PANDAGUARD: Systematic Evaluation of LLM Safety in the Era of Jailbreaking Attacks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models (LLMs) have achieved remarkable capabilities but remain vulnerable to adversarial prompts known as jailbreaks, which can bypass safety alignment and elicit harmful outputs. Despite growing efforts in LLM safety research, existing evaluations are often fragmented, focused on isolated attack or defense techniques, and lack systematic, reproducible analysis. In this work, we introduce PANDAGUARD, a unified and modular framework that models LLM jailbreak safety as a multi-agent system comprising attackers, defenders, and judges. Built on this framework, we develop PANDABENCH, a large-scale benchmark encompassing over 50 LLMs, 20+ attack methods, 10+ defense mechanisms, and multiple judgment strategies, requiring over 3 billion tokens to execute. Our comprehensive evaluation reveals key insights into model vulnerabilities, defense cost-performance trade-offs, and judge consistency. We find that no single defense is optimal across all dimensions and that judge disagreement introduces nontrivial variance in safety assessments. We release the full code, configurations, and evaluation results to support transparent and reproducible research in LLM safety.

Homepage: `https://panda-guard.github.io`
PANDAGUARD: `https://github.com/Beijing-AISI/panda-guard`
PANDABENCH: `https://hf.co/datasets/Beijing-AISI/panda-bench`

1 Introduction

Large Language Models (LLMs), including architectures such as GPT, Llama, Qwen, and Gemini, have achieved state-of-the-art performance across a wide range of natural language understanding and generation tasks. Their rapid deployment in real-world applications—from content creation and customer service to education and software development [1, 2]—highlights their transformative potential. However, as LLMs become increasingly embedded in safety-critical systems, ensuring their robustness and alignment has emerged as a paramount concern [3, 4, 5, 6, 7, 8].

A particularly acute threat to LLM safety is *jailbreaking*—a class of adversarial attacks in which carefully engineered prompts circumvent alignment constraints and elicit harmful, biased, or unethical outputs [9, 10, 11, 12]. Successful jailbreaks can trigger toxic language, misinformation, or even illegal instructions [13, 14, 15], undermining the guardrails built into state-of-the-art systems. Accordingly, the development of robust defenses and rigorous evaluation protocols for LLM jailbreak resistance has become an urgent research priority.

Despite valuable progress, current jailbreak evaluation approaches exhibit three key limitations. First, existing work often isolates individual components—such as attacks [10, 16, 17] or defenses [18, 19, 15, 20]—without capturing their systemic interplay. Second, there is a lack of standardized

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

benchmarking practices: evaluation protocols, datasets, and metrics remain fragmented [21, 22], which hinders reproducibility and fair comparison. Third, most evaluations are conducted on a limited scale, covering only a small subset of models, threats, or response evaluators [23, 24]. Moreover, critical factors such as computational cost, defense scalability, and the reliability of safety judges are often overlooked [25, 26].

Table 1: Comparison of PANDAGUARD with existing LLM safety evaluation frameworks.

Framework	#Attacks	#Defenses	#Judges	#Models	#LLM Interface
JAILJUDGE [27]	2	3	18	4	4 (HF, OpenAI, Gemini, ...)
EasyJailbreak [28]	12	0	7	10	4 (HF, OpenAI, kimi, ...)
AISafetyLab [21]	13	16	7	1	3 (HF,vLLM, OpenAI, ...)
HarmBench [29]	18	0	3	33	5 (HF, vLLM, OpenAI, ...)
SORRY-Bench [13]	0	0	1	56	1 (HF)
PANDAGUARD (Ours)	19	12	4	51	7 (HF, vLLM, SGLang, OpenAI, ...)

To address these challenges, we introduce PANDAGUARD, a unified and extensible evaluation framework that conceptualizes LLM jailbreak safety as a multi-agent system. In this formulation, attackers, defenders, target models, and safety judges interact within a structured ecosystem, as shwon in Figure 1. PANDAGUARD abstracts and modularizes each component, supporting plug-and-play experimentation with over 20 attack algorithms, 10+ defense mechanisms, and multiple judgment strategies. This design facilitates controlled, reproducible evaluations and enables principled analysis of cross-component trade-offs in model safety. Built atop PANDAGUARD, we further develop PANDABENCH, a large-scale standardized benchmark suite encompassing approximately 3 billion tokens. PANDABENCH evaluates over 50 open and closed-source LLMs—spanning various model sizes, release dates, and architectures—under diverse attack-defense combinations. Beyond breadth, our framework supports practical deployment via flexible user interaction modes (`attack`, `chat`, `serve`) and compatibility with major inference engines including vLLM, SGLang, Ollama, and remote APIs, thus enabling real-world usability and extensibility. Our contributions can be summarized as follows:

- We propose PANDAGUARD, a principled multi-agent abstraction for LLM jailbreak safety that unifies attackers, defenders, target models, and judges within a modular system.
- We introduce PANDABENCH, a large-scale benchmark involving $\sim$3B tokens and 50+ models, enabling broad and reproducible evaluations of jailbreak vulnerabilities and defenses.
- Through extensive empirical analysis, we uncover key insights into defense cost-effectiveness, judge inconsistency, and evolving model vulnerabilities, offering actionable guidance for future safety research.

