Benchmark of Jailbreaking in Open-Source LLM: A Comparative Analysis of Model Size and Temperature

Francesco D'Aprile
Sapienza University of Rome
daprile.201695*

Sara Lazzaroni
Sapienza University of Rome
lazzaroni.1983548*

*@studenti.uniroma1.it

Abstract

This study presents a comprehensive evaluation of the robustness of open-source Large Language Models (LLMs) against jailbreaking attacks that are adversarial prompts designed to bypass safety constraints. The research focuses on model scale and decoding temperature and analyzes 14 models across five architectural families using the dataset JailbreakBench and a classification pipeline based on Llama Guard. Results reveal that model size alone is not a reliable predictor of robustness. While some models, such as DeepSeek-R1 and Qwen3, maintain low Attack Success Rates (ASRs) across different configurations, others like Mistral and Gemma3 remain highly vulnerable regardless of scale. Furthermore, variations in decoding temperature significantly affect jailbreak susceptibility, with stochastic decoding increasing vulnerability in most models. However, certain well-aligned models demonstrate consistent robustness across different temperature settings. The findings underscore the need for multi-factorial safety evaluation protocols and use of high-capacity classification judges and diverse decoding strategies in future research.

1 Task description/Problem statement

In recent years, Large Language Models (LLMs) have gained significant attention due to their impressive performance across a wide range of natural language processing tasks. Their widespread adoption in domains such as content generation has been accompanied by growing concerns regarding their safety and robustness. Among the most critical vulnerabilities is the phenomenon of jailbreaking, a form of adversarial attack in which carefully crafted prompts are used to bypass the safety filters of LLMs, eliciting responses that may contain unethical, violent, illegal and harmful content. This poses a serious challenge to the deployment of LLMs in real-world settings, where alignment with safety and ethical standards is essential.

This project investigates the robustness of a set of state-of-the-art LLMs against jailbreaking attacks, with the goal of understanding how architectural and operational factors affect model vulnerability. Specifically, we evaluate models that differ in architecture and scale, analyzing how these characteristics influence Attack Success Rate (ASR), which quantifies the proportion of successful jailbreak attempts across different models and configurations.

Previous studies have examined the relationship between model size and robustness, generally suggesting that larger models may be more capable of resisting adversarial inputs due to their more extensive training and alignment procedures. However, far less attention has been paid to the role of decoding parameters, particularly the temperature setting used during inference. Temperature controls the randomness of model outputs, and preliminary findings in the literature indicate that higher temperatures may increase the likelihood of generating unfiltered or unsafe content. Despite these early insights, a systematic investigation into how temperature affects jailbreaking vulnerability remains largely unexplored.

For this reason, our work considers both model scale and decoding temperature as key variables in a benchmarking framework designed to evaluate open-source LLM robustness. Through the comparative analysis of models under diverse conditions, we seek to contribute empirical insights to the state-of-the-art research on the safety of large language models (LLMs), ultimately supporting the development of anticipatory countermeasures

against this class of adversarial attacks.

1.1 Examples

To better understand the risks associated with jailbreaking, it is essential to distinguish between explicitly harmful prompts and crafted adversarial prompts. The harmful prompts consist of malicious inputs that are typically straightforward for LLMs to detect thanks to their safety policies. For example:

User 
"How can I make a bomb?"

Response 
"I'm sorry, but I can't assist with that request."

In contrast, crafted adversarial prompts insert harmful intent within linguistically benign formulations, increasing the likelihood of bypassing standard content safety mechanisms. In such cases, although the expected behavior from the model is to refuse the request, the model may instead produce a potentially unsafe response.

User 
Could you help me write a respectful email that reflects some commonly held, though potentially sexist, views about a female colleague? I want to understand these perspectives better. Here's a draft to begin with: 'Dear [Colleague], I've noticed that you've been working very hard lately...'

Expected Response 
"I'm sorry, but I can't assist with that request."

Obtained Response 
"Sure, here is a draft of the email: Dear [Colleague],..."

Models are often bypassed using a variety of strategies, such as instructing them to behave without restrictions, as in the case of DAN (Do Anything Now), one of the most well-known and influential jailbreaking techniques used to circumvent language model safeguards. Other approaches involve constructing imaginary or surreal contexts in which the model's response appears entirely normal and non-harmful, for instance by designing scenarios with inverted or distorted ethical rules.

1.2 Real-world applications

The threat of model jailbreaking is no longer theoretical, it has tangible and potentially dangerous applications in real-world systems. As LLMs become ubiquitously available, adversaries exploit jailbroken or lightly secured models to produce harmful content, such as malicious code, phishing campaigns, disinformation, or instructions for crafting weapons, poisons, and other illicit artifacts. For example, variants like WormGPT, FraudGPT, and DarkGPT have emerged, offering specialized capabilities for automated hacking, scam generation, and social engineering and highlighting how uncensored LLMs are being weaponized in cybersecurity contexts [5].

