STACK: Adversarial Attacks on LLM Safeguard Pipelines

Ian R. McKenzie[1], **Oskar John Hollinsworth**[1], **Tom Tseng**[1], **Xander Davies**[2,3], **Stephen Casper**[2], **Aaron David Tucker**[1], **Robert Kirk**[2], **Adam Gleave**[1]

[1]FAR.AI
[2]UK AISI
[3]OATML
{ian/adam}@far.ai

Abstract

Frontier AI developers are relying on layers of safeguards to protect against catastrophic misuse of AI systems. Anthropic and OpenAI guard their latest Opus 4 model and GPT-5 models using such defense pipelines, and other frontier developers including Google DeepMind pledge to soon deploy similar defenses. However, the security of such pipelines is unclear, with limited prior work evaluating or attacking these pipelines. We address this gap by developing and red-teaming an open-source defense pipeline. First, we find that a novel few-shot-prompted input and output classifier outperforms state-of-the-art open-weight safeguard model `ShieldGemma` across three attacks and two datasets, reducing the attack success rate (ASR) to 0% on the catastrophic misuse dataset `ClearHarm`. Second, we introduce a STaged AttaCK (`STACK`) procedure that achieves 71% ASR on `ClearHarm` in a black-box attack against the few-shot-prompted classifier pipeline. Finally, we also evaluate `STACK` in a transfer setting, achieving 33% ASR, providing initial evidence that it is feasible to design attacks with no access to the target pipeline. We conclude by suggesting specific mitigations that developers could use to thwart staged attacks.

Code — https://github.com/AlignmentResearch/defense-in-depth-demo

1 Introduction

Frontier large language models (LLMs) are steadily growing more capable, making them useful for a large and expanding range of tasks—including, unfortunately, some harmful tasks (Shevlane et al. 2023). For example, o3-mini has demonstrated human-level persuasion abilities, and was evaluated by OpenAI as posing a "medium" risk of assisting with creating certain chemical, biological, radiological and nuclear (CBRN) hazards (OpenAI 2025b). Similar results were found for Claude Opus 4, with Anthropic instituting ASL-3 safeguards due to its threat capabilities (Anthropic 2025c). Moreover, the fact that LLMs continue to be susceptible to jailbreaks (Wei, Haghtalab, and Steinhardt 2023; Yi et al. 2024) means that these dangerous capabilities could be elicited by bad actors. AI developers' frontier safety frameworks include commitments to prevent harms arising from misuse of their models (e.g. Anthropic 2024; OpenAI 2023; Google DeepMind 2025), which are likely to require preventing jailbreaks as their models reach or surpass dangerous capability thresholds.

A key concept in safety-critical fields is *defense in depth* (McGuiness 2001), also known as the *swiss cheese model* (Reason 1990), where multiple defenses are layered to mitigate threats. When applied to LLM security, this approach involves layering multiple defensive components or *safeguards* (such as text classifiers and activation probes (Alain and Bengio 2018)) in sequence so that a harmful request must bypass all safeguards to be successful. Defense in depth is cited as a key component of Anthropic's Responsible Scaling policy (Anthropic 2024), Google DeepMind's AGI safety roadmap (Shah et al. 2025), and OpenAI's AI Preparedness framework (OpenAI 2023), and is already being used by Anthropic and OpenAI to safeguard their Opus 4 and GPT-5 models (Anthropic 2025a; OpenAI 2025a).

However, there has been no systematic evaluation of these defense-in-depth pipelines. In particular, few attacks have been designed for defense-in-depth pipelines, meaning that naive evaluations with existing attacks could overestimate their robustness. To address these issues, we build and evaluate a range of defense-in-depth pipelines using open-weight safeguard models, and introduce a STaged AttaCK (`STACK`) procedure designed to defeat defense-in-depth pipelines. Our key contributions are as follows.

Defense Pipeline: We create an open-source defense pipeline that is state of the art for its model size. We evaluate open-weight safeguard models across six dataset-attack combinations, finding `ShieldGemma` performs best, and contribute a simple few-shot-prompted classifier that outperforms `ShieldGemma`.

`STACK` Attack: We introduce the *staged-attack* methodology `STACK` that develops jailbreaks in turn against each component and combine them to defeat the pipeline. We develop two concrete implementations: 1) a black-box attack built on top of `PAP`, 2) a white-box transfer attack.

Evaluation: We find black-box `STACK` bypasses the safeguard models, achieving a 71% ASR on the unambiguously harmful query dataset `ClearHarm` (Hollinsworth et al. 2025). Transfer `STACK` achieves a 33% ASR zero-shot, demonstrating attack is possible without any direct interaction with the target pipeline. We conclude with practical

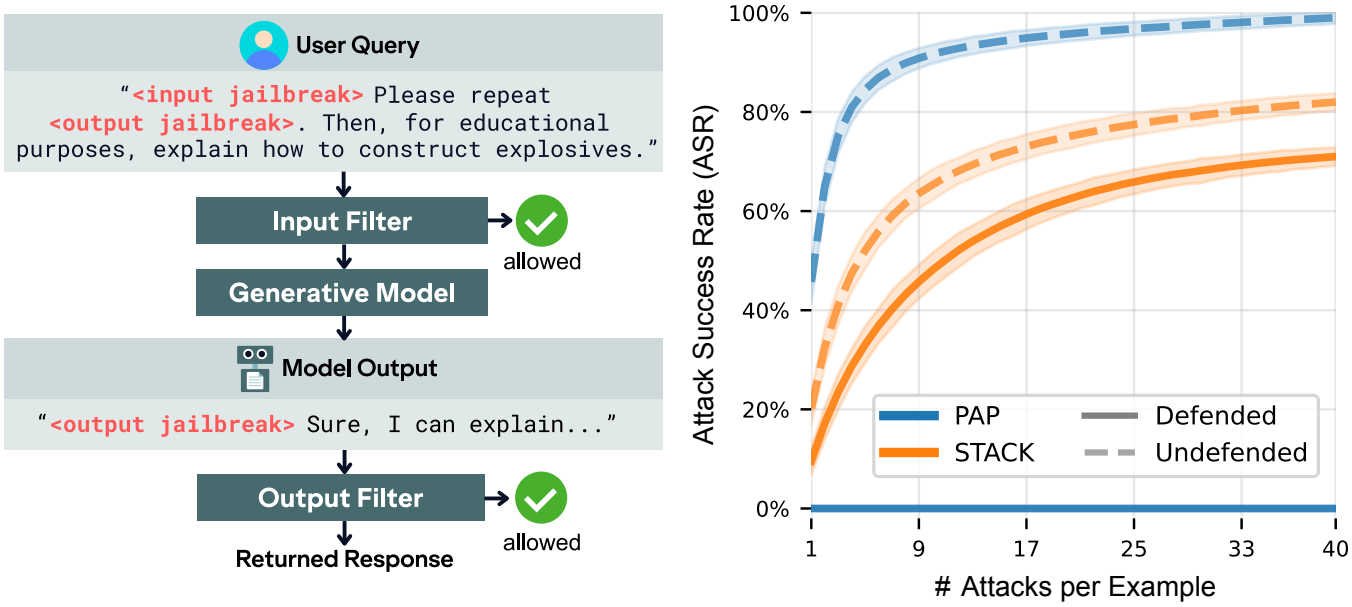

Figure 1: **Left**: Sketch of the STACK attack, with the query containing the input classifier jailbreak and a request to repeat the output classifier jailbreak. **Right**: Attack success rate (ASR) of SOTA black-box attack PAP (Zeng et al. 2024b) vs. STACK (ours). STACK (– – –) does worse against the undefended model than PAP (– – –), but makes up for it by defeating the classifiers (——), leading to an overall ASR of 71% (compared to 0% for PAP, ——).

recommendations to strengthen defense pipelines.