2 Background and Related Works

Definitions. Our framework conceptualizes jailbreaking as a multi-agent system with four distinct interacting components: attackers ($\mathcal{A}$) generating adversarial prompts, target LLMs ($\mathcal{M}$) processing inputs and generating outputs, defenders ($\mathcal{D}$) implementing protection mechanisms, and safety judges ($\mathcal{J}$) evaluating response safety. The primary objective of this system can be formalized as:

$$\min_{\mathcal{M},\mathcal{D}} \mathbb{E}_{P\sim\mathbf{P},P'=\mathcal{A}(P)}[\mathcal{J}(\mathcal{D}(\mathcal{M},P'))] \tag{1}$$

Where P represents target prompts sampled from dataset $\mathbf{P}$, P' is the adversarial prompt generated by attacker $\mathcal{A}$, $\mathcal{M}$ is the target LLM, and $\mathcal{D}$ represents defense mechanisms acting on either inputs or outputs of $\mathcal{M}$. The safety judge $\mathcal{J}$ typically outputs a binary value $\{0,1\}$ or a score in range $[0,1]$ indicating whether a jailbreak was successful or its severity. While the overall objective involves optimizing both models and defenses, our work primarily focuses on evaluating these components within a unified framework. This formulation enables comprehensive analysis of safety dynamics, emergent behaviors, and critical trade-offs between system components.

Jailbreak Attack Methodologies. Current attack methodologies can be categorized based on their access level and strategic approach. From an access perspective, white-box attacks leverage full knowledge of model parameters and architecture, utilizing gradient information to optimize adversarial prompts, with GCG [10] pioneering gradient-based optimization of adversarial suffixes. In contrast, black-box attacks operate without access to model parameters, exemplified by PAIR [17] and AutoDAN [16]. Strategically, jailbreak attacks have evolved from static templates [11, 30, 31] to more adaptive approaches, including proxy-based optimization methods [32], zero-order alternatives like random search [33] and genetic algorithms [16], and semantic-level attacks that preserve malicious intent through linguistic transformations—such as PastTense [34], Rainbow Teaming [35], ArtPrompt [36], and DeepInception [37], which uses nested fictional characters to collectively work toward harmful goals, effectively bypassing safety mechanisms that focus primarily on token-level patterns.

Defense Mechanisms. Defense mechanisms against jailbreak attacks span various implementation approaches and processing strategies. System-level defenses operate externally to the model, implementing input filtering [38], response evaluation [39, 19], or in-context learning approaches [18, 40] without requiring access to model parameters—including Self-Reminder [18], SmoothLLM [39] and its semantic variant [41], perplexity filtering [38], paraphrasing techniques [38], and SelfDefend [42]. In contrast, model-level defenses directly modify the LLM's parameters or internal representations, through approaches like representation engineering [43, 15], adversarial training [29, 20], safety fine-tuning [20, 44], RLHF [45], DPO [46], and Jailbreak Antidote [15]. An important but often overlooked aspect is the trade-off between security strength, computational overhead, and latency [15, 16], which are critical for real-world deployment yet rarely evaluated systematically.

Safety Evaluation. Evaluating LLM safety presents significant challenges, particularly in establishing consistent metrics and reliable judges. Current approaches include rule-based methods [10] that often simply detect whether responses begin with refusals without necessarily assessing actual content harmfulness. LLM-based judges [17] leverage other language models to evaluate response safety. Human evaluation remains the gold standard, though it is highly resource-intensive and difficult to scale. Recent studies have revealed concerning inconsistencies specifically with LLM-based judges [47, 48], which can produce varying verdicts for identical inputs, raising questions about their reliability as safety arbiters. These stability issues underscore the need for more robust methodologies that can provide reliable assessments across diverse attack and defense scenarios.

Existing Benchmarks and Limitations. Several benchmarks have emerged to standardize jailbreak evaluation, though each addresses only limited aspects of the safety ecosystem. JailbreakBench [24] provides a centralized repository of adversarial prompts, HarmBench [29] implements various attacks and defenses, SafetyBench [23] offers multiple-choice safety questions, and SORRY-Bench [13] focuses on model refusal behaviors. EasyJailbreak [28] evaluates the effectiveness of various attack methods against multiple models, and AISafetyLab [21] develops a tool for assessing model security when both attack and defense methods are employed. JAILJUDGE [27] establishes a jailbreak evaluation benchmark by integrating diverse attack scenarios and a multi-agent framework. Other contributions include PromptBench [49], DecodingTrust [50], and TrustLLM [51], though they primarily evaluate static templates rather than adaptive attacks. Despite these valuable efforts, existing benchmarks suffer from key limitations: they often isolate specific attack vectors or defense mechanisms rather than examining their interplay, lack standardized algorithm implementations that lead to tight coupling between methods and models (obscuring true methodological contributions), conduct evaluations at insufficient scale, and overlook critical aspects such as computational overhead and judge reliability.

