Enhancing Efficiency and Robustness in Semantic Smoothing: An Algorithmic Approach to LLM Defense

Arman Rahman
arahman@ucsd.edu

Dante Testini
dtestini@ucsd.edu

Shreya Sudan
ssudan@ucsd.edu

Donald Taggart
dtaggart@ucsd.edu

Barna Saha
bsaha@ucsd.edu

Arya Mazumdar
arya@ucsd.edu

Abstract

Due to the impressive ability of modern LLMs to respond with great detail to nearly any prompt posed by humans, there has been increasing concern over this new technology being used to inform those with malicious intent. Various defenses, including SemanticSmooth have been developed yet they often impede on the language model's ability to quickly respond to genuinely benign prompts. This inefficiency can be addressed by taking into consideration the provided prompts' semantics, allowing for a new LLM defense that can achieve a practical balance of being both robust and computationally efficient. This is our motivation for implementing a new algorithmic approach we refer to as SwiftGuard.

Website: https://donaldtaggart.github.io/SwiftGuard-Website/
Code: https://github.com/shreyasudan/SwiftGuard

1 Introduction

Large language models (LLMs) are increasingly deployed across industries, from healthcare to finance. Despite their widespread utility, these models remain vulnerable to jailbreaking attacks that subvert safety guardrails, leading to the generation of objectionable or harmful content. This vulnerability undermines trust and limits LLM deployment in sensitive domains.

SemanticSmooth, a recent innovation in LLM defense, mitigates these risks by employing semantic perturbations and aggregating predictions to neutralize adversarial attacks. (Jil et al. 2017) However, its reliance on reinforcement learning introduces significant computational overhead and performance trade-offs, particularly for benign prompts. With the majority of users using LLMs for benign reasons, the widespread installation of defenses that significantly slow down the process speed does not provide a user-friendly experience. Addressing these inefficiencies is critical for ensuring that SemanticSmooth can be both robust and computationally efficient.

Current implementations of SemanticSmooth depend heavily on reinforcement learning to adapt to adversarial prompts, increasing runtime and computational costs even for innocuous inputs. This challenge highlights the necessity of developing defenses that are both robust and computationally efficient.

Recent advances, such as the Single Pass Detection (SPD) technique (Candogan et al. 2025), offer promising alternatives. SPD detects jailbreaking attempts by leveraging output logits in a single forward pass, significantly reducing computational overhead without sacrificing detection accuracy. Inspired by this efficient approach, we have integrated SPD with a rule-based classifier to develop SwiftGuard, our novel protection system. This integration preserves the robustness of existing defenses while enhancing efficiency, enabling rapid and scalable protection against adversarial attacks.

2 Methods

Our approach builds on the observation that **differences in logit distributions** can effectively distinguish adversarial from benign prompts (Candogan et al. 2025). Leveraging this insight, we introduce SwiftGuard, an efficient, scalable system designed to detect and mitigate jailbreaking attempts on LLMs. This section details the **classifier architecture, single-pass detection pipeline, and trade-offs in pipeline length.**

2.1 Binary Classifier for Query Filtering

SwiftGuard incorporates a lightweight safety filter before the Single-Pass Detection (SPD) module to preprocess and validate user queries.

This classifier forwards each query to a filtering system. The system evaluates the prompt

for safety, legality, and ethical concerns. If the query is determined to be undoubtedly safe, it will not be sent to SPD. If the query is deemed to have any aspect that could be considered unsafe, it will be sent to SPD for further analysis. This preliminary classifier does not rely on a machine learning model but instead acts as a rule-based filter. This simple yet effective mechanism helps filters out explicitly benign prompts, such that they need not be further analyzed.

2.2 Single Pass Detection (SPD)

We implement Single-Pass Detection (SPD) to detect jailbreaking attempts in one forward pass by leveraging logit-based classification. Unlike multi-step approaches, SPD does not require auxiliary models or multiple queries, making it computationally efficient.

2.2.1 Logit-Based Feature Extraction

Given an input sequence $x_{1,n}$ the LLM generates an output sequence $o_{n+1,m}$. The logits associated with the first few output tokens provide a discriminative signal for adversarial detection (Candogan et al. 2025). We extract the following feature matrix:

$$H = [h_1, h_2, \ldots, h_r] \in \mathbb{R}^{r \times k}$$

where:

- $h_i = -\log(\sigma(l_{i,k}))$
- l_i is the logit vector of the ith token
- σ is the soft-max function
- k is the number of top-k highest probability tokens included
- r is the number of initial output tokens considered

For our implementation, we set $r = 5$ and $k = 50$, ensuring robust feature representation.

2.2.2 Binary Classification Model

SPD employs a binary classifier to distinguish between benign and jailbroken prompts based on logit shifts and entropy variations.

