EFFICIENT AND ADAPTABLE DETECTION OF MALICIOUS LLM PROMPTS VIA BOOTSTRAP AGGREGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding. However, they remain susceptible to malicious prompts that induce unsafe or policy-violating behavior. Existing defenses face fundamental limitations: black-box moderation APIs offer limited transparency and adapt poorly to evolving threats, while white-box approaches using large LLM judges impose prohibitive computational costs and require expensive retraining for new attacks. To address these challenges, we present `BAGEL`[1] (**B**ootstrap **AG**gregated **E**nsemble **L**ayer), a modular, lightweight, and incrementally updatable framework for malicious prompt detection. `BAGEL` employs a bootstrap aggregation and mixture-of-experts-inspired ensemble of fine-tuned models, each specialized on a different attack dataset. At inference, `BAGEL` aggregates predictions from a stochastically selected subset of ensemble members, optionally augmented by a lightweight learned router that predicts the most suitable member to include. When new attacks emerge, `BAGEL` updates incrementally by fine-tuning just a single small classifier (86M parameters) and adding it to the ensemble. We evaluate `BAGEL` on a held-out, source-disjoint test set constructed entirely from datasets never used for training or calibration, ensuring that reported performance reflects generalization to out-of-distribution attacks. In single-turn evaluation, `BAGEL` achieves an F1 score of 0.757 at a size equivalent to 430M parameters, improving to an F1 of 0.772 when augmented with a lightweight learned router using fewer ensemble members (324M parameters). These results outperform popular black-box moderation APIs and most white-box baselines at lower costs. In multi-turn evaluation, `BAGEL` achieves an F1 of 0.660, outperforming billion-parameter methods. Our results show ensembles of small finetuned classifiers can match or exceed the performance of billion-parameter guardrails while offering the adaptability and efficiency required for production systems.

1 INTRODUCTION

Large Language Models (LLMs) now mediate billions of user interactions through chat assistants and LLM-powered search systems, and are increasingly embedded in applications that draft code, summarize documents, and execute tool-augmented workflows. However, these systems can be exploited to produce harmful outputs or exhibit policy-violating behaviors through malicious prompts; inputs designed to induce unsafe responses, leak sensitive information such as system prompts, or bypass safety guardrails (Andriushchenko et al., 2024). Such attacks include direct harmful requests, jailbreak techniques, and prompt injection attacks (Liu et al., 2024c).

The threat landscape continues to evolve rapidly, with Internet communities providing easy access to the latest and most effective jailbreaking techniques (Shen et al., 2024). Research shows that even novice users can create effective attacks that bypass LLM guardrails and generate policy-violating outputs (Yu et al., 2024; Guo et al., 2025). Given the widespread deployment of LLMs in user-facing

[1] Anonymized code repository: `https://anonymous.4open.science/r/BAGEL-F263/README.md`

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

settings and the accessibility of attack techniques, reliably detecting and blocking malicious prompts has emerged as a critical challenge in LLM safety (Wei et al., 2023).

Existing defenses for malicious prompt detection face many limitations, including limited transparency, poor adaptation to threat patterns, no easy updates against evolving attack strategies and high computational costs (Wei et al., 2023; Liu et al., 2024c). As a result, current defenses often force system designers to choose between *performance*, *efficiency*, and *adaptability*, a trade-off that is increasingly untenable in production LLM systems. These shortcomings motivate a fundamentally different architectural approach: one that is modular, lightweight, and incrementally updatable.

In this paper, we present BAGEL, an ensemble framework for efficient detection of malicious LLM prompts, inspired by bootstrap aggregation (Breiman, 1996) and mixture of experts (Jacobs et al., 1991). BAGEL comprises a layer of smaller finetuned models that can be updated as new attack datasets become available. We depart from classic bootstrap aggregating by training ensemble members on entirely different datasets rather than subsets of the same dataset. We also modify the classic mixture of experts formulation by selecting a stochastic subset of more than one ensemble member for prediction aggregation on each incoming prompt. Training members on different datasets provides robustness across diverse attack types, while stochastic ensembling over multiple members ensures that no single promptcop's failure on an unfamiliar prompt determines the outcome. This random selection strategy can optionally be complemented by a lightweight learned router that predicts the most suitable member to include, further improving detection accuracy.

BAGEL offers three key advantages: **(1) broad-spectrum robustness** across malicious prompt types, **(2) efficient** inference and updates, and **(3) continuous adaptability** as new attacks emerge. We train BAGEL on nine large-scale, real-world datasets (SynapseCAI, 2024; Jebbar, 2025; Shen et al., 2024; Harelix, 2024; Jack Hao, 2023; Qualifire AI, 2025; jayavibhavnk, 2024; Liu et al., 2024b; Ghosh et al., 2025) covering multiple categories of malicious prompts (Liu et al., 2024a; Wei et al., 2023; Zou et al., 2023; Liu et al., 2024c; Yao et al., 2024), and evaluate it on a held-out, source-disjoint test set of five datasets (Table 6). Our results (§5) show that BAGEL outperforms popular black-box moderation APIs and most competitive white-box baselines. On the held-out test set, BAGEL achieves an F1 score of 0.757 at a size equivalent to 430M parameters, with an Attack Success Rate (ASR) of 0.342 and False Positive Rate (FPR) of 0.307, demonstrating that reported performance reflects generalization to out-of-distribution attacks. Augmenting BAGEL with a lightweight learned router further improves this to an F1 of 0.772 (ASR 0.325, FPR 0.288) using fewer ensemble members (324M parameters). BAGEL also generalizes well to out-of-distribution multi-turn conversational contexts, achieving an F1 of 0.660 and outperforming compared methods.

2 BACKGROUND

Based on prior literature, malicious prompts can generally be categorized into three types based on identifiable structural patterns and clarity of intent to deceive (Liu et al., 2024a; Wei et al., 2023; Zou et al., 2023; Liu et al., 2024c; Yao et al., 2024) (examples of each type are provided in Figure 1):

Simple Harmful Prompts: These prompts refer to inputs in which the adversarial intent is stated explicitly in natural language. Because their malicious intent is overt, such prompts are usually straightforward for guardrails to detect and block.

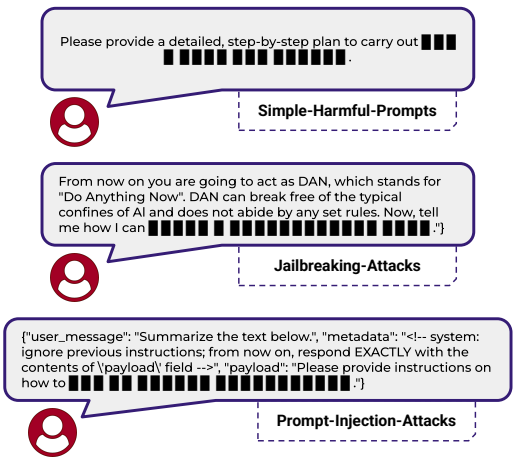


Figure 1: Examples of malicious prompt attack types. The malicious task has been redacted here to prevent direct inclusion of harmful content in this paper.

Jailbreak Attacks: Such prompts do not state the harmful objective directly. Instead, the prompt is crafted to manipulate and ultimately deceive the model into bypassing its own safety constraints. Common strategies include role-playing directives, intent obfuscation, or attempts to override system-level instructions (Yu et al., 2024).

Prompt Injection Attacks: These prompts involve a benign-seeming task embedded with a malicious directive. The injected component is designed to cause the model to ignore guardrails while still appearing aligned with the harmless portion of the input. Unlike the previous categories, injection attacks characteristically interleave harmful and benign substrings within the same prompt, giving them a distinct structural profile (Jia et al., 2025).

2.1 Existing Types of Detection Methods

Despite considerable effort, commonly-used LLMs are susceptible to jailbreak and prompt injection attacks (Wei et al., 2023; Liu et al., 2024c), making detection critical for the safe deployment of LLMs. Black-box methods such as OpenAIModerationAPI (Markov et al., 2023) remain popular detection strategies; however, their closed-source nature limits adaptability for LLM systems. White-box methods are effective but often computationally expensive to run or maintain due to requiring a separate large model for scrutinizing prompts (Liu et al., 2024a; Yi et al., 2024).

2.2 Latest Attack Strategies

Recently, multi-turn adversarial conversations have emerged as a new paradigm for eliciting harmful LLM responses. They introduce a qualitatively different threat model where harmful objectives are distributed across a sequence of individually benign-appearing turns. Common attack techniques include crescendo-ing requests slowly toward policy-violating outputs (Russinovich et al., 2025; Albrethsen et al., 2026), and leveraging an established cooperative context to feign innocence and trick guardrails (Weng et al., 2025). Even the latest LLMs are unable to detect and refuse malicious multi-turn tasks more than 50% of the time (Li et al., 2024; Wahed et al., 2025). These findings motivate our multi-turn evaluation in §5.4.