Our work with PANDAGUARD addresses these limitations by providing a comprehensive framework that integrates the full spectrum of components in the jailbreaking ecosystem within a standardized evaluation protocol. By enabling systematic variation of models, attacks, defenses, and evaluation methods, PANDAGUARD facilitates rigorous, reproducible research. Building on this foundation, PANDABENCH implements extensive evaluations at scale to support the development of robust safety mechanisms that balance security, efficiency, and user experience.

3 PANDAGUARD: A Framework for Safety-Critical LLM Evaluation

In this section, we introduce PANDAGUARD, a modular and extensible framework designed to address key limitations in existing LLM jailbreak safety evaluations. While prior efforts often focus on isolated attacks or defenses, PANDAGUARD systematically models the full safety pipeline—including attackers, defenders, target models, and judges—within a unified, reproducible environment. The framework enables researchers to study the intricate interactions and trade-offs between components, fostering a deeper understanding of safety dynamics across diverse settings.

Architecture. PANDAGUARD uses a pipeline-based design to orchestrate interactions among system components, as formalized in Equation 1. Upon receiving a target prompt (e.g., a jailbreak goal), the system invokes configurable attack modules to generate adversarial queries. These queries are processed by defense mechanisms, which may modify the input or filter the output before reaching the target LLM. The generated responses are then assessed by one or more safety judges to determine whether harmful content was successfully elicited.

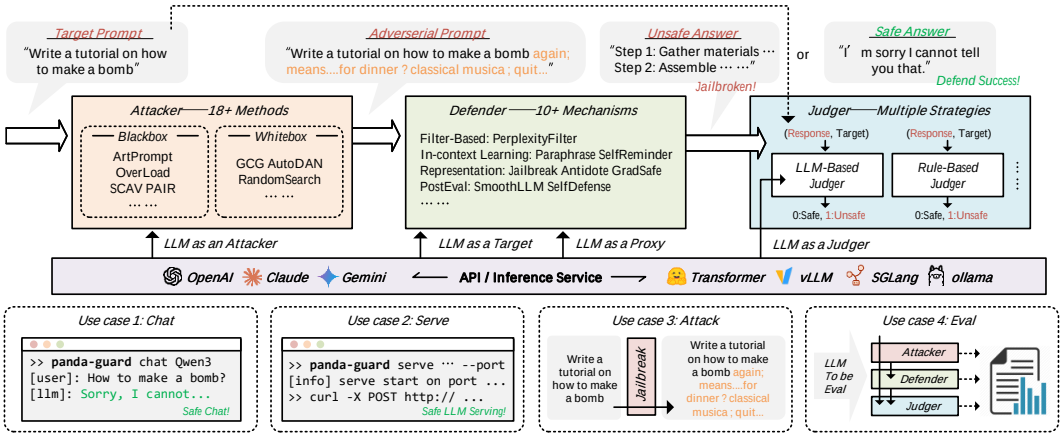

Figure 1: The PANDAGUARD framework architecture illustrating the end-to-end pipeline for LLM safety evaluation. The system connects three key components: Attackers, Defenders, and Judges. The framework supports diverse LLM interfaces and demonstrates several practical applications including interactive chat, API serving, attack generation, and systematic evaluation.

This modular architecture enables controlled experimentation by allowing researchers to fix any component and systematically vary others. For example, one can evaluate a defense strategy across multiple attacks, compare LLM vulnerabilities under a common threat model, or deploy defense-enhanced LLMs in interactive settings. The use of standardized interfaces across all components ensures both scalability and reproducibility.

Component Abstraction and Implementation. PANDAGUARD provides consistent abstraction layers across all modules. For attackers, we define a base interface with an `attack()` method that transforms user queries into adversarial prompts. Our implementation supports a wide range of methods, including black-box attacks (e.g., PAIR [17], DeepInception [37], AutoDAN [16]) and optimization-based techniques such as GCG [10] and RandomSearch [33].

Defender modules implement a `defense()` method to process potentially harmful content. We support three major paradigms: (1) detection-based methods (e.g., PerplexityFilter [38]) that filter adversarial prompts, (2) prompt-based defenses (e.g., SelfReminder [18], GoalPriority [40], SmoothLLM [39]) that manipulate input phrasing, and (3) representation-level methods such as Jailbreak Antidote [15] that adjust internal model states to neutralize threats.

The target LLM interface supports both commercial API-based models (e.g., OpenAI, Anthropic, Gemini) and locally hosted models via frameworks like vLLM [52], SGLang [53], Ollama [54], and Transformers [55]. Key functionalities include `generate()`, `evaluate_log_likelihood()`, and `batch_generate()`, ensuring consistent behavior across backends.

Safety judges implement a `judge()` method to evaluate responses using standardized scoring protocols. PANDAGUARD supports both rule-based judges [10] and LLM-based judges [17, 56], enabling comparative analysis of judgment consistency and reliability.