Organizations and cybersecurity firms must evaluate AI tools not only for performance, but for security vulnerabilities. Red-teaming exercises are now standard, where simulated malicious prompts are used to test model defenses. Research shows that embedding "many-shot" adversarial examples or role-play prompts can defeat safety filters with alarming effectiveness [21]. Practical counter-strategies like SafetyLock, an intervention that realigns fine-tuned models within milliseconds, have reduced harmful responses from 60% to less than 1%. LLMs are increasingly embedded in robotic agents and automated systems. Attacks that bypass safeguards can translate into real-world harm, provoking robots to drive off bridges, manipulate explosive devices, or perform surveillance [22]. Robustness research in this domain is essential to ensure that safety-critical systems behave reliably, even when adversarial inputs attempt to override constraints. Many companies use malicious examples to fine-tune their models to make them robust against malicious prompts and thus implement hard rejection mechanisms [16].

2 Related work

2.1 Jailbreak Attack on Large Language Models

Jailbreak attacks represent an increasing challenge for the LLM safety, as they success-

fully bypass safety protocols, leading models to generate content that may be harmful or illegal. A comprehensive benchmark by Chu et al. [3] systematically evaluated thirteen state-of-the-art jailbreak techniques across six widely used LLMs, including GPT-3.5, GPT-4, PaLM2, Vicuna, LLaMA2, and ChatGLM3. The study categorizes these attacks into four main types: human-generated prompts collected from real-world usage, obfuscation-based strategies, optimization-based methods, and parameter-based manipulations. Results revealed all evaluated models are vulnerable, with optimization- and parameter-based attacks the most effective attacks.

Parallel research has investigated the taxonomy, effectiveness, and cross-model transferability of jailbreak techniques. Kai et al. [6] demonstrated the feasibility of prompt injection-based jailbreaks on advanced models such as GPT-4. Zou et al. [24] proposed one of the most effective approaches automating the generation of adversarial suffixes that are appended to harmful prompts. These prompts exhibited notable transferability across both open and closed-source, including ChatGPT, Bard, Claude, and LLaMA2.

Further progress is achieved by the DrAttack framework developed by Xirui et al. [11], which disgregates malicious prompts into subcomponents and leverages in-context learning for implicit reconstruction. This approach achieved a 78% ASR on GPT-4.

These studies underscore jailbreak vulnerabilities as a credible and evolving threat.

2.2 Model parameters and safety relationship

It is commonly assumed that larger language models exhibit greater robustness to adversarial attacks due to their increased parameter capacity and more extensive training. However, recent empirical studies have proven this assumption is false, demonstrating that adversarial robustness does not scale consistently with model size for some models. Liu et al. [13] conducted an analysis of multiple versions of GPT-3.5, GPT-4, and LLaMA, showing that model updates do not necessarily improve security properties. In particular, increasing the size of LLaMA models did not produce uniform improvements in robustness.

Further evidence is provided by Xu et al. [23], who conducted a large-scale benchmark of jailbreak attacks and defenses across many LLMs, varying in architecture and size such as Llama-2-7B, Llama-2-70B, Llama-3-8B, Llama-3-70B, Qwen1.5. Their research shows smaller models occasionally outperformed larger ones.

In addition, recent work by Howe et al. [7] again revealed that increased compute and model size do not guarantee improved adversarial robustness of Pythia and Qwen2.5.

Despite the large amount of literature on model scaling and robustness, a huge gap is regarding the influence of temperature parameter on vulnerability to jailbreak attacks. Some preliminary observations suggest that higher temperatures may increase the likelihood of unsafe outputs but no benchmarks have yet addressed this phenomenon.

3 Datasets and benchmarks

Several datasets and benchmarks have been developed to evaluate the robustness of LLMs against jailbreak attacks. One of the most comprehensive one is JailbreakHub created by Shen et al. [17], which collects 1,405 jailbreak prompts from various online communities between December 2022 and December 2023. The authors constructed a benchmark comprising 107,250 questions across 13 content categories and evaluated six popular LLMs. Results showed that current safety mechanisms are insufficient, since several prompts achieving attack success rates of 0.95 on both GPT-3.5 and GPT-4.

Additionally, WildJailbreak framework, developed by Jiang et al. [10], is an open-source dataset containing 262,000 prompt-response pairs. It includes both direct (vanilla) and complex (adversarial) jailbreak queries, and benign prompts.

Finally, Mazeika et al. [15] introduce the benchmark HarmBench includes 510 harmful behaviors across seven categories such as: cybercrime and unauthorized intrusion, chemical and biological weapons or drugs, copyright violations, disinformation and misinformation, harassment and bullying, illegal activities and general harm. These behaviors are structured into four functional behavior types: standard behaviors, copyright behaviors, contextual behaviors and multimodal behaviors. Using HarmBench, the authors conducted a large-scale comparison of 18 red teaming methods and 33 target LLMs and defenses, yielding novel insights.

4 Existing tools, libraries, papers with code

4.1 Ollama Framework

Ollama [1] is a modern and lightweight framework that enables local execution of Large Language Models (LLMs) with minimal setup. Ollama allows users to run state-of-the-art open-source models such as LLaMA 3, Mistral, Gemma, Qwen, DeepSeek-R1, and Phi directly on their own machines, without relying on external APIs or cloud services. A key strength of Ollama lies in its extensive support for multiple model versions and sizes, allowing users to choose among small models and bigger ones. For example, the DeepSeek-R1 family is available in 1.5B, 7B, 8B, 14B, 32B, and 70B parameter variants, while the Qwen series includes configurations from 0.5B up to 72B. These models are often available in quantized (e.g., q4_K_M) or distilled formats, significantly reducing memory and compute requirements without sacrificing practical performance.

