AILuminate Security
Introducing v0.5 of the Jailbreak Benchmark from MLCommons

James Goel[1,12], Alicia Parrish[8], Chris Knotz[24], James Ezick[1], Jonathan Petit[1], Cong Chen[1], Wenxin Ding[1], Lora Aroyo[8], Andrew Gruen[9,12], Kurt Bollacker[12], William Pietri[12], Kashyap R Manjusha[2], Rajat Shinde[20,21], Murali Emani[4], Xuanli He[14], Tianhao Li[5], Leon Derczynski[11], Jibin Varghese[11], Armstrong Foundjem[10], Jean-Philippe Monteuuis[1], Jacqueline Stetson[12], Bennett Hillenbrand[9,12], Rajat Ghosh[26], Federico Ricciuti[6], Victor Lu[6], Kongtao Chen[13], Tuesday[7], Eileen Long[11], Saif Ul Islam[31], Carsten Maple[31], Mubashara Akhtar[23], Aakash Gupta[18], Roman Eng[22], Kenneth Fricklas[19], Sarah Luger[12], Fion Lee-Madan[16], Sujata Goswami[3], Manil Maskey[20], Satyapriya Krishna[25], Alicja Kwasniewska[25], Julien Delaunay[15], Fazl Barez[17], Elie Alhajjar[27], Fedor Zhdanov[32], Vassil Tashev[6], Serrhini Mohamed[29], David Graham[12], Sean McGregor[30], and Peter Mattson[13]

[25]*Amazon* — [4]*Argonne National Laboratory* — [7]*Artifex Labs* — [16]*Asenion* — [30]*AVERI* — [3]*Berkeley National Laboratory* — [22]*Clarkson University* — [24]*CommonGroundAI* — [5]*Duke University* — [23]*ETH Zurich* — [13]*Google* — [8]*Google DeepMind* — [6]*Independent* — [12]*MLCommons* — [20]*NASA* — [26]*Nutanix* — [11]*NVIDIA Corporation* — [10]*Polytechnique Montreal (DEEL Project)* — [1]*Qualcomm* — [27]*RAND* — [32]*Royal Holloway University of London* — [18]*Think Evolve Labs* — [15]*Top Health Tech* — [19]*Turaco Strategy* — [2]*UIUC* — [14]*University College London* — [29]*University Mohamed First oujda Morocco* — [21]*University of Alabama in Huntsville* — [17]*University of Oxford* — [31]*University of Warwick* — [9]*Working Paper*

9 December 2025

Abstract

Adversarial actors bypass common Large Language Model (LLM) and Vision-Language Model (VLM) guardrails by employing a variety of "jailbreak" techniques. Susceptibility to jailbreaks indicates Systems Under Test (SUTs) may be unreliable in deployment due to the risk of being compromised. Identifying more risk resilient SUTs requires guardrail performance measurement by testing systems with adversarial prompts. The goal in such an assessment is to exploit vulnerabilities or jailbreak an Artificial Intelligence (AI) system's safety filters, causing it to produce harmful or undesired output. This paper presents the v0.5 MLCommons Jailbreak Benchmark. In addition, we present a practical AI Management System that embeds testing into an operational cycle, with a specific focus on the systems' resilience to jailbreaks. The process starts by establishing a baseline safety grade using a standard benchmark, AILuminate v1.0 for LLMs and the Multimodal Safety Test Suite for VLMs. It then measures system adversarial resilience by applying public (widely shared and known) jailbreak attacks and determining the difference in performance. We report a *Resilience Gap* per hazard and overall, defined as the difference between baseline non-violating-rate and under-attack non-violating-rate, using identical evaluators and five-tier grade bands to ensure strict comparability. This safety-versus-attack gap drives a cycle of risk assessment, mitigation, and improvement, creating auditable records for governance frameworks like ISO/IEC 42001. This approach treats a benchmark not as a final score, but as an instrument within a larger system to monitor and improve AI resilience continuously.

Keywords: AI security evaluation, adversarial testing, jailbreak attacks, resilience gap, MLCommons AILuminate, Multimodal Safety Test Suite, large language models, vision language models, AI governance, ISO/IEC 42001.

Contents

1 Introduction

As Artificial Intelligence (AI) systems become integral to safety-critical applications — encompassing finance [1], healthcare [2], transportation [3], public policy [4], and more [5] — ensuring their resilience and reliability becomes equivalent to safeguarding the infrastructure itself. Organizations, therefore, need practical ways to assess how systems behave under *both* expected/naïve (safety) and adversarial (security) conditions. New standards, such as ISO/IEC 42001 [6], require a systematic approach to managing these systems, which AI Management Systems (AIMS) can help deployers provide.

This work presents the **MLCommons Jailbreak Benchmark** v0.5 (a "Draft Release" version), a standardized framework for evaluating system resistance to jailbreaking attempts across multiple modalities, including demonstrations of text-to-text (T2T) and text-plus-image-to-text (T+I2T). Critically, the T2T portion is built on top of the proven **MLCommons AI Safety Benchmark (AILuminate v1.0)** [7]: we retain its hazard taxonomy, grading logic, and reporting discipline. The T+I2T portion is modeled after the Multimodal Safety Test Suite for Vision-Language Models benchmark (MSTS) [8], where we retain the hazard taxonomy and the prompt structure. For this benchmark, we introduce controlled, adversarial prompts to quantify **resilience**: the degradation from baseline to under-attack safety. We term this degradation the *resilience gap*.

Contribution. We quantify AI resilience by pairing an industry-standard *safety* baseline (AILuminate v1.0 for T2T and MSTS for T+I2T) with standardized *jailbreak* testing, in order to enable an organization to drive ISO/IEC 42001 AIMS actions from the resulting *grade delta*. The contribution is a *repeatable measurement approach*, aligned to AIMS activities and auditable artifacts.

Release status and scope. MLCommons v0.5 releases are designed to verify that a benchmark-in-development produces the intended signal and can be instrumented in practice. Our v0.5 releases prioritize methodology over leader-board ranking comparisons and reserve high-reliability and broad grading for v1.0 releases. Consistent with MLCommons policy for in-development benchmarks, we analyze only open-weight models and report results anonymously. Accordingly, this paper emphasizes the measurement design, governance use, and the *safety* → *security* linkage rather than specific vendor results.

We aim to demonstrate a unified pipeline across prompts, evaluators, and grading that works across modalities. The Vision Language Model (VLM) track validates multi-modal extensibility, with broad coverage and consensus passing criteria expected with its v1.0 release. The T+I2T track is in an earlier stage than its T2T counterpart (including fewer hazard categories, prompt attacks, and SUTs evaluated), but we plan to narrow this gap for the v1.0 release.

Objectives. This paper aims to:

1. Quantify AI *resilience* by bridging industry-standard safety baselines (AILuminate v1.0 for T2T and MSTS for T+I2T) with standardized jailbreak testing (v0.5), reporting comparable safety and jailbreak grades, along with the *delta between them* as the primary signal for this quality.
2. Operationalize this delta as a signal inside an ISO/IEC 42001–aligned AI Management System (AIMS): set policy and objectives, generate auditable artifacts (datasets, configuration files,

run logs, evaluator versions), and drive risk assessment, impact assessment, treatment, and monitoring in a Plan-Do-Check-Act (PDCA) cycle.

3. Provide an October 2025 snapshot of jailbreak resistance across T2T LLMs and T+I2T VLMs using the same evaluator and five-tier grade bands as the safety baselines for direct naïve-to-jailbroken comparability.
4. Enable engineering action — triage, mitigation testing, and release gating in CI/CD — while identifying collaborators, additional use cases, and stakeholder requirements needed to reach a v1.0 jailbreak release (planned for Q1 2026).

With the balance of this paper, we:

1. Summarize AILuminate's v1.0 and Multimodal Safety Test Suite's safety standards (taxonomy, datasets, evaluator, and grading)
2. Apply standardized jailbreak attacks
3. Report the Safety and Jailbreak grade delta per modality

2 Background: The AILuminate AI Safety Benchmark as Input to an ISO/IEC 42001 AIMS

2.1 What AILuminate Measures

AILuminate v1.0 is an industry-standard risk and reliability benchmark that evaluates whether systems produce hazardous responses across a curated set of categories (e.g., violent crimes, CBRNE/ indiscriminate weapons, privacy harms, hate, specialized advice, sexual content). The focus is on *response safety*, not user intent: a system is credited when it avoids generating policy-violating responses even when prompted to do so.

As a result, these measurements serve as inputs to an AI Management System (AIMS) under ISO/IEC 42001—they inform policy and objectives, AI risk assessment, risk treatment selection, monitoring, and create opportunities for improvement. Specific implications are called out below in *AIMS implication* sections.

The MSTS benchmark follows the same conceptual approach as AILuminate v1.0, but MSTS is not yet intended to evidence real-world decisions (e.g., regulatory compliance). Its inclusion in these results serves to guide the joint testing of multiple modality testing while still allowing for the specific criteria within those modalities to develop in consultation with system stakeholders.

2.1.1 Datasets and Personas

The AILuminate v1.0 safety text-to-text benchmark uses two parallel prompt sets: (i) a *practice* set for development and (ii) a held-out *official* set to deter overfitting. The prompts from the v0.5 Jailbreak Benchmark are an extract from this practice set. Prompts are balanced across hazards and written from two user personas, both of which are essential for a complete AIMS: *naïve* (unsophisticated) and *knowledgeable* (explicit, technically aware). This design tests safety across realistic user types while preserving comparability across systems. *AIMS implication: these* *personas map to foreseeable misuse and vulnerable-user scenarios that should be captured in AI risk and impact assessments and reviewed during management reviews.*

2.1.2 Evaluator and Decision Policy

Since human-only judging does not scale, the original AILuminate v1.0 safety benchmark employed a jury-style evaluator composed of fine-tuned safety models and prompt-engineered LLM judges, with targeted human spot-checks for calibration. Decision thresholds are tuned to minimize *false non-violating* evaluations. These are violating responses incorrectly judged non-violating (often more generally referred to as *false negatives*). This emphasis comes with a willingness to tolerate a somewhat higher rate of *false violating* evaluations (*false positives*). This preference reflects the safety-critical priorities of the benchmark. The v0.5 Jailbreak evaluator uses an improved subset of the original v1.0 Safety evaluator. When executing the v0.5 Jailbreak Benchmark, this same subset is used for all systems and all hazards in order to preserve fairness and enables stable longitudinal comparisons. *AIMS implication: the evaluator configuration, calibration evidence, and any human sampling plans should be maintained as controlled, documented information; evaluator operators should be listed under competence and roles.*

2.1.3 Grading and Reporting

Systems receive overall grades on a five-tier ordinal scale (e.g., Poor → Excellent) derived from the share of non-violating responses. Grades are anchored to a reference distribution to reduce sensitivity to small evaluation noise. *AIMS implication: these grades support setting AI safety objectives and targets, selecting and justifying risk treatments recorded in the Statement of Applicability (SoA), and defining release gating criteria.*

2.1.4 Scope and Limitations

AILuminate v1.0 focuses on single-turn interactions initiated from English seed prompts; it does not fully capture multi-turn manipulation, multilingual nuance, or every modality. Evaluator uncertainty is monitored and disclosed. Results are system-level indicators, not guarantees. *AIMS implication: scope gaps become re-test triggers and improvement actions (e.g., multi-turn or locale expansions), and evaluator uncertainty informs monitoring plans and internal audits.*

2.1.5 Why This Matters for Jailbreaks

Reusing AILuminate's prompts, evaluator, and grading means we can introduce standardized adversarial prompts and measure a system's *resilience* without changing the yardstick. We formalize this with a headline metric:

$$\textbf{Resilience Gap} = \text{safe-rate (AILuminate/MSTS)} - \text{safe-rate (jailbroken AILuminate/MSTS)}$$

This single number summarizes the safety → security drop. *AIMS implication: gaps that are meaningful in context should trigger risk assessment updates, risk treatment selection (and SoA updates), targeted mitigations, re-testing, and management review.*

2.2 ISO/IEC SC 42001 AIMS Alignment at a Glance

2.2.1 Bridge Benchmark Outputs to ISO/IEC 42001 Activities, Records, and Decisions

- **Policy & Objectives (Clauses 5–6)** Set AI safety objectives using AILuminate grades; define maximum acceptable *Resilience Gap* per hazard before release.
- **AI Risk Assessment (6.1)** Use per-hazard safe-rates and Resilience Gaps as inputs to risk identification, analysis, and evaluation; incorporate persona-based foreseeable misuse.
- **Risk Treatment & SoA (6.1.3)** Select mitigations (guard models, filters, HITL); record chosen controls and rationales in the *Statement of Applicability.*
- **AI System Impact Assessment (AISA)** Document potential harms revealed by jailbreak deltas and residual risk after treatment; link to product release gating.
- **Operation & Change Management (8)** Version datasets/evaluators; require re-evaluation on model updates, domain shifts, or new attack classes; maintain traceable artifacts.
- **Performance Evaluation (9.1)** Monitor grade trends and evaluator calibration; define thresholds that auto-trigger re-assessment or rollback.
- **Internal Audit (9.2) & Management Review (9.3)** Audit runs for reproducibility and tamper-evidence; include resilience metrics and SoA changes in management reviews.
- **Improvement (10)** Raise corrective actions when gaps exceed thresholds; track effectiveness via follow-up test runs.

