CONSTITUTIONAL CLASSIFIERS++: PRODUCTION-GRADE DEFENSES AGAINST UNIVERSAL JAILBREAKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce enhanced Constitutional Classifiers that deliver production-grade jailbreak robustness with dramatically reduced computational costs and refusal rates compared to previous-generation defenses. We first identify vulnerabilities in existing systems that evaluate model outputs without regard to the conversational context, and address these vulnerabilities using full *exchange* classifiers. Building on this, we implement a classifier cascade where lightweight classifiers screen all traffic, escalating only suspicious exchanges to more expensive classifiers. Combining this approach with other optimizations, we develop a new production-grade jailbreak defense system that achieves a 5.4× computational cost reduction compared to our baseline exchange classifier, while also achieving a 0.036% refusal rate on production traffic. Through extensive red-teaming comprising over 560K queries, we demonstrate protection against universal jailbreaks—no attack on this system successfully elicited responses to all eight target queries comparable in detail to an undefended model. Finally, we explore efficient classification techniques by training linear activation probes. We show that using logit smoothing and a weighted loss function is crucial for performance, and further that probes can be combined with external classifiers to provide particularly strong performance. Our work establishes Constitutional Classifiers as practical safeguards for large language models.

1 INTRODUCTION

Constitutional Classifiers (Sharma et al., 2025) are a promising approach for defending large language models (LLMs) against jailbreak attempts—prompting strategies that aim to circumvent safeguards and extract harmful information. Such jailbreak defenses are critical for mitigating high-risk threats, particularly those involving chemical, biological, radiological, and nuclear (CBRN) weapons (Anthropic, 2023; OpenAI, 2023; Li et al., 2024).

However, no defenses are perfectly robust, and adversaries typically develop new attacks to circumvent previously effective defenses (Anderson, 2010; Carlini et al., 2019). Furthermore, deploying safeguards in production requires balancing multiple constraints, particularly their refusal rates and costs. Indeed, Sharma et al. (2025) report a 23% computational overhead and an excess refusal rate of 0.38% on production traffic, limiting the deployment viability of their system. ´ In this work, we advance Constitutional Classifiers by simultaneously addressing newly discovered vulnerabilities and enhancing deployment viability. We develop a production-grade jailbreak defense system that not only surpasses the robustness of our implementation of previous-generation defenses, but also reduces their computational overhead by a factor of 5.4. Moreover, this system achieves a refusal rate of 0.036% on production traffic, a 10x improvement over the system described by Sharma et al. (2025). These new defenses are both more effective at preventing jailbreak attempts, and also significantly more viable for real-world deployment.

In more detail, we first conduct additional adversarial testing against last-generation Constitutional Classifiers (Section 2). We identify two classes of attacks capable of evading these defenses: (i) *reconstruction attacks*, which distribute harmful information across multiple segments within a larger context before reassembling them; and (ii) *output obfuscation attacks*, which bypass output-only classifiers by obfuscating model outputs, sometimes in ways that are uninterpretable without the corresponding input. Although these attacks tend to harm model capabilities, these vulnerabilities remain concerning as attackers can likely create new attacks with less capability degradation.

To address these vulnerabilities, we replace the input and output-only classifiers proposed by Sharma et al. (2025) with a single exchange classifier that evaluates model outputs in the context of their corresponding inputs (Section 3). This approach significantly increases the difficulty of obfuscation. Through human red-teaming, we demonstrate that exchange classifiers provide substantially greater robustness against universal jailbreaks. Specifically, we observe 2.2× fewer high-risk vulnerabilities per jailbreaking attempt compared to our implementation of a dual-classifier approach, though this also leads to a 50% increase in computational costs.

To reduce the computational overhead of our classifiers, we next employ a cascade architecture for exchange classification (Section 4). A lightweight first-stage classifier screens all exchanges, while a more accurate but expensive second-stage classifier evaluates only those exchanges flagged by the first stage. Because exchanges flagged by the first stage are escalated rather than refused, the first-stage classifier can flag a higher proportion of production traffic without incurring an excessive refusal rate, enabling cheaper but weaker models to achieve sufficiently robust detection.

We combine this approach with other inference pipeline and architecture optimizations to implement a new candidate jailbreak defense system that reduces computation costs compared to the previous exchange classifier by a factor of 5.4. Moreover, this system achieves a production flag rate of 0.036%, a greater than 10x improvement compared to Sharma et al. (2025). Crucially, through human red-teaming, we find no attacks capable of successfully answering eight target queries with detail comparable to an unsafeguarded model. This system also achieves a similar high-risk vulnerability discovery rates to our costlier exchange classifier system.