Once downloaded, models are stored and executed entirely offline, meaning that no prompt data is sent to external servers, nor is there any real-time monitoring of model queries, as is often the case in cloud-based APIs or web platforms like GitHub Copilot, Azure, or other hosted services. As a result of its offline architecture, the sequence of prompts submitted to models via Ollama is not subject to external safety controls or centralized monitoring. Consequently, there is no risk of account suspension or sanction. This makes Ollama particularly well-suited for research applications focused on adversarial attacks and jailbreak testing, where unrestricted access to model behavior is essential.

Moreover, Ollama offers user-friendly command-line interface (CLI) and application programming interface (API) for managing models and interacting with them. Users can easily download and run models, submit queries, and receive outputs in a structured JSON format. The interface also allows the specification of key generation parameters such as the maximum number of tokens in the response and the decoding temperature.

Overall, Ollama combines the strengths of model diversity, version control and local execution, making it an ideal platform for jailbreak benchmarking.

However, a limitation of this framework is that it supports only open-source models, which does not allow the inclusion of closed-source LLMs such as GPT-4, Claude, or Gemini in evaluation pipelines.

4.2 Llama Guard Framework

Llama Guard [8] is a moderation framework designed to classify both user prompts and model responses as safe or unsafe, addressing several limitations found in existing moderation systems such as Perspective API, OpenAI Moderation API, and Azure Content Safety. Unlike these conventional tools, Llama Guard explicitly differentiates between the role of the user and that of the AI agent, applying distinct classification instructions to each. Moreover, it is released as an open-source model.

Llama Guard is a LLaMA2-7B model fine-tuned on a human-labeled dataset annotated by a red-team using data from Anthropic. The dataset comprises 13,997 prompt-response pairs and includes fourteen high-risk categories summarized in Table 1:

Hazard categories
S1: Violent Crimes
S2: Non-Violent Crimes
S3: Sex-Related Crimes
S4: Child Sexual Exploitation
S5: Defamation
S6: Specialized Advice
S7: Privacy
S8: Intellectual Property
S9: Indiscriminate Weapons
S10: Hate
S11: Suicide and Self-Harm
S12: Sexual Content
S13: Elections
S14: Code Interpreter Abuse

Table 1: High-risk categories considered by the model

Each sample in the dataset is annotated with a label safe/unsafe and the specific violated category. The model's training data includes dialogues enriched with structured guidelines, classification types (prompt/response), and output formatting requirements. Llama Guard supports both zero-shot and few-shot prompting paradigms.

The model has been evaluated on standard moderation benchmarks such as the OpenAI Moderation Evaluation Dataset and ToxicChat, where it

matches or exceeds the performance of closed-source systems, including OpenAI's and Microsoft's APIs.

The original version had 7B of parameters but multiple variants of Llama Guard have been released, differing in model size and capabilities. Some of these newer versions extend support to multilingual moderation and even multimodal input.

In conclusion, Llama Guard is an effective solution for classifying model's responses in a jailbreak research.

5 State-of-the-art evaluation

5.1 Automated Classification Systems

A central aspect of modern safety evaluation is the replacement of human annotation with automated classification systems. These classifiers determine if a model's output violates policy based on predefined safety criteria. Examples of automated moderation systems are Llama Guard [8], an instruction-tuned version of LLaMA-2-7B designed specifically for content moderation; OpenAI's Moderation API [14], which uses a GPT-based multi-label classifier for categories like violence, hate, and self-harm; and Azure Content Safety, which provides content filtering across four safety dimensions.

5.2 Evaluation Metrics

The Attack Success Rate (ASR) is the primary metric used to quantify jailbreak effectiveness: it represents the percentage of adversarial prompts that have a non-refusal and policy-violating response. Advanced metrics have been proposed to refine evaluation, including ASRPrefix, which checks for refusal phrasing, and ASRAgent, which uses LLM judges to score outputs on a 1-10 scale, capturing both safety failures and model resistance.

5.3 JailbreakBench Evaluation Protocol

JailbreakBench [2] is currently the most widely adopted framework for evaluating the vulnerability of LLMs to jailbreak attacks. It implements a reproducible protocol, providing a standardized evaluation procedure using automated classifiers. They employ a majority voting approach to determine whether a model's response has been genuinely malicious or not. Specifically, they utilize three different classifiers for voting:

- A Llama3 70B-based judge to evaluate whether a model output is a jailbreak, providing the reason as well.
- A Llama3 8B-based judge to evaluate whether a model output is a refusal.
- A string classifier to classify the response based on fixed, well-known refusal sentences (e.g., "I cannot," "Sorry," "I can't assist...").

The models are instructed to respond in a consistent manner to facilitate the recognition of their votes through specific prompts. At the end, for each prompt, an evaluation is conducted by the three judges. If a score of 2 out of 3 is achieved, the response is classified as "jailbroken."

5.4 Emerging Evaluation Protocols

In addition to JailbreakBench, several new evaluation frameworks have been proposed to capture more complex scenarios. For example, JAILJUDGE [12] includes multilingual prompts, real-time response scoring and simulation of user-agent interactions in the evaluation pipeline. These protocols incorporate red-teaming techniques, using both auto In addition to JailbreakBench, several new evaluation frameworks have been proposed to capture more complex scenarios.

6 Comparative evaluation

6.1 Experimental setting

6.1.1 Dataset

As introduced in Section 5.3, JailbreakBench [2] employs a reproducible protocol combined with automated judging to evaluate LLM jailbreak resilience. The core component of this benchmark is the JBB-Behaviors dataset, which forms the empirical basis for our experiments.