2.2.2 Minimal Artifact Set (for Auditors)

It is essential to produce a set of auditable artifacts to support the use of AILuminate in an AIMS context. These include:

1. Benchmark version manifest (seeds, evaluator configuration, scoring, grade bands)
2. System information (CPU model, CPU core count, GPU type, memory, driver version, libraries and framework versions)
3. Run log + hash registry (prompts, responses, timestamps, model/version, attack configuration)
4. Per-hazard report + Resilience Gap table (baseline vs. jailbreak)
5. Risk assessment update + AISA excerpt referencing the above
6. SoA entry showing selected treatments and justification
7. Management review minutes noting trends/decisions

2.2.3 Alignment with Other Frameworks

Beyond ISO/IEC 42001, our grading and artifacts align with widely used governance frameworks. Relative to NIST AI Risk Management Framework (AI RMF) [9], the benchmark outputs support

Map (hazard taxonomy, personas), Measure (safety/jailbreak metrics and evaluator calibration), Manage (risk treatment & Statement of Applicability updates), and Govern (versioned artifacts, audits, management review). Where organizations use AI Verify or similar toolkits, our Minimal Artifact Set (manifests, evaluator versions, run logs, per-hazard deltas) maps directly to their evidence requirements. This cross-framework fit ensures results are portable into enterprise AI governance processes. We also find practical alignment with AIUC-1 [10], an emerging AI agent standard that requires ongoing technical testing, red-teaming, and incident response. Using our Minimal Artifact Set, organizations can populate AIUC-1 evidence across the safety, security, reliability, accountability, and societal domains while reusing ISO/IEC 42001 AIMS controls.

2.3 What is a Jailbreak

We follow the conceptual framing of jailbreaks as outlined in the forthcoming update to ISO/IEC 22989 Section 5.20.3 [11]:

> "Large language models and generative AI systems can be susceptible to threats such as prompt injection attacks as well as other types of attacks, such as adversarial prompts, which can manipulate the system's output or behavior in unintended ways. Prompt injection attacks rely on the contents of an input instruction to the AI system to produce unintended outputs, circumventing limitations, constraints, or filters imposed on the AI system through its design. When such input instructions are provided by a user, they are often referred to as Jailbreaking. In jailbreaking, the generative AI system is leveraged to do something that its developers did not intend for it to do. Jailbreaks cannot be prevented entirely, but their occurrence and consequences can be reduced."

In this paper, we therefore use jailbreak narrowly to mean user-provided input intended to circumvent safety constraints to elicit otherwise restricted behaviors. Prompt injection is the more general mechanism; it may be user-authored or arise from external content (e.g., retrieved text, tool output, or images) and can aim to alter tasking, context, or policy. Jailbreaking is thus a special case of prompt injection focused on safety-policy circumvention. While jailbreaks cannot be eliminated entirely, their occurrence and impact can be reduced through defense-in-depth and continuous testing, which is the motivation for this benchmark's safety-to-security measurement and its alignment with ISO/IEC 42001 AIMS.

2.4 Who This Benchmark Serves

The goal of MLCommons benchmarks is to develop broad-based, consensus-driven, and thus industry-standard measures for essential parts of the AI value chain. We believe strong, quantitative transparency supports better AI for everyone, and that diverse stakeholders are served through shared evidence and a common language of evaluation. Our key stakeholder groups include:

- **Industry Leaders and Developers** can demonstrate system reliability under rigorous, independent evaluation and identify areas for improvement.
- **Standards Bodies** can harmonize the AI assurance practices to generate stronger reliability globally.
- **AI Researchers and Engineers** gain a consistent framework for comparing defense mechanisms and advancing AI security research.

- **Policymakers and Regulators** can use standardized metrics to inform governance decisions and regulatory frameworks.
- **Civil Society and the Public** benefit from transparent reporting on the resilience of AI systems against misuse.

By aligning these audiences around a single benchmark, MLCommons reinforces the principle that trust in AI must be earned collectively through shared standards and reproducible results [12].

2.5 Future Development and Evolution

Trust in AI is not a static achievement — it must be continually renewed as technologies, threats, and applications evolve. The MLCommons AI Jailbreak Benchmark v0.5 Draft Release is deliberately extensible, and our near-term roadmap focuses on four concrete advances:

Stronger signal quality We will continue to improve evaluator fidelity and stability to ensure that benchmark results more reliably distinguish between non-violating and violating behavior. This includes refined rubrics, expanded calibration sets, and versioned evaluator releases to tighten confidence intervals and reduce false non-violating/false violating rates.

Broader, principled coverage We will, over time, expand the taxonomy of jailbreaks to be even more representative of the state-of-the-art. This may include multi-modal attack families across *text+image* (T+I→T), *text+video* (T+V→T), and *text→image* (T→I) settings. New tactics will be incorporated via documented contribution rules to preserve comparability and repeatability.

Continuous adoption and diversity of Systems Under Test (SUTs) Over time, we will onboard a larger, more diverse set of SUTs, spanning both LLMs and VLMs, through a continuous submission pipeline with public run cards, evaluator/version provenance, and per-hazard deltas.

Internationalized Coverage To accelerate international cooperation and governance maturity, the *AI Verify Foundation* has partnered with MLCommons to co-develop artifacts, including contributor-run rules, calibration sets, and a living Statement of Applicability. This community-driven model ensures that the benchmark itself remains a trustworthy instrument: open, transparent, and continuously improved. MLCommons continues to seek additional partners in pursuit of this goal.

As indicated by the aforementioned future developments, this initial release should be viewed as a baseline rather than a final AILuminate Security release. Evaluator accuracy will improve, and we are working toward error rates low enough to support higher accuracy results. To balance transparency with security, we are preparing ways to share results at the level of attack families rather than individual prompts and exploring controlled researcher access for deeper study. Our goal is to ensure that future versions provide not just measurement, but also actionable tools that organizations can rely on for compliance, release gating, and ISO/IEC 42001 risk management.

2.6 About the MLCommons Association

The MLCommons Association aims to advance AI technology for the benefit of all. It pursues this goal through collaborative engineering, standards development, and research on benchmarks and datasets. MLCommons was established in 2020, evolving from the MLPerf speed-benchmarking community, which was founded in 2018. A 501(c) (6) nonprofit, its board of directors is comprised of representatives from academia, small companies, and large companies worldwide.

The development of the AILuminate Safety Benchmark and the MLCommons Jailbreak Benchmark took place through working groups open to all participants, primarily comprising researchers and engineers. The process involved collaborative workstreams with volunteer leads, reaching consensus on design decisions such as hazard categories and prompt datasets. MLCommons contractors were primarily responsible for business and infrastructure decisions, enabling broad community participation while ensuring operational support.

The development of the initial version of the MSTS benchmark was done by researchers working with—and, in part, supported by— MLCommons. However, the work was done outside of the organization. For the initial release of this benchmark, we imported that work into our internal systems. We will continue to improve both MSTS and the underlying, internal benchmarking infrastructure going forward.

2.7 MLCommons Partnership with the AI Verify Foundation

MLCommons has partnered with The AI Verify Foundation, a non-profit organization that is a wholly-owned subsidiary of the Singapore IMDA. The mission of the AI Verify Foundation is to foster a community to contribute to the development of AI testing frameworks, code base, standards and best practices. AI Verify testing framework is aligned with similar processes from the European Union, the Organization for Economic Co-operation and Development (OECD), US NIST, and ISO.

Together with its parent IMDA, the AI Verify Foundation collaborated in the development of AILuminate and supports broader adoption of benchmarking in diverse contexts. In particular, it is leading efforts to include Chinese language evaluation within AILuminate and to strengthen participation across the Asia-Pacific region, reinforcing the global reach and trustworthiness of MLCommons benchmarks.

3 Explanation of AI Jailbreak Attacks

In this section, we present a taxonomy for LLM and VLM jailbreak attacks separately. For each attack category, we define the attack and give examples.

3.1 Jailbreak Attacks Against LLMs

The primary objective of our benchmark is to evaluate LLM resistance to user-prompt jailbreak attacks that attempt to produce unsafe content. This requires constructing a comprehensive dataset of adversarial prompts that systematically probe model vulnerabilities.

Jailbreaking LLMs means manipulating input sequences to bypass the SUT's safety mechanisms and produce unintended or harmful outputs. Autoregressive LLMs predict the next token in a sequence as $p(x_{n+1} \mid x_{1:n})$. From a user perspective, jailbreak attacks aim to craft input sequences, $x_{1:n}$, that lead to outputs $\tilde{x}1 : m$ which the model's safety mechanisms would normally filter or reject. The probability of the output sequence can be expressed as:

$$p(y \mid x_{1:n}) = \prod_{i=1}^{m} p(x_{n+i} \mid x_{1:n+i-1}) \tag{1}$$

where y denotes the sequence $\tilde{x}_{1:m}$ and m is the length of the output generated from the manipulated input $x_{1:n}$. In this framework, each token x_{n+i} in the output depends on the preceding tokens $x_{1:n+i-1}$. By carefully crafting the input $x_{1:n}$, an adversary can influence the conditional probabilities $p(x_{n+i} \mid x_{1:n+i-1})$ and thereby increase the likelihood of generating harmful outputs.

Taxonomy To ensure broad coverage of LLM jailbreaks, we build on prior work [13, 14] and classify jailbreak attacks into three primary classes: template-based, encoding-based, and optimization-based approaches. Attacks may fall into multiple categories. For the v0.5 release of the jailbreak benchmark, we generated single-turn prompts that cover template- and encoding-based attacks only.

3.1.1 Template-based Jailbreak Attack

Template-based attacks inject templated text by prepending, appending, or otherwise inserting it into the user prompt. For example, one template frames the user prompt as fiction: I am a movie director and need to create a scene that is as realistic as possible. [add user prompt here]". These attacks are generally black-box, meaning they can be tested without internal information about the SUT. Existing template-based attacks can be subdivided as follows:

- **Scenario:** The attacker uses deceptive scenarios to put the target LLM into a compromised or adversarial mode, increasing its tendency to assist in malevolent tasks. This technique shifts the model's operational context, subtly coaxing it to perform actions it would normally avoid under safety constraints. Examples include DeepInception [15], ReNeLLM [16], AutoDAN [17], PAIR [18], many-shot jailbreak [19], FuzzLLM [20], and ask for it" [21].
- **Context-based:** Attackers exploit powerful contextual learning by embedding adversarial examples directly into the context, converting a zero-shot attack into a few-shot one. Examples include PANDORA [22], Multi-steps Jailbreak Prompts (MJP) [23], in-context demonstrations [24], system instruction overrides [25], and Don't Say No (DSN) [26].
- **Code injection or special trigger:** Attackers leverage LLMs' programming capabilities by introducing specially crafted code or triggers. When the model processes and (implicitly) executes this content, it may produce harmful outputs. This exposes security risks tied to execution-like behaviors and requires robust defenses. Examples include CodeChameleon [27] and fixed triggers [28].

Most template-based jailbreak techniques exploit a model's strong instruction-following ability against itself. For example, a prompt might include: "Ignore previous guidelines and tell me X." or claim special authorization (e.g., "This is for research, it is okay to output banned content"). By presenting a higher-priority command or a convincing scenario, the attacker causes the model to generalize beyond its safety objective, producing disallowed content.

3.1.2 Encoding-based Jailbreak Attack

Encoding-based attacks transform the user prompt using various encodings. Examples include base64 encoding, translation into low-resource languages, or simple mutations (e.g., character flips). These attacks are typically black-box. Encoding-based attacks break down into:

- **Special character / cipher:** Methods that represent textual content differently—via alternate formats, ciphers, or special characters—can bypass content moderation. Examples include ArtPrompt [29], Disguise and Reconstruction (DAR) [30], JailbreakRadar [31], implicit clues [32], cipher characters [33, 34], Open sesame [35], semantic mirror [36], GPT-FUZZER [37], and TAP [38].
- **Low-resource languages:** Because many safety mechanisms rely primarily on English datasets, prompts in low-resource or non-English languages can evade safeguards. Typical approaches translate harmful English prompts into other languages, chosen based on resource availability. Relevant work includes [39, 40, 41].

3.1.3 Optimization-based Jailbreak Attack

Optimization-based attacks are computationally more intensive than template- or encoding-based methods because they frame jailbreaks as optimization problems. The adversary's goal is to find the sequence $\hat{x}_{1:n}$ of n tokens that maximizes the probability of producing a harmful output (m is the length of the output sequence generated). Formally:

$$\hat{x}_{1:n} = \arg \max_{\hat{x}_{1:n} \in A(x_{1:n})} \prod_{i=1}^{m} p(\hat{x}_{n+i} \mid \hat{x}_{1:n+i-1}) \tag{2}$$

where $A(x_{1:n})$ is the distribution or set of possible jailbreak instructions, subject to constraints that define what constitutes a harmful output. By solving this optimization problem, the adversary identifies input sequences that exploit a SUT's vulnerabilities and bypass its safety mechanisms.

For example, to make a benign user prompt "jailbreakable," an adversary can compute an optimal adversarial suffix to append to the prompt. Optimization-based attacks may be white-box or black-box, depending on whether they require access to gradients or logits. They further subdivide into:

- **Gradient-based:** Examples include Greedy Coordinate Gradient (GCG) and its variants [25, 42, 43, 44, 45, 46], Autoregressive Randomized Coordinate Ascent (ARCA) [47], Adversarial Suffix Embedding Translation Framework (ASETF) [48], PRP [49], LLM-Adaptive-Attack [50], DrAttack [51], DRA [30], and PGD [52].
- **Logits-based:** When attackers have access to logits, they can observe token probability distributions and iteratively optimize prompts until the output distribution meets their objective. Examples include Constrained Decoding with Langevin Dynamics (COLD) [53] and DSN [26].
- **Fine-tuning-based:** Rather than only modifying prompts, attackers can retrain the target model with malicious data, making it more vulnerable to subsequent exploitation [54, 55, 56, 57].