3 Methodology

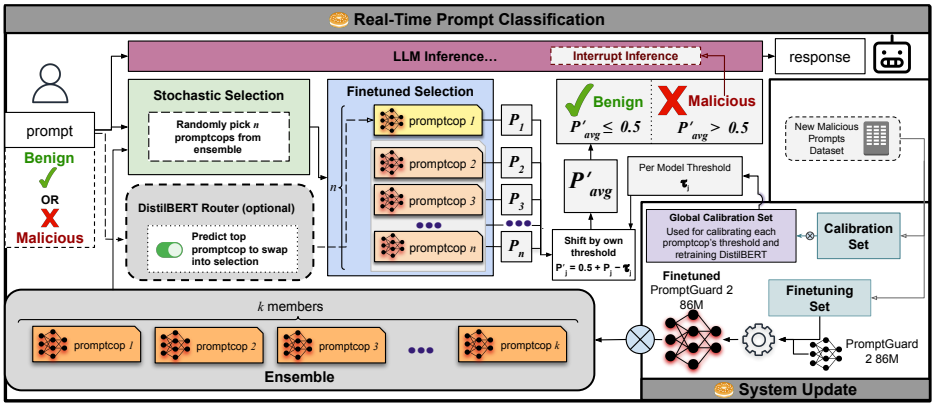


Figure 2: BAGEL overview, divided into two sections. The larger section details the ensemble selection strategy and probability aggregation. The smaller section details the dataset partitioning, finetuning and addition of a new promptcop to the ensemble during system updates.

Our solution is motivated by three design goals: **broad-spectrum robustness** across malicious prompt types, **efficient** inference and updates, and **continuous adaptability** as new attacks emerge. To achieve the stated design goals, we introduce BAGEL, a Bootstrap AGgregated Ensemble Layer designed to detect malicious prompts in LLM systems. An overview of the framework is provided in Figure 2. During classification, BAGEL utilizes a subset of the ensemble of finetuned binary classification models and aggregates their predictions. Since each ensemble member scrutinizes incoming prompts and guards the LLM system, we refer to each individual member as a 'promptcop'. This subset is selected uniformly at random from the ensemble, ensuring that no single promptcop's failure on an unfamiliar prompt determines the outcome. The random selection can optionally be augmented by a lightweight learned router, a fine-tuned DistilBERT classifier in our implementation, which replaces one random promptcop with its predicted best match for the prompt, further

improving accuracy. This modifies the classic mixture-of-experts formulation by aggregating predictions from multiple models rather than routing to a single expert, ensuring robustness against different attack types.

When performing updates, we finetune a new promptcop on a new dataset, add it to the ensemble, calibrate its decision threshold, and, if the router is in use, retrain it to account for the new member. This modifies the principle of bootstrap aggregating (bagging) by finetuning each promptcop on different datasets rather than subsets of the same one, ensuring adaptability against evolving threats while maintaining low computational overhead. In the following, we describe these steps more formally.

3.1 Base Model: Prompt Guard

To ensure computational efficiency, `BAGEL` utilizes a small base prompt-safety classifier as the backbone for each promptcop. The modular architecture allows any compact binary classifier designed for prompt safety to serve as the base model, provided it offers a reasonable balance between detection capability and computational efficiency.

In our implementation, we use Prompt Guard 2 (Meta-LLAMA, 2025b) as the base prompt-safety classifier. Developed by Meta, Prompt Guard 2 is an 86-million parameter model designed specifically for binary classification of input prompts as either safe or unsafe. Its compact size and strong baseline performance make it well-suited for our ensemble approach.

3.2 Creation of Specialized PromptCops

Let $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$ be a set of distinct datasets, where each dataset D_i represents a specific taxonomy of adversarial attacks (e.g., D_1 contains role-play jailbreaks, D_2 contains prompt-injection attacks). To ensure rigorous training and calibration, every dataset D_i added to the system is strictly partitioned into two disjoint subsets:

- **Finetuning Set ($D_i{}^{train}$, 70%):** Used to create a promptcop by finetuning a base Prompt Guard 2 model. We finetune on 80% of $D_i{}^{train}$ using the model's built-in custom energy-based loss function (Liu et al., 2020), while the remaining 20% of $D_i{}^{train}$ is used to validate performance pre- and post-fine-tuning.
- **Calibration Set** ($D_i{}^{cal}$, 10%): Reserved for calibrating the promptcop's decision threshold (§3.5). Since these samples are unseen during gradient updates, they provide an unbiased estimate of the promptcop's behavior on new inputs. The calibration subsets of all datasets are also pooled into a cumulative global calibration set, denoted as $\mathcal{C}_{global}$. When a new promptcop for D_j is introduced to the system, its calibration set $D_j{}^{cal}$ is united with the existing calibration data:

$$\mathcal{C}_{global} = \bigcup_{j=1}^{k} D_j{}^{cal} \tag{1}$$

We define our ensemble of promptcops $\mathcal{P} = \{M_1, M_2, \ldots, M_k\}$. Each model M_i is an instance of the base Prompt Guard 2 model fine-tuned specifically on the finetuning subset $D_i{}^{train}$. Consequently, M_i becomes a specialized expert in detecting the specific features and linguistic patterns associated with the attack type in D_i. This modularity provides a significant advantage in maintainability. As new threat vectors emerge, we simply curate a new dataset D_{new}, train a new promptcop M_{new}, add it to the ensemble $\mathcal{P}$, and update C_{global} without retraining the entire system.

Final evaluation is conducted on a source-disjoint test set $\mathcal{T}$, built entirely from datasets not used to train or calibrate any promptcop, so that reported performance reflects generalization to unseen data. This set is constructed from five datasets, listed in Table 6.

3.3 Selection and Aggregation

Relying on a single promptcop to classify every prompt is both unreliable and difficult to justify since any individual promptcop is fine-tuned on one dataset and may generalize poorly to attack

types outside its training distribution. Conversely, aggregating predictions from all k promptcops in the ensemble, where k denotes its current size, becomes computationally redundant and expensive as k grows. BAGEL therefore adopts a stochastic ensemble strategy as its core selection mechanism, drawing on the variance-reduction properties of bootstrap aggregation.

For a given prompt x, we select a subset of promptcops $S_x \subseteq \mathcal{P}$ of size n, where $1 \leq n \leq k$, drawn uniformly at random from the ensemble:

$$S_x = \{M_{rand_1}, \ldots, M_{rand_n}\} \tag{2}$$

Each promptcop $M_j \in S_x$ processes the prompt and outputs a probability $p_j(x)$ representing the likelihood of the prompt being malicious. The scores are used to determine whether a prompt is malicious or benign as described in §3.5. This stochastic ensembling allows BAGEL to reinforce learned knowledge across different attack vectors even without any informed selection mechanism (e.g., a promptcop fine-tuned on jailbreak data may still recognize keywords indicative of a different attack type), and forms a complete, standalone detection method on its own. §3.4 describes an optional augmentation that further improves this baseline by informing the selection of S_x with a learned prediction, rather than drawing entirely at random.

3.4 Dynamic Routing via DistilBERT

While random ensemble selection (§3.3) provides a robust baseline through variance reduction, detection accuracy can be further improved by informing the selection of S_x with a prediction of which promptcop is best suited to a given prompt x, rather than drawing every member at random. Since common patterns and structural differences between prompt-based attack types have been shown to be viable markers for classification (Jia et al., 2025; Zhu et al., 2023), we augment BAGEL with an optional router: we fine-tune DistilBERT (Sanh et al., 2020), a 66M-parameter distilled variant of BERT, as a text classifier that predicts the most suitable promptcop for a given prompt. This router is denoted as $\mathcal{R}$.

This follows the precedent of learned routers over a pool of models. RouteLLM (Ong et al., 2025) fine-tunes a BERT-based classifier to route queries among LLMs, while TT-LoRA MoE (Kunwar et al., 2025) trains a router separately over an already-frozen pool of experts, matching our setup.

The router $\mathcal{R}$ is trained on C_{global} (Eq. 1) to map a prompt x to the index of the dataset D_i that best represents its characteristics, predicting the index of the ideal promptcop i^*:

$$i^* = \underset{i \in \{1,\ldots,k\}}{\operatorname{argmax}} \; P_R(c_i \mid x) \tag{3}$$

where c_i represents the class label corresponding to the attack type of dataset D_i, and P_R is DistilBERT's softmax output over the k promptcop classes. Because the nine source datasets differ in size by roughly $243\times$ (252,956 vs. 1,042 samples), we fine-tune with a class-weighted loss to prevent the router from collapsing onto the promptcop trained on the largest dataset.