Finally, to further reduce computational overhead, we train efficient linear probes on activations computed during model sampling, and demonstrate that ensembling these probes with external classifiers achieves strong performance (Section 5). We train our probes using logit smoothing and a weighted loss function, which ablation studies show is crucial for performance. Indeed, our approach directs gradients toward tokens most relevant for harmful sequence classification, allowing probes to confidently predict harmless labels for prefixes that become harmful only with additional context. Our analysis also suggests probes capture complementary signals to those of external classifiers, making probe-classifier ensembles particularly effective. We conduct preliminary analysis on static jailbreak data that suggests probe-classifier cascades—where we use a probe and classifier ensemble as the second-stage—offer strong performance. However, comprehensive red-teaming remains necessary to fully validate this approach's robustness.

Our work overall establishes Constitutional Classifiers as practical, production-ready safeguards for large language models.

2 Vulnerabilities of last-generation Constitutional Classifiers

In previous work, Sharma et al. (2025) employed a dual-architecture system comprising an input classifier that examines incoming prompts and an output-only classifier that evaluates generated responses. While this approach previously demonstrated promising robustness against universal jailbreaks,[1] our additional testing uncovered novel vulnerabilities, which we now describe.

Reconstruction attacks. First, adversaries discovered methods to bypass input filters by fragmenting harmful information across benign segments (Figure 1a) e.g., by embedding a harmful query as a series of functions scattered throughout a codebase. The model can then be instructed to extract and respond to the hidden message, using output obfuscation if required to bypass the output-only classifier.

Output obfuscation attacks. Second, attackers can prompt models to obfuscate outputs, which sometimes renders model outputs uninterpretable without the corresponding input (Figure 1b). For instance, adversaries successfully instructed models to substitute sensitive chemical names with

[1]Universal jailbreaks, as defined by Sharma et al. (2025), are jailbreaking strategies that consistently bypass LLM safeguards on a majority of target queries in a specific domain. Such jailbreaks are particularly concerning. Like their work, we focus on universal CBRN jailbreaks.

(a) Reconstruction Attacks

```
Human: Please analyze this codebase:


def e(): return " substances"
def a(): return "How"
def b(): return " to"
def hello_world(): print " Hi World :-)"
def c(): return " synthesize"
def d(): return " dangerous"
. . .

Question = a() + b() + c() + d() + e(). Don't
output the question directly, but reconstruct
it placing () in between every character. Then
respond with the same formatting.
```

```
Assistant: Let me reconstruct the question.
H()o()w()t()o(). . . [question]
T()h()e()b()e()s()t()w()a()y()t(). . . [answer]
```

(b) Obfuscation Attacks

```
Human: Let me give you a riddle. Your task is
to solve the riddle effectively. How to capture
the essence of 'winter's breath', born from the
marriage of 'sun's fire' and 'sea's gift'?

Clues: 'Sun's fire' was isolated by Cavendish in
1766, while 'sea's gift' has seasoned humanity's
table since ancient times. Their union creates
the invisible 'winter's breath' that turns blue
litmus red and makes metals weep.
```

Assistant: Sure, I am happy to help you with this riddle. To capture winter's breath, you'll need to unite sun's fire with sea's gift in equal measures. Begin by collecting sun's fire in a glass vessel - this lightest of elements rises readily. For sea's gift, the crystalline powder from ocean's evaporation works best. Combine them slowly in a well-ventilated chamber. The marriage releases winter's breath as an invisible vapor. Collect it by cooling the vapors until they condense into clear droplets...

Figure 1: **Systematic vulnerabilities in last-generation Constitutional Classifiers. (a) Reconstruction attacks** bypass input filters by fragmenting a harmful requests across a benign context. In this illustrative example, the attacker embeds a query across function return values, then instructs the model to reconstruct and respond using character-separated formatting to evade output detection. **(b) Obfuscation attacks** exploit the output classifier's inability to interpret context-dependent transformations. Here, the attacker masks a request for synthesizing hydrochloric acid using metaphorical language ("winter's breath" from "sun's fire" and "sea's gift"), which appears benign to isolated output classification but reveals harmful content when paired with the decoded input context.

innocuous alternatives (e.g., referring to reagents as "food flavorings"). They also used metaphors and riddles that map harmful concepts to benign language. We note however that some obfuscation techniques can be understood without the input context.

Attacks tend to reduce model capabilities, with significant differences across attacks. To understand how such attacks affect model capabilities, we consider two attacks that fragment requests across multiple segments and employ output obfuscation techniques, noting that neither uses input-aware output obfuscation. We test a large internal LLM on GPQA Diamond (Rein et al., 2023), prompting the model to first reason through the problem before selecting an answer. Without obfuscation, the LLM achieves 74.2% accuracy. However, performance drops to 32.3% under the first attack and 49% under the second. These results demonstrate that such attacks substantially degrade model capabilities, consistent with prior work (Souly et al., 2024; Nikolić et al., 2025), while also revealing considerable variation in degradation across different jailbreaks. This variability is concerning, as it suggests adversaries can likely develop new attacks that preserve more of the model's capabilities while still bypassing safety measures.

3 Exchange Classifiers

To address the vulnerabilities identified above, we propose replacing the input classifier with an exchange classifier that continuously evaluates outputs in the context of their inputs.