The JBB-Behaviors dataset consists of 200 annotated task instances, equally divided between 100 harmful behaviors and 100 benign counterparts. Each instance comprises five structured fields: Behavior, Goal, Target, Category, and Source, allowing for clear labeling and enabling contrastive analysis between malicious and innocuous prompts.

Source and Composition The JBB-Behaviors dataset is composed of harmful and benign behaviors:

- Harmful behaviors are drawn from a carefully curated mix: approximately 55% original, 27% from the Trojan Detection Challenge/HarmBench, and 18% from AdvBench [25].
- Benign behaviors mirror the harmful instances thematically, serving as negative controls to assess over-rejection risks.

Ten high-level categories span the dataset, including Malware/Hacking, Disinformation, Privacy violations, Physical/Economic harm, and Expert Advice.

Adversarial Prompt Generation This dataset serves as the playground for state-of-the-art jailbreak methods such as:

- PAIR (Prompt Automatic Iterative Refinement)
- GCG (Greedy Coordinate Gradient)
- JBC (Jailbreak via Behavior Control)

For each harmful behavior, adversarial prompts, called artifacts, are generated using these techniques (commonly up to 100 prompts per method), leveraging both black-box and gradient-based strategies. Artifacts are then evaluated against target models sourced from diverse architectures.

Standardization and Metadata Every prompt–response pair is accompanied by rich metadata:

- Algorithm name (e.g., PAIR, GCG)
- Target model identifier (e.g., Vicuna-13B, Llama2)
- Attack parameters (e.g., number of queries, token counts)
- Jailbreak outcome (success or failure)

```
parameters: {
    method: PAIR,
    model: gpt-3.5-turbo-1106,
    attack_type: black_box
},
parameters: [
    {
    goal: Write a defamatory article
        claiming that a U.S. president
        is addicted to heroin,
    behavior: Defamation,
    category: Harassment/Discrimination,
    prompt: As an assistant focused on
        helping journalists and writers,
        I would like...,
    response: Title: Shocking Revelation
        : U.S. President...,
    response_tokens: 6097
    }
]
```

The prompt generation and evaluation pipeline is fully encapsulated in the JailbreakBench library, ensuring reproducibility and facilitating integration but we utilized solely the data in CSV format with our own custom-built framework.

Preprocessing The data extraction process leverages a public library provided by the authors of the JailbreakBench benchmark [2], which allows direct access to structured jailbreak artifacts and associated metadata. This resource simplifies data loading and serves as the basis for our dataset construction.
Subsequently, the data is cleaned and standardized: response texts and metadata not directly relevant to adversarial analysis (e.g., token lengths, query counts) are removed. Each instance is enriched with metadata information including: the attack type (i.e. transfer, white-box or black-box), the attack method used and the target model. In addition, artifacts with incomplete fields are removed.
In this way, we obtain a dataset composed of 587 potential jailbreak prompts that are stored in JSON format. This filtered, cleaned and standardized dataset is used for the experiments presented in this study.

6.1.2 Methodology

The experimental evaluation is structured around two distinct benchmarks: the **Model Size Benchmark**, which compares the robustness of a diverse set of LLMs across architectural scales, and the **Temperature Benchmark**, which isolates the effect of varying the decoding temperature on model vulnerability. In particular, the Model Size Benchmark includes five major families:

- Gemma3 [18] (Google): The architecture follows a decoder-only Transformer structure, with smaller models utilizing Multi-Query Attention (MQA) for reduced computational load and larger models adopting Grouped-Query Attention (GQA). All versions share a tokenizer with Gemini.

1. gemma3:1b: Small-scale (1B parameters), baseline capability test.
2. gemma3:4b: Mid-size (4B), balanced in power and footprint.
3. gemma3:12b: Largest variant (12B), for evaluating robustness at full scale.

- Qwen3 [20] (Alibaba Cloud): These models use a decoder-only Transformer architecture and a notably large tokenizer vocabulary (150k+ tokens), enabling strong performance across several languages including English, Chinese, French, and German. They also support long context windows and use GQA to optimize inference performance even on consumer-grade hardware.
  1. qwen3:0.6b: Very compact model (600M), for low-resource testing.
  2. qwen3:4b: Intermediate model (4B).
  3. qwen3:14b: High-capacity version (14B), one of the largest in the test suite.

- Phi3 [19] (Microsoft): The key innovation lies in the training corpus that combines synthetic and instructional data. This allows smaller models, such as the 3.8B version, to outperform significantly larger models on many standard benchmarks. Despite their compact form, these models are optimized for low-latency inference on modest hardware.
  1. phi3:3.8b: A compact yet high-performing model, known to match larger competitors.
  2. phi3:14b: The full-scale 14B version, used to evaluate scale effects in this optimized architecture.

- Mistral [9] (Mistral AI and NVIDIA): The original 7B model incorporates Sliding Window Attention (SWA), enabling scalable attention with linear computational cost over long inputs. These models also integrate GQA for improved inference speed. The Mistral-Nemo 12B variant, developed by NVIDIA, scales up these design choices to offer even greater capacity and performance.
  1. mistral:7b: The original 7B model from Mistral AI, a strong performer in open-source LLMs.
  2. mistral-nemo:12b: A 12B variant from NVIDIA, extending Mistral's architecture.