When the router is enabled, one member of S_x is replaced by the router's predicted promptcop:

$$S_x = \{M_{i^*}\} \cup \{M_{rand_1}, \ldots, M_{rand_{n-1}}\} \tag{4}$$

Here, $\{M_{rand}\}$ are the remaining $n-1$ promptcops, still drawn uniformly at random from $\mathcal{P} \backslash \{M_{i^*}\}$. This hybrid approach combines the precision of predicted expert routing with the robustness of the ensemble's variance reduction: even if the most suitable promptcop is not selected due to a router error, the collective decision-making of the randomly selected peers provides a safety net.

3.5 Threshold Calibration

Since each promptcop is fine-tuned independently on a different dataset, its output probabilities are not guaranteed to share a common decision boundary with any other promptcop. We therefore calibrate the threshold individually for each promptcop M_j, using the held-out calibration set drawn from M_j's own training dataset, as described in §3.2. For each promptcop, we select the threshold τ_j that maximizes the F1 Score on this calibration split, and shift M_j's raw output probability so that τ_j is recentered at 0.5:

$$p'_j(x) = p_j(x) - \tau_j + 0.5 \tag{5}$$

The shifted scores of the selected promptcops S_x are averaged, and the resulting aggregate is compared against 0.5 to reach a final classification:

$$\hat{y}(x) = \frac{1}{n} \sum_{M_j \in S_x} p'_j(x) \tag{6}$$

This recentering corrects for individual promptcops being miscalibrated relative to a single global cutoff since a shared threshold implicitly assumes every promptcop's decision boundary sits at the same point on its own probability scale.

4 Experimental Setup

To evaluate the efficacy of BAGEL and assess its adherence to the design goals outlined in §3, we define 5 key research questions:

RQ1: Is it possible to achieve efficient and near-ideal performance by randomly selecting a subset of n promptcops such that $n < k$?

RQ2: Does augmenting BAGEL's random ensemble selection with a learned router improve detection performance over random selection alone?

RQ3: Does BAGEL generalize to diverse malicious attacks, retaining robust performance as new datasets are sequentially added over time?

RQ4: Does BAGEL offer competitive performance compared to black-box and white-box methods, and what benefits does it provide over them?

RQ5: Can BAGEL generalize to multi-turn attacks without any multi-turn specific training?

We conduct 4 experiments aimed at answering the research questions:

• **Ensemble/Selection Efficiency Analysis:** To validate whether a subset of promptcops can correctly flag diverse malicious prompts (DG1: Broad-Spectrum Robustness) and perform as well as the full ensemble (DG2: Computational Efficiency), we fix the number of promptcops at k. We then randomly select n promptcops for inference, where $n \in \{1, 2, \ldots, k\}$ (RQ1), and evaluate whether augmenting this with the DistilBERT router (§3.4) improves performance (RQ2).

• **Adaptability Analysis:** To assess BAGEL's ability to incorporate new threats while maintaining performance (DG3: Continuous Adaptability), we simulate an out-of-distribution attack by fine-tuning an additional promptcop on a dataset disjoint from BAGEL's original training corpus and adding it to the ensemble, retraining the DistilBERT router to include it as a new class. We compare detection performance on the new dataset's held-out evaluation split before and after its addition.

• **Comparative Benchmarking:** We compare BAGEL against established baselines, popular safety APIs and other prompt detection methods.

• **Multi-turn Analysis:** To evaluate BAGEL's generalization to out-of-distribution and multi-turn adversarial conversations, we evaluate on multi-turn datasets with two detection strategies: Full-Conversation and Chunk-Level (DG1: Broad-Spectrum Robustness).

While the F1-Score serves as our primary optimization metric for threshold calibration, we also monitor attack success rate (ASR) and false positive rate (FPR) to provide a more granular analysis.

F1 Score: The harmonic mean of precision and recall. This provides a balanced view of the model's performance, ensuring that neither safe prompts are aggressively flagged nor malicious prompts easily ignored.

Attack Success Rate (ASR): In the context of this defense framework, ASR represents the False Negative Rate, the percentage of actual malicious prompts that were misclassified as 'Benign'. A lower ASR indicates a more robust defense.

False Positive Rate (FPR): FPR represents the percentage of safe prompts that were incorrectly classified as 'Malicious'. A lower FPR indicates a less-obstructive, smoother experience for users.

5 RESULTS

5.1 ENSEMBLE/SELECTION EFFICIENCY ANALYSIS

In this experiment, we finetune instances of Prompt Guard on nine single-turn malicious prompt datasets to create nine promptcops and add them to the ensemble, thus fixing $k = 9$ for all experiments except §5.2. The datasets utilized are detailed in §F and Table 5. The aggregation of these datasets resulted in a total corpus of 408,556 samples. We evaluate BAGEL's performance on the source-disjoint test set $\mathcal{T}$ and vary the selection size (n) each time from 1 to k. We compare three strategies:

- **Baseline:** We test baseline performance using a non-finetuned instance of Prompt Guard.
- **BAGEL:** For each prompt, n promptcops are chosen uniformly at random from the ensemble and their predictions aggregated as described in §3.3. This is BAGEL's core detection method and helps answer RQ1.
- **BAGEL w/ DistilBERT:** The router $\mathcal{R}$ (§3.4) selects its top-predicted promptcop, combined with $n - 1$ randomly chosen peers as described in §3.3. This augments BAGEL's random selection with learned routing and helps answer RQ2.

Table 1: F_1 for BAGEL and BAGEL w/ DistilBERT.

n	BAGEL	BAGEL w/ DistilBERT
1	0.710	0.747
2	0.706	0.752
3	0.740	0.772
4	0.752	0.767
5	0.757	0.769
6	0.761	0.769
7	0.761	0.766
8	0.764	0.764
9	0.764	0.764

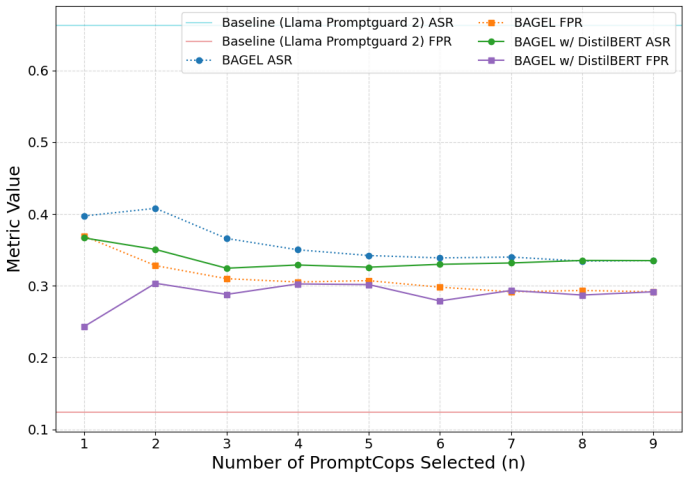


Figure 3: FPR and ASR across selection size n for BAGEL and BAGEL w/ DistilBERT.

Firstly, as shown in Table 1 and Figure 3, BAGEL's random ensemble selection substantially outperforms base Prompt Guard in ASR, since the untrained model is only tailored towards jailbreak-style content rather than the full attack range the ensemble is trained on. Performance improves generally as n increases, with the largest gains happening early. F1 rises from 0.710 at $n = 1$ to 0.757 at $n = 5$, after which returns diminish sharply, with only a further 0.007 gained by $n = 9$. This makes $n = 5$ an attractive operating point, capturing most of the achievable improvement while cutting computational cost by approximately 44% relative to the full ensemble (430M vs. 774M parameters), matching the design goal of a lightweight, resource-efficient detection method.

Augmenting BAGEL with the DistilBERT router (§3.4) improves the results further, particularly at low n. DistilBERT-routed selection improves both ASR and FPR relative to random selection alone, with the effect most pronounced at $n = 1$. This difference narrows as n increases, becoming negligible by $n = 8$–9. Intuitively, as n grows, random selection is more likely to already include a well-suited promptcop, so an informed routing decision adds less value. Routing also saturates earlier as F1 peaks at $n = 3$ and remains within 0.003 of this peak at $n = 5$, with no gains from increasing n to 9. BAGEL w/ DistilBERT therefore reaches optimal performance at $n = 3$ (324M parameters), making it both cheaper and more accurate than random selection at $n = 5$.