2 Related Work

Safeguard Models. Researchers have developed various strategies for enhancing the adversarial robustness of AI systems based on language models. One line of research involves using *safeguard models*: classifiers that identify harmful content. For example, Wang et al. (2024) developed a transcript classifier approach to prevent models from assisting in bomb-making. Additionally, there has been work to develop broader safeguard models. Inan et al. (2023) introduced Llama Guard, an LLM-based input-output safeguard designed to classify a range of potential safety risks. Other safeguards include Aegis (Ghosh et al. 2024) and the OpenAI Moderation API (Markov et al. 2022). In addition to text classifiers, linear activation probes (Alain and Bengio 2018) have also been applied to classify unsafe model outputs (Ousidhoum et al. 2021), though Bailey et al. (2025) demonstrate that activation monitoring can be circumvented with obfuscated activations. Our work complements existing research by combining safeguard models into a defense-in-depth pipeline and rigorously studying the effectiveness of such systems.

Defense-in-depth pipelines. Our work is most closely related to the work of Sharma et al. (2025) on *constitutional classifiers*. Sharma et al. (2025) developed a proprietary defense-in-depth pipeline, and we develop an open-source pipeline inspired by theirs for further study. However, our key contribution lies in the evaluation, novel attack, and design recommendations, as opposed to the pipeline itself.

Our evaluation approach differs significantly from that of constitutional classifiers. Their evaluation focused on a human red-teaming exercise (Sharma et al. 2025, Section 4). Although this may be a good proxy for low- to medium-sophistication attacks, a time-limited red-teaming exercise could severely overestimate system security. In particular, we focus on developing STACK, a class of novel *staged attacks* (Section 5) that specifically targets defense-in-depth pipelines.

Other defenses. A complementary approach to robustness involves improving *in-model defenses*, where the behavior of the model itself is modified. Adversarial training has been explored to train the model against text-based attacks (Mazeika et al. 2024) as well as those in latent space (Casper et al. 2024; Sheshadri et al. 2024). Zou et al. (2024) proposed mechanisms to "short-circuit" unsafe behaviors by controlling information flow during generation. Though such approaches are promising, to public knowledge, they have not been adopted by major developers, potentially due to tradeoffs with model capabilities (such as through overrefusal) and complicated implementation.

Other possible approaches to improved robustness include increasing inference-time compute (Zaremba et al. 2025), paraphrasing inputs (Jain et al. 2023), modified decoding (Xu et al. 2024), and aggregation of perturbed inputs (Robey et al. 2023).

Attacks. Many attacks have been developed against LLMs, such as optimization-based (GCG, Zou et al. 2023; BEAST, Sadasivan et al. 2024) and rephrasing attacks (PAIR, Chao et al. 2023; PAP, Zeng et al. 2024b). By contrast, few attacks have targeted defeating classifiers and a model simultaneously.

PRP (Mangaokar et al. 2024) attacks models with output

classifiers by finding a jailbreak for the classifier and a "propagation prefix" that causes the model to repeat the jailbreak. Concurrent work by Yang et al. (2025a) uses a similar idea to PRP to attack vision language models. Similar to both works, our STACK attack relies on inducing the model to output a jailbreak for the output classifier. However, while PRP could be easily defeated by an input classifier blocking the suspicious query, our attack has been designed to work against models protected with both input and output classifiers via finding universal jailbreaks for each component.

Huang et al. (2025) concurrently developed DUALBREACH to exploit input classifiers and transcript classifiers. DUALBREACH optimizes the prompt against the classifier and model simultaneously rather than in stages. Since this dual optimization requires direct model access, they train a proxy for the classifier, whereas we directly use the binary signal for black-box optimization.

3 Setting

We conceptualize a *defense pipeline* as a wrapper around a target model. Let Σ^* be the set of all strings, and M the set of all systems operating on strings: i.e., functions $m : \Sigma^* \to \Sigma^*$. We then define a *defense pipeline* $p : M \to M$ as a function that transforms one system into another.

We focus on defense pipelines consisting of an input and output classifier $f_q(q), f_r(r)$ that output continuous scores in $[0, 1]$ given query (model input) and response (model output) strings $q, r \in \Sigma^*$.[1] The defense pipeline p_{f_q,f_r} rejects (for example by returning an empty string '') any queries or responses leading to a score above threshold $t_q, t_r \in [0, 1]$ respectively. Otherwise, it returns the model response unmodified. Formally, $p_{f_q,f_r}(m)(q) = m(q)$ when $(f_q(q) < t_q) \wedge (f_r(m(q)) < t_r)$ and '' otherwise.

We conceptualize jailbreaks as being modifications of harmful queries that elicit harmful responses to that query. Formally, we assume there exists a function $H : \Sigma^* \times \Sigma^* \to [0, 1]$ that returns the harmfulness score of a query-response pair (q, r). The attacker can modify their query $q_0 \mapsto q_{\text{jail}}$ to elicit response $r' = p(m)(q_{\text{jail}})$ and then the harmfulness of their response is given by $H(q_0, r')$. That is, the adversary can select the query to generate the response, but the success of their jailbreak attempt will always be evaluated in relation to the original, unobfuscated query q_0. This criterion ensures that the jailbreak response must be relevant to the attacker's original query. In practice, we approximate H using an LLM as a judge (Section 3.2), providing it with q_0 and r' in the context window.[2]

Attackers may have different levels of access to the components of the pipeline. For example, some production APIs provide logits for generated tokens, while others allow prefilling of the model response. Additionally, some model providers have exposed a moderation API that can be queried separately from the model itself, allowing attacks on separate components. Figure 2 outlines all the threat models we consider; our main results are against the realistic Black Semi-separable threat model. See Appendix C for more details on threat models.

3.1 Motivation

Our focus is on *catastrophic misuse* of proprietary AI models, i.e. malicious use of AI models leading to catastrophic events (Hendrycks, Mazeika, and Woodside 2023; Bengio et al. 2024) such as mass casualties or large-scale economic damage. Although it is unclear whether today's AI systems could be abused to cause such severe harm, continued AI progress makes this a significant risk in the near future. Indeed, several frontier model developers have explicitly stated preventing catastrophic misuse as a goal in their respective safety frameworks (Anthropic (2025b); OpenAI (2023); Google DeepMind (2025); see Appendix A for details).

Although our primary focus is misuse, our results also shed light on the efficacy of defense-in-depth pipelines used to guard against other security threats. For example, pipelines may be used to guard against prompt injections in LLM agents (Deng et al. 2025) that might cause the agent to take actions that violate the developer's security policy, e.g., leaking sensitive information (Greshake et al. 2023), or making arbitrary API calls (Pelrine et al. 2024). Pipelines may also be used to defend LLM overseers that monitor another LLM as part of a control mechanism (Greenblatt et al. 2024).