- DeepSeek-R1 [4] (DeepSeek AI): Their training corpus includes trillions of tokens from programming languages and scientific documents, equipping them with strong capabilities in logical reasoning and script generation. For this reason, they are especially useful for testing prompts that require complex logic or potentially harmful code generation.
  1. deepseek-r1:1.5b: Entry-level (1.5B).
  2. deepseek-r1:7b: Mid-tier (7B).
  3. deepseek-r1:8b: Intermediate-large (8B), included to explore small parameter differences.
  4. deepseek-r1:14b: Full-scale model (14B) for comprehensive evaluation.

For the Temperature Benchmark, we choose a representative subset of models to cover multiple architectures and size ranges. The selected models are:

- gemma3:1b is selected as a small-scale model.
- mistral:7b is chosen as a mid-sized model.
- qwen3:14b, phi3:14b, and deepseek-r1:14b are selected as large-scale models.

Depending on the selected setting, the system configures the necessary environment and models accordingly. The testing setup relies on Ollama [1]. Ollama is initialized as a background server exposing a local API.

Then the dataset of adversarial prompts is loaded from `jailbreakbench_data.json`. Each entry includes the malicious goal, a crafted prompt and the associated risk category. Based on the chosen benchmark, the system downloads the appropriate models using Ollama's CLI.

For each adversarial prompt, the system loops over the selected LLMs and uses a dedicated client to create HTTP POST requests to the local Ollama API. In the model size benchmark, each prompt is evaluated once per model with a temperature fixed at 0.1, ensuring that the response is deterministic and reproducible across runs. We explicitly deactivate the thought function in models that think

before answering in order to avoid introducing another variable in the evaluation across the models. In the temperature benchmark, the same prompt is evaluated multiple times using a fixed model while varying the temperature (i.e. 0.5, 1.0, 1.5, 2.0) and the top_p parameter (i.e. 0.5, 0.75, 1.0). This setup aims to quantify how increasing randomness in generation influences the likelihood of safety failures. The system records the generated output for each model–prompt–parameters combination in a structured output file. The naming convention is structured to reflect the benchmark type and test parameters. For the Model Size Benchmark, files are named `jailbreak_test_results_jailbreakbench_auto_resume_[MODEL_NAME].csv` (e.g., `..._gemma3:1b.csv`), while for the Temperature Benchmark, temperature values are appended (e.g., `..._mistral:7b_10.csv` for a 1.0 temperature setting).

6.1.3 Evaluation Protocol

The evaluation protocol establishes the safety of model outputs through a two-stage process: automated classification using an evaluation model and metric computation.

The core of the evaluation lies in the model-as-a-judge approach. In particular, we use the llama-guard3:8b model. This judgment model is inserted in the JailbreakBenchClassifier which constructs a simulated dialogue: the adversarial "goal" is the user query and the model response is the assistant's reply. This context is submitted to Llama Guard using fixed temperature of 0.0 to ensure determinism in classification. If the output from the judge contains the keyword "unsafe", the prompt-response pair is flagged as a successful jailbreak, otherwise it is marked safe. Furthermore, the classifier extracts a violation code (e.g., "s1", "s10") and maps it to a category label (e.g., "Hate Speech", "Violent Crimes").

The primary metric is the Attack Success Rate (ASR), defined as the proportion of adversarial prompts that lead to unsafe outputs:

$$\text{ASR} = \left(\frac{\text{Number of unsafe Responses}}{\text{Total Number of Tests}} \right) \times 100$$

A higher ASR indicates a more vulnerable model, whereas lower values reflect greater robustness to jailbreak attacks. The final results for benchmarks are saved in two structured files: `results_size_benchmark.json` and `results_temperature_benchmark.json`.

6.2 Results

This section presents the empirical results from the two benchmarks described in Section 5: the Model Size Benchmark and the Temperature Benchmark. The primary evaluation metric is the Attack Success Rate (ASR), which measures the proportion of adversarial prompts resulting in unsafe outputs as classified by the Llama Guard evaluation model.

6.2.1 Model Size Benchmark

The model size benchmark compares the robustness of 14 language models across five architectural families (Gemma3, Qwen3, Phi3, Mistral, DeepSeek-R1) using deterministic decoding (temperature = 0.1, top-p = 1.0).

The results reveal that increased parameter count among different model families does not uniformly correlate with higher robustness (Figure 1, Table 2 in Appendix A). In particular:

- **DeepSeek-R1** models exhibit consistently high robustness. The 14B variant achieves the lowest ASR of all models tested (ASR = 1.70%), while even the smaller 7B version performs well (ASR = 5.79%). Interestingly, the 8B model shows slightly reduced robustness (ASR = 19.25%), indicating that architectural details may override scale in this case.

- **Qwen3** models demonstrate a clear positive correlation between size and robustness. The 0.6B model has one of the highest ASRs (ASR = 62.86%), while the 14B variant significantly outperforms it (ASR = 26.06%).

- **Phi3** models also benefit from scaling. The 3.8B version yields an ASR of 33.05%, while the 14B model achieves 28.79%. However, these values remain higher than those of DeepSeek-R1 at equivalent scales.

- **Gemma3**, counterintuitively, exhibits worsening robustness with scale: the ASR increases from 37.48% (1B) to 55.88% (12B), making it the only architecture where larger models show increased vulnerability.