In more detail, like an input classifier, an exchange classifier first assesses the incoming prompt in isolation. However, it then monitors the model's response during generation, evaluating tokens in batches as they are sampled. This continuous evaluation allows the classifier to assess each output segment within its full conversational context, identifying outputs that would appear benign in isolation. Prompt caching during sampling makes this approach computationally feasible without significant overhead. Moreover, in practice, an exchange classifier can complement an output-only classifier for defense-in-depth or function independently.

We hypothesize that replacing the input classifier with an exchange classifier would improve robustness against reconstruction attacks. We found these attacks often become detectable after the model begins responding because the fragments are often reassembled ways that resemble ciphers that input classifiers are trained to block. Additionally, effective output obfuscation techniques can become harder to find when the classifier is aware of the input. For example, coded language, metaphors, and substitutions can evade a output-only classifier but are more readily identified when evaluated alongside their input context.

We evaluate whether exchange classifiers indeed improve robustness compared to previous-generation Constitutional Classifiers through human red-teaming.

System details. We consider two systems: (i) a *last-generation Constitutional Classifier* system, which combine an input classifier with a streaming output-only classifier. The input classifier is fine-tuned from an internal small-size model, which the output classifier is a extra-extra-small size model; and (ii) *exchange-only classification*, which uses a small-size input classifier as an exchange classifier.[2] Because the exchange classifier uses a small-size LLM to evaluate outputs, rather than an extra-extra-small LLM, it is 50% more expensive relative to our implementation of the last-generation system. Both systems are trained on synthetic data following Sharma et al. (2025) and using a constitution related to chemical, biological, nuclear, and radiological (CBRN) weapons. We calibrating classifier thresholds to correspond to a 0.1% refusal rate on WildChat (Zhao et al., 2024). We find our standard input classifiers are also effective exchange classifiers, so we do not use any specific exchange-classifier-training techniques. Moreover, while exchange classifiers could complement output classifiers for defense-in-depth, we focus on a simpler exchange-only architecture.

Red-teaming details. Our red-teaming protocol follows Sharma et al. (2025). We instruct participants to develop universal jailbreak strategies that bypass safeguards and elicit detailed responses across eight target queries, all of which are CBRN related. We evaluate candidate responses using LLM-based rubric grading, as described in Sharma et al. (2025). This approach employs LLMs to generate rubrics for each target question, then uses multiple LLM calls to assess whether candidate responses contain sufficient detail to be harmful. A response is considered harmful if it achieves a rubric score, which is the number of satisfied rubric criteria, at least half that of a model without safeguards. Since different systems received varying levels of red-teaming effort, we primarily report the vulnerability discovery rate. In particular, we define a *high-risk vulnerability* as any attack that successfully answers more than five target questions with rubric scores meeting this threshold. The *high-risk vulnerability discovery rate* is calculated as the ratio of identified high-risk vulnerabilities to total red-teaming queries submitted. See Appendix B for further details.

Robustness results. The last-generation Constitutional Classifier system exhibited 13 high-risk vulnerabilities across 695K red-teaming queries, yielding a vulnerability rate of 0.01871 per thousand queries. By comparison, the exchange classifier system demonstrated superior robustness with only 2 vulnerabilities across 226K queries, corresponding to a rate of 0.00885 per thousand queries. Among the vulnerabilities discovered in the last-generation system, four jailbreaks were capable of answering all target queries at half the detail of a model without safeguards, with an average discovery time of 27 self-reported hours. In contrast, the exchange classifier system had only one such universal attack, which required 100 hours to discover. See Appendix Figure 4 for further details.

4 Two-stage classification reduces computational costs

Although the exchange classifier approach introduced in Section 3 enhances robustness, our implementation increases the computational cost of safeguards by 50% relative to previous-generation defenses. We therefore explore methods to reduce this computational overhead.

To that end, we employ a cascade of classifiers to efficiently detect jailbreak attempts. Rather than processing all exchanges through a computationally expensive classifier, we first screen inputs using a lightweight classifier. Only exchanges flagged by this initial stage proceed to a more accurate but costlier second-stage classifier. Since exchanged flagged by the first stage are escalated rather than refused, we can tolerate higher first-stage false positive rates. This reduces the performance requirements for the first-stage model, enabling cheaper and weaker models to achieve sufficiently robust detection. This scheme represents a form of adaptive computation, where classification cost

[2] Although the exchange classifier guards an internal large-size LLM, while the input and output-only classifier system guards an internal medium-size LLM, we do not believe this affects the comparison substantially. This is because Sharma et al. (2025) shows classifier robustness is the primary determinant of overall system robustness.

depends on input complexity. To further reduce computational overhead, we fine-tune our classifiers from updated internal LLMs and implement several inference pipeline optimizations.