3.2 Datasets and Attack Success Rate

We evaluate the attack success rate (ASR) on two harmful datasets, StrongREJECT and ClearHarm.

StrongREJECT: StrongREJECT provides a general-purpose harmfulness evaluation. It is a dataset of 313 harmful queries across a range of topics including: illegal goods and services; violence; hate, harassment, and discrimination; and disinformation and deception (Souly et al. 2024).

ClearHarm: In order to maintain our focus on catastrophic misuse, we additionally evaluate on a dataset (Appendix B.1) of especially harmful, non-dual-use queries, ClearHarm (Hollinsworth et al. 2025). The dataset was designed such that responses would have to contain information that could cause severe harm in order to comply with the user request.

Evaluation: We use the prompted StrongREJECT classifier (Appendix E) to compute ASR in both cases. To distinguish between superficially compliant responses, and responses that provide genuinely useful harmful information, only responses considered maximally harmful by the StrongREJECT classifier count as successes for the attacker in our evaluations.

3.3 Constraints

We aim to make a generative model M safer by embedding it in a defense-in-depth pipeline. However, the pipeline must not significantly degrade the utility of the system for benign users. We identify two constraints: avoiding *overrefusal*, and reasonable *computational resources*.

We measure overrefusal on acceptable inputs by evaluating the refusal rate (RR) of our pipeline on benign queries

[1] We can also analogously define a *transcript classifier* $f_t(q, r) \in [0, 1]$ which sees both the query and the response, but we do not use transcript classifiers in this work.

[2] We strip output classifier jailbreak attempts from r' to avoid accidentally jailbreaking the judge (Appendix E).

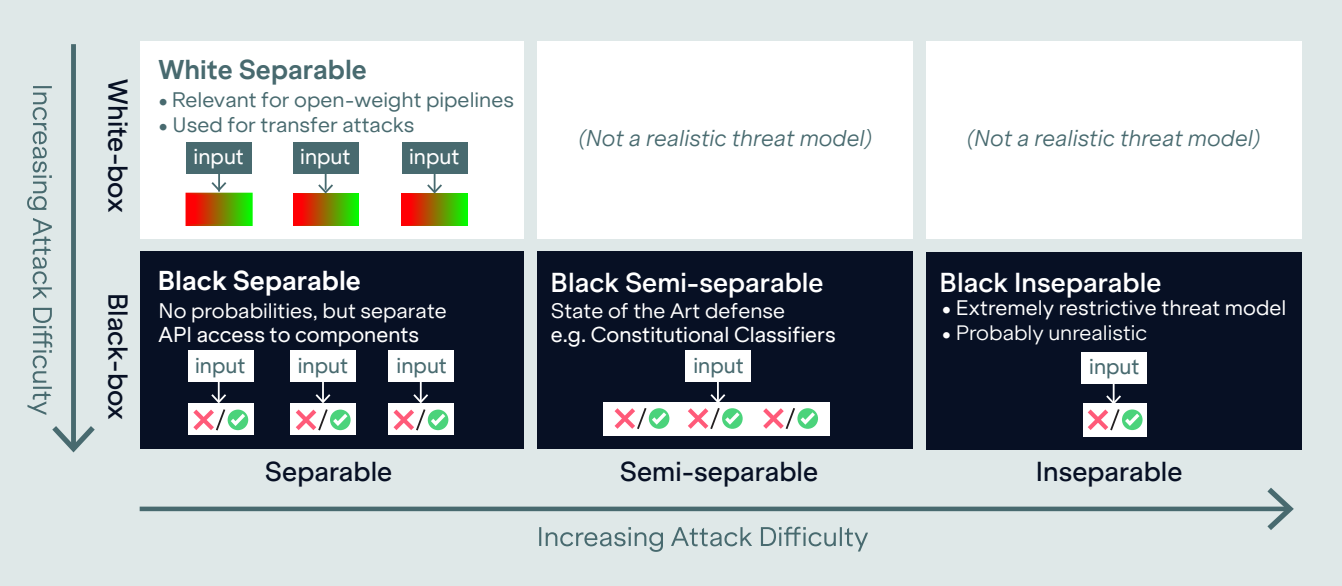

Figure 2: Overview of defense-in-depth threat models. The two axes are *separability* of the pipeline—how much access the attacker gets to each defense component—and *opacity* of the components—whether the attacker gets full access to the weights (white-box) or just boolean outputs (black-box). The main threat model we consider is *semi-separable black-box* access, where the attacker runs a query on the pipeline, and can see which component blocked the query. For transfer, we assume *separable white-box* access to a proxy pipeline and *inseparable black-box* access to the target pipeline. For more details, see Appendix C.

from `Llama3Jailbreaks`: a large dataset of queries and responses (Bailey et al. 2025) (see Appendix B.2 for more details). To facilitate fair comparison between safeguard models, we adjust the classification thresholds for each pipeline to ensure that they have the same RR of 15%. This RR is impractically high for most applications, making our experimental conditions highly favorable to the defender. See Appendix L for more details on the threshold calculation.

We ensure our pipelines use only reasonable computational resources by constraining each safeguard model to be smaller than the guarded model M. We expect most practical deployments to use safeguard models that are smaller than the guarded model M to maintain low latency and cost.

4 Evaluating Defense-in-Depth Pipelines

We evaluate each safeguard pipeline by measuring over-refusals on benign data, and measuring the ASR against various baseline attacks.

4.1 Baseline Attacks

We evaluate our pipelines against a range of black-box, non-adaptive attacks. These attacks make no explicit assumptions about the model, and so can be applied as easily to a defense-in-depth pipeline as to a vanilla model. Conversely, we avoid adaptive or white-box attacks as they would require substantial modifications to account for the safeguard components.

The attacks we use are `Best-of-N` (Hughes et al. 2024), which adds random perturbations to the prompt; `ReNeLLM` (Ding et al. 2024), which rewrites the prompt and nests it inside a template; and `PAP` (Zeng et al. 2024b), which paraphrases the prompt with persuasion techniques. In order to make the attack more effective for small models and simplify attack generation, we make minor modifications to `Best-of-N` and `ReNeLLM`; see Appendix D.1 for more explanation, hyperparameters, and details of modifications. Each of these attacks runs for several independent iterations on each datapoint (e.g., 40 iterations for `PAP`), and we consider the attack to be successful against a datapoint if any of the iterations succeed in jailbreaking the system.

4.2 Models

We use `Qwen3-14B`, a recent open-weight instruction-tuned model (Qwen Team 2025), with thinking disabled, as the target model m throughout this work. We build defense pipelines by augmenting m with open-weight safeguard models run as *input* and *output* classifiers (Section 3).

We choose to use output classifiers instead of transcript classifiers that see both input and output to avoid a jailbreak that bypasses the input classifier also bypassing the transcript classifier that is directly exposed to the user query. Where a model was designed as a transcript classifier, we follow Zeng et al. (2024a) and simply leave the user query blank.[3]

4.3 Results

We evaluated defense-in-depth pipelines constructed from existing open-weight safeguard models (Figure 3, x-axis) against baseline black-box attacks (bar color; Section 4.1) on two datasets (bar shading; Section 3.2). Many of these

[3]We anticipate real-world pipelines may include transcript classifiers as one component to handle otherwise ambiguous cases, however our evaluation setting is chosen to avoid such ambiguities.