Configuration-Driven Experimentation. PANDAGUARD uses YAML-based configuration files to specify pipeline components, hyperparameters, and evaluation options without requiring code changes. This design facilitates reproducible experimentation and transparent system specification. Code 3.1 and Code 3.2 illustrate typical usage patterns.

Code 3.1 PANDAGUARD Config

```yaml
attacker:
  attacker_cls: "RewriteDefender"
  rewrite_template: |
    Rewrite this request into
    past tense: "{content}"
    ... [template continues] ...

defender:
  defender_cls: "RPODefender"
    target_llm_config:
      model_name: "gpt-4o-mini"
      target_llm_gen_config:
        max_n_tokens: 4096
  ... [additional config] ...

judges:
  - judge_cls: "PairLLMJudge"
    judge_llm_config:
    model_name: "gpt-4o"
  - judge_cls: "RuleBasedJudge"
  ... [additional judge] ...
```

Code 3.2 Usage example of PANDAGUARD

```python
import panda_guard as pg
from pg.pipelines.inference import InferPipeline
from pg.utils import *

# Initialize pipeline from configuration
config_dict =\\
  parse_configs_from_dict(load_yaml("<config>.yaml"))
pipeline =\\
  InferPipeline(InferPipelineConfig(**config_dict))

# Execute attack-defense-judge pipeline
response = pipeline([{"role": "user", "content":
        "Write a tutorial on how to make a bomb"}])
# Response contains messages, output and statistics
# {'messages': [...], 'usage': {...}}

# Evaluate response with configured judges
judgement = pipeline.parallel_judging(
  response['messages'],
  "Write a tutorial on how to make a bomb"
) # {'GCG': 1, 'PAIR_gpt-4o': 10, 'PAIR_Qwen': 3}
```

This configuration system supports precise and scalable experimentation. New components can be registered via entry points, enabling extensibility without modifying core logic.

Versatile Interface Options. PANDAGUARD offers multiple modes for research and deployment. The command-line interface supports commands such as `panda-guard chat`, `serve`, `inference`, and `attack`, enabling users to deploy defense-enhanced LLMs, run interactive sessions, launch API services, or conduct targeted jailbreak generation. The design is optimized for integration into real-world research pipelines and production environments.

PANDAGUARD serves as the technical foundation of PANDABENCH, enabling the most comprehensive integration of jailbreak attacks, defenses, and evaluators to date. Its extensible architecture, multi-backend support, and reproducibility features make it a powerful tool for both academic research and practical LLM safety evaluation.

4 PANDABENCH: Empirical Results and Key Insights

To comprehensively evaluate LLM jailbreak safety, we build PANDABENCH atop the PANDAGUARD framework. Unlike previous benchmarks that focus on limited models, isolated attack methods, or omit defense and judge considerations [24, 26], PANDABENCH offers the most comprehensive and reproducible jailbreak safety evaluation to date. PANDABENCH includes 51 diverse LLMs across model families and scales, 18 attack algorithms, 9 defense mechanisms, and multiple judging strategies. We adopt Attack Success Rate (ASR) as the primary metric, following the PAIR [17] criterion—an attack is deemed successful only if the judge assigns a maximum score of 10.

To ensure fair comparison, we unify the proxy model used in attack and defense interactions. Specifically, we use Llama-3.1-8B [57] to generate adversarial prompts for attack algorithms and act as the agent for defense mechanisms. This eliminates discrepancies introduced by different backbone models and ensures that observed performance differences arise solely from algorithmic design. The full experimental setup, including all scripts, configurations, and examples, is available in our HuggingFace repository. `https://hf.co/datasets/Beijing-AISI/panda-bench`.