- Classifier Architecture:
  - A Support Vector Machine (SVM) with an RBF kernel is trained on the logit-derived feature matrix.
  - The classifier outputs a binary decision: benign (0) or adversarial (1).
- Decision Function: The classifier function $f_{class}(H)$ maps the extracted feature matrix to a decision, such that

$$f_{class}(H) : \mathbb{R}^{r \times k} \to \{0, 1\}$$

  - $0 \to$ Benign input

- 1 → Jailbreaking attempt detected

The decision threshold is determined via ROC curve analysis, optimizing the true positive rate (TPR) and false positive rate (FPR).

2.3 Dataset and Evaluation Metrics

2.3.1 Training and Testing Data

The evaluation of SwiftGuard was conducted using GPT-4o-mini as the target model. The dataset consists of adversarial jailbreak prompts, benign user queries, and natural language inference samples. The data is divided into training and testing sets to assess both model generalization and performance stability.

Table 1: Dataset breakdown for training and evaluation.

Dataset	Train Set Size	Test Set Size	Total Samples
PAIR	20	100	120
GCG	154	378	532
Alpaca	395	395	790
QNLI	495	494	989

Dataset Notes:

- PAIR (Chao et al. 2019) (or Prompt Automatic Iterative Refinement) is a prompt-level attack which uses an auxiliary attack LLM to iteratively generate prompts that consider its previously failed jailbreaking attempts. This attack commonly involves 'role-playing' attempts, creating a story to trick the LLM to accepting the prompt as benign.
- GCG (or Greedy Coordinate Gradient) is a token-level attack which appends the prompt with a suffix optimized to elicit the target output.
- Alpaca and QNLI (or Question-Answering for Natural Language Inference) Datasets include a sample of benign prompts, similar to queries commonly received by LLMs on a daily basis. These datasets have become the standard when evaluating jailbreaking.
- While SPD was trained using the above described number and distribution of test samples, the pipeline involving the rule-based classifier was tested on a random sample of these prompts, containing 200 adversarial and 200 benign prompts.

2.3.2 Evaluation Metrics

The performance of SwiftGuard was assessed using standard classification and computational efficiency metrics.

1. **True Positive Rate (TPR) / Recall:** Evaluates the system's ability to detect adversarial prompts:
$$TPR = \frac{TP}{TP + FN}$$
2. **False Positive Rate (FPR):** Measures the rate at which benign queries are incorrectly classified as adversarial, crucial for ensuring usability:
$$FPR = \frac{FP}{FP + TN}$$
3. **F1-Score:** Balances precision and recall to provide a comprehensive performance measure:
$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
4. **Run Time:** Average time in seconds for each prompt to be processed

3 Results

Table 2: Performance Metrics

Method	TPR ↑	FPR ↓	F1 Score ↑	Time (s) ↓
SwiftGuard	88.08%	9.67%	85.48%	0.3
SwiftGuard with Preliminary Classifier	86.5%	0.5%	92.51%	4.62

4 Discussion

4.1 Pipeline 1 - SwiftGuard

On the dataset, SwiftGuard achieved a True Positive Rate of 88.08%, meaning that 88.08% of adversarial prompts that would have normally produced a harmful response from GPT-4o, would get rejected. The model's success is comparable to other, more complex state of the art models. For example, one version of SemanticSmooth that utilizes an adapting policy network has a True Positive Rate of 80% from PAIR attacks on GPT-3.5-turbo. (Jil et al. 2017)

SwiftGuard also achieved a False Positive Rate of 9.67%. This means that with the implementation of this defense, 9.67% of benign prompts got marked falsely as adversarial, and therefore would be rejected. Although this is also comparable to SemanticSmooth (with a 9.7% False Positive Rate)(Jil et al. 2017), it still does mean that approximately 10 percent of user's day-to-day queries would be rejected, despite not eliciting any harmful results.

The highlight of SwiftGuard's results is the time. On average, a query, be it adversarial or benign, takes 0.3 seconds to go through the SwiftGuard pipeline, with the majority of this time being spent computing the logits. This time is negligible, and an additional 0.3 seconds to a GPT query would not delay the output by a time that would hurt the user.

4.2 Pipeline 2 - SwiftGuard with Preliminary Classifier

With the addition of the Preliminary Classifier, the True Positive Rate of SwiftGuard is 86.5%. This is also comparable to SemanticSmooth, producing a solid defense against harmful prompts. This is slightly less than SwiftGuard itself, but this slight drop is most likely due to the fact that this pipeline was only tested on a subset of the full test data.