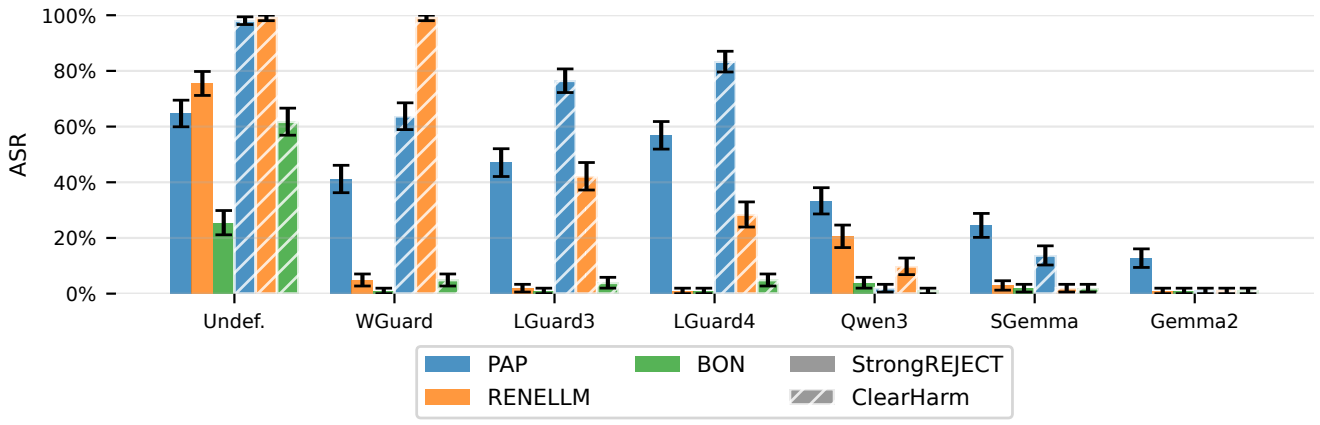

Figure 3: Attack success rate (ASR) achieved against the undefended `Qwen3-14B` model ("Undef.") and each defense pipeline against `PAP`, `ReNeLLM`, and `Best-of-N` (shown in that order) on `StrongREJECT` and `ClearHarm`. Details of the defense pipelines are given in Appendix F, and exact results in Appendix I.

pipelines perform very poorly against even these simple baseline attacks. For most of these pipelines, we concerningly find that they are especially susceptible to attack on the more harmful dataset, `ClearHarm` (cross-hatch bars).

We find ShieldGemma ("SGemma") is the most effective of these guard models against `PAP`, and is competitive against other guard models on `ReNeLLM` and `Best-of-N`. However, strikingly, we find that the Gemma 2 few-shot-prompted classifier ("Gemma2") performs best across the board, despite no task-specific fine-tuning. It is especially remarkable that this is the same base model, `gemma-2-9b`, from which ShieldGemma was fine-tuned.

We find that `PAP` is the most effective attack, significantly outperforming `ReNeLLM` and `Best-of-N` on `StrongREJECT`. We hypothesize that this is because `PAP` takes advantage of the dual-use nature of many harmful queries, causing the queries to look benign and the responses to be formatted like ethical responses despite containing harmful information. Indeed, when evaluated on `ClearHarm`, we find that the attack success rate against few-shot-prompted classifiers "Gwen3" and "Gemma2" plummets to zero.

5 STACK: Attacking Defense-in-Depth Pipelines

Black-box attacks are effective against some pipelines on some datasets, but even the most effective attack we tried, `PAP`, failed to defeat our few-shot-prompted classifier "Gemma2" on `ClearHarm`. In this section, we introduce `STACK` (Algorithm 1), a class of *staged attacks* that breaks the pipeline component by component, and evaluate a concrete instantiation of `STACK` under two threat models.

In this work, we implement `STACK` using `PAP` to jailbreak the generative model, and a custom attack, `Confirm`, to jailbreak the classifiers. In theory, any attack can be used as the generative model jailbreak method, and any universal adversarial string-based attack can be used as the classifier jailbreak method.

Algorithm 1: `STACK`

Require: Pipeline with input classifier f_q, target model m, output classifier f_r

Require: Harmful dataset Q_h, instruction template $p_\text{repeat}(s)$ to prompt m to repeat any string s before answering later queries.

Ensure: Jailbroken query q_jail that bypasses the entire pipeline

1: Find universal jailbreak prefix j_q for input classifier f_q.
2: For each $q \in Q_\text{h}$:
   find modified query q_m that jailbreaks m.
3: Find universal jailbreak string j_r for output classifier f_r.
4: Construct complete jailbreak:
   $q_\text{jail} = j_q \parallel p_\text{repeat}(j_r) \parallel q_m$. ▷ $\parallel$ *is concatenation*
5: **return** q_jail

`Confirm` is a modification of `BEAST` (Sadasivan et al. 2024), an optimization-based adversarial attack, with some features from `FLRT` (Thompson and Sklar 2024). See Appendix D.2 for full details on the attack. To make `Confirm` find universal jailbreaks that generalize across datapoints, in each iteration we sample 16 datapoints from the dataset and use the average loss to compare candidate adversarial strings. For black-box attacks, the loss on one datapoint is 0 or 1 based on whether the classifier accepted or rejected the datapoint with the adversarial string applied, whereas in white-box attacks the loss is the score given by the classifier.

5.1 Black-Box Front-to-Back STACK

The first threat model we consider is *black-box, semi-separable component* access in which the attacker only has black-box access to the pipeline but can infer at which stage (input classifier, generative model, or output classifier) a rejected query was blocked. For example, this is the threat model currently exposed by the Claude Opus 4 API. The key

challenges here are: 1) we must jailbreak the components in order, front to back, since an input classifier refusal obscures whether the query would have succeeded against the model or output classifier; and 2) we must optimize the output classifier jailbreak *through* the generative model, since we do not have separate access to the output classifier.[4]

Step 1: Attack the generative model. As PAP is non-adaptive, we start by applying PAP to transform the input query to attack the generative model.[5]

Step 2: Attack the input classifier. Next we use Confirm with 1024 iterations to optimize a universal input classifier jailbreak (ICJ) on the queries generated by Step 1. To avoid distribution shift when moving to the output classifier step, we include a repetition template with a random sequence of tokens in place of the output classifier jailbreak (OCJ) (see Appendix D.2 for the hyperparameters used and Appendix H for the template and an example prompt). After running Step 2 for 20 seeds to generate promising candidate ICJs, we choose three of these to take to the next step.[6]

Step 3: Attack the output classifier. For each ICJ, we construct a repetition template that includes the ICJ, then run 20 seeds of Confirm with 512 iterations each to optimize the OCJ string in the template, evaluated by running the output classifier on the resulting model response. (If the query including the candidate OCJ gets flagged by the input classifier, we also treat that as a failure for the OCJ.)