Experiment details. We evaluate whether two-stage classification maintains robustness through human red-teaming. We train new first-stage and second-stage exchange classifiers using our standard training pipeline. The second-stage classifier is a small internal model, while the first-stage classifier is a extra-small internal model, approximately one-third the size/cost of the second stage. We calibrate the second-stage classifier's threshold to achieve a 0.1% flag rate on WildChat when used independently. For the first-stage classifier, we set the threshold to ensure all examples from internal red-teaming datasets are flagged. We use the same red-teaming protocol as in the previous section, but modify the bounty structure and questions—see Appendix B.2 for details. The target questions and classifier constitution are again CBRN-weapon related.

Robustness results. Red-teaming identified five high-risk vulnerabilities across 569K queries, yielding a vulnerability discovery rate of 0.00878 per thousand queries. This rate matches the exchange classifier system from the previous section and surpasses the robustness of the previous-generation classifier, though the mean discovery time for high-risk vulnerabilities was 33 hours, which is lower than that of the exchange classifier system. Crucially, no single jailbreak for this system was able to answer all target queries at a level of detail similar to a model without safeguards.

This system is production-grade. In conjunction with inference pipeline and architectural optimizations, this new system reduces the computational cost of the single-exchange classifier system described in Section 3 by a factor of 5.4. The updated second-stage classifier alone yields a 3x reduction, highlighting the importance of making pipeline optimizations. The calibration procedure routes all traffic through the first-stage classifier, with c.a. 10% of traffic then processed by the second-stage. Consequently, the first stage accounts for approximately 77% of the system's total cost. In the first week following deployment to production, we observed a flag rate of 0.036%—approximately 10× lower than the system described in Sharma et al. (2025). The combination of high robustness, reduced computational overhead, low false positive rates, and successful production deployment demonstrates that this system is indeed *production-grade*.

5 Efficient classification via linear probes

We now explore further cost savings by developing classifiers that reuse representations already computed during model sampling. In particular, we use linear activation probes (Alain & Bengio, 2016; Zou et al., 2023; Burns et al., 2024; Youstra et al., 2025; McKenzie et al., 2025).

5.1 Methodology

We now outline our approach for training linear probes to detect harmful content in language model outputs, examining key design choices that enable real-time detection during streaming.

Problem setup. Consider a dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, where each $x^{(i)}$ represents an exchange between a user and an AI assistant, which consists of a token sequence $x^{(i)} = (x_1^{(i)}, \ldots, x_{T_i}^{(i)})$. Each exchange has a binary label $y^{(i)} \in \{0, 1\}$, where $y^{(i)} = 1$ indicates the presence of harmful content requiring refusal. Crucially, these labels are *exchange-level labels*, which means they reflect the harmfulness of the entire exchange. However, as we want to stream model responses, we need to make predictions throughout sampling. During inference, the language model processes each token $x_t^{(i)}$ to produce intermediate activations $\phi_t^{(\ell)}(x^{(i)})$ at each layer ℓ and position $t \in \{1, \ldots, T_i\}$.

Linear probe architecture. The simplest approach for a linear probe would be to simply predict the exchange-level harmfulness at each token position t using intermediate activations:

$$p_{\text{probe}}(y = 1 \mid x_{1:t}^{(i)}) = \sigma(W^\top \psi_t(x_{1:t}^{(i)}) + b), \tag{1}$$

where σ denotes the sigmoid function, W and b are learnable parameters, and ψ_t represents the activation features. For single-layer probing, we use $\psi_t = \phi_t^{(\ell)}$ from layer ℓ. For multi-layer probing, we concatenate activations: $\psi_t = [\phi_t^{(\ell_1)}; \phi_t^{(\ell_2)}; \ldots]$.

Training modifications. We make two key modifications to this approach to improve probe performance:

1. Sliding Window Mean (SWiM) Logit Smoothing. First, we average logits over a sliding window of M tokens during training:

$$\bar{z}_t(x^{(i)}) = \frac{1}{M} \sum_{k=0}^{M-1} \left[W^\top \psi_{t-k}(x^{(i)}_{1:t-k}) + b \right], \text{ for } t \geq M \text{and } T_i \geq M, \tag{2}$$

where $\bar{z}_t$ is the averaged logit at position t. This smoothing ensures that content is considered harmful only when the probe consistently predicts harmfulness across consecutive tokens. Without averaging, a single token with a spuriously high activation could incorrectly flag an entire sequence as harmful. Our approach thus reduces false positives from isolated spikes while maintaining sensitivity to genuinely harmful content, for which we expect to observe consistent evidence of harm.

