXiv preprint arXiv:2406.11285*, 2024.

[38] Nathaniel Li, Ziwen Han, Ian Steneker, Willow Primack, Riley Goodside, Hugh Zhang, Zifan Wang, Cristina Menghini, and Summer Yue. Llm defenses are not robust to multi-turn human jailbreaks yet. *arXiv preprint arXiv:2408.15221*, 2024.

[39] Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. Drattack: Prompt decomposition and reconstruction makes powerful llm jailbreakers. *arXiv preprint arXiv:2402.16914*, 2024.

[40] Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191*, 2023.

- [41] Yakai Li, Jiekang Hu, Weiduan Sang, Luping Ma, Jing Xie, Weijuan Zhang, Aimin Yu, Shijie Zhao, Qingjia Huang, and Qihang Zhou. Prefill-based jailbreak: A novel approach of bypassing llm safety boundary. *arXiv preprint arXiv:2504.21038*, 2025.
- [42] Yifan Li, Hangyu Guo, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. Images are achilles' heel of alignment: Exploiting visual vulnerabilities for jailbreaking multimodal large language models. In *ECCV*, 2024.
- [43] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.
- [44] Xiaogeng Liu, Peiran Li, Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. In *ICLR*, 2025.
- [45] Xiaogeng Liu and Chaowei Xiao. Autodan-reasoning: Enhancing strategies exploration based jailbreak attacks with test-time scaling. *arXiv preprint arXiv:2510.05379*, 2025.
- [46] Xiaogeng Liu, Nan Xu, Muham Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *ICLR*, 2024.
- [47] Xin Liu, Yichen Zhu, Yunshi Lan, Chao Yang, and Yu Qiao. Query-relevant images jailbreak large multi-modal models. *arXiv preprint arXiv:2311.17600*, 2023.
- [48] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023.
- [49] Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, and Bryan Hooi. Flipattack: Jailbreak llms via flipping. *arXiv preprint arXiv:2410.02832*, 2024.
- [50] Teng Ma, Xiaojun Jia, Ranjie Duan, Xinfeng Li, Yihao Huang, Xiaoshuang Jia, Zhixuan Chu, and Wenqi Ren. Heuristic-induced multimodal risk distribution jailbreak attack for multimodal large language models. In *ICCV*, 2025.
- [51] Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, Ye Sun, Yifan Ding, Hengyuan Xu, Yunhao Chen, Yunhan Zhao, et al. Safety at scale: A comprehensive survey of large model safety. *arXiv preprint arXiv:2502.05206*, 2025.
- [52] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [53] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

- [54] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*, 2023.
- [55] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. In *NeurIPS*, 2024.
- [56] MiniMax. Minimax-m2, 2025.
- [57] Gary D Lopez Munoz, Amanda J Minnich, Roman Lutz, Richard Lundeen, Raja Sekhar Rao Dheekonda, Nina Chikanov, Bolor-Erdene Jagdagdorj, Martin Pouliot, Shiven Chawla, Whitney Maxwell, et al. Pyrit: A framework for security risk identification and red teaming in generative ai system. *arXiv preprint arXiv:2410.02828*, 2024.
- [58] OpenAI. Gpt-5.1 instant and gpt-5.1 thinking system card addendum, 2025.
- [59] OpenAI. Update to gpt-5 system card: Gpt-5.2, 2025.
- [60] Zhenyu Pan, Yiting Zhang, Yutong Zhang, Jianshu Zhang, Haozheng Luo, Yuwei Han, Dennis Wu, Hong-Yu Chen, Philip S Yu, Manling Li, et al. Evo-marl: Co-evolutionary multi-agent reinforcement learning for internalized safety. *arXiv preprint arXiv:2508.03864*, 2025.
- [61] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.
- [62] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. In *AAAI*, 2024.
- [63] Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. In *ICLR*, 2025.
- [64] Salman Rahman, Liwei Jiang, James Shiffer, Genglin Liu, Sheriff Issaka, Md Rizwan Parvez, Hamid Palangi, Kai-Wei Chang, Yejin Choi, and Saadia Gabriel. X-teaming: Multi-turn jailbreaks and defenses with adaptive multi-agents. *arXiv preprint arXiv:2504.13203*, 2025.
- [65] Qibing Ren, Chang Gao, Jing Shao, Junchi Yan, Xin Tan, Wai Lam, and Lizhuang Ma. Codeattack: Revealing safety generalization challenges of large language models via code completion. In *ACL*, 2024.
- [66] Qibing Ren, Hao Li, Dongrui Liu, Zhanxu Xie, Xiaoya Lu, Yu Qiao, Lei Sha, Junchi Yan, Lizhuang Ma, and Jing Shao. Derail yourself: Multi-turn llm jailbreak attack through self-discovered clues. *arXiv preprint arXiv:2410.10700*, 2024.