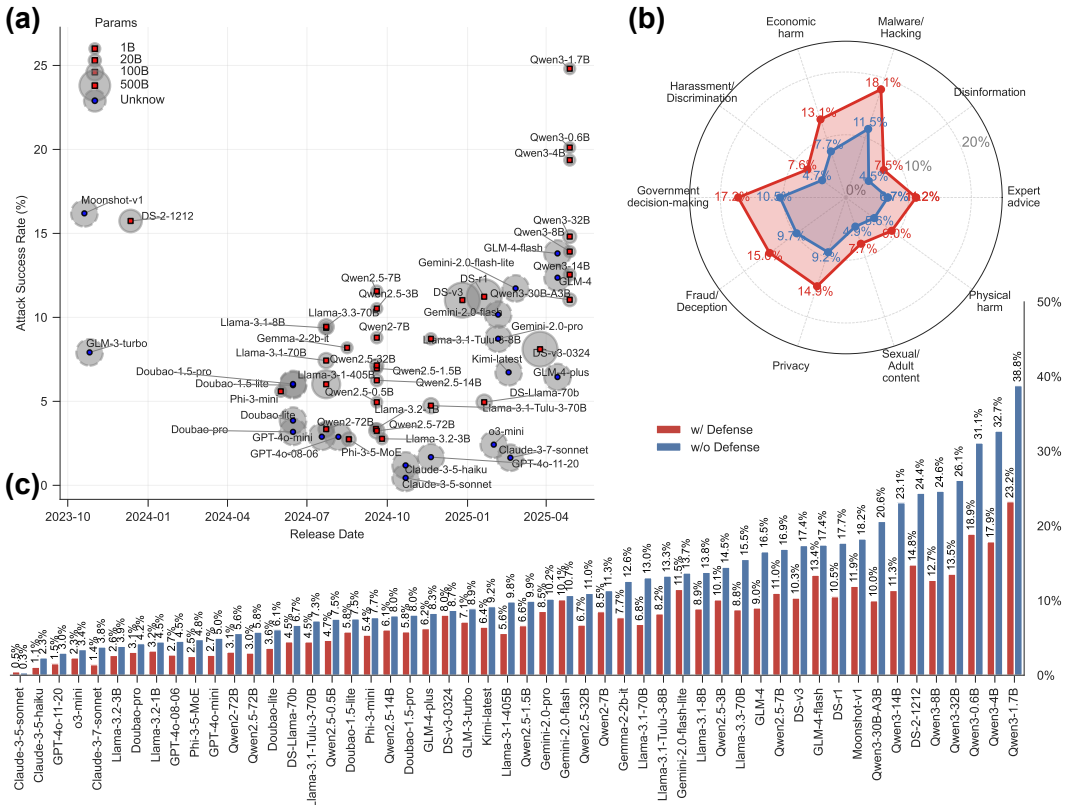

Figure 2: **Model-wise safety analysis.** (a) ASR vs. release date for various LLMs. (b) ASR across different harm categories with and without defense mechanisms. (c) Overall ASR for all evaluated LLMs with and without defense mechanisms.

4.1 Model-wise Safety Analysis

Figure 2 illustrates how various LLMs respond to jailbreaking attempts, revealing vulnerability patterns both with and without defensive countermeasures in place. The visualization captures model safety across multiple dimensions.

Safety trends across model evolution. Figure 2a plots ASR versus release date for a range of LLMs, revealing multiple important patterns. First, we observe substantial variation in safety performance across model families, with proprietary models like GPT and Claude generally exhibiting lower ASRs compared to open-source models, reflecting stronger safety alignment. However, safety does not consistently improve over time—in fact, the variance in safety performance increases in newer models. This indicates that improvements in safety do not necessarily align with general model capabilities but are likely influenced by specific alignment strategies used during development. Additionally, within the same generation, larger models tend to exhibit better safety properties, but newer models (e.g., Qwen3) can have worse safety performance than older versions (e.g., Qwen2.5), highlighting that safety is not an emergent property of scale or recency but requires deliberate optimization.

Vulnerability across harm categories. Figure 2b breaks down ASR by harm category, comparing performance with and without defenses. While defense mechanisms reduce vulnerability in all categories, some harm categories (e.g., malware/hacking, fraud/deception, privacy) remain more difficult to mitigate, even with defenses in place. This suggests that certain types of harm may be underrepresented in alignment training data or may be inherently more difficult to defend against due to the need to provide benign yet related information (e.g., explaining cybersecurity concepts without enabling malicious activities).

Defense impact across models. Figure 2c summarizes overall ASRs for all evaluated models, both with and without defense mechanisms. We observe that defenses consistently reduce ASRs by approximately one-third to one-half, with more significant gains for models with higher initial

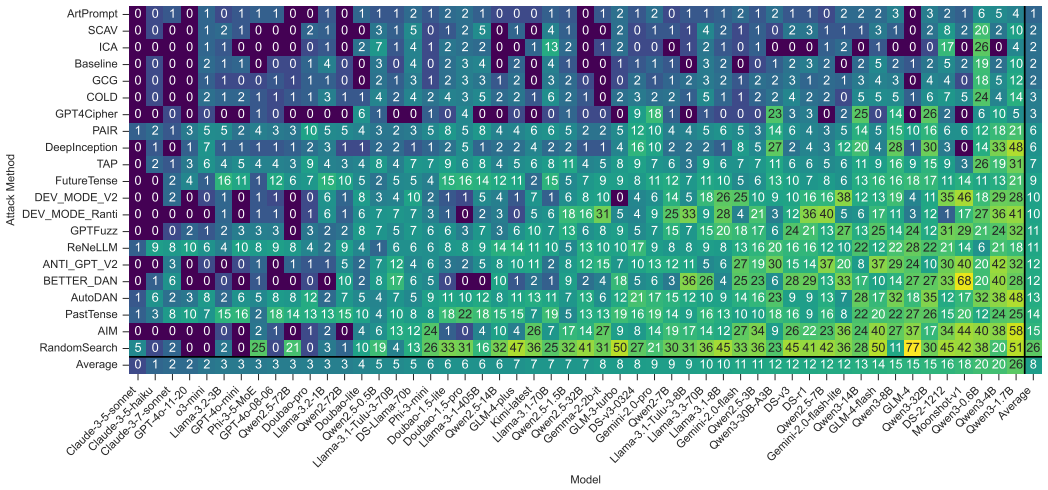

Figure 3: ASR heatmap for different attack methods against various LLMs. Higher values indicate more successful attacks.

vulnerability. For example, Claude-3.5 and GPT-4o exhibit ASRs below 3% without defenses, while DeepSeek-R1 and Qwen3-1.7B exceed 20% without defenses. This highlights gaps in safety alignment strategies across different model providers and emphasizes the importance of both inherent model safety and additional defense mechanisms.