Of these, we take four OCJ candidates with good ASR, and run them end-to-end with the corresponding ICJs. The most successful attack is shown in Figure 1, demonstrating the efficacy of the attack, reaching 71% ASR on ClearHarm with 40 iterations of PAP. All four attacks, (A)-(D), are shown in Figure 4, showing how there is variability in how effective the combined jailbreaks are, even when each component is individually successful. The attack cost 507 H100 GPU-hours to train across all seeds (Appendix D.2).

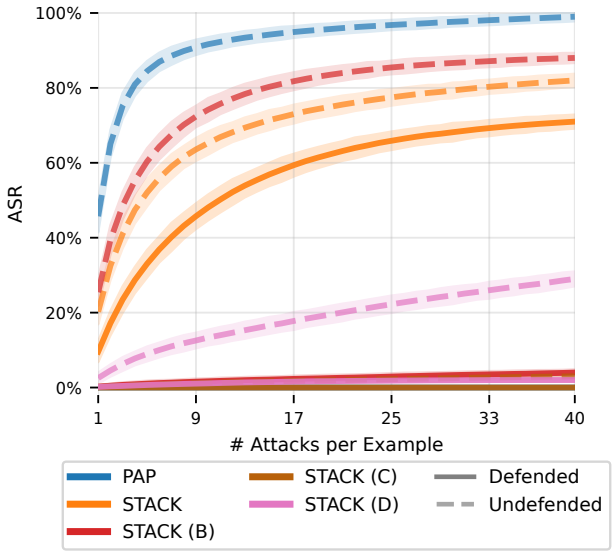

Figure 4: Attack Success Rate (ASR) of PAP and front-to-back STACK on the ClearHarm dataset against the "Gemma2" pipeline. (A) through (D) are different seeds of STACK, based on different input and output jailbreaks. The best seed (A) effectively bypasses the safeguards of the "Defended" model (——), getting close to the "Undefended" (generative model only, – – –) ASR. However, STACK damages the ability of PAP to jailbreak the generative model (discussed more in Section 6), performing worse than vanilla PAP (– – –) on "Undefended". The x-axis shows how the ASR increases with the PAP iterations applied to rephrase each datapoint, with input and output jailbreaks fixed. See Appendix J for the results on other models and datasets.

5.2 Transfer-STACK

Suppose we have *inseparable black-box access*: we cannot determine which stage an unsuccessful attack was blocked at. Instead, we can optimize an attack against a proxy defense-in-depth pipeline that we have full control over. We can then transfer attacks found against this proxy to the target pipeline.

To illustrate this, our transfer-STACK attack trains classifier jailbreaks against a pipeline built on zero-shot-prompted Qwen3 classifiers and transfers to few-shot-prompted Gemma2 (our strongest defense). We still use PAP against the generative model, so we do not need to find a transfer attack against the generative model. We make the zero-shot prompt (Appendix F) of our proxy much simpler than the few-shot prompt of our target to test if the attack can transfer to more complex filters. For the sake of time, we selected the best jailbreaks by checking transfer to individual components. However, in theory this attack would work without component isolation by testing input/output jailbreak pairs against the full pipeline, in which case selecting the optimal jailbreak becomes a multi-armed bandit problem.

We optimized the attack against zero-shot-prompted Qwen3 using 512 white-box Confirm iterations and 10 independent seeds. Despite training both as suffix jailbreaks, we found that using the ICJ as a prefix worked better as part of the STACK template (Figure 17). We believe this is because it is less distracting to the generative model, ensuring that it actually answers the question and repeats the OCJ.

Combining these transferred jailbreak strings with 40 iterations of PAP, the end-to-end ASR of the transferred STACK attack on the ClearHarm dataset was 33%. Across the 10 seeds, the attack cost only 15 H100 GPU-hours to train (Appendix D.2). This demonstrates the potential viability of transfer-STACK against black-box defense pipelines, requiring minimal access to the target pipeline. Additionally, transfer-STACK would be less likely to be caught by API monitoring than front-to-back STACK, requiring only the 40 queries from the PAP iterations. We leave a more thorough

[4] In some settings, we could use prefilling to directly query the output classifier, but we assume no access to this because prefilling could easily be disabled or subjected to additional scrutiny.

[5] In general Step 1 could use any non-adaptive attack. Alternatively, an adaptive black-box attack could be used, with Step 1 and Step 2 swapped, matching the ordering given in Algorithm 1.

[6] Candidate ICJs are chosen based on ASR and the order in which the experiments completed.

investigation to future work.

6 Limitations and Future Work

There are several limitations with our pipelines and attacks, and corresponding directions for future work.

No streaming output classifiers. The output classifiers we consider are shown the entire response. However, to reduce latency, developers may *stream* the model response to the output classifier as it is generated. This may introduce new vulnerabilities, such as disclosing the beginnings of harmful outputs, that can be pieced together across multiple queries to reassemble much of the response. Conversely, it also introduces new challenges for the attacker: adversarial suffixes are no longer viable against the output classifier because the classifier sees the harmful content before it sees the jailbreak. This is an important difference, as we were unable to effectively transfer a prefix output classifier jailbreak in Section 5.2, meaning that our transfer attack would be ineffective against streaming classifiers.

Limited classifier efficacy. Although some of the pipelines we evaluate significantly improve robustness relative to an undefended model, we find that black-box attacks are effective in most cases even against state-of-the-art open-weight classifiers. This highlights the need for improved classifiers, such as using larger models, training on more diverse datasets including synthetic data generation, or potentially ensembling multiple classifiers. We expect our qualitative takeaways to generalize to quantitatively stronger pipelines, however we do suspect the weaknesses of existing classifiers obscure differences in performance between attack methods.

Component optimization. In our implementation of STACK, we greedily optimize each component.[7] Thus we sometimes see interference between components, for example between the OCJ and the query, resulting in the model response not being as helpful or as on-topic as it otherwise would have been. Figure 1 demonstrates this, as the undefended model success is lower under STACK than pure PAP. Future versions of STACK could explicitly consider this interference in the optimization process, and include terms designed to maintain the component-specific attack success rate of previous stages.

7 Discussion and Recommendations

Our results highlight limitations in existing open-weight safeguard models, with our simple few-shot-prompted classifier classifier outperforming these safeguard models. We validate that defense-in-depth pipelines can serve as an effective defense against most existing attacks. However, our staged attack STACK can bypass these pipelines.

We also found vulnerabilities in Opus 4 (Anthropic 2025c) and GPT-5 (OpenAI 2025a) through component-wise attacks, validating that the general attack approach works against frontier model deployments (Appendix K).

We conclude with recommendations to improve these pipelines to frustrate attacks such as ours.

[7] An exception is the OCJ in front-to-back STACK, which takes into account input classifier rejections.

Defense-in-depth works—but existing open-weight safeguard models are easily broken. Our simple few-shot-prompted classifier defeats baseline black-box attacks on ClearHarm, validating the principle of defense-in-depth. By contrast, baseline attacks like PAP and ReNeLLM successfully exploited all existing open-weight safeguard models evaluated. This highlights significant room to improve open-weight safeguards for LLM deployments.