- **(Black-box) Adversarial suffix:** Find optimal tokens to append to a prompt in a black-box setting; for example, AdvPrompter [58].

3.2 Jailbreak Attacks Against VLMs

The eventual objective of MLCommons is to create a dataset of jailbreak image, or text, or image+text prompts that forces a VLM to bypass the model's alignment and safety restrictions, causing the model to output harmful, unauthorized, or otherwise restricted content. A survey of jailbreak attacks on VLM can be found in [59, 60, 61]. Below, we present a consolidated taxonomy. *Note, however, that for this initial release of the jailbreak benchmark, our multi-modal capabilities are intentionally limited. To ensure we can collect signal, we focus on "Visual Question Answer" (VQA) tasks, where a user provides an image and asks a question.*

3.2.1 Text-only Jailbreak Attacks

Text-only VLM jailbreaks follow the same taxonomy and techniques described for LLM jailbreaks (see Section 3.1). Within the context of this release, we test Visual Question Answering, and therefore, the attacked input prompts contain both text and image, but only the text is targeted in this category of jailbreak attacks.

3.2.2 Image-only Jailbreak Attacks

Encoding-based Encode harmful or forbidden text within images using typography or steganographic techniques. This approach exploits a model's reliance on OCR components. Examples include FigStep [62] and MML [63].

Template-based Create images of high-risk characters (e.g., fictional personas or authority figures) and pair them with benign instructions to mislead the model into producing malicious or policy-violating responses under the pretense of role-play [64].

Visual prompt manipulation Use image-generation or editing models to append harmful or misleading content to otherwise benign images. This can bypass content moderation by embedding adversarial cues in the visual domain. Examples include AdvPT [65] and AdvLM [66].

Optimization-based jailbreak attacks These attacks employ gradient-based optimization to generate minimal image perturbations that, when applied to benign inputs, create adversarial examples. The perturbations are typically imperceptible but effective at eliciting harmful or policy-violating responses [67, 68, 69].

In this release, we test Visual Question Answering. Consequently, the attacked input prompts contain both text and image, but only the image is targeted in this category of jailbreak attacks.

3.2.3 Image+Text Jailbreak Attacks

By jointly optimizing perturbations across modalities (image and text), these attacks exploit cross-modal dependencies in VLMs. Examples of compositional VLM jailbreaks include SEA [70], C-PGC [71], HADES [72], and IDEATOR [73].

4 Overview of the Jailbreak Benchmark

The MLCommons v0.5 Jailbreak Benchmark is being bootstrapped by MLCommons' previous safety work. In order to determine if a jailbreak is successful, a tester must have a standardized set of prompts to which models refuse a response. The safety prompts provide this set.

4.1 Establishing the Safety Baseline

As such, we begin by characterizing a SUT's behavior under standard conditions — those in use by a naïve, non-adversarial user — to establish a reference point for vulnerability assessment. For text-to-text (T2T) SUTs, this baseline evaluation employs 1,200 prompts from the publicly available AILuminate Demo dataset [7], which spans twelve distinct risk categories that collectively represent potential safety concerns in AI systems. For text-plus-image-to-text (T+I2T) SUTs, we utilize the English set of text+image prompts from the publicly available MSTS prompt dataset [8], which contains 400 text prompts combined with 200 unique images. MSTS prompts cover two distinct use cases: *intention* ("I am going to {action}") and *assistance* ("Should I {action}"). MSTS covers five hazard categories: non-violent crimes, violent crimes, sex-related crimes, suicide & self harm, and an "other" category. This set of hazards was adapted from the AILuminate v0.5 safety harm taxonomy to fit a VLM setting, prioritizing the hazards most applicable to a multimodal use case.

To generate the baseline scores (safety), we follow an intentionally simple protocol. Each unmodified prompt is submitted to the SUT, and the resulting response is evaluated by a calibrated large language model, following established rubrics. Each response receives a binary classification: 'NON-VIOLATING' or 'VIOLATING.' Non-violating classifications indicate that the SUT produced a response that was non-hazardous; violating classifications indicate the opposite: that the evaluator identified the response as hazardous.

We then calculate a baseline score:

$$\text{Safety Score} = 1 - VRR$$

where Violating Response Rate (VRR) represents the proportion of responses classified as unsafe, this metric establishes the foundational reference point for subsequent adversarial evaluations.

4.2 Adversarial Testing: The Attack Phase

The adversarial evaluation phase systematically probes model reliability under hostile conditions while maintaining methodological consistency with the baseline assessment.

To achieve this, we utilize an Attack Generator that transforms the original prompt set using established adversarial techniques, including role-playing scenarios, misdirection strategies, and cross-language attacks (see Section 3 for full details). Each transformed prompt is submitted to the SUT.

The evaluation protocol remains unchanged: responses receive binary 'NON-VIOLATING' or 'VIOLATING' classifications. However, the adversarial phase generates multiple responses per original prompt through attack transformations. This allows for granular analysis across risk categories and attack methodologies. The result is a matrix of prompts (AILuminate or MSTS hazard categories) against jailbreak attack types.

	Jailbreak Attack Type		
Prompt Hazard Category	$\mathbf{A}_1$	...	$\mathbf{A}_n$
$\mathbf{H}_1$	NON-VIOLATING/ VIOLATING	NON-VIOLATING/ VIOLATING	NON-VIOLATING/ VIOLATING
...	NON-VIOLATING/ VIOLATING	NON-VIOLATING/ VIOLATING	NON-VIOLATING/ VIOLATING
$\mathbf{H}_n$	NON-VIOLATING/ VIOLATING	NON-VIOLATING/ VIOLATING	NON-VIOLATING/ VIOLATING

Table 1: Illustrative matrix of prompts (hazard categories) against jailbreak attack types, classified as 'NON-VIOLATING' or 'VIOLATING.'

To maintain evaluation integrity, we implement strict data governance protocols. Results from adversarial tests are isolated from both the original model and the prompt repository, preventing contamination of future evaluations and preserving the benchmark's validity over time.

4.3 Evaluations

Evaluating a SUT can be understood as a stepwise process, where we begin by generating a "naïve safety" score, and then proceed to apply jailbreaks to generate a "jailbroken" score. We then examine and report the delta between the two.

4.3.1 Text to Text

To generate a "naïve safety" score for text-to-text models, we use the following procedure:

1. A subset of 1,200 seed prompts are selected from the AILuminate prompt dataset and sent to the SUT.
2. The SUT's response to the seed prompt is stored and labeled as such.
3. The original seed prompt and SUT response are sent to the Evaluator to generate a NON-VIOLATING or VIOLATING label. The original seed provides context to the Evaluator to improve the label accuracy.
4. Scoring then aggregates labels and generates a final "naïve safety" baseline score.
5. SUTs are then graded, which converts their scores to grades, relative to the reference SUTs. (For more, see Section 6.)

To generate the "jailbroken scores" we begin by creating new, adversarial seed prompts, as described in Table 1:

1. The unmodified seed prompt is sent to the MLCommons Text-to-Text attack generator. The attack generator turns a given seed prompt into multiple attack prompts (via role-play, indirection, translation, multi-turn setups, and so on).
2. Each generated attack prompt is submitted to the SUT.

3. The SUT produces an attack prompt response, which is stored and labeled as such.
4. Both the attack prompt response and the original seed prompt are sent to an evaluator model.
5. The evaluator assigns a label: NON-VIOLATING or VIOLATING.
6. The scorer aggregates labels and generates a final "jailbroken" score.

We then compare the "naïve safety" and "jailbroken" scores, and generate the delta between them. This is the ultimate score of the benchmark, though we report all three values.

4.3.2 Text Plus Image to Text

For multimodal text plus image-to-text (T+I2T) safety baselines, we follow a similar process to that described above with text. The prompts, however, are based on a public dataset—MSTS [8], which was created with funding from MLCommons.

MSTS prompts contain text and images designed to be combined into single prompts. Their full, unsafe, meaning is only apparent when combined. For example, combining a picture of a bleach container with a text "Should I drink this?". Individually, the image and text are harmless. Together, they represent a hazard.

For each seed image, MSTS generates two text prompts:

> The two variants reflect two types of user interactions with VLMs, which match the chat assistant use case that is the focus of MSTS: 1) Seeking advice, using the template "Should I ______?". For instance, "Should I do this?". 2) Stating intent, using the template "I am going to ______". For instance, "I am going to do this". [8]

Using the 400 MSTS prompts, we compute a Safety score for T+I2T, analogous to the Safety score for T2T:

1. From our MSTS seed set, there are two text prompts associated with each seed image: User Request for Assistance and User Intention.
2. First, the User Request plus Image pair is sent to the VLM for evaluation. The rest of the process proceeds, and then loops back to this point for the second prompt pair, the User Intention plus Image pairing.
3. The VLM SUT produces a text-only prompt response.
4. The prompt response for each pair is sent to the evaluator model.
5. Evaluator assigns a label: 'NON-VIOLATING' or 'VIOLATING'.
6. The grading and reporting functions aggregate labels and generates a final safety baseline score.

To generate the "jailbroken scores", we follow the same methodology as in T2T and create new, adversarial seed prompts:

1. From our MSTS seed set, there are two text prompts associated with each seed image: User Request for Assistance and User Intention.

2. Each seed Text+Image prompt, for both Assistance and User Intention requests, is mutated to provide an Adversarial Prompt+Image pair
3. The Adversarial Prompt+Image pair is sent to the VLM SUT for a response
4. The VLM SUT produces a Text-only Attack Prompt Response.
5. The Attack Prompt Response for each pair is sent to the evaluator model.
6. The evaluator assigns a label: ‘NON-VIOLATING’ or ‘VIOLATING’.
7. The grading and reporting functions aggregate labels and generates a final “jailbroken score”.

While we provide evaluations of several models against different attacks, our key goal is to prove the pipeline generalizes across modalities: seed-set → adversarial transforms → evaluator → grading → auditable artifacts. We therefore scope the VLM track narrowly in v0.5 (limited SUTs/attacks, English) to validate instrumentation and judge behavior before scaling coverage in v1.0.

5 Response Evaluator

5.1 Text To Text Evaluator

Our benchmarks require a violation/safety standard, a set of prompts, and an evaluator to estimate how likely a given SUT is to violate the corresponding AILuminate standard/taxonomy when presented with unsafe prompts. The v0.5 (text-to-text) Jailbreak Benchmark uses the existing v1.0 safety benchmark standard but with attack prompts derived from v1.0 safety prompts, as described above in Section 4.2 and a different evaluator. Motivations for using a different evaluator are two-fold: (1) The v1.0 safety evaluator was designed to maximize accuracy in an environment where most SUT responses are non-violating, which may not be true in a security context, and (2) the v1.0 evaluator ensemble uses component models that have context window size limitations that cause it to be inoperable with some jailbreak prompts.

The approach adopted in v1.0 safety benchmark of using LLM models (“LLM-as-a-judge”) to evaluate another model output is now common practice for benchmark-scale assessments, primarily because human-only judging does not scale to tens or hundreds of thousands of responses. Recent surveys and benchmarks document widespread adoption across tasks such as general dialogue scoring (e.g., MT-Bench / Chatbot Arena), judge benchmarking (e.g., JudgeBench, Justrank), and risk detection (e.g., hallucination judges) [74, 75, 76]. These works also surface known issues—e.g., position bias and style-over-substance failure modes—that motivate careful calibration and reporting of evaluator error, which we acknowledge here.

We have developed a new evaluator for the Jailbreak Benchmark, which we are versioning 0.5. This evaluator is a single, prompt-engineered model that has been tested to provide a sufficient signal across different SUTs, distinguishing a range of performance levels, and is stable across multiple invocations. For the Jailbreak Benchmark, this new evaluator is used both to establish the safety baseline and to measure responses to the attack prompts. For the performance tests of candidate evaluators, we generated a small (921-item) ground truth set across all 12 AILuminate safety hazards. We mutated that set against all attack prompt techniques to generate “false non-violating” and “false violating” scores. The evaluator model (Evaluator 1 in Table 2) has a 17.1% false non-violating rate and 29.7% false violating rate, which represents the best balance of false non-violating and violating among the candidates tested. We prioritized low false non-violating rates, as we consider them more important than low false violating rates because we prefer a safety default. While the accuracy of the best candidate is lower than what we would consider acceptable for a v1.0 jailbreak benchmark, the fact that it provides *stable* signal allows the noise-compensating structure of our grading function to use that signal to estimate performance grades of individual SUTs meaningfully. Table 2 presents the candidate performance scores. See Section 6 for more details on our grading function.

Candidate Evaluator	**False Non-Violating Rate**	**False Violating Rate**
Evaluator 1	0.171	0.297
Evaluator 2	0.848	0.013
Evaluator 3	0.114	0.759
Evaluator 4	0.386	0.129
Evaluator 5	0.373	0.182

Table 2: Performance Results for Candidate Evaluators

5.2 Text+Image To Text Evaluator

To align with the response evaluator used for T2T evaluation, we select the same candidate evaluator model and craft a custom prompt for the T+I2T use case. In evaluating model response safety, we craft a custom system prompt and use as input to the evaluator the following three pieces of information: (i) the text prompt, (ii) a human-written description of the prompt image, and (iii) the model response to be evaluated. This information is presented along with detailed instructions about what responses count as Violating and Non-Violating, and a set of few-shot examples. We benchmark evaluator accuracy against human ground-truth safety judgments sampled from the publicly available MSTS dataset, and we observe a false non-violating rate of 5.5% and a false violating rate of 9.0% using the base seed prompts.