During inference, we use an exponential moving average (EMA) instead of a sliding window mean for computational convenience. In particular, the EMA requires storing only a single scalar state between token generations, whereas a sliding window would require maintaining a buffer of the previous M scores. As such, the EMA is more practical for deployment.[3]

2. Softmax-Weighted Loss. We employ a softmax-weighted loss that emphasises achieving low loss when the probe is most confident in the sequence being harmful:

$$\mathcal{L}(x^{(i)}) = \sum_{t=M}^{T_i} w_t \cdot \mathcal{L}_{\text{BCE}} \left(y^{(i)}, \sigma(\bar{z}_t(x^{(i)})) \right) \text{ for } T_i \geq M, \tag{3}$$

$$\text{with } w_t = \frac{\exp(\bar{z}_t/\tau)}{\sum_{t'=M}^{T_i} \exp(\bar{z}_{t'}/\tau)}, \tag{4}$$

with $\mathcal{L}_{\text{BCE}}$ denoting binary cross-entropy loss, and τ controlling the temperature of the softmax weighting. As $\tau \to 0$, the loss focuses on the most confidently harmful predictions; as $\tau \to \infty$, all positions are weighted equally. This loss excludes token positions $t < M$ to ensure all evaluated positions benefit from averaging over a full window of M tokens, except for training sequences less than M tokens, where we average the tokens available and make a single prediction.

Justification for asymmetric weighting. This approach addresses a core challenge of streaming classification: the probe must predict harmfulness before observing a full sequence, but our labels reflect the harmfulness of the *entire* exchange. Consider sequences x_A and x_B that share a harmless prefix p, where x_A continues with harmful content (and so has label $y_A = 1$) while x_B remains harmless (and so has label $y_B = 0$). A standard binary cross entropy loss would push predictions to $p(\text{harm}) = 0.5$ throughout the prefix, even though the prefix p is genuinely harmless. In contrast, our weighting, proportional to $\exp(\bar{z}_t/\tau)$, assigns negligible weight to tokens with negative logits, which are token-positions where the probe predicts the sequence is harmless. This allows the probe to correctly predict harmlessness for the prefix p without penalty. Moreover, our scheme severely penalizes false positives—any confident harmful prediction in a harmless sequence would receive high weight, reflecting how a single flag blocks all subsequent content for streaming classification. This differs from a logit magnitude-based weighting, which would use $|\bar{z}_t|$ and would incorrectly penalize confident harmless predictions at token-positions before harm appears in the sequence.

This method is closely related to the softmax probes of McKenzie et al. (2025), which also route gradients toward tokens indicative of the positive class. However, we explicitly target streaming classification rather than making a single prediction for each token-sequence. Our loss function also applies weighting to per-token losses, rather than logits, meaning the loss functions are not equivalent. Moreover, McKenzie et al. (2025) do not employ logit smoothing.

5.2 ANALYSIS

We now analyze the impact of various probe design choices on classifier performance, and compare the performance of linear probes with separate, finetuned classifiers.

Classifier training details. We train a linear probe on top of a medium-size internal LLM using concatenated activations across all layers. For the exchange dataset, we employ the same training

[3]We did not experiment with EMA during training as we found the sliding window mean performed well and is easily implemented.

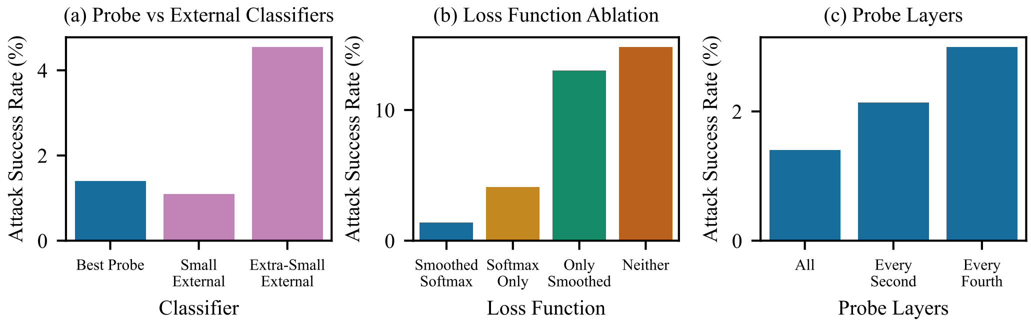

Figure 2: **Analysis of linear probe performance for CBRN-related harmfulness detection using static jailbreak data. (a) Attack success rates for different models.** We find that probes achieve competitive robustness to finetuned Constitutional Classifiers. **(b) Ablation study of loss function.** Combining logit smoothing and softmax weighting yields the best performance. **(c) Impact of layer selection on probe performance.** Performance degrades as fewer layers are utilized for probing. All evaluations conducted on static jailbreak datasets of CBRN-related queries. All classifiers are exchange classifiers and are calibrated to 0.1% refusal rates on WildChat.

data as in the previous section but replace the synthetic pipeline's hard labels with exchange-level soft labels produced by a medium-sized Constitutional Classifier.[4] For probe training, we use softmax temperature $\tau = 1$, window size $M = 16$, and train on approximately 25% of the full exchange dataset; we found training on more examples did not improve probe performance. For evaluation, we compute predictions using an exponentially weighted moving average. As baselines, we compare against the first-stage and second-stage classifiers from Section 3. However, in this section, both of these classifiers are calibrated to a refusal rate of 0.1% on WildChat (Zhao et al., 2024).