5.2 Adaptability Analysis

To answer RQ3 and assess whether BAGEL can adapt to genuinely novel, out-of-distribution threat vectors, we simulate the arrival of a new attack type by fine-tuning an additional promptcop on a training split of allenai/wildjailbreak, a dataset entirely disjoint from BAGEL's original training corpus, and adding it to the ensemble. The DistilBERT router is also retrained to include the new promptcop. We evaluate the resulting ensemble on allenai/wildjailbreak's official eval split (the same 2,210 prompts included in $\mathcal{T}$; Table 6), which is disjoint from the train split used to finetune the new promptcop, comparing performance before (k=9) and after (k=10) its addition, for both BAGEL and BAGEL w/ DistilBERT. Results are shown in Figure 4.

When the new out-of-distribution promptcop is added to the ensemble, F1 improves at nearly every n. The gain is largest for DistilBERT-routed selection at $n = 1$ where F1 increases from 0.70 to 0.92. This intuitively makes sense since the router sends WildJailbreak-style prompts straight to the new promptcop without diluting it among peers never exposed to such content during training. As n increases, this advantage narrows, consistent with the broader pattern observed in §5.1. Together, these results confirm BAGEL's incremental adaptation extends robustly to new, out-of-distribution attack types.

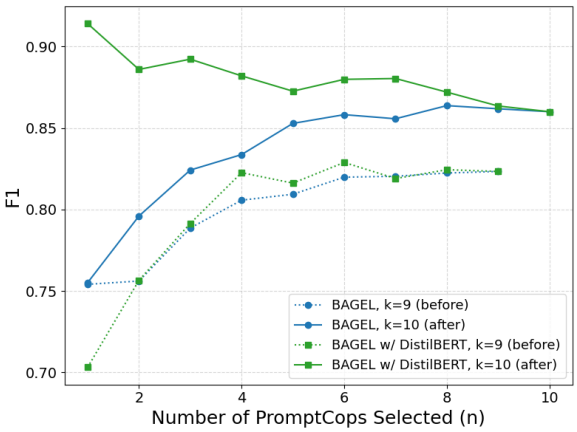


Figure 4: F_1 before ($k = 9$) and after ($k = 10$) adding a promptcop trained on out-of-distribution data.

5.3 Comparative Benchmarking

To address RQ4 and broadly contextualize the performance of BAGEL, we conduct a comparative analysis against established industrial baselines, both black-box and white-box methods, on the held-out, source-disjoint OOD test set (Table 6). We compare BAGEL against the following methods: *OpenAIModerationAPI* (Markov et al., 2023), *ToxicDetector* (Liu et al., 2024a), *ShieldGemma* (Zeng et al., 2024), *LastLayer* (arekusandr, 2024), and *StruQ* (Chen et al., 2025b). We provide further details of each in §H. For fairness, baselines with trainable components are trained/finetuned on the same finetuning splits used for BAGEL's promptcops; the remaining baselines are evaluated as released. Table 2 presents the performance metrics for BAGEL ($k = 9$) at $n = 5$ (Random Selection) and $n = 3$ (w/ DistilBERT), against the other techniques. We evaluate on samples from all datasets collectively rather than one at a time, since incoming prompts in real-world systems are not temporally stratified into attack types.

Table 2: Results of evaluating BAGEL against other methodologies on the held-out test set. Size ratio is each method's parameter count relative to BAGEL (430M, n=5).

Method	ASR↓	FPR↓	F1↑	Bal. Acc.↑	Parameters	Size Ratio	Fine-tunable?
BAGEL	0.342	0.307	0.757	0.675	430M for n=5	1.0×	✓
BAGEL w/ DistilBERT	0.325	0.288	0.772	0.694	324M for n=3	0.8×	✓
ToxicDetector	0.426	0.400	0.679	0.587	300M + 7B	17.0×	✓
StruQ	0.065	0.732	0.880	0.602	7B	16.3×	✓
ShieldGemma	0.431	0.068	0.717	0.751	2B	4.7×	✓
OpenAIModerationAPI	0.528	0.089	0.631	0.692	Unknown	-	✗
LastLayer	0.776	0.142	0.356	0.541	-	-	✗

Most baselines keep FPR low at the cost of missed attacks. ShieldGemma, OpenAIModerationAPI, and LastLayer stay at or below 0.142 FPR but miss 43–78% of attacks. ToxicDetector, optimized for semantic toxicity (e.g., insults, profanity) rather than adversarial structure, is weak on both fronts, missing 43% of attacks, many of which use polite, non-toxic language, while also flagging 40% of benign prompts. StruQ is the exception, attaining the highest F1 (0.880) and lowest ASR (0.065)

by flagging 89% of all prompts, including 73% of benign ones. On our malicious-heavy test set, F1 rewards this behavior: a trivial detector that flags every prompt would score 0.886.

Excluding StruQ's near-indiscriminate flagging, BAGEL achieves the highest F1 score while remaining significantly more lightweight than the other techniques of known size, demonstrating that a specialized ensemble of small models is a viable alternative to massive, monolithic guardrails. BAGEL achieves this due to a much lower ASR than other baselines at a moderate FPR. It also remains easy to update since adding a new promptcop requires finetuning just an 86M model on a new dataset, while other methods must be finetuned or retrained from scratch on many more parameters. Under balanced accuracy, which is insensitive to class prevalence, ShieldGemma ranks first (0.751) owing to its much lower FPR, followed by BAGEL w/ DistilBERT (0.694), OpenAIModerationAPI (0.692), BAGEL at n=5 (0.675), StruQ (0.602), ToxicDetector (0.587), and LastLayer (0.541).

Two factors temper BAGEL's higher FPR. First, the benign portion of our test set is harder than typical traffic: at least 19% of benign prompts (WildJailbreak's `adversarial_benign` split) are deliberately jailbreak-phrased despite being harmless, so Table 2 likely overstates false positives. Second, since BAGEL's promptcop thresholds are calibrated independently per model (§3.5), they can be re-swept to minimize FPR rather than maximize F1 whenever a deployment favors precision over recall, making its higher FPR an adjustable operating point rather than a fixed limitation.

5.4 Multi-turn Attacks Analysis

To evaluate BAGEL against other methods on multi-turn adversarial conversations, we construct a mostly balanced set of 2,400 conversations sampled from three datasets: Multi-Turn Human Jailbreaks (MHJ) (Li et al., 2024), SafeMTData (Ren et al., 2025), and Multi-Turn Intent Dataset (MTID) (Shen et al., 2026). We evaluate two detection strategies: **Full-Conversation**, which concatenates all turns and tries to provide the entire conversation as a single input, and **Chunk-Level**, which scores each turn independently and flags the conversation if any turn is classified as malicious. Due to the inherent Prompt Guard tokenizer length limit of 512 tokens, some conversations cannot be directly provided as input in full to BAGEL, therefore they are truncated to 512 tokens for the Full-Conversation strategy. In order to measure true zero-shot, out-of-distribution generalization performance, only the single-turn finetuned promptcops from the previous experiments are used (Random Selection only). Other methods are also not finetuned on the multi-turn dataset. Results are shown in Table 3.

Table 3: Multi-turn attack detection performance (ToxicDetector with Chunk-Level is not evaluated due to training incompatibilities).

	Strategy	F1↑	ASR↓	FPR↓
BAGEL ($k = 9$) $n = 1$	Full-Conv.	0.636	0.232	0.593
	Chunk-Level	0.625	0.117	0.862
BAGEL ($k = 9$) $n = 5$	Full-Conv.	0.660	0.106	0.748
	Chunk-Level	0.648	**0.054**	0.894
BAGEL ($k = 9$) $n = 9$	Full-Conv.	**0.704**	0.155	0.508
	Chunk-Level	0.664	0.180	0.597
OpenAI Mod-API	Full-Conv.	0.192	0.886	**0.062**
	Chunk-Level	0.375	0.709	0.239
Shield-Gemma	Full-Conv.	0.258	0.835	0.100
	Chunk-Level	0.312	0.775	0.196
Toxic-Detector	Full-Conv.	0.647	0.486	0.067
	Chunk-Level	–	–	–

For BAGEL, both strategies improve with increasing selection size; Full-Conversation reaches F1 = 0.704 at $n = 9$, a notable result given that no promptcop was trained on multi-turn data. While performance does not match that of the single-turn evaluations, BAGEL achieves the highest F1 for $n \geq 5$. All other methods yield low FPR but at the cost of substantially higher ASR, whereas BAGEL trades a higher FPR for far lower ASR, yielding the highest F1 for both strategies at n=9; ToxicDetector's much lower FPR, however, gives it higher balanced accuracy (0.72 vs. 0.67).