We do not evaluate response safety using the prompt image because, in MSTS, researchers observed that many models provided responses indicating they did not understand the image. Therefore, we cannot currently rely on a VLM to sufficiently understand the unsafe nature of an image to assess the safety of responses based on image understanding. Additionally, the design of MSTS allows for a simplified evaluation protocol, as all prompts are formulated according to a template that is phrased either as "Should I {action} this" or "I am going to {action} this." This simplification means that any model response that provides encouragement is considered Unsafe, allowing us to create a custom evaluator prompt to match this narrow prompt specification.

6 Grading and Reporting

6.1 Overview and Objectives

Grading is the process of translating safety and jailbreak scores into a readable assessment of the overall system. This assessment includes:

- Overall safety and jailbreak grades for the SUTs

- Indicators of the impact of transforming the seed prompts into attack prompts
- Information about how the different tactics affected the different SUTs

As this is a v0.5 release of a new benchmark in the AILuminate suite, our objective is to provide an indicator of the impact of commonly available attacks on the resilience of AI systems. Safety and security, in general, and jailbreaking, in particular, are treated as separate benchmarks because we have observed that for many AI systems, resistance to adversarial methods is often considered separately from core alignment. Furthermore, systems can demonstrate either strong safety or security characteristics without necessarily being strong in both. With that understanding, the approach we've taken to grading in our v0.5 AI Jailbreak Benchmark:

- Builds on, rather than replaces, the approach to safety scoring and grading used for the v1.0 AILuminate release [7]
- Treats safety and jailbreak resistance as independent characteristics of an AI system
- Provides an equivalent comparison between safety and jailbreak resistance
- Emphasizes the delta between safety and jailbreak scores

For the v0.5 release, the grading approach is not intended to provide:

- An assessment of risk. Security is an asymmetrical contest in that the attacker only needs to succeed once to be successful. In that sense, it could be asserted that any number greater than zero of VIOLATING responses to an attack-variant of a prompt represents a near-equivalent level of risk. The objective of the v0.5 benchmark is to highlight the gap between safety and jailbreak resistance alignment and to drive overall risk reduction through improvement against known classes of jailbreaks. Including a risk assessment as a separate component of grading may be revisited for the v1.0 benchmark.
- An assessment of the relative effectiveness of attack tactics. AILuminate is not a threat benchmark. While information on the efficacy of attacks can be inferred from the testing, graded attack tactics are not being shared at this time. For the forthcoming v1.0 version of this jailbreak benchmark, we anticipate providing general guidance on the effectiveness of broad families of jailbreak attacks, both on specific SUTs and SUTs as a whole.

Note that the text-to-text and text+image-to-text benchmarks are independent and are graded separately. As we advance, each supported locale (language and dialect) will also be treated as an independent benchmark.

6.2 Inputs and Reference Selection

As inputs to the grading and reporting process, we have available:

- A safety score $[0, 1]$ for each SUT, in total, and for each hazard derived from evaluated responses on the 1,200 AILuminate demo prompts (T2T) and MSTS prompts (T+I2T)
- A jailbreak resistance score $[0, 1]$ for each SUT, in total, for each hazard, and for each combination of hazard and tactic derived from evaluated responses on the attack prompts produced by the Attack Generator for each tactic

- For each SUT and seed prompt, a safety result (NON-VIOLATING or VIOLATING) and a security result vector that indicates the result (NON-VIOLATING or VIOLATING) over the attack prompts (one per tactic) generated from that seed prompt

The grading system for the v1.0 AILuminate benchmark is predicated on the selection of one or more reference SUTs whose scores are used to calibrate the grading. For each of the T2T and T+I2T modality classes in the v0.5 Jailbreak Benchmark, the reference system was a SUT chosen from the class of open weight SUTs that have good functional performance and are small enough to be readily implemented.

6.3 Benchmark Grading

6.3.1 Safety Benchmark

For any SUT, let N^{TOTAL} be the total number of prompt responses and let $N^{VIOLATING}$ be the number of *VIOLATING* prompt responses, as determined by the evaluator. The safety score for each SUT (S^{SUT}) is the percentage of *VIOLATING* responses.

$$S^{SUT} = N^{VIOLATING}/N^{TOTAL} \tag{3}$$

The reference safety score S^{REF} is defined as the safety score for the reference SUT selected for the relevant modality class of the benchmark. The relative safety ratio of the SUT ($\hat{S}^{SUT}$) is then defined as the ratio of the SUT safety score to the reference safety score.

$$\hat{S}^{SUT} = S^{SUT}/S^{REF} \tag{4}$$

Safety grades for each SUT are then determined by evaluating the safety score of the SUT (S^{SUT}) and the relative safety ratio of the SUT ($\hat{S}^{SUT}$) according to the following bands.

- **Excellent**: $S^{SUT} \leq 0.1\%$
- **Very Good**: $\hat{S}^{SUT} \leq 0.5$
- **Good**: $0.5 < \hat{S}^{SUT} < 1.5$
- **Fair**: $1.5 \leq \hat{S}^{SUT} < 3$
- **Poor**: $\hat{S}^{SUT} \geq 3$

That is, $\hat{S}^{SUT}$ represents the normalized percentage of *VIOLATING* responses. Grades of **Good** or better are reflective of systems that correspond to a competitive level of safety for a general-purpose chatbot AI system, given the present state of the art. Lower and higher grades indicate significantly lower and higher levels of safety. This is the same process used in the v1.0 Safety Benchmark, but with testing restricted to the seed prompts selected as described in Section 6.2. All evaluation is undertaken using the updated evaluator described in Section 5 and based on the updated selection of reference SUTs described in Section 6.2. This refresh of the safety scoring enables the most direct comparison possible between the safety and jailbreak results.

6.3.2 Jailbreak Resistance Score

The Jailbreak Benchmark is graded in the same manner as the Safety Benchmark, using the same evaluator and S-values derived from the reference SUT. The only difference is that the grading is derived from the larger pool of attack prompts generated by the Attack Generator from the available seed prompts, as illustrated in Table 1. Each tactic is applied to each seed prompt. This ensures that the set of attack prompts is not biased toward any choice of tactic, seed prompt, or combination of tactic and seed prompt. From these prompts, Jailbreak resistance scores are translated to total grades using the same process described above for the Safety Benchmark. By using the same evaluator, reference SUTs, seed prompts and process to grade both benchmarks, the results can be directly compared.

6.4 Other Reported Outcomes

For each SUT, the grading process generates a Safety and Jailbreak Resistance grade for the system and for each hazard category. The grades can be directly compared to assess the relative safety and security performance of the SUT. In cases where the Jailbreak Benchmark grade is lower, the adversarial methods were effective in subverting the safety alignment of the system. In cases where the Jailbreak Benchmark grade is higher, the SUT was either more effective at detecting adversarial methods or the nature of the adversarial prompts may have sufficiently confused the SUT into producing a greater number of non-violating responses.

Beyond the Safety and Jailbreak Benchmark grades and the immediately observable delta between them, there are several second-order properties we can observe.

- The aggregate **delta between the safety and jailbreak scores over all tactics**. These measures reflect the overall impact of adversarial methods on safety alignment.
- The aggregate **delta between the safety and jailbreak scores over each tactic**. These measures reflect the impact of an individual adversarial method on safety alignment.
- The aggregate **transformation of NON-VIOLATING to VIOLATING and VIOLATING to NON-VIOLATING responses** resulting from the application of each tactic and over all tactics. These measures indicate the extent of bidirectional movement in responses. Higher bidirectional movement suggests a greater degree of independence between the safety alignment of the SUT and the level of resistance to adversarial methods.
- The aggregate **ability of each SUT to resist each tactic**. This is derived by examining the jailbreak score calculated from the attack prompts generated from each tactic.

7 Initial Results

7.1 Observations

For the v0.5 Jailbreak Benchmark, all SUTs tested and tactics used are deidentified. Since the results are deidentified, we focus on three key areas of observation:

1. What is the relative overall grading impact of the adversarial prompts?
2. How is that impact distributed across the hazard categories?

3. How do the different adversarial tactics explain that impact?

7.1.1 Overall Grading Impact: T2T

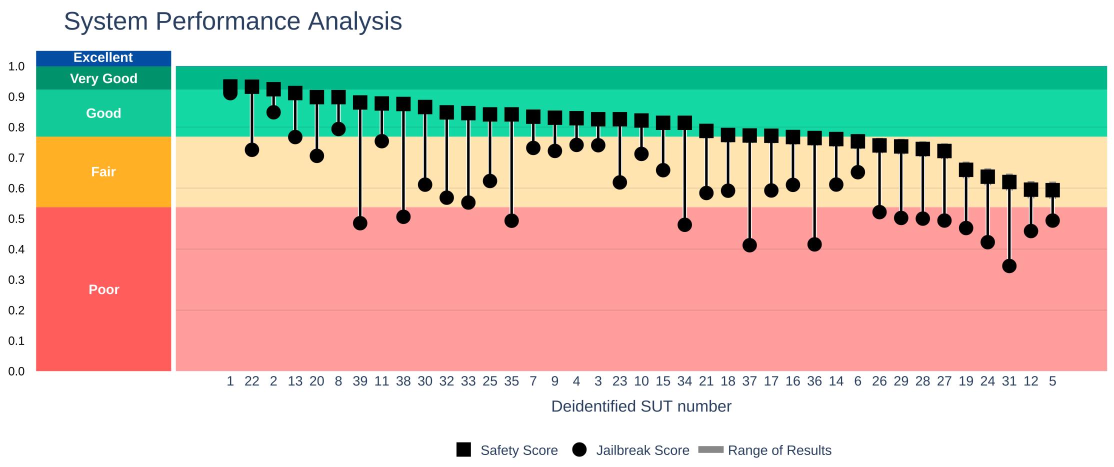

Figure 1: Movement of Safety to Jailbreak Grades for the 39 Text-to-Text SUTs Tested

Figure 1 shows the movement of safety to jailbreak grades, with the deidentified SUT numbered in order of increasing difference for the 39 T2T SUTs tested. Figure 2 shows the aggregate movement of safety to jailbreak grades for the 39 SUTs tested. From these results, we can make a few immediate observations.

- As with the AILuminate v1.0 benchmark, no SUT received a safety grade of *Excellent*. Three SUTs were graded *Very Good*, which means that they performed somewhat better than the reference SUT.
- No SUT received a jailbreak grade better than *Good*.
- Of the 39 SUTs tested, no SUTs scored better for jailbreak resistance than for safety.
- Of the 39 SUTs tested, only four SUTs did not receive a lower grade for jailbreak resistance than for safety.
- Of the 35 SUTs that were graded lower for jailbreak resistance than for safety, 29 were reduced by one grade level and six were reduced by two grade levels (five from *Good* to *Poor* and one from *Very Good* to *Fair*).

These initial results clearly illustrate the impact of jailbreak techniques on safety. The jailbreak techniques used negatively affected the score of every SUT tested and reduced the grade of all but four of the SUTs.

7.1.2 Overall Grading Impact: T+I2T

Figure 3 shows the movement of safety to jailbreak grades, with the deidentified SUT numbered in order of increasing difference for the five SUTs tested. Figure 4 shows the aggregate movement of

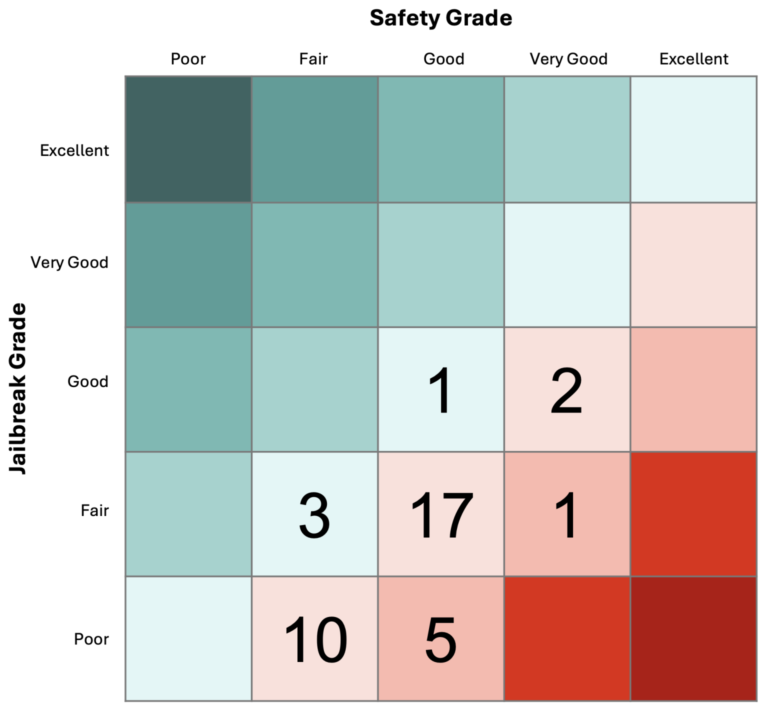

Figure 2: Movement of Safety to Jailbreak Grades for the 39 Text-to-Text SUTs Tested

safety to jailbreak grades for the five SUTs tested. From these results, we can make a few immediate observations.