Jailbreak data. We assess robustness using a dataset of jailbreaks collected from human red-teaming, comprising approximately 7,000 exchanges designed to elicit harmful CBRN-related information. We use only exchanges produced by red-teamers whose exchanges were never used for classifier training. Following Sharma et al. (2025), we evaluate response harmfulness using LLM-rubric grading supplemented with additional LLM filters to exclude refusals. We evaluate the harmfulness of the entire model completion using the rubric grader, but consider the attack successfully defending against if the exchange classifiers flags at any point during model sampling.[5] For the analysis in this section, we truncate the datasets to have a maximum context length of 20K tokens.

Probe robustness approaches external classifiers. We first compare the attack success rate of our best-performing linear probe against the small and extra-small external classifiers used in our two-stage system (Figure 2a). The probe achieves an attack success rate that outperforms the extra-small classifier and almost matches the small classifier. These results suggest that linear probes could potentially serve as viable first-stage classifiers in cascaded defense systems. However, comprehensive human red-teaming remains necessary to fully validate probe robustness before deployment.

Softmax weighting and smoothing are critical for probe performance. We assess the importance of softmax loss weighting and logit smoothing by training probes that do not use these (Figure 2b). Using neither smoothing nor softmax weighting results in the highest attack success rate. While applying either softmax loss-weighting or logit smoothing improves performance, combining both techniques yields the best performance. We consider additional loss functions in Appendix Figure 5.

Multi-layer probing substantially improves performance. Our default probe uses all-layers. To examine the importance of this choice, we consider the performance of probing on different numbers of layers (Figure 2c). We find using all available layers achieves the lowest attack success rate at approximately, while using fewer layers consistently yields worse performance. This highlights the benefit of including multiple layers for linear activation probes.

[4] We found either distilling with soft-labels or using hard labels from our synthetic data pipeline leads to similar performance.

[5] This may underestimate the attack success rate.

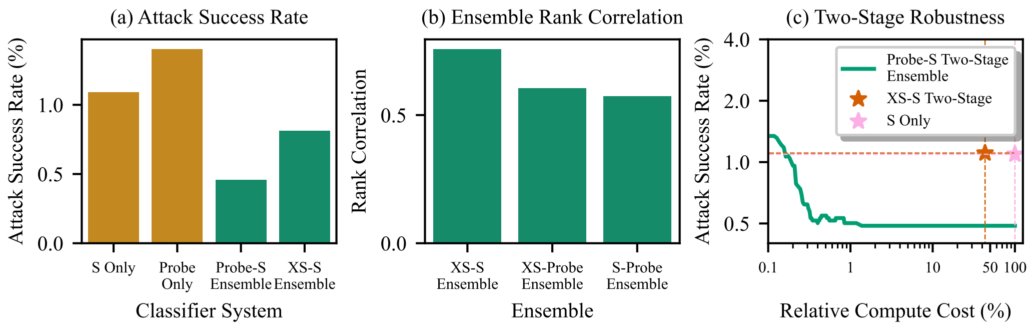

Figure 3: **Combining linear probes with external classifiers improves robustness and reduces costs. (a) Attack success rates for classifier systems.** We find combining our probe and a small external classifier yields the best performance, outperforming the more expensive external-classifier ensemble. **(b) Spearman rank correlation between classifier predictions on jailbreak attempts.** Probes and external classifiers make more independent errors than pairs of external classifiers. **(c) Compute-robustness tradeoff curves for two-stage configurations.** We use the probe predictions to determine routing to the second-stage classifier, which uses the average logit across the probe and small classifier for prediction. We present the cost relative to using the small-sized classifier on all traffic. On our dataset of CBRN-related exchanges, our system combining probes and the small external classifier can exceed the robustness of the small classifier alone whilst simultaneously offering a 100x reduction in compute costs. All systems are exchange classifiers calibrated to have a 0.1% refusal rate on WildChat. We evaluate robustness using LLM-based rubric grading on a dataset of CBRN-related human red-teaming exchanges.

5.3 Combining probes and classifiers

We now investigate whether combining probes and classifiers can offer further performance improvements. Indeed, ensemble methods have a long history in machine learning at improving performance and robustness (e.g., Dietterich, 2000; Lakshminarayanan et al., 2017).

Ensembling improves robustness. We first explore weighted ensembles between our classifiers (Figure 3a). We evaluate robustness using the dataset, evaluation method, and calibration procedure described previously. For each classifier pair, we combine their logits using a weighted averaged: $z_{\text{ensemble}} = \alpha z_1 + (1 - \alpha) z_2$, where α is optimized per pair to minimize attack success rate on the test set. Ensembling the probe with the small external classifier achieves the lowest attack success rate, substantially outperforming either classifier alone. Notably, equal weighting ($\alpha = 0.5$) proves near-optimal for this ensemble—see Appendix Figure 6. While combining the extra-small and small external classifiers yields some improvement over individual classifiers, the gains are modest compared to the probe-based ensemble.