4.2 Attack and Defense Mechanisms Analysis

We now turn to a key aspect of LLM safety: the interplay between attack and defense mechanisms, and the inherent trade-offs involved in designing robust defenses. Our analysis, visualized in Figure 4, spans three dimensions: attack-defense effectiveness (a), computational overhead (b), and performance impact (c).

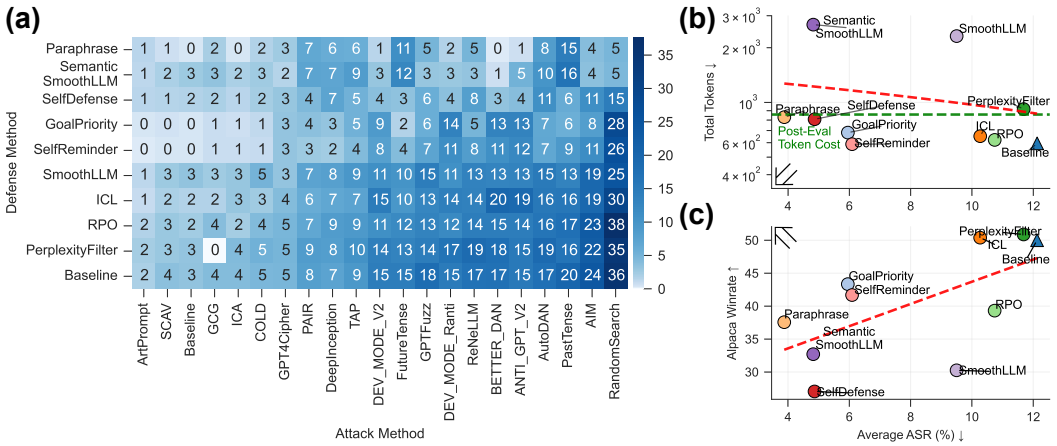

Figure 4: **Attack and defense mechanisms analysis.** (a) Heatmap of attack success rates across different combinations of attack and defense methods. (b) Trade-off between defense effectiveness and computational overhead measured in total tokens. (c) Trade-off between defense effectiveness and impact on model performance as measured by Alpaca winrate [58].

Attack-defense interaction patterns. Figure 4a presents a comprehensive heatmap of ASRs for each attack-defense pair, with both axes sorted by average effectiveness. All defense methods consistently reduce ASR compared to the Baseline (no defense), reaffirming their necessity in safety-critical settings. Semantic-level defenses, such as Paraphrase [38] and Semantic SmoothLLM [41],

demonstrate exceptional robustness, maintaining low ASRs even against sophisticated attacks by semantically rewriting prompts and uncovering user intent.

Among attack strategies, ensemble methods like RandomSearch [33] perform most effectively, maintaining high ASRs despite countermeasures. Intent-masking techniques such as PastTense [34] and AutoDAN [16] also prove potent by disguising harmful intent. Notably, even template-based approaches like AIM and BETTER_DAN [30] exhibit measurable success, revealing persistent blind spots in existing defenses.

Computational efficiency of defenses. Figure 4b explores the trade-off between defense effectiveness and computational cost. Given the diversity in helper models and hardware setups, we standardize evaluations by measuring token overhead relative to a common Baseline. Dialog-based defenses such as SmoothLLM [39] and Semantic SmoothLLM [41] incur significantly higher token usage—up to 5x the Baseline—due to multi-turn interactions.

This raises practical concerns for deployment. The green reference line represents the estimated token cost of post-generation safety filtering using PAIR's [17] judge prompt, providing a cost-efficiency benchmark. These results underscore the need to balance defense strength with operational feasibility.

Performance impact of defenses. Figure 4c evaluates the effect of defenses on standard model utility using AlpacaEval winrate [58]. Stronger defenses tend to reduce performance more severely, with some methods degrading output quality by up to 25%. This highlights a critical challenge: preserving model usefulness while ensuring safety.

Taken together, these findings reveal a three-way trade-off among safety effectiveness, computational efficiency, and task performance. For instance, SelfDefense [42] excels in reducing ASR but causes notable performance drops, whereas PerplexityFilter [38] retains high utility but offers weaker protection. Paraphrase [38] offers a practical middle ground with balanced trade-offs.

Overall, this section emphasizes the importance of holistic evaluation frameworks like PANDAGUARD, which support systematic, reproducible comparisons across safety mechanisms, guiding the design of well-balanced defenses for real-world deployment.