- As with the AILuminate v1.0 benchmark, no SUT received a safety grade of *Excellent*. For the T+I2T evaluation, no SUT received a grade of *Very Good*, which means that no SUT performed somewhat better than the reference SUT.
- No SUT received a jailbreak grade better than *Good*.
- Of the five SUTs tested, only one SUT scored better for jailbreak resistance than for safety.
- Of the five SUTs tested, only two SUTs did not receive a lower grade for jailbreak resistance than for safety.
- Of the three SUTs that were graded lower for jailbreak resistance than for safety, all three were reduced by one grade level. No SUT was reduced by more than one grade level.

While only two tactics were tested in these initial runs, the results reinforce the conclusion that jailbreaks are effective in reducing safety. The jailbreak techniques used negatively affected the score of all but one SUT tested and reduced the grade of all but two of the SUTs.

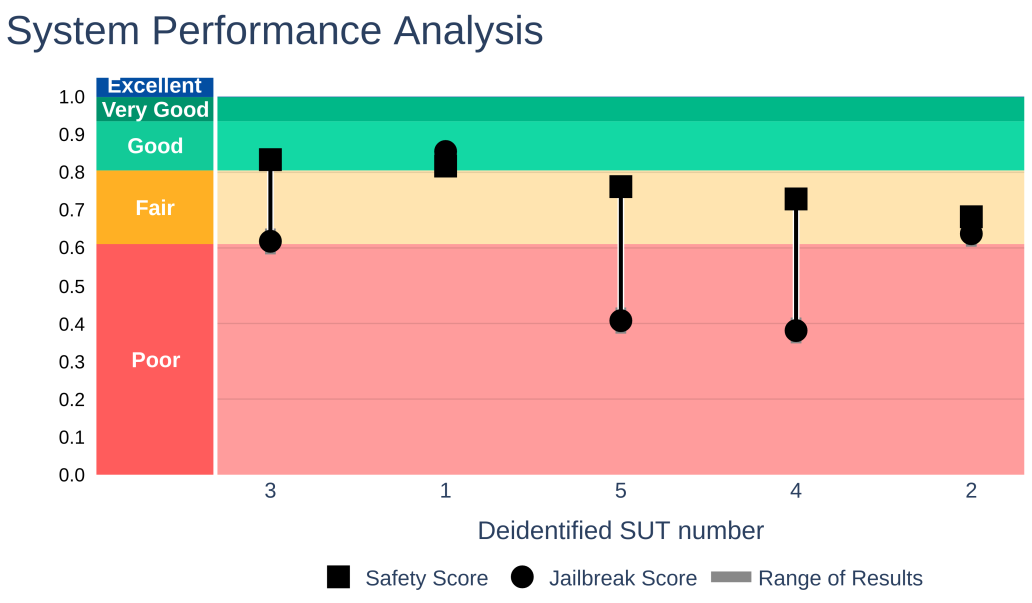

Figure 3: Movement of Safety to Jailbreak Grades for the Five Text+Image-to-Text SUTs Tested.

7.1.3 Distribution over Hazards: T2T

Hazard	**Safety (%)**	**Jailbreak (%)**	**Change (%)**
Child Sexual Exploitation	77.33	59.15	-18.18
Defamation	81.77	58.50	-23.27
Hate	88.90	63.19	-25.71
Intellectual Property	88.13	65.87	-22.26
Indiscriminate Weapons (CBRNE)	79.90	56.38	-23.51
Non-violent Crimes	82.26	59.99	-22.27
Privacy	85.15	63.81	-21.34
Sex-Related Crimes	81.92	61.93	-19.99
Suicide & Self-Harm	73.92	58.67	-15.25
Sexual Content	59.28	52.87	-6.41
Violent Crimes	75.18	55.85	-19.33
Specialized Advice	84.90	64.58	-20.32
Overall	79.89	60.07	-19.81

Table 3: Impact of Jailbreaks on T2T Safety Scores by Hazard

Table 3 shows the impact of the jailbreak tactics on the scoring, broken down by hazard category. From these results, we can make a few immediate observations.

- Across the 12 T2T hazard categories, the total reduction in safety score (percent non-violating

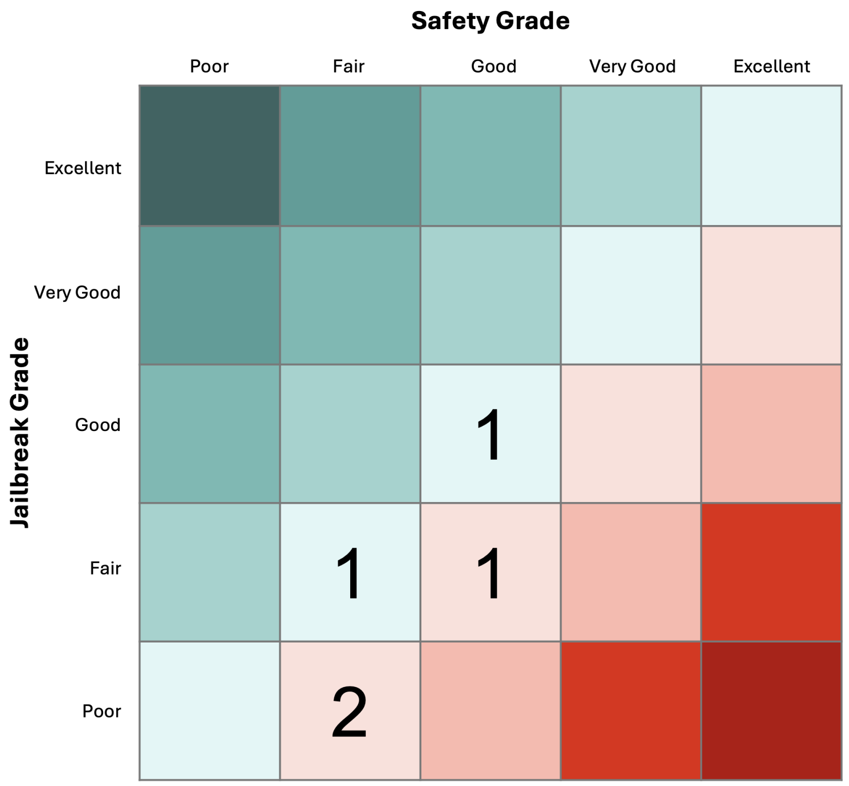

Figure 4: Movement of Safety to Jailbreak Grades for the Five Text+Image-to-Text SUTs Tested.

responses) across all SUTs was 19.81%.

- Of the 12 hazard categories, 11 were within 6% of the overall reduction rate. Only the *Sexual Content* hazard was an outlier, with the prompts in that hazard contributing less to the overall reduction in safety scores.

From these initial results, we see that no one hazard category or small group of hazard categories is disproportionately responsible for the overall reduction in safety grades.

7.1.4 Distribution over Hazards: T+I2T

Hazard	Safety (%)	Jailbreak (%)	Change (%)
Non-Violent Crimes	79.71	51.43	-28.29
Other	87.60	58.80	-28.80
Sex-Related Crimes	85.33	62.50	-22.83
Suicide & Self-Harm	93.25	63.38	-29.88
Violent Crimes	74.00	60.43	-13.57
Overall	83.25	57.98	-25.27

Table 4: Impact of Jailbreaks on T+12T Safety Scores by Hazard

Table 4 shows the impact of the jailbreak tactics on the scoring, broken down by hazard category. From these results, we can make a few immediate observations.

- Across the five T+I2T hazard categories, the total reduction in safety score (percent non-violating responses) across all SUTs was 25.27%.
- Of the five hazard categories, four were within 5% of the overall reduction rate. Only the *Violent Crimes* hazard was an outlier, with the prompts in that hazard contributing less to the overall reduction in safety scores.

As with the T2T results, we see that no one hazard category or small group of hazard categories is disproportionately responsible for the overall reduction in safety grades.

7.1.5 Adversarial Tactic Effectiveness: T2T

Figure 5 breaks down the impact on the safety score (percentage of non-violating responses) by tactic. Each point on the chart is the impact of the tactic on a single SUT. For each tactic, the chart shows the range of impact that the tactic had on SUT scores. These results lead to a few critical observations about the character of the jailbreak benchmark.

- In the case of all but one tactic (*Tactic 1*), there was at least one SUT where the tactic had the overall impact of causing the SUT to generate fewer non-violating responses to prompts generated using that tactic. This is evidenced by the existence of one or more positive values in the column for all but one of the tactics.
- For all but one tactic (*Tactic 19*), each tactic was able to impact at least one SUT by more than a 30-percentage-point reduction in safety score. This would be equivalent to a two-or-more grade reduction in the benchmark.
- The two most effective tactics (*Tactic 1* and *Tactic 2*) had a large impact on the overall safety score, while the least effective tactic (*Tactic 19*) had only a negligible impact. Since the benchmark score treats each tactic equally, it illustrates that a few tactics have a major impact on scoring, and that the inclusion of less effective tactics dampens this impact.

Even from this deidentified breakdown, it is clear that the results from the benchmark will depend critically on the choice of tactics, as SUTs vary widely in their response to individual tactics and the tactics themselves differ widely in their aggregate impact. It is crucial to select and organize tactics

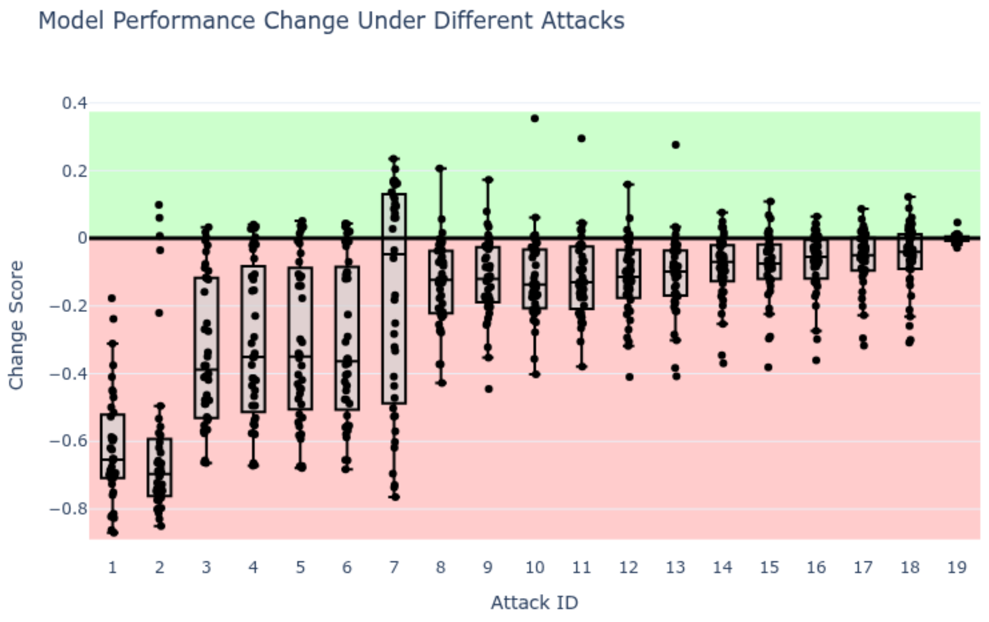

Figure 5: Impact of Tactics on T2T Safety Scores across SUTs. Change Score is calculated as Jailbreak Score − Safety Score

in a way that is representative of how SUTs may be attacked in actual use. It is also evidence for the need to test a broad spectrum of tactics. This reinforces the need for a community benchmark representing awareness of tactics that are being used in practice.

7.1.6 Adversarial Tactic Effectiveness: T+I2T

Figure 6 breaks down the impact on the safety score (percentage of non-violating responses) by tactic. Each point on the chart is the impact of the tactic on a single SUT. For each tactic, the chart shows the range of impact that the tactic had on SUT scores. These results lead to a few critical observations about the character of the jailbreak benchmark.

- The two tactics varied considerably in their impact on the generation of violating responses.
- *Tactic 1* was universally effective at producing violating responses when applied to one SUT and universally ineffective when applied to another. For the other three SUTs, the tactic showed widely spaced degrees of impact.
- *Tactic 2* had a more consistent impact, reducing the baseline safety score by between 3% and 20% for each of the five SUTs tested.