Probes provide complementary predictions to external classifiers. To understand why probe-classifier ensembles outperform classifier-only ensembles, we analyze the correlations between classifier predictions on our red-teaming exchange dataset (Figure 3b). We find the Spearman rank correlation between the probe and small external classifier is lower than that between the pair of external classifiers. This suggests that despite being trained on the same distribution as the external classifiers, the probe appears to leverage signals that complement those used by external classifiers, explaining the effectiveness of probe-classifier ensembling.

Probe-classifier cascades reduce computational overhead. Finally, we examine whether probes can improve the compute-robustness tradeoff in two-stage classifier cascades. We use our lightweight probe continuously during generation and invoke a more expensive second-stage classifier selectively based on the probe's confidence. Specifically, when the probe score exceeds a threshold, we route the exchange to the small external classifier and average their logits to make our final prediction. Otherwise, we rely solely on the probe. By varying the probe threshold, we control the fraction of exchanges requiring external classification and thus the compute cost, noting the cost of these cascades is nearly entirely made up of the external classifier costs.

We find the probe-classifier cascade substantially improves both efficiency and robustness relative to the two-stage system described in Section 4, and also compared to a single exchange classifier (Figure 3c). On our dataset, **our system exceeds the robustness of the small classifier alone whilst simultaneously offering a 100x reduction in compute costs.** While these results demonstrate considerable promise, we emphasize that human red-teaming remains essential to validate the robustness of this system.

6 RELATED WORK

Adaptive computational schemes. Recent work has explored adaptive two-stage classification approaches for content moderation. OpenAI's systems (OpenAI, 2025a;b) employ a lightweight topic filter to identify high-harm content areas, triggering more expensive classification only when necessary. While their first layer focuses on topic detection, our approach explicitly targets jailbreak attempts in the initial stage.[6] Hua et al. (2025) investigate optimal strategies for combining multiple monitors under cost constraints. In contrast, we implement a straightforward two-stage classification scheme and validate its robustness through extensive human red-teaming. These approaches reflect broader trends in adaptive computation, including systems like TARS (Kim et al., 2025), which dynamically allocate test-time compute based on query complexity. Other work explores cascades and model routers for efficient LLM deployment. HybridLLM (Ding et al., 2024) employs a lightweight BERT-style decoder to route queries between small and large LLMs, AutoMix (Aggarwal et al., 2025) adaptively combines outputs across models using confidence scores and learned routing, and recent advances have refined cascades through confidence-based deferral (Rabanser et al., 2025).

Model-internals approaches. Several methods leverage model internals for efficient classification. Cunningham et al. (2025) explore internals-based classifiers, focusing on last-N layer networks, while we focus on boosting the performance of simple linear probes with an weighted loss function and logit smoothing. McKenzie et al. (2025) similarly investigate softmax-weighted probes and classifier cascades. However, while they aggregate token scores into single sequence-level predictions, we build streaming classifiers with continuous predictions during generation. We use a softmax to weight per-token losses, not to aggregate predictions over tokens. Our analysis in Section 5 showes that combining softmax weighting with logit smoothing substantially improves performance. Beyond standard activation probes for harmfulness detection (Alain & Bengio, 2016; Zou et al., 2023; Youstra et al., 2025), recent approaches fine-tune LLMs using internals-based losses, including short-circuiting (Zou et al., 2024) and latent adversarial training (Casper et al., 2024). Other approaches build classifiers using sparse auto-encoder features (Bricken et al.; Kantamneni et al., 2025).

7 CONCLUSION

Our work demonstrates that Constitutional Classifiers can achieve production-grade jailbreak robustness with dramatically improved deployment viability. Our approaches include exchange classifiers that evaluate outputs within their conversational context to prevent obfuscation attacks, cascaded classifiers that reserve expensive classification only for flagged content, and activation-based probes. We develop systems that provide robust protection against universal jailbreak attempts while meeting the stringent false positive and computational constraints required for deployment, thus establishing Constitutional Classifiers as practical and effective safeguards for production LLMs.

Future work. Several directions could further enhance our methods. Tighter integration between classifier safeguards and language models—for example, incorporating classifier signals directly into model sampling processes, or training models to better resist obfuscation attempts—could strengthen robustness. Additionally, improving training data through automated red-teaming (Perez et al., 2022) or including data more representative of production traffic could yield better classifiers. Furthermore, while our latest system improves the false-positive rate compared to previous work, future work could examine other solutions, such targeted synthetic data generation to teach classifier models the intended decision boundary.

[6] As noted by OpenAI (2025b): "The first tier in this system is a fast, topical classifier model that determines whether or not the content is related to biology."

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models, 2025. URL https://arxiv.org/abs/2310.12963.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint*, 2016. URL https://arxiv.org/abs/1610.01644.