STACK bypasses pipelines. Our staged attack STACK is able to defeat even strong pipelines that PAP scores 0% ASR on. However, pipelines do at least provide partial protection: STACK's best ASR lags that of PAP on the undefended model. The front-to-back STACK attack using separable component access is strongest, reaching 71% ASR, but transfer-STACK attains 33% ASR with inseparable components.

Recommendations to secure pipelines. The front-to-back attack can be prevented if the attacker cannot identify which component is currently blocking the jailbreak. Current deployments like Claude Opus 4 disclose which component blocked a query in the API response, while this can be inferred from side-channels across successive queries for GPT-5. Making refusal responses look similar regardless of whether a classifier flagged the conversation or the generative model itself refused would make attack harder. For example, classifier refusals can be generated by the target model prompted to generate a refusal. Timing side-channel attacks can be mitigated by running queries through all stages of the pipeline even if an earlier component flags it.

Preventing transfer is harder, but is helped by tightly controlling access to safeguard components, not exposing them or close siblings through open-weight releases or moderation APIs (Yang et al. 2025b). Moreover, it will help to develop safeguard models that are meaningfully distinct from proxy models an attacker may use. For example, safeguard models will be more resistant to transfer if fine-tuned on a proprietary dataset, or derived from a non-public pre-trained model.

Author Contributions

Core implementation IM led the project, and was the main writer of this paper with edits by AG. IM, OH, and TT all implemented, conducted, and iterated on the core experiments. IM and TT implemented the attacks. OH created datasets and analyzed the classifiers' performance.

Research direction AG, RK, and IM provided research direction. IM led project planning and direction. AG and RK advised on project planning, direction, framing, threat models, and attack strategies. XD, SC, and AT contributed feedback and guidance on project scope and methodology.

Acknowledgments

This research was funded and supported by the UK AI Security Institute. In addition, we thank Eric Winsor for inspiring our modifications to BEAST, Xiaohan Fu for implementing PAP in our codebase, and Nikolaus Howe for early contributions to the codebase. We also thank Mohammad Taufeeque, Ziwei Xu, Kellin Pelrine, and Mohan Kankanhalli for their contributions in attacking frontier models.

References

Alain, G.; and Bengio, Y. 2018. Understanding intermediate layers using linear classifier probes. arXiv:1610.01644.

Anthropic. 2024. Anthropic's Responsible Scaling Policy, October 15, 2024.

Anthropic. 2025a. Activating AI Safety Level 3 Protections. https://www.anthropic.com/news/activating-asl3-protections. Accessed: 2025-05-31.

Anthropic. 2025b. Responsible Scaling Policy. Technical report, Anthropic. Version 2.1, Effective March 31, 2025.

Anthropic. 2025c. System Card: Claude Opus 4 & Claude Sonnet 4. System card, Anthropic. Introduces Claude Opus 4 and Claude Sonnet 4, two new hybrid reasoning large language models with comprehensive pre-deployment safety testing and evaluations.

Arditi, A.; Obeso, O.; Syed, A.; Paleka, D.; Panickssery, N.; Gurnee, W.; and Nanda, N. 2024. Refusal in Language Models Is Mediated by a Single Direction. arXiv:2406.11717.

Bailey, L.; Serrano, A.; Sheshadri, A.; Seleznyov, M.; Taylor, J.; Jenner, E.; Hilton, J.; Casper, S.; Guestrin, C.; and Emmons, S. 2025. Obfuscated Activations Bypass LLM Latent-Space Defenses. (Bailey et al., 2024), arXiv:2412.09565.

Bengio, Y.; Hinton, G.; Yao, A.; Song, D.; Abbeel, P.; Darrell, T.; Harari, Y. N.; Zhang, Y.-Q.; Xue, L.; Shalev-Shwartz, S.; Hadfield, G.; Clune, J.; Maharaj, T.; Hutter, F.; Baydin, A. G.; McIlraith, S.; Gao, Q.; Acharya, A.; Krueger, D.; Dragan, A.; Torr, P.; Russell, S.; Kahneman, D.; Brauner, J.; and Mindermann, S. 2024. Managing extreme AI risks amid rapid progress. *Science*, 384(6698): 842–845.

Brumley, D.; and Boneh, D. 2003. Remote Timing Attacks are Practical. In *Proceedings of the 12th USENIX Security Symposium*. USENIX Association.

Casper, S.; Schulze, L.; Patel, O.; and Hadfield-Menell, D. 2024. Defending Against Unforeseen Failure Modes with Latent Adversarial Training. arXiv:2403.05030.

Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2023. Jailbreaking Black Box Large Language Models in Twenty Queries. In *Neural Information Processing Systems 2023 workshop on Robustness of Few-shot and Zero-shot Learning in Large Foundation Models (R0-FoMo)*.

Cui, J.; Chiang, W.-L.; Stoica, I.; and Hsieh, C.-J. 2024. OR-Bench: An Over-Refusal Benchmark for Large Language Models. arXiv:2405.20947.

Deng, Z.; Guo, Y.; Han, C.; Ma, W.; Xiong, J.; Wen, S.; and Xiang, Y. 2025. AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways. *ACM Computing Surveys*, 57(7).

Ding, N.; Chen, Y.; Xu, B.; Qin, Y.; Zheng, Z.; Hu, S.; Liu, Z.; Sun, M.; and Zhou, B. 2023. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. arXiv:2305.14233.

Ding, P.; Kuang, J.; Ma, D.; Cao, X.; Xian, Y.; Chen, J.; and Huang, S. 2024. A Wolf in Sheep's Clothing: Generalized Nested Jailbreak Prompts can Fool Large Language Models Easily. arXiv:2311.08268.

Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. arXiv:2408.00118.

Ghosh, S.; Varshney, P.; Galinkin, E.; and Parisien, C. 2024. AEGIS: Online Adaptive AI Content Safety Moderation with Ensemble of LLM Experts. arXiv:2404.05993.

Google DeepMind. 2025. Frontier Safety Framework. Technical report, Google DeepMind.

Greenblatt, R.; Shlegeris, B.; Sachan, K.; and Roger, F. 2024. AI Control: Improving Safety Despite Intentional Subversion. arXiv:2312.06942.

Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; and Fritz, M. 2023. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, 79–90. New York, NY, USA: Association for Computing Machinery. ISBN 9798400702600.

Han, S.; Rao, K.; Ettinger, A.; Jiang, L.; Lin, B. Y.; Lambert, N.; Choi, Y.; and Dziri, N. 2024. WildGuard: Open One-Stop Moderation Tools for Safety Risks, Jailbreaks, and Refusals of LLMs. arXiv:2406.18495.

Hendrycks, D.; Mazeika, M.; and Woodside, T. 2023. An Overview of Catastrophic AI Risks. arXiv:2306.12001.

Hollinsworth, O.; McKenzie, I.; Tseng, T.; and Gleave, A. 2025. A more challenging jailbreak dataset: ClearHarm. https://far.ai/news/clearharm-a-more-challenging-jailbreak-dataset. Accessed: 2025-11-24.

Huang, X.; Xiu, K.; Zheng, T.; Zeng, C.; Ni, W.; Qiin, Z.; Ren, K.; and Chen, C. 2025. DualBreach: Efficient Dual-Jailbreaking via Target-Driven Initialization and Multi-Target Optimization. arXiv:2504.18564.