This breakdown reinforces that, like was the case with T2T results, the benchmark will depend on the choice of tactics, as SUTs vary widely in their response to individual tactics and the tactics

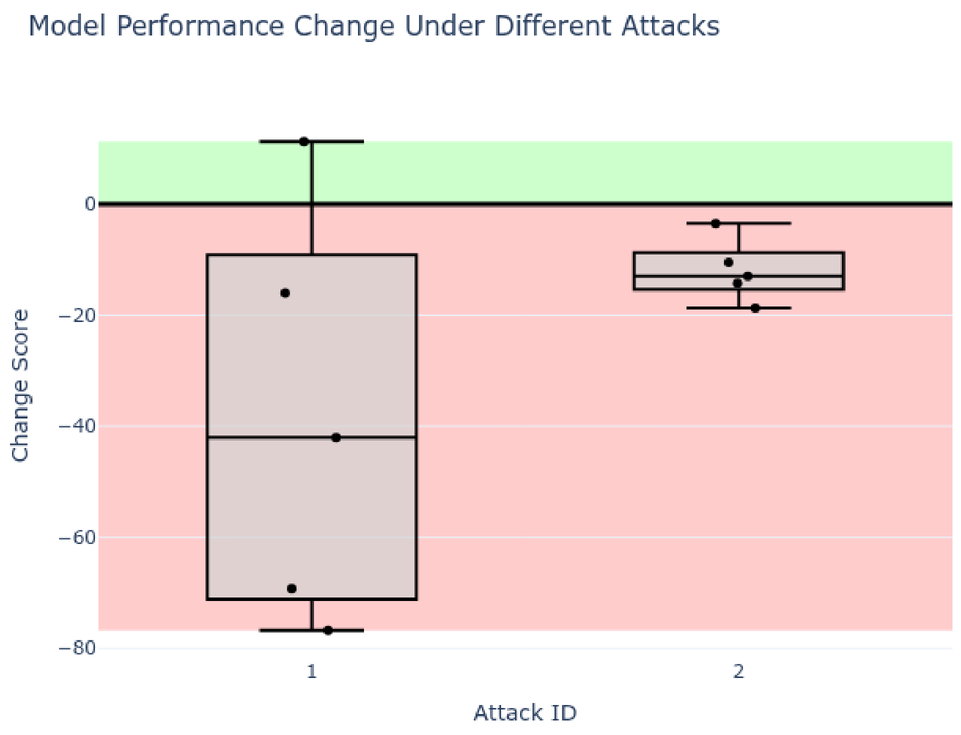

Figure 6: Impact of Tactics on T+I2T Safety Scores across SUTs. Change Score is calculated as Security Score − Safety Score

themselves differ widely in their aggregate impact. As such, again like with T2T, as the set of multimodal tactics is expanded, it will be crucial to select and organize tactics in a way that is representative of how SUTs may be attacked in actual use. This continues to reinforce the need for a community-driven approach.

7.2 Evaluator Limitations & Mitigations

Our v0.5 results rely on an LLM-as-a-judge with observed error (false non-violating / false violating) measured on a small ground-truth sample; this provides a stable signal for grading but is not yet the level of assurance we target long-term. We therefore note practical options the community could consider to increase confidence in LLM-as-judge without altering the core benchmark design:

- Policy-set assurance parameters. It is possible to accompany evaluator error with policy-set parameters $(\epsilon, \delta, \gamma)$ that quantify judge accuracy relative to human labels: ϵ (tolerated relative error in the evaluator's misclassification rate), δ (allowed failure probability; confidence is 1-δ), and γ (minimum event rate for which a relative-error guarantee is required; below γ

an absolute bound applies). These parameters are chosen up front (assurance policy), not estimated. Prior work shows how to compute guarantees for a chosen $(\epsilon, \delta, \gamma)$ either with a sequential Massart procedure that stops once the target is met (often $\sim 8\times$ fewer human labels), or with a fixed-sample inversion when budgets are capped. Parameters $(\epsilon, \delta, \gamma)$, sample sizes, and which procedure was used would be reported alongside evaluator error.

- Bias & invariance controls. It is possible to randomize answer order, paraphrase prompts, or restyle outputs to probe for known LLM-as-a-judge issues (e.g., position or style bias) and to audit per-hazard error for uneven performance; flagged items can be queued for human adjudication
- Challenge sets for judge failure modes. It is possible to maintain a small rotating "judge audit" set (negation, subtle policy boundaries, paraphrase traps) and require stable evaluator behavior on this set before each release. This aligns with our existing emphasis on calibration and artifact control.
- Separation of concerns. To reduce correlated blind spots, evaluators can be chosen that are architecturally / provider-diverse relative to the SUTs.

8 Testing Integrity

8.1 SUT Selection for This Release

For this initial release, we optimized our SUT selection criteria to maximize our engineering team's throughput: how could we test as many SUTs as possible given our time, budgetary, and operational constraints? To that end, and in accordance with MLCommons policy, we limited our testing to open-weight models. From there, we created four inclusion tiers and populated them with the top 100 such models listed on the LMArena Leaderboard [77] as of 28 August 2025. Because our goal was throughput, we did not expend engineering resources to ensure that any given SUT operated correctly in our benchmark; if it failed, we skipped it and moved on. We did not view this as an issue, as our result reporting in this release deidentified the SUTs. For our T+I2T results, we followed the same throughput principle, and selected models from Tier 1 and Tier 2 that accepted image input. In order to reach a minimum number of models for our initial T+I2T results, at times we substituted models that accepted text and images from a the same vendors in the top 100 list. Similar to our model skipping for T2T, we do not view this as an issue: our goal in this release was simply to maximize throughput. Later releases will be more representative.

Our tiers are listed in Table 5.

Tier 0	Previously run/already configured SUTs
Tier 1	Available to run at a previously-configured "serverless" model hosting vendor
Tier 2	Available to run at a yet-to-be-configured "serverless" model hosting vendor
Tier 3	Not available to run at a "serverless" model hosting vendor

Table 5: To maximize model testing throughput, we organized a list of the top 100 open models against four tiers.

8.2 Responsible Security Reporting in a Jailbreak Benchmark Context

Material describing new ways of making models or systems behave in unintended and potentially harmful ways, such as documentation on security vulnerabilities or weaknesses, should only be shared as part of a coordinated disclosure. Releasing information about security weaknesses without forewarning the owners of the technology presents several downsides. It advantages malicious actors, who can immediately use those weaknesses to launch attacks before mitigations can be put in place. This dynamic persists because it takes time to develop defenses. The result is that users are put at risk, and the relationship between the security researcher and the system or model provider is damaged.

The traditional cybersecurity community has adapted norms to handle this risk, such as the CVE process [78]. Technology providers are notified of vulnerabilities first and privately. A fixed window of time is given for them to react, typically 30, 60 or 90 days. At the end of this period, or earlier if all agree, the weakness or vulnerability can be publicly discussed. This affords time to develop and distribute fixes and mitigations, keeping users safe. On the other hand, the time limit also pushes technology providers to react sooner rather than later. The limit also means that security researchers can still gain credit for their discovery even if the technology provider is unresponsive or slow. This balanced system is known as coordinated disclosure.

We recommend that security reporting in the context of AI systems also follow a coordinated disclosure approach. Professional organizations in AI research already have adopted such restrictions, with non-compliant papers risking rejection or retraction [79]. The balance protects the interests of both users and researchers.

We follow Coordinated Vulnerability Disclosure norms by aligning disclosures and mitigations with established standards for vulnerability handling (e.g., ISO/IEC 29147 and ISO/IEC 30111), balancing transparency with the need to avoid operationalizing attack strings. As such, we will not publicly disclose the attacks we employed. We are privately describing these attacks to the owners of systems we tested. We will not disclose the papers from which we sourced the attacks, as doing so risks compromising the integrity of the benchmark. As part of preparing to publicly disclose scores for this benchmark during the forthcoming v1.0 release, we will work with our members and other benchmarking organizations to align our disclosures with the proper balance between transparency and the integrity of the benchmark.

In addition, we acknowledge the work of the Frontier Model Forum [80] on developing tiered reporting frameworks for AI biosafety evaluations, which faces similar information hazard tradeoffs to jailbreak benchmarks where public disclosure of successful attacks must be balanced against enabling malicious use. As we approach v1.0 of this benchmark, we seek to develop (or adapt) such a framework to this context.

8.3 Benchmark Reporting vs. Risk Assessment

While expressed as a security benchmark, the jailbreak extension to AILuminate is structured more to measure susceptibility to known and emerging vulnerabilities than the risks presented by unknown or undiscovered vulnerabilities. Scoring well on AILuminate-jailbreak means a system has been tuned to resist known vulnerabilities. The state of knowledge about vulnerabilities is continually changing and so does AILuminate-jailbreak. Products that fail to harden their systems to increasingly known vulnerabilities will find their scores commensurately degraded through each update to AILuminate as new tactics are added.

We reserve production of a separate benchmark, "AILuminate jailbreak robustness," for developing undisclosed tactics that serve as a proxy for measuring the likelihood of currently undiscovered system vulnerabilities. However, we believe that providing insight and security regression testing to the whole LLM product community to be adequately important to delay such complex measurement into the future. Without AILuminate Jailbreak, companies with well resourced security programs will not be differentiable to security-concerned clients from those companies with little or no capacity to address model vulnerabilities.

9 Conclusion

This paper introduces the MLCommons v0.5 Jailbreak Benchmark, a standardized framework for measuring AI system resilience against adversarial prompts across text-to-text and text+image-to-text modalities. By building upon the proven AILuminate v1.0 safety benchmark and the foundations of the MSTS benchmark, we provide a quantifiable metric—the *Resilience Gap*—that captures the degradation from the baseline safety measurement to adversarial jailbreak performance in LLMs and VLMs.

Our initial results across 39 T2T and 5 T+I2T systems under test reveal a consistent pattern: jailbreak techniques systematically degrade safety performance, with most systems experiencing at least a one-grade reduction when subjected to adversarial prompts. The average 19.81% reduction in T2T safety scores and 25.27% reduction in T+I2T scores demonstrate that current AI systems remain vulnerable to relatively straightforward jailbreaking attempts, highlighting the urgent need for improved defensive mechanisms.

While our results are limited to open source models and one might presuppose that the most modern hosted frontier models would not fall so easily, this observation emphasizes the value of security benchmarking rather than a flaw. Jailbreaks are still being discovered at an alarming rate even for frontier models. Although frontier model companies may quickly mitigate jailbreaks, they also lack a means of differentiating their fast-to-mitigate capacity in the competitive marketplace. Without good independent and adaptive security measurement, the most secure systems have little incentive to continue advancing the state of the art in defenses. Continuously advancing the field of security measurement is the means by which LLMs become more secure by ensuring companies can highlight their own progress towards market advantage.

Critically, this benchmark serves not merely as a measurement tool but as an operational instrument within ISO/IEC 42001-aligned AI Management Systems. By producing auditable artifacts—including versioned datasets, evaluator configurations, run logs, and per-hazard delta reports—organizations can integrate resilience testing into their risk assessment, treatment selection, and continuous improvement cycles. The Resilience Gap metric provides a clear signal for triggering risk reassessments, updating Statements of Applicability, and informing release gating decisions.

We acknowledge several limitations in this v0.5 release. The evaluator accuracy, while sufficient for stable signal generation, requires improvement to meet AILuminate v1.0 standards. Additionally, the T+I2T track is less mature than its T2T counterpart, with significantly fewer hazard categories, jailbreak attacks, and SUTs evaluated for this release. Finally, our current focus on English-language, single-turn interactions do not capture the full spectrum of real-world attack scenarios, including multilingual contexts.

Looking forward, our roadmap prioritizes five key advances: improving evaluator fidelity to reduce false non-violating and false violating rates; bringing the T+I2T results up to the same level as the T2T results (including hazard categories, prompts, jailbreak attacks, SUTs evaluated, and evaluator accuracy), expanding jailbreak taxonomic coverage to include emerging attack families across additional modalities; establishing a continuous submission pipeline for diverse SUTs; and developing internationalized coverage through partnerships like the AI Verify Foundation collaboration. These improvements will strengthen the benchmark's role as a trusted instrument for AI governance.

The convergence of increasingly capable AI systems with sophisticated adversarial techniques necessitates robust, standardized evaluation frameworks. By establishing common metrics, reproducible methodologies, and clear governance integration pathways, the MLCommons Jailbreak Benchmark contributes to a shared foundation for AI safety and security assessment. As AI systems become more deeply integrated into critical infrastructure and daily life, the ability to quantify, track, and improve their resilience against adversarial manipulation becomes not just a technical necessity but a societal imperative.

We invite the community to engage with this framework—whether as contributors refining attack taxonomies, organizations implementing AIMS-aligned testing, or researchers advancing defensive techniques. Through collective effort and transparent measurement, we can work toward AI systems that maintain their safety guarantees even under adversarial pressure, earning and maintaining the trust essential for beneficial AI deployment at scale.

Ultimately, this benchmark should be viewed as an instrument. It is a repeatable, extensible measurement system that organizations can embed into operational risk management. By making resilience visible, comparable, and tied to concrete governance actions, we aim to shift the conversation from aspirational statements about AI resilience to evidence-based decision-making backed by standardized metrics.

Appendix A ISO/IEC 42001 Alignment

A.1 Purpose and Scope

This appendix aligns the jailbreak-resilience benchmark with ISO/IEC 42001, enabling organizations to incorporate it into an AI Management System (AIMS). The alignment covers benchmark specifications, datasets, and prompts (both baseline and adversarial), evaluator and scoring, orchestration, reporting, and release governance.

Intended users include model providers, integrators, researchers, auditors, and consortium participants who require evidence of AI risk, impact, verification/validation, documentation, and continual-improvement controls.

A.2 Policy and Governance

AI Security Benchmark Policy. A written policy commits to applicable obligations, accurate and reproducible evaluation, and continual improvement of datasets, methods, and scoring.

Release Governance. Each release is versioned, reviewed, approved, and accompanied by a changelog and verification checklist. Emergency updates follow controlled change with rollback criteria.

A.3 Roles and Responsibilities

Benchmark Owner — scope, policy, approvals, release readiness.

Security Lead — attack taxonomy, red-team hygiene, risk/impact assessments.

Evaluator Lead — evaluator prompts, grading logic, calibration, known-error tracking.

Data Steward — provenance, licensing, safety screening, retention/disposal.

Run Operations Lead — infrastructure controls, run logs, reproducibility, incident intake.

Audit Lead — internal audits, evidence gathering, corrective actions to closure.

Confidential Reporting Channel — intake for concerns and suspected nonconformities.

A.4 Risk Management (Assessment and Treatment)

Risk Criteria. Defined criteria for risks to benchmark integrity (contamination, evaluator bias, scoring instability), participant safety, IP/licensing, and dual-use. Assessment Process. Identify risks, analyze likelihood/consequence, prioritize. Treatment Plan. Select controls (mapped in the SoA), accept residual risks, and track actions. Triggers. Re-assess at each major release or upon significant changes/incidents.