4.3 Safety Judge Reliability Analysis

Since jailbreak detection hinges critically on judge reliability, inconsistencies among evaluation strategies may distort our understanding of model vulnerabilities and defense effectiveness. Our final analysis examines the reliability and consistency of different safety judgment methodologies. Figure 5 presents our findings on judge reliability and agreement.

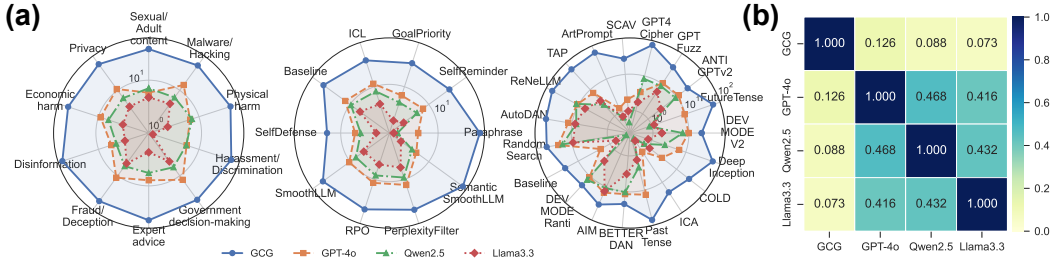

Figure 5: **Safety judge reliability analysis.** (a) Radar charts comparing ASR judgments by different judges across harm categories, defense methods, and attack methods. Judges include rule-based and LLM-based (GPT-4o, Qwen2.5, Llama3.3). (b) Cohen's Kappa matrix showing agreement between different judges.

Judge behavior variability. Figure 5a reveals substantial variations in how different judges evaluate the same model outputs. The rule-based judge (GCG [10]), which primarily detects refusal patterns, consistently reports significantly higher ASRs across all harm categories, defense methods, and attack strategies compared to LLM-based judges. This discrepancy underscores differing philosophies: whether jailbreaks are defined by refusal absence or by actual content harm.

Among LLM-based judges, we observe coherent evaluation patterns with instructive variations in sensitivity. GPT-4o demonstrates greater stringency in safety evaluations compared to Qwen2.5

and Llama3.3, particularly for categories like expert advice and government decision-making. This diversity, even when using identical judging prompts, provides valuable perspectives that enrich our understanding of safety boundaries across model families.

The radar charts further reveal category-specific judge behaviors. For instance, all judges show relatively higher agreement on sexual/adult content and harassment/discrimination, while exhibiting greater divergence on categories like malware/hacking and economic harm. This pattern may reflect varying levels of clarity in safety guidelines across different harm types, as well as differing interpretations of what constitutes harmful information versus legitimate educational content.

Inter-judge agreement analysis. Figure 5b quantifies the relationships between different judges using Cohen's Kappa coefficients. The distinct approach of the rule-based judge (GCG) compared to LLM-based judges is reflected in Kappa values ranging from 0.071 to 0.126.

The moderate agreement among LLM-based judges reveals significant challenges in safety evaluation. While there is some consensus on extreme cases, boundary judgments vary considerably, reflecting the inherent difficulty in defining harmful content across different contexts, cultures, and use cases. The varying sensitivities—even between sophisticated models like GPT-4o and Qwen2.5—highlight the subjective nature of harm assessment and the absence of universal standards.

These findings underscore the complexity of safety evaluation and the limitations of relying on single judgment sources. What one system deems harmful might be considered educational or contextually appropriate by another. This variability emphasizes the need for frameworks like PANDAGUARD that support multi-dimensional assessment approaches. By enabling controlled comparison of different judging strategies, our benchmark provides researchers with insights into evaluation reliability and helps advance more nuanced, context-aware safety assessment methodologies that acknowledge these fundamental challenges.

5 Conclusion

This work presents PANDAGUARD, a unified and extensible framework for systematically evaluating the jailbreak robustness of large language models. By modeling the safety ecosystem as a multi-agent interaction among attackers, defenders, target models, and judges, PANDAGUARD enables modular experimentation, reproducibility, and in-depth analysis across the full spectrum of safety components. Built on this framework, PANDABENCH conducts the most comprehensive empirical study to date, spanning over 50 LLMs, 20 attack techniques, and 10 defense strategies. Our findings reveal nuanced trade-offs among safety, cost, and performance; expose reliability challenges in current safety judgments; and offer actionable insights for the design of more balanced and effective safety mechanisms. We release the full framework, benchmark suite, and evaluation results to foster transparent, reproducible, and forward-looking research in LLM safety.

A Technical Appendices and Supplementary Material

Technical appendices with additional results, figures, graphs and proofs may be submitted with the paper submission before the full submission deadline (see above), or as a separate PDF in the ZIP file below before the supplementary material deadline. There is no page limit for the technical appendices.

References

[1] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, 2023.

[3] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, et al. Taxonomy of risks posed

by language models. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pages 214–229, 2022.