- [67] Steven I Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D Weisz. The programmer’s assistant: Conversational interaction with a large language model for software development. In *IUI*, 2023.
- [68] Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The crescendo {Multi-Turn}{LLM} jailbreak attack. In *USENIX Security*, 2025.
- [69] Mikayel Samvelyan, Sharath C Raparthi, Andrei Lupu, Eric Hambro, Aram H Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow teaming: Open-ended generation of diverse adversarial prompts. In *NeurIPS*, 2024.
- [70] Seed. Doubao-seed-1.6, 2025.
- [71] Kartik Sharma, Yiqiao Jin, Vineeth Rakesh, Yingtong Dou, Menghai Pan, Mahashweta Das, and Srijan Kumar. Sysformer: Safeguarding frozen large language models with adaptive system prompts. *arXiv preprint arXiv:2506.15751*, 2025.
- [72] Kimi Team. Kimi k2: Open agentic intelligence, 2025.
- [73] Qwen Team. Qwen3-235b-a22b, 2025.
- [74] Qwen Team. Qwen3-max: Just scale it, 2025.
- [75] Qwen Team. Qwen3-next-80b-a3b-instruct, 2025.
- [76] V Team. Gbm-4.5v and gbm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025.
- [77] Tencent Hunyuan Team. Hunyuan-a13b-instruct, 2025.
- [78] Pliny the Promter. L1B3RT45: Jailbreaks for All Flagship AI Models, 2024.
- [79] Han Wang, An Zhang, Nguyen Duy Tai, Jun Sun, Tat-Seng Chua, et al. Ali-agent: Assessing llms’ alignment with human values via agent-based evaluation. In *NeurIPS*, 2024.
- [80] Ruofan Wang, Juncheng Li, Yixu Wang, Bo Wang, Xiaosen Wang, Yan Teng, Yingchun Wang, Xingjun Ma, and Yu-Gang Jiang. Ideator: Jailbreaking and benchmarking large vision-language models using themselves. In *ICCV*, 2025.
- [81] Xin Wang, Kai Chen, Xingjun Ma, Zhenheng Chen, Jingjing Chen, and Yu-Gang Jiang. Advqdet: Detecting query-based adversarial attacks with adversarial contrastive prompt tuning. In *ACM MM*, 2024.
- [82] Xin Wang, Kai Chen, Jiaming Zhang, Jingjing Chen, and Xingjun Ma. Tapt: Test-time adversarial prompt tuning for robust inference in vision-language models. In *CVPR*, 2025.
- [83] Xin Wang, Jie Li, Zejia Weng, Yixu Wang, Yifeng Gao, Tianyu Pang, Chao Du, Yan Teng, Yingchun Wang, Zuxuan Wu, et al. Freezenva: Action-freezing attacks against vision-language-action models. *arXiv preprint arXiv:2509.19870*, 2025.