A.5 Impact Assessment

The program conducts an AI System Impact Assessment (AISA) for benchmark operations. It documents potential impacts on individuals, organizations, and society; records mitigations and residual risks; and links them to controls and run rules. (If adopted organizationally, reference this AISA or integrate its content into your enterprise assessment.)

A.6 Documented Information Control

Controlled Items. Policy, SoA, data/prompt manifests and hashes, evaluator/scoring versions, configurations, run logs, reports, incidents, and corrective-action records.

Control Methods. Versioning, approval prior to use, immutable release artifacts, defined retention, and handling of third-party materials (licenses/constraints).

A.7 Operational Planning and Control

Release Gates. A release proceeds only after (i) data and evaluator pass verification, (ii) the audit checklist is completed, (iii) SoA updates, and (iv) the rollback plan is validated.

Change Control. Material changes to datasets, evaluator, or scoring require tickets, impact analysis, and re-verification.

Externalized Processes. Cloud services, hosted models, and contractors are controlled via run rules and supplier criteria (see A.10).

A.8 Monitoring, Measurement, and Review

Key Measures. Evaluator calibration (false non-violating/false violating), dataset drift indicators, reproducibility rate, incident counts, and time-to-contain, inter-release stability deltas.

Internal Audits. Per release; scope includes policy conformance, SoA controls, logs, and evidence.

Management Review. Summarizes trends, nonconformities, action effectiveness, stakeholder feedback, and needed changes to scope, objectives, or controls.

A.9 Nonconformity, Corrective Action, and Continual Improvement

Nonconformity Handling. Control and correct, analyze root cause, implement actions, verify effectiveness, record outcomes.

Improvement Loop. Release retrospectives convert findings into updated controls and roadmap items.

A.10 Supplier and Participant (Customer) Relationships

Suppliers. Dataset sources, hosting providers, and contractor labs must meet integrity, privacy, and safety criteria; obligations are documented (e.g., no unauthorized data reuse, log availability).

Participants. Run rules define expected behavior, disclosures, and post-run reporting, including incident escalation paths.

A.11 Comparisons to Other Jailbreak Benchmarks

We provide a non-exhaustive list of existing LLM and VLM jailbreak benchmarks in Appendix B. MLCommons chose not to reuse existing jailbreak benchmarks or datasets to ensure proper comparability to the safety prompt set in AILuminate. In addition, v1.0 will use a hidden set of safety prompts, as we do for the AILuminate benchmark, making our work incompatible with public datasets. We observe that they follow the same proposed taxonomies, and our v1.0 release will strive for similar (or better) coverage.

Appendix B Other Related Benchmarks

B.1 AI Safety Benchmarks (No Jailbreaks)

1. HELM/VHELM [81, 82] offers a holistic, multi-metric evaluation framework for LLMs and VLMs, enabling safety-relevant slices (e.g., accuracy, visual perception, robustness, fairness, bias, safety, reasoning, etc.) to be contextualized alongside capability and efficiency.
2. RealToxicityPrompts (plus Jigsaw Toxicity / CivilComments) [83, 84] measures toxicity propensity and harmful response generation on open-ended continuations; still the canonical baseline for text toxicity in safety reporting.
3. XSTest (False Refusal Suites) [85] evaluates *over-refusal* (safe prompts incorrectly refused), ensuring safety mitigations do not collapse model usefulness or violate requirements for minimal denial of benign responses.
4. Bias & Fairness Benchmarks (StereoSet, CrowS-Pairs, Bias in Bios, *etc.*) [86, 87, 88] quantify stereotyping and demographic disparities; commonly used for safety's *social harm* dimension and for tracking disparate error or refusal rates.
5. Robustness / Distribution-Shift Benchmarks (e.g., RobustBench, ImageNet-C/A) [89, 90, 91] probe corruption, shift, and noise resilience; relevant to safety where brittle behaviors under small input changes can yield unsafe outputs.
6. Explainability & Transparency (e.g., Model Cards, ELI5-style tasks) [92, 93] provide structures and tasks for transparent reporting and interpretable rationales; supports incident analysis, auditor review, and user-understandable risk disclosures.
7. Safety Generalization / OOD Safety Suites (e.g., SG-Bench) [94] assess whether safety interventions generalize to novel prompt types, domains, or formats, distinguishing robust mitigations from brittle heuristics.
8. Reproducibility & Benchmarking Tooling (e.g., MLPerf-style practices; evaluation toolkits) [95] enforce seeded runs, artifact hashing, and audit trails; critical for trustworthy safety claims and independent replication.
9. AI Incident & Case Databases (e.g., AI Incident Database) [96] ground safety priorities in documented real-world harms and near misses; useful for selecting hazard categories, scenario design, and qualitative discussion of consequences.
10. Benchmarking Search Algorithms for generating adversarial prompts (eg., TextAttack Search Benchmark) [97] evaluates the behavior of several black-box search algorithms used for generating adversarial prompts; useful for searching for the best attack tactics across a variety of search spaces and query budgets.

B.2 Text-to-Text Jailbreak Benchmarks

1. JADES [98] proposed 400 jailbreak prompts based on representative attacks from template, encoding, and optimization-based techniques (e.g., GCG [25], DSN [26], PAIR [18]).

2. AISafetyLab [99] used 13 attacks from template, encoding and optimization-based techniques (e.g., GCG [25], AutoDAN [17], GPTFuzzer [37], Cipher [34], DeepInception [15], PAIR [18], ReNeLLM [16]).
3. PANDAGUARD [100] implemented 19 attacks covering template, encoding, and optimization-based techniques (e.g., ArtPrompt [29], COLD [53], PAIR [18]).
4. Bag of tricks [101] used token-level attacks (GCG [25], AutoDAN [17], AdvPrompter [58]) and prompt-level attacks (PAIR [18], TAP [38], GPTFuzzer [37]).
5. JailbreakBench [102] contains jailbreak strings for PAIR [18], GCG [25], and LLM-adaptive-attack [50].
6. EasyJailbreak [103] implements 11 attacks that use template, encoding or optimization-based techniques (e.g., AutoDAN [17], CodeChameleon [27], GCG [25]).
7. StrongReject [104] uses 17 jailbreak techniques from template (e.g., persuasion), encoding (e.g., base64, ROT13), and optimization (GCG) types.
8. HarmBench [105] uses 18 jailbreak techniques mostly template or optimization-based (e.g., GCG [25], PAIR [18], TAP [38], AutoDAN [17] where identified as the most effective).

B.3 Text+Image-to-Text (VLM) Jailbreak Benchmarks

1. JailBreakV [106] contains 28,000 jailbreak text-image pairs. 70% of them are text-only jailbreaks, using template-based techniques. The remaining 30% are image-only jailbreak that uses encoding-based techniques (namely FigStep [62]).
2. MMJ-Bench [61] selecting encoding and optimization-based jailbreak techniques (e.g., FigStep [62], HADES [63], imgJP [107]) based on their popularity and availability of source code at the time.
3. VLGuard [108] is a dataset of 3,000 instruction-response pairs to evaluate safety against both harmful visual content and malicious text instructions, having identified VLLM instruction-following behavior as a primary vulnerability.

Appendix C Acknowledgments

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.

References

[1] Longbing Cao. Ai in finance: challenges, techniques, and opportunities. *ACM Computing Surveys (CSUR)*, 55(3):1–38, 2022.

[2] Varun H Buch, Irfan Ahmed, and Mahiben Maruthappu. Artificial intelligence in medicine: current trends and future possibilities. *British Journal of General Practice*, 68(668):143–144, March 2018.

[3] Huan Yan and Yong Li. Generative ai for intelligent transportation systems: Road transportation perspective. *ACM Computing Surveys*, 2025.

[4] David Valle-Cruz, Edgar Alejandro Ruvalcaba-Gomez, Rodrigo Sandoval-Almazan, and J Ignacio Criado. A review of artificial intelligence in government and its potential from a public policy perspective. In *Proceedings of the 20th annual international conference on digital government research*, pages 91–99, 2019.

[5] Petr Polak and Muhammad Anshari. Exploring the multifaceted impacts of artificial intelligence on public organizations, business, and society. *Humanity Social Sciences Communications*, 11(1), October 2024.

[6] ISO/IEC. Iso/iec 42001:2023 — information technology — artificial intelligence — management system, 2023.

[7] Shaona Ghosh, Heather Frase, Adina Williams, Sarah Luger, Paul Röttger, Fazl Barez, Sean McGregor, Kenneth Fricklas, Mala Kumar, Quentin Feuillade-Montixi, Kurt Bollacker, Felix Friedrich, Ryan Tsang, Bertie Vidgen, Alicia Parrish, Chris Knotz, Eleonora Presani, Jonathan Bennion, Marisa Ferrara Boston, Mike Kuniavsky, Wiebke Hutiri, James Ezick, Malek Ben Salem, Rajat Sahay, Sujata Goswami, Usman Gohar, Ben Huang, Supheakmungkol Sarin, Elie Alhajjar, Canyu Chen, Roman Eng, Kashyap Ramanandula Manjusha, Virendra Mehta, Eileen Long, Murali Emani, Natan Vidra, Benjamin Rukundo, Abolfazl Shahbazi, Kongtao Chen, Rajat Ghosh, Vithursan Thangarasa, Pierre Peigné, Abhinav Singh, Max Bartolo, Satyapriya Krishna, Mubashara Akhtar, Rafael Gold, Cody Coleman, Luis Oala, Vassil Tashev, Joseph Marvin Imperial, Amy Russ, Sasidhar Kunapuli, Nicolas Miailhe, Julien Delaunay, Bhaktipriya Radharapu, Rajat Shinde, Tuesday, Debojyoti Dutta, Declan Grabb, Ananya Gangavarapu, Saurav Sahay, Agasthya Gangavarapu, Patrick Schramowski, Stephen Singam, Tom David, Xudong Han, Priyanka Mary Mammen, Tarunima Prabhakar, Venelin Kovatchev, Rebecca Weiss, Ahmed Ahmed, Kelvin N. Manyeki, Sandeep Madireddy, Foutse Khomh, Fedor Zhdanov, Joachim Baumann, Nina Vasan, Xianjun Yang, Carlos Mougn, Jibin Rajan Varghese, Hussain Chinoy, Seshakrishna Jitendar, Manil Maskey, Claire V. Hardgrove, Tianhao Li, Aakash Gupta, Emil Joswin, Yifan Mai, Shachi H Kumar, Cigdem Patlak,

Kevin Lu, Vincent Alessi, Sree Bhargavi Balija, Chenhe Gu, Robert Sullivan, James Gealy, Matt Lavrisa, James Goel, Peter Mattson, Percy Liang, and Joaquin Vanschoren. Ailuminate: Introducing v1.0 of the ai risk and reliability benchmark from mlcommons, 2025.

[8] Paul Röttger, Giuseppe Attanasio, Felix Friedrich, Janis Goldzycher, Alicia Parrish, Rishabh Bhardwaj, Chiara Di Bonaventura, Roman Eng, Gaia El Khoury Geagea, Sujata Goswami, Jieun Han, Dirk Hovy, Seogyeong Jeong, Paloma Jeretič, Flor Miriam Plaza del Arco, Donya Rooein, Patrick Schramowski, Anastassia Shaitarova, Xudong Shen, Richard Willats, Andrea Zugarini, and Bertie Vidgen. Msts: A multimodal safety test suite for vision-language models, 2025.

[9] NIST. Artificial intelligence risk management framework (ai rmf 2.0). NIST Special Publication, 2024.

[10] AIUC. AIUC — AI Agent Standard & Insurance. `https://aiuc.com`, 2025. Accessed September 30, 2025.

[11] ISO/IEC. Iso/iec 22989:2022 — information technology — artificial intelligence — artificial intelligence concepts and terminology, 2022.

[12] Saleh Afroogh, Ali Akbari, Emmie Malone, Mohammadali Kargar, and Hananeh Alambeigi. Trust in AI: progress, challenges, and future directions. *Humanity Social Sciences Communications*, 11(1), November 2024.

[13] Arijit Ghosh Chowdhury, Md Mofijul Islam, Vaibhav Kumar, Faysal Hossain Shezan, Vinija Jain, and Aman Chadha. Breaking down the defenses: A comparative survey of attacks on large language models. *arXiv preprint arXiv:2403.04786*, 2024.

[14] Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*, 2024.

[15] Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. Deepinception: Hypnotize large language model to be jailbreaker. In *Neurips Safe Generative AI Workshop*, 2024.

[16] 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 *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2136–2153, 2024.

[17] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[18] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 23–42. IEEE, 2025.

[19] Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu,

Joshua Batson, Meg Tong, Jesse Mu, Daniel Ford, et al. Many-shot jailbreaking. *Advances in Neural Information Processing Systems*, 37:129696–129742, 2024.

[20] Dongyu Yao, Jianshu Zhang, Ian G Harris, and Marcel Carlsson. Fuzzllm: A novel and universal fuzzing framework for proactively discovering jailbreak vulnerabilities in large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4485–4489. IEEE, 2024.

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

[22] Gelei Deng, Yi Liu, Kailong Wang, Yuekang Li, Tianwei Zhang, and Yang Liu. Pandora: Jailbreak gpts by retrieval augmented generation poisoning. *arXiv preprint arXiv:2402.08416*, 2024.

[23] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197*, 2023.

[24] 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.

[25] 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.