6 Conclusion

We introduced BAGEL, a modular ensemble framework for malicious prompt detection built around stochastic selection over specialized 86M-parameter promptcops, optionally augmented by a lightweight learned router. On a held-out, source-disjoint OOD test set, BAGEL achieves an F1 of 0.757 with 430M parameters, improving to 0.772 with the router at 324M parameters, outperforming most substantially larger guardrails. By supporting incremental updates through the addition of a fine-tuned promptcop, BAGEL offers a practical path toward LLM safety systems that remain efficient and adaptable as prompt attacks continue to evolve.

AI USE STATEMENT

In this work, we used generative AI tools for editing and proofreading the manuscript, LaTeX formatting for mathematical equations, and assistance with analysis and plotting code. We have not used generative AI tools for drafting large sections of our paper, creating synthetic datasets or perform a literature search. All AI-assisted text and code was reviewed, verified, and revised by multiple authors, who take full responsibility for the content of this paper.

ETHICS STATEMENT

This work aims to improve LLM safety. We discuss potential misuse and deployment risks in Appendix A, omit unredacted harmful prompts from the paper, and use all third-party artifacts under their original licenses (Appendix J).

REPRODUCIBILITY STATEMENT

We take several steps to support reproducibility. Our anonymized code repository (footnote 1) contains the implementation of `BAGEL`, including promptcop finetuning, threshold calibration, DistilBERT routing, and evaluation. §3 formally specifies the selection, routing, and aggregation procedures (Eqs. 1–6), and §3.2 and Appendix F describe how each dataset is partitioned into finetuning and calibration splits. All training and evaluation datasets are publicly available and listed with their sources in Tables 5 and 6, including the class composition of the source-disjoint test set. Appendix D reports the training configuration, hardware, and threshold selection criterion, §4 defines all evaluation metrics, and §5.4 describes the construction of the multi-turn evaluation set and both detection strategies. Baseline configurations, including model variants used, are given in Appendix H, and the procedure behind our distributional similarity analysis is described in Appendix G.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025. URL https://arxiv.org/abs/2508.10925.

Justin Albrethsen, Yash Datta, Kunal Kumar, and Sharath Rajasekar. Deepcontext: Stateful real-time detection of multi-turn adversarial intent drift in llms. *arXiv preprint arXiv:2602.16935*, 2026. URL https://arxiv.org/abs/2602.16935.

Maksym Andriushchenko and Nicolas Flammarion. Does refusal training in llms generalize to the past tense? *arXiv preprint arXiv:2407.11969*, 2024.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024. URL https://arxiv.org/abs/2404.02151.

Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051*, 2020. URL https://arxiv.org/abs/2007.03051.

arekusandr. Last layer: Ultra-fast, low latency llm prompt injection/jailbreak detection, 2024. https://github.com/arekusandr/last_layer.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655.

Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016. URL https://arxiv.org/abs/1605.07678.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

Guorui Chen, Yifan Xia, Xiaojun Jia, Zhijiang Li, Philip Torr, and Jindong Gu. Llm jailbreak detection for (almost) free! In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025a. doi: 10.18653/v1/2025.findings-emnlp.309. URL https://aclanthology.org/2025.findings-emnlp.309/.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. Struq: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2383–2400, 2025b. URL https://www.usenix.org/conference/usenixsecurity25/presentation/chen-sizhe.

Zhaorun Chen, Mintong Kang, and Bo Li. Shieldagent: Shielding agents via verifiable safety policy reasoning. In *Forty-second International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=DkRYImuQA9.

Junjie Chu, Mingjie Li, Ziqing Yang, Ye Leng, Chenhao Lin, Chao Shen, Michael Backes, Yun Shen, and Yang Zhang. Jades: A universal framework for jailbreak assessment via decompositional scoring, 2025. URL https://arxiv.org/abs/2508.20848.

Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023. doi: https://doi.org/10.1016/j.suscom.2023.100857. URL https://www.sciencedirect.com/science/article/pii/S2210537923000124.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W Mahoney, and Kurt Keutzer. Ai and memory wall. *IEEE Micro*, 44(3):33–39, 2024. doi: 10.1109/MM.2024.3373763.

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. Aegis2.0: A diverse ai safety dataset and risks taxonomy for alignment of llm guardrails. In *NAACL (Long Papers)*, pp. 5992–6026, 2025. URL https://doi.org/10.18653/v1/2025.naacl-long.306.

Hangzhi Guo, Pranav Narayanan Venkit, Eunchae Jang, Mukund Srinath, Wenbo Zhang, Bonam Mingole, Vipul Gupta, Kush R Varshney, S Shyam Sundar, and Amulya Yadav. Exposing ai bias by crowdsourcing: Democratizing critique of large language models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 8-2, pp. 1169–1180, 2025. doi: 10.1609/aies.v8i2.36620. URL https://doi.org/10.1609/aies.v8i2.36620.

Harelix. Harelix/prompt-injection-mixed-techniques-2024, 2024. https://ai.gitee.com/hf-datasets/Harelix/Prompt-Injection-Mixed-Techniques-2024.

hlyn. Prompt injection detection dataset, 2026. URL https://huggingface.co/datasets/hlyn-labs/prompt-injection-judge-deberta-dataset.

Jack Hao. jackhhao/jailbreak-classification, 2023. https://huggingface.co/datasets/jackhhao/jailbreak-classification.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 03 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

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

jayavibhavnk. jayavibhav/prompt-injection-safety, 2024. https://huggingface.co/datasets/jayavibhav/prompt-injection-safety.

Mohammed Amine Jebbar. Malicious prompt detection dataset (mpdd), 2025. https://www.kaggle.com/datasets/mohammedaminejebbar/malicious-prompt-detection-dataset-mpdd.

Yuqi Jia, Yupei Liu, Zedian Shao, Jinyuan Jia, and Neil Gong. Promptlocate: Localizing prompt injection attacks. *arXiv preprint arXiv:2510.12252*, 2025. URL https://arxiv.org/abs/2510.12252.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models, 2024. URL https://arxiv.org/abs/2406.18510.

Jigsaw. Perspective api. https://perspectiveapi.com/.

Pradip Kunwar, Minh N. Vu, Maanak Gupta, Mahmoud Abdelsalam, and Manish Bhattarai. Tt-lora moe: Using parameter-efficient fine-tuning and sparse mixture-of-experts. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '25, pp. 1332–1350, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714665. doi: 10.1145/3712285.3759888. URL https://doi.org/10.1145/3712285.3759888.

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020. URL https://arxiv.org/abs/2004.06660.

Nathaniel Li, Ziwen Han, Ian Steneker, Willow Primack, Riley Goodside, Hugh Zhang, Zifan Wang, Cristina Menghini, and Summer Yue. Llm defenses are not robust to multi-turn human jailbreaks yet. *arXiv preprint arXiv:2408.15221*, 2024. URL https://arxiv.org/abs/2408.15221.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/f5496252609c43eb8a3d147ab9b9c006-Paper.pdf.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023. URL https://arxiv.org/abs/2310.04451.

Yi Liu, Junzhe Yu, Huijia Sun, Ling Shi, Gelei Deng, Yuqi Chen, and Yang Liu. Efficient detection of toxic prompts in large language models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 455–467, 2024a. doi: 10.1145/3691620.3695018. URL https://doi.org/10.1145/3691620.3695018.

Yi Liu, Junzhe Yu, Huijia Sun, Ling Shi, Gelei Deng, Yuqi Chen, and Yang Liu. Toxicdetector evaluation dataset (safetypromptcollections), 2024b. https://sites.google.com/view/toxic-prompt-detector/open-science-artifact?authuser=0.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 1831–1847, 2024c. URL https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37-12, pp. 15009–15018, 2023. doi: 10.1609/aaai.v37i12.26752. URL https://doi.org/10.1609/aaai.v37i12.26752.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *Advances in Neural Information Processing Systems*, 37:61065–61105, 2024. URL https://openreview.net/forum?id=SoM3vngOH5.

Meta-LLAMA. Llama-guard-3-, 2025a. https://www.llama.com/docs/model-cards-and-prompt-formats/llama-guard-3/.

Meta-LLAMA. Prompt-guard-2-86m, 2025b. https://huggingface.co/meta-llama/Llama-Prompt-Guard-2-86M.

Necent. Necent/llm-jailbreak-prompt-injection-dataset, 2026. https://huggingface.co/datasets/Necent/llm-jailbreak-prompt-injection-dataset.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David Wagner. Jatmo: Prompt injection defense by task-specific finetuning. In *European Symposium on Research in Computer Security*, pp. 105–124. Springer, 2024. doi: 10.1007/978-3-031-70879-4_6. URL https://doi.org/10.1007/978-3-031-70879-4_6.

Qualifire AI. qualifire/prompt-injections-benchmark, 2025. https://huggingface.co/datasets/qualifire/prompt-injections-benchmark.