- [84] Yixu Wang, Jiaxin Song, Yifeng Gao, Xin Wang, Yang Yao, Yan Teng, Xingjun Ma, Yingchun Wang, and Yu-Gang Jiang. Safevid: Toward safety aligned video large multimodal models. In *NeurIPS*, 2025.
- [85] Yixu Wang, Xin Wang, Yang Yao, Xinyuan Li, Yan Teng, Xingjun Ma, and Yingchun Wang. Safeevalagent: Toward agentic and self-evolving safety evaluation of llms. *arXiv preprint arXiv:2509.26100*, 2025.
- [86] Yu Wang, Xiaofei Zhou, Yichen Wang, Geyuan Zhang, and Tianxing He. Jailbreak large vision-language models through multi-modal linkage. In *ACL*, 2025.
- [87] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In *NeurIPS*, 2023.
- [88] Zeming Wei, Yifei Wang, Ang Li, Yichuan Mo, and Yisen Wang. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*, 2023.
- [89] Laura Weidinger, Maribeth Rauh, Nahema Marchal, Arianna Manzini, Lisa Anne Hendricks, Juan Mateos-Garcia, Stevie Bergman, Jackie Kay, Conor Griffin, Ben Bariach, et al. Sociotechnical safety evaluation of generative ai systems. *arXiv preprint arXiv:2310.11986*, 2023.
- [90] xAI. Grok 4.1 fast, 2025.
- [91] Yunjia Xi, Jianghao Lin, Yongzhao Xiao, Zheli Zhou, Rong Shan, Te Gao, Jiachen Zhu, Weiwen Liu, Yong Yu, and Weinan Zhang. A survey of llm-based deep search agents: Paradigm, optimization, evaluation, and challenges. *arXiv preprint arXiv:2508.05668*, 2025.
- [92] Zeguan Xiao, Yan Yang, Guanhua Chen, and Yun Chen. Tastle: Distract large language models for automatic jailbreak attack. *arXiv preprint arXiv:2403.08424*, 2024.
- [93] Huiyu Xu, Wenhui Zhang, Zhibo Wang, Feng Xiao, Rui Zheng, Yunhe Feng, Zhongjie Ba, and Kui Ren. Redagent: Red teaming large language models with context-aware autonomous language agent. *arXiv preprint arXiv:2407.16667*, 2024.
- [94] Zhao Xu, Fan Liu, and Hao Liu. Bag of tricks: Benchmarking of jailbreak attacks on llms. In *NeurIPS*, 2024.
- [95] Xikang Yang, Xuehai Tang, Songlin Hu, and Jizhong Han. Chain of attack: a semantic-driven contextual multi-turn attacker for llm. *arXiv preprint arXiv:2405.05610*, 2024.
- [96] Yan Yang, Zeguan Xiao, Xin Lu, Hongru Wang, Xuetao Wei, Hailiang Huang, Guanhua Chen, and Yun Chen. Seqar: Jailbreak llms with sequential auto-generated characters. In *NAACL*, 2025.
- [97] Zuopeng Yang, Jiluan Fan, Anli Yan, Erdun Gao, Xin Lin, Tao Li, Kanghua Mo, and Changyu Dong. Distraction is all you need for multimodal large language model jailbreaking. In *CVPR*, 2025.

[98] Yang Yao, Xuan Tong, Ruofan Wang, Yixu Wang, Lujundong Li, Liang Liu, Yan Teng, and Yingchun Wang. A mousetrap: Fooling large reasoning models for jailbreak with chain of iterative chaos. *ACL Findings*, 2025.

[99] Zonghao Ying, Deyue Zhang, Zonglei Jing, Yisong Xiao, Quanchen Zou, Aishan Liu, Siyuan Liang, Xiangzheng Zhang, Xianglong Liu, and Dacheng Tao. Reasoning-augmented conversation for multi-turn jailbreak attacks on large language models. *arXiv preprint arXiv:2502.11054*, 2025.

[100] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023.

[101] Miao Yu, Junfeng Fang, Yingjie Zhou, Xing Fan, Kun Wang, Shirui Pan, and Qingsong Wen. Llm-virus: Evolutionary jailbreak attack on large language models. *arXiv preprint arXiv:2501.00055*, 2024.

[102] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. In *ICLR*, 2024.

[103] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*, 2020.

[104] Yuyou Zhang, Miao Li, William Han, Yihang Yao, Zhepeng Cen, and Ding Zhao. Safety is not only about refusal: Reasoning-enhanced fine-tuning for interpretable llm safety. *arXiv preprint arXiv:2503.05021*, 2025.

[105] Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*, 2025.

[106] Shiji Zhao, Ranjie Duan, Fengxiang Wang, Chi Chen, Caixin Kang, Shouwei Ruan, Jialing Tao, YueFeng Chen, Hui Xue, and Xingxing Wei. Jailbreaking multimodal large language models via shuffle inconsistency. In *ICCV*, 2025.

[107] Andy Zhou, Kevin Wu, Francesco Pinto, Zhaorun Chen, Yi Zeng, Yu Yang, Shuang Yang, Sanmi Koyejo, James Zou, and Bo Li. Autoredteamer: Autonomous red teaming with lifelong attack integration. *arXiv preprint arXiv:2503.15754*, 2025.