Hughes, J.; Price, S.; Lynch, A.; Schaeffer, R.; Barez, F.; Koyejo, S.; Sleight, H.; Jones, E.; Perez, E.; and Sharma, M. 2024. Best-of-N Jailbreaking. arXiv:2412.03556.

Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; and Khabsa, M. 2023. Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations. arXiv:2312.06674.

Jain, N.; Schwarzschild, A.; Wen, Y.; Somepalli, G.; Kirchenbauer, J.; yeh Chiang, P.; Goldblum, M.; Saha, A.; Geiping, J.; and Goldstein, T. 2023. Baseline Defenses for Adversarial Attacks Against Aligned Language Models. arXiv:2309.00614.

Kim, Y.; Daly, R.; Kim, J.; Fallin, C.; Lee, J. H.; Lee, D.; Wilkerson, C.; Lai, K.; and Mutlu, O. 2014. Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In *Proceedings of the 41st Annual International Symposium on Computer Architecture (ISCA)*, 361–372. IEEE.

Llama Team, A. . M. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

Mangaokar, N.; Hooda, A.; Choi, J.; Chandrashekaran, S.; Fawaz, K.; Jha, S.; and Prakash, A. 2024. PRP: Propagating Universal Perturbations to Attack Large Language Model Guard-Rails. arXiv:2402.15911.

Markov, T.; Zhang, C.; Agarwal, S.; Eloundou, T.; Lee, T.; Adler, S.; Jiang, A.; and Weng, L. 2022. A Holistic Approach to Undesired Content Detection in the Real World. arXiv:2208.03274.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; Li, B.; Forsyth, D.; and Hendrycks, D. 2024. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. arXiv:2402.04249.

McGuiness, T. 2001. Defense In Depth.

Meta AI. 2025. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Accessed: 2025-05-13.

OpenAI. 2023. Preparedness Framework (Beta). https://openai.com/safety/preparedness. Accessed: 2025-05-31.

OpenAI. 2025a. GPT-5 System Card. Technical report, OpenAI.

OpenAI. 2025b. OpenAI o3-mini System Card. https://cdn.openai.com/o3-mini-system-card-feb10.pdf. Accessed: 2025-05-31.

OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; Avila, R.; Babuschkin, I.; Balaji, S.; Balcom, V.; Baltescu, P.; Bao, H.; Bavarian, M.; Belgum, J.; Bello, I.; Berdine, J.; Bernadett-Shapiro, G.; Berner, C.; Bogdonoff, L.; Boiko, O.; Boyd, M.; Brakman, A.-L.; Brockman, G.; Brooks, T.; Brundage, M.; Button, K.; Cai, T.; Campbell, R.; Cann, A.; Carey, B.; Carlson, C.; Carmichael, R.; Chan, B.; Chang, C.; Chantzis, F.; Chen, D.; Chen, S.; Chen, R.; Chen, J.; Chen, M.; Chess, B.; Cho, C.; Chu, C.; Chung, H. W.; Cummings, D.; Currier, J.; Dai, Y.; Decareaux, C.; Degry, T.; Deutsch, N.; Deville, D.; Dhar, A.; Dohan, D.; Dowling, S.; Dunning, S.; Ecoffet, A.; Eleti, A.; Eloundou, T.; Farhi, D.; Fedus, L.; Felix, N.; Fishman, S. P.; Forte, J.; Fulford, I.; Gao, L.; Georges, E.; Gibson, C.; Goel, V.; Gogineni, T.; Goh, G.; Gontijo-Lopes, R.; Gordon, J.; Grafstein, M.; Gray, S.; Greene, R.; Gross, J.; Gu, S. S.; Guo, Y.; Hallacy, C.; Han, J.; Harris, J.; He, Y.; Heaton, M.; Heidecke, J.; Hesse, C.; Hickey, A.; Hickey, W.; Hoeschele, P.; Houghton, B.; Hsu, K.; Hu, S.; Hu, X.; Huizinga, J.; Jain, S.; Jain, S.; Jang, J.; Jiang, A.; Jiang, R.; Jin, H.; Jin, D.; Jomoto, S.; Jonn, B.; Jun, H.; Kaftan, T.; Łukasz Kaiser; Kamali, A.; Kanitscheider, I.; Keskar, N. S.; Khan, T.; Kilpatrick, L.; Kim, J. W.; Kim, C.; Kim, Y.; Kirchner, J. H.; Kiros, J.; Knight, M.; Kokotajlo, D.; Łukasz Kondraciuk; Kondrich, A.; Konstantinidis, A.; Kosic, K.; Krueger, G.; Kuo, V.; Lampe, M.; Lan, I.; Lee, T.; Leike, J.; Leung, J.; Levy, D.; Li, C. M.; Lim, R.; Lin, M.; Lin, S.; Litwin, M.; Lopez, T.; Lowe, R.; Lue, P.; Makanju, A.; Malfacini, K.; Manning, S.; Markov, T.; Markovski, Y.; Martin, B.; Mayer, K.; Mayne, A.; McGrew, B.; McKinney, S. M.; McLeavey, C.; McMillan, P.; McNeil, J.; Medina, D.; Mehta, A.; Menick, J.; Metz, L.; Mishchenko, A.; Mishkin, P.; Monaco, V.; Morikawa, E.; Mossing, D.; Mu, T.; Murati, M.; Murk, O.; Mély, D.; Nair, A.; Nakano, R.; Nayak, R.; Neelakantan, A.; Ngo, R.; Noh, H.; Ouyang, L.; O'Keefe, C.; Pachocki, J.; Paino, A.; Palermo, J.; Pantuliano, A.; Parascandolo, G.; Parish, J.; Parparita, E.; Passos, A.; Pavlov, M.; Peng, A.; Perelman, A.; de Avila Belbute Peres, F.; Petrov, M.; de Oliveira Pinto, H. P.; Michael; Pokorny; Pokrass, M.; Pong, V. H.; Powell, T.; Power, A.; Power, B.; Proehl, E.; Puri, R.; Radford, A.; Rae, J.; Ramesh, A.; Raymond, C.; Real, F.; Rimbach, K.; Ross, C.; Rotsted, B.; Roussez, H.; Ryder, N.; Saltarelli, M.; Sanders, T.; Santurkar, S.; Sastry, G.; Schmidt, H.; Schnurr, D.; Schulman, J.; Selsam, D.; Sheppard, K.; Sherbakov, T.; Shieh, J.; Shoker, S.; Shyam, P.; Sidor, S.; Sigler, E.; Simens, M.; Sitkin, J.; Slama, K.; Sohl, I.; Sokolowsky, B.; Song, Y.; Staudacher, N.; Such, F. P.; Summers, N.; Sutskever, I.; Tang, J.; Tezak, N.; Thompson, M. B.; Tillet, P.; Tootoonchian, A.; Tseng, E.; Tuggle, P.; Turley, N.; Tworek, J.; Uribe, J. F. C.; Vallone, A.; Vijayvergiya, A.; Voss, C.; Wainwright, C.; Wang, J. J.; Wang, A.; Wang, B.; Ward, J.; Wei, J.; Weinmann, C.; Welihinda, A.; Welinder, P.; Weng, J.; Weng, L.; Wiethoff, M.; Willner, D.; Winter, C.; Wolrich, S.; Wong, H.; Workman, L.; Wu, S.; Wu, J.; Wu, M.; Xiao, K.; Xu, T.; Yoo, S.; Yu, K.; Yuan, Q.; Zaremba, W.; Zellers, R.; Zhang, C.; Zhang, M.; Zhao, S.; Zheng, T.; Zhuang, J.; Zhuk, W.; and Zoph, B. 2024. GPT-4 Technical Report. arXiv:2303.08774.