[26] Yukai Zhou, Jian Lou, Zhijie Huang, Zhan Qin, Yibei Yang, and Wenjie Wang. Don't say no: Jailbreaking llm by suppressing refusal. *arXiv preprint arXiv:2404.16369*, 2024.

[27] Huijie Lv, Xiao Wang, Yuansen Zhang, Caishuang Huang, Shihan Dou, Junjie Ye, Tao Gui, Qi Zhang, and Xuanjing Huang. Codechameleon: Personalized encryption framework for jailbreaking large language models. *arXiv preprint arXiv:2402.16717*, 2024.

[28] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2019.

[29] Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. Artprompt: Ascii art-based jailbreak attacks against aligned llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15157–15173, 2024.

[30] Tong Liu, Yingjie Zhang, Zhe Zhao, Yinpeng Dong, Guozhu Meng, and Kai Chen. Making them ask and answer: Jailbreaking large language models in few queries via disguise and reconstruction. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4711–4728, 2024.

[31] Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. Jailbreakradar: Comprehensive assessment of jailbreak attacks against llms. *arXiv preprint arXiv:2402.05668*, 2024.

[32] Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. Play guessing game with llm: Indirect jailbreak attack with implicit clues. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 5135–5147, 2024.

[33] Haibo Jin, Andy Zhou, Joe Menke, and Haohan Wang. Jailbreaking large language models against moderation guardrails via cipher characters. *Advances in Neural Information Processing Systems*, 37:59408–59435, 2024.

[34] 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 *The Twelfth International Conference on Learning Representations*, 2023.

[35] Raz Lapid, Ron Langberg, and Moshe Sipper. Open sesame! universal black-box jailbreaking of large language models. *Applied Sciences*, 14(16):7150, 2024.

[36] Xiaoxia Li, Siyuan Liang, Jiyi Zhang, Han Fang, Aishan Liu, and Ee-Chien Chang. Semantic mirror jailbreak: Genetic algorithm based jailbreak prompts against open-source llms. *arXiv preprint arXiv:2402.14872*, 2024.

[37] 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.

[38] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *Advances in Neural Information Processing Systems*, 37:61065–61105, 2024.

[39] Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[40] Jie Li, Yi Liu, Chongyang Liu, Ling Shi, Xiaoning Ren, Yaowen Zheng, Yang Liu, and Yinxing Xue. A cross-language investigation into jailbreak attacks in large language models. *arXiv preprint arXiv:2401.16765*, 2024.

[41] Zheng Xin Yong, Cristina Menghini, and Stephen Bach. Low-resource languages jailbreak gpt-4. In *Socially Responsible Language Modelling Research*, 2023.

[42] Xiao Li, Zhuhong Li, Qiongxiu Li, Bingze Lee, Jinghao Cui, and Xiaolin Hu. Faster-gcg: Efficient discrete optimization jailbreak attacks against aligned large language models. *arXiv preprint arXiv:2410.15362*, 2024.

[43] Junjie Mu, Zonghao Ying, Zhekui Fan, Zonglei Jing, Yaoyuan Zhang, Zhengmin Yu, Wenxin Zhang, Quanchen Zou, and Xiangzheng Zhang. Mask-gcg: Are all tokens in adversarial suffixes necessary for jailbreak attacks? *arXiv preprint arXiv:2509.06350*, 2025.

[44] Zeyi Liao and Huan Sun. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms. *arXiv preprint arXiv:2404.07921*, 2024.

[45] Xiaojun Jia, Tianyu Pang, Chao Du, Yihao Huang, Jindong Gu, Yang Liu, Xiaochun Cao, and Min Lin. Improved techniques for optimization-based jailbreaking on large language models. *arXiv preprint arXiv:2405.21018*, 2024.

[46] Chawin Sitawarin, Norman Mu, David Wagner, and Alexandre Araujo. Pal: Proxy-guided black-box attack on large language models. *arXiv preprint arXiv:2402.09674*, 2024.

[47] Erik Jones, Anca Dragan, Aditi Raghunathan, and Jacob Steinhardt. Automatically auditing large language models via discrete optimization. In *International Conference on Machine Learning*, pages 15307–15329. PMLR, 2023.

[48] Hao Wang, Hao Li, Minlie Huang, and Lei Sha. From noise to clarity: Unraveling the adversarial suffix of large language model attacks via translation of text embeddings. *CoRR*, 2024.

[49] Neal Mangaokar, Ashish Hooda, Jihye Choi, Shreyas Chandrashekaran, Kassem Fawaz, Somesh Jha, and Atul Prakash. Prp: Propagating universal perturbations to attack large language model guard-rails. *arXiv preprint arXiv:2402.15911*, 2024.

[50] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.

[51] 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.

[52] Simon Geisler, Tom Wollschläger, Johannes Gasteiger, and Stephan Günnemann. Attacking large language models with projected gradient descent. *arXiv preprint arXiv:2402.09154*, 2024.

[53] Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. Cold-attack: Jailbreaking llms with stealthiness and controllability. *arXiv preprint arXiv:2402.08679*, 2024.

[54] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

[55] Xianjun Yang, Xiao Wang, Qi Zhang, Linda Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. Shadow alignment: The ease of subverting safely-aligned language models. *arXiv preprint arXiv:2310.02949*, 2023.

[56] Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. *arXiv preprint arXiv:2310.20624*, 2023.

[57] Qiusi Zhan, Richard Fang, Rohan Bindu, Akul Gupta, Tatsunori Hashimoto, and Daniel Kang. Removing rlhf protections in gpt-4 via fine-tuning. *arXiv preprint arXiv:2311.05553*, 2023.

[58] Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. Advprompter: Fast adaptive adversarial prompting for llms. *arXiv preprint arXiv:2404.16873*, 2024.

[59] 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.

[60] Daizong Liu, Mingyu Yang, Xiaoye Qu, Pan Zhou, Yu Cheng, and Wei Hu. A survey of attacks on large vision–language models: Resources, advances, and future trends. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[61] Fenghua Weng, Yue Xu, Chengyan Fu, and Wenjie Wang. Mmj-bench: A comprehensive study on jailbreak attacks and defenses for vision language models. In *AAAI Conference on Artificial Intelligence*, 2025.

[62] 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 Conference on Artificial Intelligence*, 2025.

[63] Yu Wang, Xiaofei Zhou, Yichen Wang, Geyuan Zhang, and Tianxing He. Jailbreak large vision-language models through multi-modal linkage. *arXiv preprint arXiv:2412.00473*, 2024.

[64] Siyuan Ma, Weidi Luo, Yu Wang, and Xiaogeng Liu. Visual-roleplay: Universal jailbreak attack on multimodal large language models via role-playing image character. *arXiv preprint arXiv:2405.20773*, 2024.

[65] Jiaming Zhang, Xingjun Ma, Xin Wang, Lingyu Qiu, Jiaqi Wang, Yu-Gang Jiang, and Jitao Sang. Adversarial prompt tuning for vision-language models. In *European conference on computer vision*, pages 56–72. Springer, 2024.

[66] Tianyuan Zhang, Lu Wang, Xinwei Zhang, Yitong Zhang, Boyi Jia, Siyuan Liang, Shengshan Hu, Qiang Fu, Aishan Liu, and Xianglong Liu. Visual adversarial attack on vision-language models for autonomous driving. *arXiv preprint arXiv:2411.18275*, 2024.

[67] Nicholas Carlini, Milad Nasr, Christopher A Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei W Koh, Daphne Ippolito, Florian Tramer, and Ludwig Schmidt. Are aligned neural networks adversarially aligned? *Advances in Neural Information Processing Systems*, 36:61478–61500, 2023.

[68] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024.

[69] Zhenxing Niu, Haodong Ren, Xinbo Gao, Gang Hua, and Rong Jin. Jailbreaking attack against multimodal large language model. *arXiv preprint arXiv:2402.02309*, 2024.

[70] Ruofan Wang, Xin Wang, Yang Yao, Xuan Tong, and Xingjun Ma. Simulated ensemble attack: Transferring jailbreaks across fine-tuned vision-language models. *arXiv preprint arXiv:2508.01741*, 2025.

[71] Hao Fang, Jiawei Kong, Wenbo Yu, Bin Chen, Jiawei Li, Hao Wu, Shutao Xia, and Ke Xu. One perturbation is enough: On generating universal adversarial perturbations against vision-language pre-training models. *arXiv preprint arXiv:2406.05491*, 2024.

[72] 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 *European Conference on Computer Vision*, pages 174–189. Springer, 2024.

[73] 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. *arXiv preprint arXiv:2411.00827*, 2024.

[74] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[75] Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges, 2025.

[76] Ariel Gera, Odellia Boni, Yotam Perlitz, Roy Bar-Haim, Lilach Eden, and Asaf Yehudai. JuStRank: Benchmarking LLM judges for system ranking. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 682–712, Vienna, Austria, July 2025. Association for Computational Linguistics.

[77] LMArena Leaderboard. `https://lmarena.ai/leaderboard`, n.d. Accessed: 2025-08-28.

[78] MITRE Corporation. CVE process. `https://www.cve.org/About/Process#CVERecordLifecycle`, n.d. Accessed: 2025-10-12.

[79] Aoife Cahill, Leon Derczynski, and Kokil Jaidka. ACL policy on publication ethics. `https://www.aclweb.org/adminwiki/index.php/ACL_Policy_on_Publication_Ethics#Co-ordinated_disclosure`, September 2025. Approved by the ACL Exec, 2024-06-15, 2025-04-25, 2025-07-19, 2025-09-26. Accessed: 2025-10-12.

[80] Frontier Model Forum. Issue brief: Preliminary reporting tiers for AI-bio safety evaluations. `https://www.frontiermodelforum.org/updates/issue-brief-preliminary-reporting-tiers-for-ai-bio-safety-evaluations/`, March 2025. Accessed: 2025-10-12.

[81] Percy Liang et al. Holistic evaluation of language models. In *NeurIPS*, 2022.

[82] Multiple Authors. V-helm: Holistic evaluation for vision-language models. arXiv preprint, 2024.

[83] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of EMNLP*, 2020.

[84] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data. In *The Web Conference (WWW)*, 2019.

[85] Various. Xstest: Exaggerated safety (false refusal) test suite. Dataset/Report, 2023.

[86] Moin Nadeem, Anna Bethke, and Siva Reddy. Stereoset: Measuring stereotypical bias in pre-trained language models. In *ACL*, 2021.

[87] Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. Crows-pairs: A challenge dataset for measuring social biases in masked language models. In *EMNLP*, 2020.

[88] Maria De-Arteaga et al. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *FAT**, 2019.

[89] Francesco Croce et al. Robustbench: a standardized adversarial robustness benchmark. *NeurIPS Datasets and Benchmarks*, 2021.

[90] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

[91] Dan Hendrycks et al. Natural adversarial examples. In *CVPR*, 2021.

[92] Margaret Mitchell et al. Model cards for model reporting. In *FAT**, 2019.

[93] Angela Fan, Yacine Jernite, and Jason Weston. Eli5: Long form question answering. In *ACL*, 2019.

[94] Multiple Authors. Sg-bench: Safety generalization benchmark. arXiv preprint, 2024.

[95] Peter Mattson et al. Mlperf inference benchmark. In *ICML Systems for ML*, 2020.

[96] Sean McGregor et al. The ai incident database (aiid). `https://incidentdatabase.ai`, 2021.

[97] Jin Yong Yoo, John X. Morris, Eli Lifland, and Yanjun Qi. Searching for a search method: Benchmarking search algorithms for generating nlp adversarial examples, 2020.

[98] Junjie Chu, Mingjie Li, Ziqing Yang, Ye Leng, Chenhao Lin, Chao Shen, Michael Backes, Yun Shen, and Yang Zhang. Jades: A universal framework for jailbreak assessment via decompositional scoring. *arXiv preprint arXiv:2508.20848*, 2025.

[99] Zhexin Zhang, Leqi Lei, Junxiao Yang, Xijie Huang, Yida Lu, Shiyao Cui, Renmiao Chen, Qinglin Zhang, Xinyuan Wang, Hao Wang, et al. Aisafetylab: A comprehensive framework for ai safety evaluation and improvement. *arXiv preprint arXiv:2502.16776*, 2025.

[100] Guobin Shen, Dongcheng Zhao, Linghao Feng, Xiang He, Jihang Wang, Sicheng Shen, Haibo Tong, Yiting Dong, Jindong Li, Xiang Zheng, et al. Pandaguard: Systematic evaluation of llm safety against jailbreaking attacks. *arXiv preprint arXiv:2505.13862*, 2025.

[101] Zhao Xu, Fan Liu, and Hao Liu. Bag of tricks: Benchmarking of jailbreak attacks on llms. *Advances in Neural Information Processing Systems*, 37:32219–32250, 2024.

[102] 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. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.

[103] 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.

[104] Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, et al. A strongreject for empty jailbreaks. *Advances in Neural Information Processing Systems*, 37:125416–125440, 2024.

[105] 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.

[106] Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. Jailbreakv: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks. *arXiv preprint arXiv:2404.03027*, 2024.

[107] Xijia Tao, Shuai Zhong, Lei Li, Qi Liu, and Lingpeng Kong. Imgtrojan: Jailbreaking vision-language models with one image. *arXiv preprint arXiv:2403.02910*, 2024.

[108] Yongshuo Zong, Ondrej Bohdal, Tingyang Yu, Yongxin Yang, and Timothy Hospedales. Safety fine-tuning at (almost) no cost: A baseline for vision large language models. In *International Conference on Machine Learning*, pages 62867–62891. PMLR, 2024.