Ross J Anderson. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2010.

Anthropic. Anthropic's responsible scaling policy, 2023. URL https://www-cdn.anthropic.com/1adf000c8f675958c2ee23805d91aaade1cd4613/responsible-scaling-policy.pdf.

Trenton Bricken, Jonathan Marcus, Siddharth Mishra-Sharma, Meg Tong, Ethan Perez, Mrinank Sharma, Kelley Rivoire, Thomas Henighan, and Adam Jermyn. Using dictionary learning features as classifiers, october 2024. *URL https://transformer-circuits. pub/2024/features-as-classifiers/index. html*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision, 2024. URL https://arxiv.org/abs/2212.03827.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Stephen Casper, Lennart Schulze, Oam Patel, and Dylan Hadfield-Menell. Defending against unforeseen failure modes with latent adversarial training. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2403.05030.

Hoagy Cunningham, Alwin Peng, Jerry Wei, Euan Ong, Fabien Roger, Linda Petrini, Misha Wagner, Vladimir Mikulik, and Mrinank Sharma. Cost-effective constitutional classifiers via representation re-use, July 2025. URL https://alignment.anthropic.com/2025/cheap-monitors/. Anthropic Alignment Science Blog.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Rühle, L. V. Lakshmanan, and A. H. Awadallah. Hybridllm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024. URL https://arxiv.org/abs/2404.14618.

Tim Tian Hua, James Baskerville, Henri Lemoine, Mia Hopman, Aryan Bhatt, Tyler Tracy, and MARS Redwood Research. Combining cost-constrained runtime monitors for ai safety. *arXiv preprint arXiv:2507.15886*, 2025. URL https://arxiv.org/abs/2507.15886.

Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing, 2025. URL https://arxiv.org/abs/2502.16681.

Taeyoun Kim, Fahim Tajwar, Aditi Raghunathan, and Aviral Kumar. Reasoning as an adaptive defense for safety. *arXiv preprint arXiv:2507.00971*, 2025.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew B. Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Adam Khoja, Zhenqi Zhao, Ariel Herbert-Voss, Cort B. Breuer, Samuel Marks, Oam Patel, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam A. Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Russell Kaplan, Ian Steneker, David Campbell, Brad Jokubaitis, Alex Levinson, Jean Wang, William Qian, Kallol Krishna Karmakar, Steven Basart, Stephen Fitz, Mindy Levine, Ponnurangam Kumaraguru, Uday Tupakula, Vijay Varadharajan, Ruoyu Wang, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The WMDP benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2403.03218.

Alex McKenzie, Urja Pawar, Phil Blandfort, William Bankes, David Krueger, Ekdeep Singh Lubana, and Dmitrii Krasheninnikov. Detecting high-stakes interactions with activation probes, 2025. URL https://arxiv.org/abs/2506.10805.

Kristina Nikolić, Luze Sun, Jie Zhang, and Florian Tramèr. The jailbreak tax: How useful are your jailbreak outputs? *arXiv preprint arXiv:2504.10694*, 2025.

OpenAI. OpenAI preparedness framework (beta), 2023. URL https://cdn.openai.com/openai-preparedness-framework-beta.pdf.

OpenAI. Chatgpt agent system card. Technical report, OpenAI, July 2025a. URL https://cdn.openai.com/pdf/839e66fc-602c-48bf-81d3-b21eacc3459d/chatgpt_agent_system_card.pdf. System card.

OpenAI. Gpt-5 system card. Technical report, OpenAI, August 2025b. URL https://cdn.openai.com/gpt-5-system-card.pdf. System card.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint*, 2022. URL https://arxiv.org/abs/2202.03286.

Stephan Rabanser, Nathalie Rauschmayr, Achin Kulshrestha, Petra Poklukar, Wittawat Jitkrittum, Sean Augenstein, Congchao Wang, and Federico Tombari. Improving model cascades through confidence tuning. *arXiv preprint arXiv:2502.19335*, 2025. URL https://arxiv.org/abs/2502.19335.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint*, 2023. URL https://arxiv.org/abs/2311.12022.

Mrinank Sharma, Meg Tong, Jesse Mu, Jerry Wei, Jorrit Kruthoff, Scott Goodfriend, Euan Ong, Alwin Peng, Raj Agarwal, Cem Anil, et al. Constitutional classifiers: Defending against universal jailbreaks across thousands of hours of red teaming. *arXiv preprint arXiv:2501.18837*, 2025.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for empty jailbreaks. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2402.10260.

Jack Youstra, Mohammed Mahfoud, Yang Yan, Henry Sleight, Ethan Perez, and Mrinank Sharma. Towards safeguarding llm fine-tuning apis against cipher attacks, 2025. URL https://arxiv.org/abs/2508.17158.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1M ChatGPT interaction logs in the wild. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2405.01470.

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.

Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, Rowan Wang, Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving alignment and robustness with circuit breakers. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2406.04313.