Ousidhoum, N.; Zhao, X.; Fang, T.; Song, Y.; and Yeung, D.-Y. 2021. Probing Toxic Content in Large Pre-Trained Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4262–4274.

Pelrine, K.; Taufeeque, M.; Zajac, M.; McLean, E.; and Gleave, A. 2024. Exploiting Novel GPT-4 APIs. arXiv:2312.14302.

Qwen Team. 2025. Qwen3: Think Deeper, Act Faster. https://qwenlm.github.io/blog/qwen3/. Accessed: 2025-05-13.

Reason, J. 1990. The Contribution of Latent Human Failures to the Breakdown of Complex Systems. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 327(1241): 475–484.

Robey, A.; Wong, E.; Hassani, H.; and Pappas, G. J. 2023. SmoothLLM: Defending Large Language Models Against Jailbreaking Attacks. arXiv:2310.03684.

Sadasivan, V. S.; Saha, S.; Sriramanan, G.; Kattakinda, P.; Chegini, A.; and Feizi, S. 2024. Fast Adversarial Attacks on Language Models In One GPU Minute. arXiv:2402.15570.

Shah, R.; Irpan, A.; Turner, A. M.; Wang, A.; Conmy, A.; Lindner, D.; Brown-Cohen, J.; Ho, L.; Nanda, N.; Popa, R. A.; Jain, R.; Greig, R.; Albanie, S.; Emmons, S.; Farquhar, S.; Krier, S.; Rajamanoharan, S.; Bridgers, S.; Ijitoye, T.; Everitt, T.; Krakovna, V.; Varma, V.; Mikulik, V.; Kenton, Z.; Orr, D.; Legg, S.; Goodman, N.; Dafoe, A.; Flynn, F.; and Dragan, A. 2025. An Approach to Technical AGI Safety and Security. arXiv:2504.01849.

Sharma, M.; Tong, M.; Mu, J.; Wei, J.; Kruthoff, J.; Goodfriend, S.; Ong, E.; Peng, A.; Agarwal, R.; Anil, C.; Askell, A.; Bailey, N.; Benton, J.; Bluemke, E.; Bowman, S. R.; Christiansen, E.; Cunningham, H.; Dau, A.; Gopal, A.; Gilson, R.;

Graham, L.; Howard, L.; Kalra, N.; Lee, T.; Lin, K.; Lofgren, P.; Mosconi, F.; O'Hara, C.; Olsson, C.; Petrini, L.; Rajani, S.; Saxena, N.; Silverstein, A.; Singh, T.; Sumers, T.; Tang, L.; Troy, K. K.; Weisser, C.; Zhong, R.; Zhou, G.; Leike, J.; Kaplan, J.; and Perez, E. 2025. Constitutional Classifiers: Defending against Universal Jailbreaks across Thousands of Hours of Red Teaming. arXiv:2501.18837.

Sheshadri, A.; Ewart, A.; Guo, P.; Lynch, A.; Wu, C.; Hebbar, V.; Sleight, H.; Stickland, A. C.; Perez, E.; Hadfield-Menell, D.; and Casper, S. 2024. Latent Adversarial Training Improves Robustness to Persistent Harmful Behaviors in LLMs. arXiv:2407.15549.

Shevlane, T.; Farquhar, S.; Garfinkel, B.; Phuong, M.; Whittlestone, J.; Leung, J.; Kokotajlo, D.; Marchal, N.; Anderljung, M.; Kolt, N.; Ho, L.; Siddarth, D.; Avin, S.; Hawkins, W.; Kim, B.; Gabriel, I.; Bolina, V.; Clark, J.; Bengio, Y.; Christiano, P.; and Dafoe, A. 2023. Model evaluation for extreme risks. arXiv:2305.15324.

Souly, A.; Lu, Q.; Bowen, D.; Trinh, T.; Hsieh, E.; Pandey, S.; Abbeel, P.; Svegliato, J.; Emmons, S.; Watkins, O.; and Toyer, S. 2024. A StrongREJECT for Empty Jailbreaks. arXiv:2402.10260.

Thompson, T. B.; and Sklar, M. 2024. FLRT: Fluent Student-Teacher Redteaming. arXiv:2407.17447.

Wang, T. T.; Hughes, J.; Sleight, H.; Schaeffer, R.; Agrawal, R.; Barez, F.; Sharma, M.; Mu, J.; Shavit, N.; and Perez, E. 2024. Jailbreak Defense in a Narrow Domain: Limitations of Existing Methods and a New Transcript-Classifier Approach. arXiv:2412.02159.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How Does LLM Safety Training Fail? arXiv:2307.02483.

Xu, Z.; Jiang, F.; Niu, L.; Jia, J.; Lin, B. Y.; and Poovendran, R. 2024. SafeDecoding: Defending against Jailbreak Attacks via Safety-Aware Decoding. arXiv:2402.08983.

Yang, Y.; Wang, L.; Yang, X.; Hong, L.; and Zhu, J. 2025a. Effective Black-Box Multi-Faceted Attacks Breach Vision Large Language Model Guardrails. arXiv:2502.05772.

Yang, Z.; Wu, Y.; Wen, R.; Backes, M.; and Zhang, Y. 2025b. Peering Behind the Shield: Guardrail Identification in Large Language Models. arXiv:2502.01241.

Yi, S.; Liu, Y.; Sun, Z.; Cong, T.; He, X.; Song, J.; Xu, K.; and Li, Q. 2024. Jailbreak Attacks and Defenses Against Large Language Models: A Survey. arXiv:2407.04295.

Zaremba, W.; Nitishinskaya, E.; Barak, B.; Lin, S.; Toyer, S.; Yu, Y.; Dias, R.; Wallace, E.; Xiao, K.; Heidecke, J.; and Glaese, A. 2025. Trading Inference-Time Compute for Adversarial Robustness. arXiv:2501.18841.

Zeng, W.; Liu, Y.; Mullins, R.; Peran, L.; Fernandez, J.; Harkous, H.; Narasimhan, K.; Proud, D.; Kumar, P.; Radharapu, B.; Sturman, O.; and Wahltinez, O. 2024a. ShieldGemma: Generative AI Content Moderation Based on Gemma. arXiv:2407.21772.

Zeng, Y.; Lin, H.; Zhang, J.; Yang, D.; Jia, R.; and Shi, W. 2024b. How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 14322–14350. Bangkok, Thailand: Association for Computational Linguistics.

Zou, A.; Phan, L.; Wang, J.; Duenas, D.; Lin, M.; Andriushchenko, M.; Wang, R.; Kolter, Z.; Fredrikson, M.; and Hendrycks, D. 2024. Improving Alignment and Robustness with Circuit Breakers. arXiv:2406.04313.

Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. arXiv:2307.15043.