[108] Weikang Zhou, Xiao Wang, Limao Xiong, Han Xia, Yingshuang Gu, Mingxu Chai, Fukang Zhu, Caishuang Huang, Shihan Dou, Zhiheng Xi, et al. Easyjailbreak: A unified framework for jailbreaking large language models. *arXiv preprint arXiv:2403.12171*, 2024.

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

[110] Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. A reinforced generation of adversarial examples for neural machine translation. *arXiv preprint arXiv:1911.03677*, 2019.

## A Appendix: Usage and Extensibility

This section explains how to use the **OpenRT** framework for security evaluations and how to extend it with your own attack methods.

### A.1 Basic Usage

**Installation** To install the framework, run:

```
pip install -r requirements.txt
python setup.py install
```

**Listing 1** Installing **OpenRT** from source.

**Running Experiments.** You can run experiments using configuration files or Python scripts. The configuration-driven method only requires a YAML file. For example:

```
python main.py --config configs/autodan_turbo_experiment.yaml
```

**Listing 2** Running an experiment from a YAML configuration file.

Alternatively, you can set up the experiment programmatically:

```
from openrt import (
    OpenAIModel, StaticDataset, PAIR,
    LLMJudge, JudgeEvaluator, Orchestrator
)

# Initialize components
model = OpenAIModel(model_name="gpt-5.1", api_key="...")
helper_model = OpenAIModel(model_name="gpt-4o", api_key="...")
dataset = StaticDataset(prompts=["harmful query 1", ...])
judge = LLMJudge(judge_model=model, success_threshold=5)
attack = PAIR(model, helper_model, judge, max_iterations=5)
evaluator = JudgeEvaluator(judge=judge)

# Run experiment
orchestrator = Orchestrator(model, dataset, attack, evaluator)
metrics, results = orchestrator.run()

print(f"Attack Success Rate: {metrics.ASR:.2%}")
```

**Listing 3** Programmatic setup of an OpenRT red teaming experiment.

**Viewing Results** Results are saved in a folder with detailed logs:

```
results/
baseline/
gpt-5.1_20251207T074428Z/
metrics/gpt-5.1_PAIR_metrics.json
details/gpt-5.1_PAIR_results.jsonl
```

**Listing 4** Example output directory structure and result files.

## A.2 Extending Attacks

You can add new attack methods by implementing the `BaseAttack` interface and registering it with the framework. Here is an example of how to create a custom attack:

```
from openrt.attacks.base_attack import BaseAttack, AttackResult
from openrt.core.registry import attack_registry

@attack_registry.register("my_attack")
class MyAttack(BaseAttack):
    def __init__(self, model, max_iters=10, **kwargs):
        super().__init__(model, **kwargs)
        self.max_iters = max_iters

    def attack(self, target: str) -> AttackResult:
        for i in range(self.max_iters):
            adv_prompt = self._craft_prompt(target, i)
            response = self.model.query(adv_prompt)
            if self._is_successful(response):
                return AttackResult(
                    target=target,
                    success=True,
                    final_prompt=adv_prompt,
                    output_text=response,
                    method="my_attack",
                )
        return AttackResult(target=target, success=False)
```

**Listing 5** Implementing and registering a custom attack in `OpenRT`.

Once registered, your custom attack will be available for use in experiments:

```
attack:
  name: "my_attack"
  args:
    max_iters: 15
```

**Listing 6** Using the custom attack via configuration.

### A.3 Configuration

Complete experiments can also be declared purely through YAML configuration files, upon which the Orchestrator dynamically instantiates and wires together the corresponding components (model, dataset, attack, judge, and evaluator). An example configuration for a complete experiment is shown in Listing 7.

```
experiment_name: "Comprehensive_Safety_Evaluation"

model:
  name: "openai"
  args:
    model_name: "gpt-5.1"
    temperature: 0.7
    api_key: "${OPENAI_API_KEY}"

dataset:
  name: "jsonl"
  args:
    file_path: "data/advbench.jsonl"

attack:
  name: "autodan_turbo"
  args:
    epochs: 5
    warm_up_iterations: 2
    lifelong_iterations: 3
    break_score: 8.5

evaluator:
  name: "judge"
  args:
    judge:
      name: "llm_judge"
      args:
        success_threshold: 5
```

**Listing 7** YAML configuration for a complete OpenRT red-teaming experiment.