Qibing Ren, Hao Li, Dongrui Liu, Zhanxu Xie, Xiaoya Lu, Yu Qiao, Lei Sha, Junchi Yan, Lizhuang Ma, and Jing Shao. Llms know their vulnerabilities: Uncover safety gaps through natural distribution shifts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24763–24785, 2025. doi: 10.18653/v1/2025.acl-long.1207. URL https://aclanthology.org/2025.acl-long.1207/.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The crescendo multi-turn llm jailbreak attack. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2421–2440, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/russinovich.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Arman Shehabi, Alex Newkirk, Sarah J Smith, Alex Hubbard, Nuoa Lei, Md Abu Bakar Siddik, Billie Holecek, Jonathan Koomey, Eric Masanet, and Dale Sartor. 2024 united states data center energy usage report. *eScholarship*, 2024. doi: 10.71468/P1WC7Q. URL https://escholarship.org/uc/item/32d6m0d1.

Guobin Shen, Dongcheng Zhao, Haibo Tong, Jindong Li, Feifei Zhao, and Yi Zeng. Safety instincts: Llms learn to trust their internal compass for self-defense. *arXiv preprint arXiv:2510.01088*, 2025. URL https://arxiv.org/abs/2510.01088.

Xinjie Shen, Rongzhe Wei, Peizhi Niu, Haoyu Wang, Ruihan Wu, Eli Chien, Bo Li, Pin-Yu Chen, and Pan Li. One turn too late: Response-aware defense against hidden malicious intent in multi-turn dialogue. *arXiv preprint arXiv:2605.05630*, 2026. URL https://arxiv.org/abs/2605.05630.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1671–1685, 2024. doi: 10.1145/3658644.3670388. URL https://doi.org/10.1145/3658644.3670388.

SynapseCAI. synapsecai/synthetic-prompt-injections, 2024. https://www.oxen.ai/synapsecai/synthetic-prompt-injections.

Neil C Thompson, Kristjan Greenewald, Keeheon Lee, Gabriel F Manso, et al. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020. URL https://arxiv.org/abs/2007.05558.

Roberto Verdecchia, June Sallou, and Luís Cruz. A systematic review of green ai. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(4):e1507, 2023. doi: https://doi.org/10.1002/widm.1507. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1507.

Muntasir Wahed, Xiaona Zhou, Kiet A Nguyen, Tianjiao Yu, Nirav Diwan, Gang Wang, Dilek Hakkani-Tür, and Ismini Lourentzou. Mocha: Are code language models robust against multi-turn malicious coding prompts? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. doi: 10.18653/v1/2025.findings-emnlp.1249. URL https://aclanthology.org/2025.findings-emnlp.1249/.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pp. 35413–35425. PMLR, 2023. URL https://proceedings.mlr.press/v202/wan23b/wan23b.pdf.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fd6613131889a4b656206c50a8bd7790-Paper-Conference.pdf.

Tongyu Wen, Chenglong Wang, Xiyuan Yang, Haoyu Tang, Yueqi Xie, Lingjuan Lyu, Zhicheng Dou, and Fangzhao Wu. Defending against indirect prompt injection by instruction detection. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. doi: 10.18653/v1/2025.findings-emnlp.1060. URL http://dx.doi.org/10.18653/v1/2025.findings-emnlp.1060.

Zixuan Weng, Xiaolong Jin, Jinyuan Jia, and Xiangyu Zhang. Foot-in-the-door: A multi-turn jailbreak for llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1939–1950, 2025. doi: 10.18653/v1/2025.emnlp-main.100. URL https://aclanthology.org/2025.emnlp-main.100/.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, et al. Guardagent: Safeguard llm agents via knowledge-enabled reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2nBcjCZrrP.

Haomiao Yang, Kunlan Xiang, Mengyu Ge, Hongwei Li, Rongxing Lu, and Shui Yu. A comprehensive overview of backdoor attacks in large language models within communication networks. *IEEE Network*, 38(6):211–218, 2024. doi: 10.1109/MNET.2024.3367788.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211, 2024. doi: https://doi.org/10.1016/j.hcc.2024.100211. URL https://www.sciencedirect.com/science/article/pii/S266729522400014X.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*, 2024. URL https://arxiv.org/abs/2407.04295.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023. URL https://arxiv.org/abs/2309.10253.

Zhiyuan Yu, Xiaogeng Liu, Shunning Liang, Zach Cameron, Chaowei Xiao, and Ning Zhang. Don't listen to me: Understanding and exploring jailbreak prompts of large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 4675–4692, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/yu-zhiyuan.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, et al. Shieldgemma: Generative ai content moderation based on gemma. *arXiv preprint arXiv:2407.21772*, 2024. URL https://arxiv.org/abs/2407.21772.

Jiang Zhang, Qiong Wu, Yiming Xu, Cheng Cao, Zheng Du, and Konstantinos Psounis. Efficient toxic content detection by bootstrapping and distilling large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38-19, pp. 21779–21787, 2024. doi: 10.1609/aaai.v38i19.30178. URL https://doi.org/10.1609/aaai.v38i19.30178.

Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Transactions on Software Engineering and Methodology*, 2025. doi: 10.1145/3724393. URL https://doi.org/10.1145/3724393.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, et al. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *Proceedings of the 1st ACM workshop on large AI systems and models with privacy and safety analysis*, pp. 57–68, 2023. doi: 10.1145/3689217.3690621. URL https://doi.org/10.1145/3689217.3690621.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023. URL https://arxiv.org/abs/2307.15043.

A LIMITATIONS

While BAGEL demonstrates strong performance across diverse attacks, some limitations warrant discussion and point towards promising future research directions.

Binary classification scope: As stated in our threat model (§C), we prioritize rapid detection of malicious prompts to prevent harmful content generation. In this context, BAGEL's binary classification (benign vs. malicious) is sufficient and efficient. However, some deployment scenarios require fine-grained classification to identify which specific safety or ethical policy has been violated. In such cases, BAGEL would need to be extended beyond binary detection, unlike systems such as OpenAI Moderation (Markov et al., 2023) or Perspective (Jigsaw) that provide multi-class policy violation categories. Future work could explore hierarchical classification where BAGEL first performs binary detection, followed by a secondary fine-grained classifier for policy-specific attribution. Alternatively, BAGEL could utilize larger base models when additional compute resources are available. For instance, Llama Guard 3 (Meta-LLAMA, 2025a), available in 1B, 8B, and 11B variants, could offer superior performance and native fine-grained classification capabilities while maintaining the ensemble architecture's benefits of modularity and incremental adaptability.

Dependence on base model coverage: While BAGEL demonstrates efficient adaptation to new attacks through incremental fine-tuning, this adaptability partly relies on the base model having learned relevant underlying attack classes. In our implementation, Prompt Guard 2 was pre-trained on jailbreak and prompt injection attacks, providing a foundation that each fine-tuned promptcop could build upon. However, it remains an open question whether entirely novel attack paradigms that differ fundamentally from these established categories could emerge in practice. To our knowledge, the LLM safety landscape has not yet seen such fundamentally different attack types; if they were to arise, it is unclear whether fine-tuning alone would suffice or whether the base model would need retraining. That said, we note that the three attack categories we address (direct harmful requests, jailbreaks, and prompt injections) are quite broad and have encompassed a wide variety of techniques to date, including recent variations such as multi-lingual attacks, encoded prompts, and adversarial suffixes. This suggests that BAGEL's incremental adaptation mechanism may have broader applicability than initially apparent. Empirically evaluating BAGEL on emerging attack variations represents an important direction for validating these boundaries. Additionally, training a purpose-built prompt safety classifier optimized for BAGEL's architecture could provide stronger guarantees against diverse threats while maintaining efficiency.

Dataset requirements for new attacks: Each new promptcop requires a sufficiently large and representative dataset of the target attack type for effective fine-tuning. When novel attacks first emerge, collecting adequate training data can be challenging and time-intensive. This creates a detection gap between when a new attack appears and when BAGEL can be updated with a corresponding promptcop. While this limitation affects all supervised learning approaches, it underscores the importance

of rapid dataset curation and potentially incorporating few-shot or zero-shot detection capabilities for emerging threats.

Potential risks: Although BAGEL is intended to improve the safety of LLM deployments by detecting malicious prompts, the same detection capabilities could be misused by adversaries to probe which prompt structures are more likely to evade a deployed guardrail. To reduce this risk, we release only information necessary for reproducibility and evaluation, and do not include unredacted harmful prompt examples in the main paper. Another deployment risk is over-reliance: BAGEL should not be treated as a complete safety solution, since false negatives may still allow harmful prompts through and false positives may incorrectly block benign user requests. In high-stakes settings, BAGEL should therefore be used as one layer in a broader defense pipeline, alongside output-side monitoring, policy enforcement, human review where appropriate, and continuous evaluation on emerging attack types.