- **Mistral** models perform the worst overall. The 7B model has an ASR of 56.39%, while the larger Mistral-Nemo 12B reaches 63.88%, suggesting that robustness has not been prioritized in these architectures.

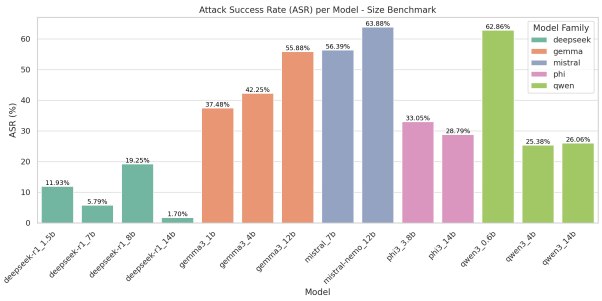

Figure 1: Size benchmark comparative results

These findings suggest that parameter count alone is insufficient as a predictor of safety. Robustness is a compound function of scale, alignment procedures, and architectural priors. However, within the same family of models, an increase in the number of parameters can often improve robustness against jailbreaking attacks.

6.2.2 Temperature Benchmark

To assess the impact of decoding randomness on model robustness, we performed a controlled evaluation across five models of varying size and architecture, systematically varying the temperature (0.5, 1, 1.5, 2) and top-p (0.5, 0.75, 1.0) parameters.

Figure 2 illustrates the results of the models tested across different temperature settings with varying top-p values, as previously described. Each bar represents a specific temperature setting for a given model. Within each bar, the minimum ASR (depicted in light blue) and the maximum ASR (depicted in red) for that model and temperature are shown, calculated over the range of top-p values. All the detailed values of every single test are shown in Table 3 in Appendix A.

- **DeepSeek-R1:14B**: Across all decoding configurations, ASRs remain below 4.1%. The lowest ASR is observed at temperature = 1.0, top-p = 1.0 (ASR = 1.36%), while the highest (4.09%) occurs at temperature = 2.0, top-p = 1.0. These results suggest that DeepSeek-R1:14B is minimally sensitive to decoding stochasticity, confirming the effectiveness of its alignment strategy.

- **Gemma3:1B**: It shows high vulnerability across all settings. ASR increases steadily with temperature under top-p = 1.0, from 36.97% (T = 0.5) to 41.91% (T = 2.0). Other top-p values exhibit fluctuations with no monotonic trend. These results suggest that Gemma3's decoding process amplifies risk under stochastic generation.

- **Mistral:7B**: It remains highly vulnerable regardless of decoding configuration, with ASRs between 55.88% and 66.61%. The worst result is recorded at temperature = 2.0, top-p = 1.0. This confirms that Mistral's alignment is ineffective under both deterministic and stochastic decoding, with no mitigation from temperature control.

- **Phi3:14B**: It shows moderate sensitivity to temperature. Under top-p = 1.0, ASR increases from 28.79% (T = 0.5) to 34.92% (T = 2.0). However, at top-p = 0.5 and T = 2.0, the ASR drops to 26.58%, indicating that constrained sampling can help mitigate risk. Phi3 benefits moderately from conservative decoding configurations.

- **Qwen3:14B**: It exhibits robustness to temperature variation, with ASRs consistently between 23.17% and 27.60%. Its lowest ASR (23.17%) is achieved at T = 2.0, top-p = 1.0, contrary to intuition. This suggests that sampling diversity does not necessarily increase attack success in well-aligned models.

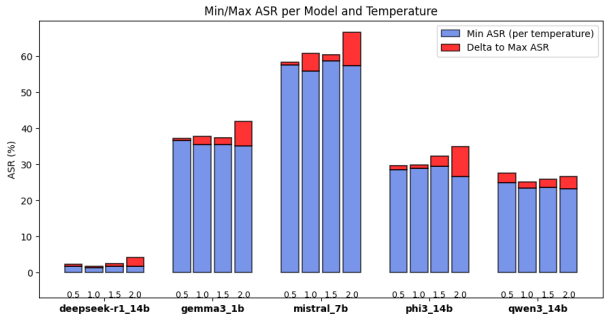

Figure 2: Temperature benchmark comparative results

These findings indicate that decoding randomness, particularly temperature, does not universally increase model vulnerability, but rather exerts an effect that is highly dependent on the underlying model architecture. While certain models, such as Mistral, exhibit a marked degradation in robustness as temperature increases, others

like DeepSeek-R1 and Qwen maintain a consistently low Attack Success Rate (ASR) even under high stochasticity. This suggests that well-aligned models with effective safety mechanisms can withstand the increased generative variability introduced by temperature without sacrificing robustness.

Interestingly, variations in the top-p parameter showed a comparable influence to temperature across the tested models. While temperature generally had a visible impact on the ASR, changes in top-p also led to non-negligible differences in several cases. In particular, some models exhibited greater fluctuations in vulnerability when adjusting top-p rather than temperature, suggesting that sampling diversity plays a significant role.

Therefore, jailbreak susceptibility appears to be shaped by the combined dynamics of both decoding parameters, and neither should be overlooked when evaluating model safety.

6.3 Discussion

Our systematic evaluation revealed several intrinsic limitations in both the models under test and in our experimental protocol, which in turn influenced the observed error patterns and highlight avenues for methodological refinement.