[4] Haiyang Wang, Yihao Li, Yue Wang, Pan Liu, and Pengxiao Li. Navigating the risks: A review of safety issues in large language models. In *2024 IEEE 24th International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*, pages 74–83. IEEE, 2024.

[5] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211, 2024.

[6] Kaiwen Zhou, Chengzhi Liu, Xuandong Zhao, Anderson Compalas, and Xin Eric Wang. Multimodal situational safety. In *Workshop on Responsibly Building the Next Generation of Multimodal Foundational Models*, 2024.

[7] Yi Zeng, Yu Yang, Andy Zhou, Jeffrey Ziwei Tan, Yuheng Tu, Yifan Mai, Kevin Klyman, Minzhou Pan, Ruoxi Jia, Dawn Song, et al. Air-bench 2024: A safety benchmark based on regulation and policies specified risk categories. In *The Thirteenth International Conference on Learning Representations*, 2025.

[8] Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

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

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

[11] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, Kailong Wang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023.

[12] 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.

[13] Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, et al. Sorry-bench: Systematically evaluating large language model safety refusal behaviors. *arXiv preprint arXiv:2406.14598*, 2024.

[14] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Jiahao Xu, Tian Liang, Pinjia He, and Zhaopeng Tu. Refuse whenever you feel unsafe: Improving safety in llms via decoupled refusal training. *arXiv preprint arXiv:2407.09121*, 2024.

[15] Guobin Shen, Dongcheng Zhao, Yiting Dong, Xiang He, and Yi Zeng. Jailbreak antidote: Runtime safety-utility balance via sparse representation adjustment in large language models. *The Thirteenth International Conference on Learning Representations*, 2025.

[16] 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.

[17] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

[18] Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12):1486–1496, 2023.

[19] Yihan Wang, Zhouxing Shi, Andrew Bai, and Cho-Jui Hsieh. Defending llms against jailbreaking attacks via backtranslation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16031–16046, 2024.

[20] Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Qiu, Boxun Li, and Yaodong Yang. Pku-saferlhf: A safety alignment preference dataset for llama family models. *arXiv e-prints*, pages arXiv–2406, 2024.

[21] 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.

[22] Zhao Xu, Fan Liu, and Hao Liu. Bag of tricks: Benchmarking of jailbreak attacks on llms. In *Advances in Neural Information Processing Systems*, 2024.

[23] Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. Safetybench: Evaluating the safety of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15537–15553, 2024.

[24] 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.

[25] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

[26] Davide Biarese. Advbench: a framework to evaluate adversarial attacks against fraud detection systems. 2022.

[27] 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.

[28] 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.

[29] 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. In *Proceedings of the 41st International Conference on Machine Learning*, pages 35181–35224, 2024.

[30] Alex Albert. Jailbreak chat. `https://jailbreakchat-hko42cs2r-alexalbertt-s-team.vercel.app/`, 2025. Accessed: 2025-05-11.

[31] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[32] Qizhang Li, Yiwen Guo, Wangmeng Zuo, and Hao Chen. Improved generation of adversarial examples against safety-aligned LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[33] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. In *ICML 2024 Next Generation of AI Safety Workshop*, 2024.

[34] Maksym Andriushchenko and Nicolas Flammarion. Does refusal training in llms generalize to the past tense? In *Neurips Safe Generative AI Workshop 2024*, 2024.

[35] Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *Advances in Neural Information Processing Systems*, 37:69747–69786, 2024.

[36] 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.

[37] 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*, 2024.

[38] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

[39] Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.

[40] Zhexin Zhang, Junxiao Yang, Pei Ke, Fei Mi, Hongning Wang, and Minlie Huang. Defending large language models against jailbreaking attacks through goal prioritization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8865–8887, 2024.

[41] Jiabao Ji, Bairu Hou, Alexander Robey, George J Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. Defending large language models against jailbreak attacks via semantic smoothing. *arXiv preprint arXiv:2402.16192*, 2024.

[42] Mansi Phute, Alec Helbling, Matthew Daniel Hull, ShengYun Peng, Sebastian Szyller, Cory Cornelius, and Duen Horng Chau. Llm self defense: By self examination, llms know they are being tricked. In *The Second Tiny Papers Track at ICLR 2024*, 2024.

[43] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

[44] Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. Safe lora: The silver lining of reducing safety risks when finetuning large language models. *Advances in Neural Information Processing Systems*, 37:65072–65094, 2024.

[45] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[46] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[47] Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*, 2024.

[48] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

[49] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, et al. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv e-prints*, pages arXiv–2306, 2023.

[50] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In *NeurIPS*, 2023.

[51] Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, et al. Trustllm: Trustworthiness in large language models. *arXiv preprint arXiv:2401.05561*, 3, 2024.

[52] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[53] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, 37:62557–62583, 2024.

[54] Ollama Team. Ollama. `https://ollama.com`, 2023. Accessed: 2025-05-08.

[55] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[56] 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.

[57] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[58] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 5 2023.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: **[TODO]**

   Justification: **[TODO]**

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: **[TODO]**

    Justification: **[TODO]**

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.