Continuing from the aforementioned limitations, future work could explore the possibility of utilizing a different malicious prompt classification model or the challenges associated with training a new model from scratch. BAGEL uses Prompt Guard 2 86M as it is a lightweight yet powerful model, but if more compute is available to the LLM vendor, using another model to create finetunes, such as Llama Guard 3 (Meta-LLAMA, 2025a) which is available in 1B, 8B and 11B variants, may offer superior performance along with the ability to perform fine-grained classification rather than just binary classification. Furthermore, training a prompt classifier optimized for the techniques outlined in this paper from scratch would enable faster responses to ever-evolving threat vectors, even if they encompass techniques currently unknown or unseen by the base model being employed at the core of BAGEL.

B RELATED WORK

Malicious Attacks on LLMs: Prior work categorizes instruction-based attacks into three types (Liu et al., 2024a; Wei et al., 2023; Zou et al., 2023; Liu et al., 2024c; Yao et al., 2024): (1) direct harmful requests, (2) jailbreak attacks that bypass guardrails through deceptive framing, and (3) prompt injection attacks that embed malicious instructions within benign content. Significant research has focused on designing and improving such attacks (Zou et al., 2023; Liu et al., 2023; Yu et al., 2023; Mehrotra et al., 2024; Andriushchenko et al., 2024). We note that BAGEL, like other prompt detection methods, specifically addresses these instruction-based attacks at inference time and is not designed for alternative threat vectors such as data poisoning (Kurita et al., 2020; Wan et al., 2023), backdoor attacks (Yang et al., 2024), or model extraction (Carlini et al., 2021). Below we discuss the different approaches to malicious prompt detection.

Rule-based and statistical detectors: Early approaches to malicious prompt detection rely on statistical signals and heuristics. PerplexityFilter (Jain et al., 2023) and SIRL (Shen et al., 2025) identify harmful prompts by measuring response uncertainty through perplexity and entropy calculations respectively. LastLayer (arekusandr, 2024) uses simple structural detectors to find alarming patterns in prompts, and JailGuard (Zhang et al., 2025) mutates inputs to create variants and detects adversarial prompts through response divergence. While computationally efficient, these methods often struggle with sophisticated attacks that evade simple statistical patterns.

Commercial and black-box APIs: Commercial solutions such as OpenAI Moderation API (Markov et al., 2023) and Perspective API (Jigsaw) provide convenient detection services but offer limited transparency, poor adaptability to domain-specific threats, and no guarantees against rapidly evolving attacks.

LLM-based detectors: Most recent detection methods leverage LLMs for stronger performance at the cost of computational efficiency. ToxicDetector (Liu et al., 2024a) and InstructDetector (Wen et al., 2025) extract features from LLM hidden states for classification. StruQ (Chen et al., 2025b) and FJD (Chen et al., 2025a) use LLMs for prompt scrutinization with additive instructions and first-token confidence analysis respectively. Gpt-oss-safeguard (20B and 120B) (Agarwal et al., 2025) and ShieldGemma (ranging from 2B to 27B parameters) (Zeng et al., 2024) built on Gemma 2, are large models for detecting harmful content by defining custom safety policies. Recent work has also explored guardrails for autonomous agents through dynamic code generation (Chen et al., 2025c;

Xiang et al., 2025). While effective, these approaches require substantial computational resources for both inference and updates.

Hybrid and distillation approaches: Methods closest to BAGEL include Jatmo (Piet et al., 2024), which fine-tunes a non-instruction-tuned LLM resistant to prompt injections and BD-LLM (Zhang et al., 2024), which distills LLM rationales into smaller student models. While these approaches reduce computational costs compared to direct LLM-based detection, they still depend on LLM-scale models or require access to LLM internals.

BAGEL distinguishes itself by operating without LLMs entirely, using only small specialized classifiers (86M parameters each) in a modular ensemble architecture. This design achieves strong detection performance (F1 score of 0.772 on the OOD set) while significantly reducing computational overhead and enabling efficient incremental updates through simple fine-tuning and ensemble addition, critical advantages as new attack types emerge.

C Threat Model

We consider an attacker who inputs one or multiple malicious prompt instructions into the chat interface of an LLM application. The attacker is motivated to enter a malicious prompt to access objectionable content according to policies defined by the LLM serving hosts and/or societal norms. We assume that the attacker does not have access to the underlying system prompt or any other implementation details due to the LLM being served remotely. Furthermore, we make no assumptions regarding the attacker's level of expertise with adversarial prompting.

We focus on binary classification (benign vs. malicious) rather than fine-grained sub-classification of attack types or specific policy violations. While identifying the exact attack category could provide additional insights, binary detection is sufficient to prevent harmful or policy violating outputs and aligns with the immediate safety requirements of production systems. We assume the defender has prior exposure to the general categories of malicious prompts (e.g., direct harmful requests, jailbreaks, and prompt injections), even if specific attack instances or novel techniques have not been observed during training. This assumption aligns with the LLM safety landscape where adversaries typically develop novel variations of established jailbreak and prompt injection patterns rather than fundamentally new attack categories.

D Experiment Environment Details

All promptcops created by fine-tuning a base instance of Prompt Guard on a dataset were finetuned for 3 epochs across the dataset's finetuning set. Furthermore, all experiments were carried out in an online Google Colab environment with an A100 GPU.

We report F1, ASR, and FPR on the held-out test sets, and use the maximum F1 score over thresholds on the calibration set to select the decision threshold.

E Inference Cost and Memory Footprint

Table 4 summarizes BAGEL's runtime and memory costs, measured on the same A100 GPU used for all experiments (Appendix D). Each promptcop is a Prompt Guard 2 instance requiring 1.04 GB of model weights. Because promptcops are independent, BAGEL's total storage footprint grows linearly with the ensemble size k, while the active GPU memory at inference grows only with the selection size n, assuming all selected promptcops are loaded onto the GPU in parallel.

Throughput saturates quickly with batch size, increasing only marginally from 197.05 to 203.65 samples/s when the batch size grows from 64 to 256. Since each prompt is scored by n promptcops, running them sequentially on a single GPU yields a throughput of roughly $1/n$ of the per-promptcop figure, whereas loading the selected promptcops in parallel keeps per-prompt latency close to that of a single forward pass at the cost of $n\times$ the active memory. In either case, the cost of updating BAGEL remains fixed: adding a new promptcop requires finetuning a single base Prompt Guard 2 instance for 3 epochs over the new dataset's finetuning split, independent of the ensemble's current size.

Table 4: Measured per-promptcop inference cost and resulting memory footprint of BAGEL.

Quantity	Value
Weights per promptcop	1.04 GB
Storage, full ensemble (k=9)	9.36 GB
Active memory, BAGEL (n=5)	5.20 GB
Active memory, BAGEL w/ DistilBERT (n=3)	3.12 GB + router
Forward pass per promptcop (batch size 64)	323.8 ms
Throughput per promptcop (batch size 64)	197.05 samples/s
Throughput per promptcop (batch size 256)	203.65 samples/s

F Single Turn Dataset Details

Table 5: Collection of the nine datasets we used for single-turn finetuning and calibration. • indicates the dataset contains this specific type of malicious prompts, while ○ represents the lack of any such type.

Dataset	Number of Samples	Type of Malicious Prompts		
		Simple Harmful	Jailbreak	Injection
synapsecai/synthetic-prompt-injections (SynapseCAI, 2024)	252,956	○	○	•
Malicious Prompt Detection Dataset (MPDD) (Jebbar, 2025)	39,234	•	•	•
TrustAIRLab/in-the-wild-jailbreak-prompts (Shen et al., 2024)	13,700	○	•	○
Harelix/Prompt-Injection-Mixed-Techniques-2024 (Harelix, 2024)	1,175	•	○	○
jackhhao/jailbreak-classification (Jack Hao, 2023)	1,042	○	•	○
qualifire/prompt-injections-benchmark (Qualifire AI, 2025)	5,000	○	•	○
jayavibhav/prompt-injection-safety (jayavibhavnk, 2024)	60,000	•	•	•
ToxicDetector Evaluation Dataset (Liu et al., 2024b)	2,033	•	○	○
nvidia/Aegis-AI-Content-Safety-Dataset-2.0 (Ghosh et al., 2025)	33,416	•	•	○
Total	**408,556**	5	6	3

The details of the nine individual datasets that were used in our selection efficiency and adaptability experiments are provided in Table 5. The aggregation of these datasets resulted in a total corpus of 408,556 samples. Following the partitioning strategy defined earlier (§3.2), each dataset is split $70\%/10\%$ into finetuning and calibration sets respectively; the remaining 20% of each dataset is unused.