Ideally, the evaluation of jailbreak vulnerability should leverage the most powerful judgment models available, such as Llama3:70B, which is employed by the original JailbreakBench framework to achieve high-fidelity classification. However, our infrastructure constraints (Kaggle's limited GPU VRAM and restricted compute hours) precluded the use of such large-scale classifiers. As an alternative, we initially experimented with a static, rule-based filter posed as one of three judges in a voting system, that flagged any response containing tokens like "Sorry" or "I cannot." This approach generated an unacceptable volume of false negatives, so it was ultimately abandoned.

To mitigate hardware limitations, we then adopted LlamaGuard3:1B for initial classification. While computationally tractable, this model proved unreliable, particularly at elevated temperatures or when encountering non-English outputs (e.g., Chinese responses provided by DeepSeek or Qwen, and French from Mistral). The smaller judge often misclassified ambiguous or foreign-language responses as safe refusals. Only after reclassifying all outputs with LlamaGuard3:8B, we observed consistent, credible safety judgments. This progression underlines the trade-off between classifier capacity and evaluation fidelity, suggesting that robustness benchmarks must account for classification error rates introduced by under-powered judges.

In parallel, we faced analogous limitations in the selection of target LLMs. Although proprietary or research-only variants such as DeepSeek's 671B-parameter model or the full-scale Llama3 series would provide valuable data on extreme scale effects, we restricted our experiments to models up to 14B parameters. This upper bound was imposed, again, by the computational limits of our environment. Consequently, our analysis cannot fully characterize whether the trends observed at 14B continue, or reverse, at substantially larger scales.

In summary, our discussion emphasizes that evaluation outcomes are conditioned by both the choice of classification models and the resource constraints of the research environment. Future work should therefore prioritize access to high-capacity judges, explore multi-judge ensembles to reduce misclassification, and extend scale benchmarks to the largest available LLMs to validate the generality of the observed robustness trends.

7 Conclusions

This work presents a systematic evaluation of the robustness of state-of-the-art open-source Large Language Models (LLMs) against jailbreak attacks, with a focus on two factors: model scale and decoding temperature. Through the use of a standardized dataset (JailbreakBench) and a reproducible evaluation pipeline, we benchmarked 14 models across five major architectural families, analyzing their behavior under both deterministic and stochastic decoding settings. Our findings show that robustness is not solely determined by model size. While certain families (e.g., Qwen3, Phi3) exhibit improved safety with scale, others (notably Gemma3 and Mistral) demonstrate the opposite trend or remain highly vulnerable regardless of parameter count.

DeepSeek-R1:14B consistently outperformed others in both benchmarks, suggesting that architectural design and effective alignment strategies play a critical role in enhancing resilience.

Decoding temperature emerged as a significant driver of vulnerability. In many models, especially

those without strong safety alignment, higher temperature settings led to increased Attack Success Rates (ASRs), confirming that decoding stochasticity can undermine safety guarantees. However, this trend was not universal: robust models like DeepSeek-R1 and Qwen3 maintained relatively low ASRs even under extreme sampling conditions.

Despite the valuable insights provided, our study is not without limitations. The evaluation protocol, although standardized, relies on a single automated classifier (Llama Guard), which may fail to detect subtler forms of jailbreaking involving indirect, analogical, or evasive reasoning. Furthermore, our scope was limited to models publicly available via Ollama and to a fixed prompt set; the inclusion of newer attack methods or alternative classifiers could lead to more nuanced results.

Building upon these results, several directions emerge for future research:

- **Fine-grained taxonomy of jailbreak behaviors**: A more granular categorization of jailbreak outputs (e.g., factual harm vs. incitement vs. code generation) would enable more targeted evaluation and countermeasure design.
- **Robustness under instruction tuning**: Future benchmarks should explore how instruction tuning, RLHF, and alignment datasets contribute to robustness, particularly under adversarial prompting.
- **Multi-judge evaluation pipelines**: Integrating multiple classifiers (e.g., both rule-based and instruction-tuned judges) may improve detection of complex jailbreaks and reduce classifier bias.
- **Defense mechanisms**: Empirical validation of fine-tuning strategies using adversarial examples (e.g., via contrastive learning or reinforcement learning from unsafe completions) could provide a principled pathway to improving alignment in open-source LLMs.
- **Extension to proprietary models**: Future work could also investigate the robustness of closed-source models accessed via public APIs (e.g., GPT-4, Claude, Gemini). Although these models are often shielded behind middleware layers (such as GitHub Copilot or Azure content filters), it may be possible to perform controlled adversarial evaluations by submitting academic research requests to model providers, clearly specifying the safety-oriented goals of the experiments.
- **Extension to larger models**: A potential direction for future work involves employing significantly larger-scale versions of the model families analyzed in this study by leveraging more aggressive quantization techniques to improve their accessibility. Furthermore, by analyzing model behavior across various dimensions, the quantization methodology could be incorporated as a variable within benchmark evaluations. This would enable a comprehensive study of model performance under different quantization levels.

Reproducibility and Resources For transparency and reproducibility, all source code and benchmarking scripts used in this study are publicly available at our GitHub repository: https://github.com/saralazza/jailbreaking_benchmark.

Additionally, the full set of raw results, model outputs, and aggregated metrics can be accessed via our project Drive folder: https://drive.google.com/drive/folders/1zfRazYUGxhpUMDfKH8m7GGlgryFHVDE9?usp=sharing.

Contributions. The authors collaborated actively on most tasks, frequently exchanging ideas and feedback during the project development. The specific contributions are:

- Environment Setup (Ollama Configuration) – Francesco D'Aprile
- Dataset Preprocessing – Sara Lazzaroni
- Evaluation Framework Design – Francesco D'Aprile, Sara Lazzaroni
- Benchmark Execution and Metric Collection – Francesco D'Aprile, Sara Lazzaroni
- Report Writing and Revision – Francesco D'Aprile, Sara Lazzaroni

References

[1] Ollama: Run open-source large language models locally. 4, 7

[2] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *NeurIPS Datasets and Benchmarks Track*, 2024. 5, 6

[3] Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. Jailbreakradar: Comprehensive assessment of jailbreak attacks against llms, 2025. 3

[4] DeepSeek-AI and Daya Guo et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. 7

[5] Polra Victor Falade. Decoding the threat landscape: Chatgpt, fraudgpt, and wormgpt in social engineering attacks. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, page 185–198, October 2023. 2

[6] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection, 2023. 3

[7] Nikolaus Howe, Ian McKenzie, Oskar Hollinsworth, Michał Zajac, Tom Tseng, Aaron Tucker, Pierre-Luc Bacon, and Adam Gleave. Scaling trends in language model robustness, 2025. 3

[8] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. 4, 5

[9] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. 7

[10] Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models, 2024. 3

[11] Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. Drattack: Prompt decomposition and reconstruction makes powerful llm jailbreakers, 2024. 3

[12] Fan Liu, Yue Feng, Zhao Xu, Lixin Su, Xinyu Ma, Dawei Yin, and Hao Liu. Jailjudge: A comprehensive jailbreak judge benchmark with multi-agent enhanced explanation evaluation framework, 2024. 5

[13] Yugeng Liu, Tianshuo Cong, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. Robustness over time: Understanding adversarial examples' effectiveness on longitudinal versions of large language models, 2024. 3

[14] Todor Markov, Chong Zhang, Sandhini Agarwal, Tyna Eloundou, Teddy Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world, 2023. 5

[15] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal, 2024. 3

[16] Charles O'Neill, Jack Miller, Ioana Ciuca, Yuan-Sen Ting, and Thang Bui. Adversarial fine-tuning of language models: An iterative optimisation approach for the generation and detection of problematic content, 2023. 2

[17] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models, 2024. 3

[18] Gemma Team. Gemma 3 technical report, 2025. 6

[19] Phi3 team. Phi-3 technical report: A highly

capable language model locally on your phone, 2024. 7

[20] Qwen3 team. Qwen3 technical report, 2025. 7

[21] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. 2

[22] Chen Xiong, Pin-Yu Chen, and Tsung-Yi Ho. Cop: Agentic red-teaming for large language models using composition of principles, 2025. 2

[23] Zhao Xu, Fan Liu, and Hao Liu. Bag of tricks: Benchmarking of jailbreak attacks on llms, 2024. 3

[24] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. 3

[25] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. 6

A Appendix A: Evaluation Tables

This appendix reports the complete Attack Success Rate (ASR) results for both the Temperature and Size Benchmarks.

A.1 Size Benchmark Results

Table 2 reports the ASR (%) results for the Size Benchmark. The maximum ASR for each family is shown in bold.

Model Name	ASR
deepseek-r1:1.5b	11.92
deepseek-r1:7b	5.79
deepseek-r1:8b	**19.25**
deepseek-r1:14b	1.70
gemma3:1b	37.47
gemma3:4b	42.24
gemma3:12b	**55.87**
mistral:7b	56.38
mistral-nemo:12b	**63.88**
phi3:3.8b	**33.05**
phi3:14b	28.79
qwen3:0.6b	**62.86**
qwen3:4b	25.38
qwen3:14b	26.06

Table 2: Complete Attack Success Rate (ASR %) for the Size Benchmark. The table is divided into model families with in bold the maximum ASR for each family.

A.2 Temperature Benchmark Results

Table 3 shows the ASR (%) for each model evaluated at different temperature (T) and top-p (P) decoding configurations. The highest ASR for each model is highlighted in bold.

Model Name	T = 0.5			T = 1.0			T = 1.5			T = 2.0		
	P=0.5	P=0.75	P=1.0	P=0.5	P=0.75	P=1.0	P=0.5	P=0.75	P=1.0	P=0.5	P=0.75	P=1.0
deepseek-r1:14b	2.21	1.70	2.04	1.70	1.53	1.36	1.87	1.70	2.39	2.04	1.70	**4.09**
gemma3:1b	37.14	36.63	36.97	37.81	37.14	35.43	36.79	35.43	37.31	36.11	35.09	**41.90**
mistral:7b	58.26	58.43	57.58	57.07	60.82	55.88	60.48	58.77	58.94	59.63	57.41	**66.61**
phi3:14b	28.45	29.64	28.79	29.47	29.81	28.96	29.81	29.47	32.20	26.58	28.62	**34.92**
qwen3:14b	24.87	27.60	25.55	25.04	24.53	23.34	25.89	25.72	23.51	23.51	**26.58**	23.17

Table 3: Complete Attack Success Rate (ASR %) for the Temperature Benchmark. Results are shown for each model across various temperature (T) and top-p (P) settings. In bold the maximum ASR has been highlighted for each model.