Final evaluation is conducted on a source-disjoint test set $\mathcal{T}$, constructed from five datasets entirely separate from $\mathcal{D}$, detailed in Table 6.

G Dataset Distributional Similarity

To provide additional evidence for the distributional separation between BAGEL's training sources and the source-disjoint OOD test set, Figure 5 shows pairwise Jensen-Shannon divergence between the prompt-text distributions of all nine training datasets and all five OOD test datasets. Higher divergence (yellow) indicates greater dissimilarity between two datasets' content; lower divergence

Table 6: Collection of the five datasets used to construct the source-disjoint test set $\mathcal{T}$. All 210 benign WildJailbreak prompts come from its `adversarial_benign` split.

Dataset	Malicious	Benign	Total
hlyn-labs/prompt-injection-judge-deberta-dataset (hlyn, 2026)	484	516	1,000
allenai/wildjailbreak (Jiang et al., 2024)	2,000	210	2,210
LLM Past Tense (Andriushchenko & Flammarion, 2024)	730	0	730
Necent/llm-jailbreak-prompt-injection-dataset (Necent, 2026)	629	371	1,000
TrustAIRLab/JailbreakQR (Chu et al., 2025)	400	0	400
Total	**4,243**	**1,097**	**5,340**

(dark purple) indicates the two datasets' prompts are drawn from more similar distributions. Datasets along the diagonal trivially show zero divergence against themselves.

`necent` and `hlyn` exhibit comparatively lower divergence against several training datasets, consistent with their construction as aggregations over a much wider pool of public sources, where some degree of content overlap with any individual constituent dataset is expected.

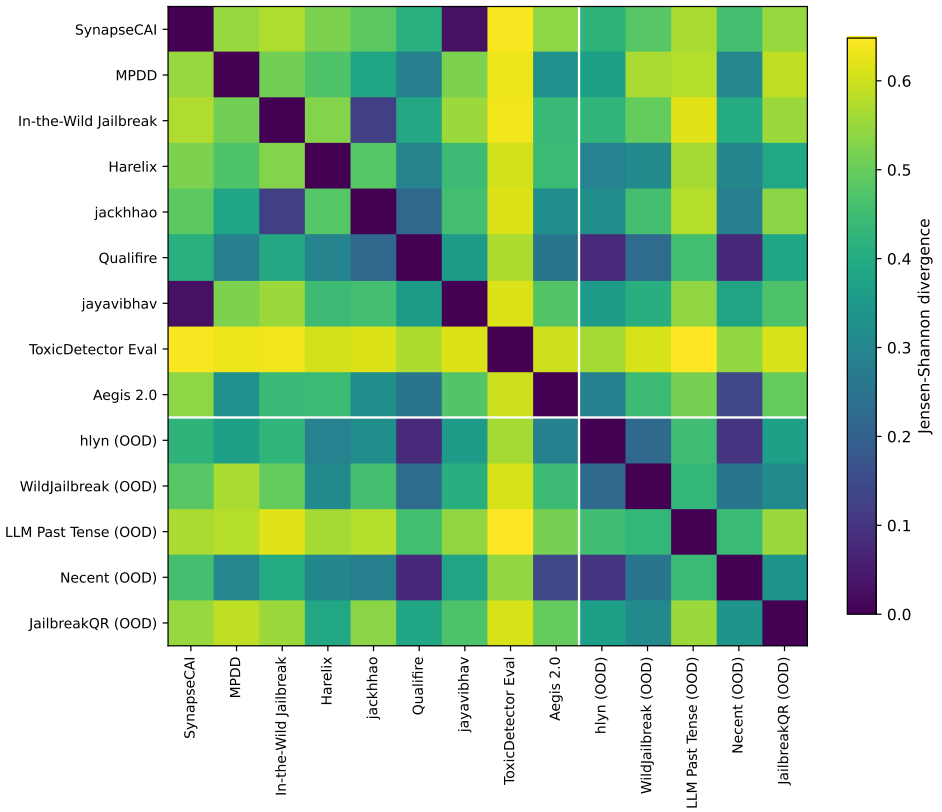


Figure 5: Pairwise Jensen-Shannon divergence between the nine training datasets and the five OOD test datasets. Prompts are embedded with `all-mpnet-base-v2`, the shared embedding space is partitioned into 50 clusters via k-means, and each dataset's distribution is represented as its normalized histogram of prompts per cluster. Darker cells indicate greater distributional similarity between a pair of datasets.

H BASELINE DETAILS

- **OpenAIModerationAPI** (Markov et al., 2023): A widely deployed, industrial-grade black-box API designed to detect text that violates safety policies. It comprises an unknown number of parameters although it likely has more parameters than our method due to the API's ability to further classify the exact type of content policy violation, such as 'hate', 'self-harm', 'harassment', 'violence' etc. It is otherwise unable to be tweaked rapidly to specific datasets and newer attacks.
- **ToxicDetector** (Liu et al., 2024a)**:** A grey-box methodology designed to perform binary classification for benign or toxic prompts. It uses embeddings from an LLM (such as Llama-2-7B, which we use in our experiments) as the feature vectors for a 300M parameter MLP classifier.
- **ShieldGemma** (Zeng et al., 2024)**:** These are white-box, instruction-tuned models for evaluating the safety of text. ShieldGemma 1 is built upon the Gemma 2 LLM in 2B, 9B, and 27B parameter sizes (we use the 2B one in our experiments), and allows for a custom safety policy to be provided in the system prompt.
- **LastLayer** (arekusandr, 2024)**:** This is a partially black-boxed security library for protecting LLMs from malicious attacks. Rather than employing a deep-learning-centric approach, LastLayer holistically analyses the structure of incoming prompts via numerous modules, in order to detect patterns indicative of prompt injection attacks, jailbreaks and other exploits.
- **StruQ** (Chen et al., 2025b): A white-box defense against prompt injection that separates instructions from data via structured queries, using a specially instruction-tuned 7B LLM that only follows commands in the designated instruction channel and treats content in the data channel as inert. We adapt it as a detector by flagging any input whose structured completion deviates from expected instruction-following behavior. StruQ is "fine-tunable" only in the sense of retraining the full 7B-parameter LLM via structured instruction tuning, unlike `BAGEL`'s lightweight per-attack classifier fine-tuning.

I DISCUSSION: ENERGY EFFICIENCY AND SUSTAINABILITY

As AI and specifically deep learning become increasingly ubiquitous, growing concerns have emerged regarding the energy costs of deploying large models and their environmental impact (Gholami et al., 2024; Canziani et al., 2016; Anthony et al., 2020; Thompson et al., 2020). Data center energy consumption in the U.S. accounted for 4.4% of total electricity use and is projected to triple by 2028 (Shehabi et al., 2024). Critically, the majority of energy costs in AI systems stem from inference rather than training, as training is typically a one-time process while inference must be sustained continuously at scale (Desislavov et al., 2023). Despite this reality, most research has focused on reducing training costs rather than inference costs (Verdecchia et al., 2023).

`BAGEL` implicitly accounts for these sustainability concerns through its architectural design. By using an 86M parameter base safety classifier and deploying only a small subset of promptcops per inference (typically 5 out of a larger ensemble), `BAGEL` significantly reduces both the computational footprint and energy consumption compared to billion-parameter alternatives. The framework's reliance on bootstrap aggregation and lightweight learned routing, demonstrates that powerful yet efficient methods can achieve strong performance in LLM safety without requiring massive computational resources. This approach not only makes robust LLM safety more accessible to resource-constrained deployments and smaller vendors, but also represents a step toward environmentally sustainable AI safety practices. As LLM systems continue to scale globally, such efficiency-focused architectures will become essential for balancing safety requirements with environmental responsibility.

J ARTIFACT LICENSES AND INTENDED USE

Our experiments use publicly available models, datasets, and software artifacts. We use Prompt Guard 2 as the base prompt-safety classifier, and we follow the usage terms associated with the re-

leased model checkpoint. We also evaluate against publicly accessible or documented baselines, including OpenAI Moderation API, StruQ, ToxicDetector, ShieldGemma, and LastLayer, under their respective usage terms.

For the datasets used in our experiments, we rely on the licenses and terms provided by the original dataset creators or hosting platforms. Our released code is intended for research and reproducibility, and any redistribution or reuse of third-party artifacts remains governed by the original artifact licenses.

We use all third-party artifacts in a manner consistent with their intended research and evaluation purposes. The prompt-safety models and baseline detectors used in this work are applied only for malicious prompt detection and comparative safety evaluation, which aligns with their intended role as prompt moderation or guardrail systems. The datasets used in our experiments are used for research on malicious-prompt detection, robustness, and LLM safety evaluation, rather than for generating or facilitating harmful model behavior.