This pipeline also received a False Positive Rate of 0.5%, which is drastically lower than SwiftGuard alone. This is intentional, as the Preliminary Classifier is there to identify obviously benign prompts, meaning that most of the benign prompts did not get the chance to be mislabeled as adversarial by SPD. The implementation of this infrastructure to a market LLM would result in only half a percent of prompts being incorrectly denied. The preliminary classifier also increased the F1 Score from 85.48% to 92.51%, showing the comprehensive improvement of the model.

The low point of this Preliminary Classifier Pipeline is the time it takes to run. On average this pipeline takes 4.62 seconds for each prompt, with the majority of the time being taken by the LLM call of the Preliminary Classifier. This takes away from the speed of SwiftGuard itself, causing all users to have to wait almost an extra 5 seconds for the result of a query. If this pipeline were to be implemented on a market LLM, the swiftness of the additional LLM call would have to be optimized better in order for the standard user's experience to not be harmed.

5 Conclusion

With SwiftGuard's True Positive Rate of 88% it appears to classify attacks well, while keeping a low False Positive Rate and a quick runtime. These rates are along the lines of SemanticSmooth, but with a much more simple structure and a quick runtime. When a preliminary classifier is added, the False Positive Rate plummets, while maintaining a similar True Positive Rate. This comes at the price of time, as the overall pipeline takes on average 15 times longer than the logit production and classification alone. The choice between adding a preliminary classifier in practice would come down to whether time or False Positive Rate is valued more by the industry of focus.

Overall, through utilizing the differences in the logits of adversarial and benign responses, SwiftGuard protects the LLM with comparable accuracy and improved speed relative to other cutting-edge methods such as SemanticSmooth, preserving the experience of the standard user.

References

Candogan, Leyla Naz, Yongtao Wu, Elias Abad Rocamora, Grigorios G. Chrysos, and Volkan Cevher. 2025. "Single-pass Detection of Jailbreaking Input in Large Language Models." *Transactions on Machine Learning Research*. [Link]

Chao, Patrick, Alexander Robey, Edgar Dobriban, Hamed Hassani, George Pappas, and Eric Wong. 2019. "Jailbreaking Black Box Large Language Models in Twenty Queries." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics. [Link]

Jil, Jiabao, Bairu Hou, Alexander Robey, George Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. 2017. "DEFENDING LARGE LANGUAGE MODELS AGAINST JAILBREAK ATTACKS VIA SEMANTIC SMOOTHING." In *Advances in Neural Information Processing Systems*. [Link]

Appendices

A.1 Preliminary Classifier

The details for the Preliminary Classifier's prompt that is designed to point out and discard obviously benign prompts. This also appears in check_text.txt in the repository.

You are an assistant responsible for ensuring that user queries do not lead to illegal, unsafe, or unethical outcomes when processed by Large Language Models. Your task is to evaluate the given query for any potential violations of safety, legality, or ethics. Please carefully review the query and re-assess it multiple times to ensure it adheres to all relevant guidelines. If you determine that the query might result in harmful, illegal, or unethical responses, reply with 'NO'. If the query is entirely safe and does not present any ethical concerns, reply with 'YES'. Only respond with 'YES' or 'NO' and nothing else.

Below is the query to evaluate: "QUERY"

A.2 Project Proposal

Our most recent Project Proposal

A.2.1 Problem Statement

Large language models (LLMs) are increasingly deployed across industries, from healthcare to finance. Despite their widespread utility, these models remain vulnerable to jailbreaking attacks that subvert safety guardrails, leading to the generation of objectionable or harmful content. This vulnerability undermines trust and limits LLM deployment in sensitive domains.

SemanticSmooth, a recent innovation in LLM defense, mitigates these risks by employing semantic perturbations and aggregating predictions to neutralize adversarial attacks. (Jil et al. 2017) However, its reliance on reinforcement learning introduces significant computational overhead and performance trade-offs, particularly for benign prompts. With the majority of users using LLMs for benign reasons, the widespread installation of defenses

that significantly slow down the process speed does not provide a user-friendly experience. Addressing these inefficiencies is critical for ensuring that SemanticSmooth can be both robust and computationally efficient.

Current implementations of SemanticSmooth rely heavily on reinforcement learning to adapt to adversarial prompts. This approach increases runtime and computational costs even for benign inputs, making it less practical for real-world applications. The challenge lies in maintaining the robustness of SemanticSmooth while reducing its reliance on computationally expensive techniques.

Our proposal focuses on replacing the reinforcement learning component with an algorithmic framework that is more efficient and does not have to rely on an ever changing policy network. This framework will identify attacks based on detectable semantic perturbations and apply appropriate transformations selectively. This method will not only preserve robustness but also enhance efficiency. Preliminary research from PAIR (Chao et al. 2019) and SemanticSmooth papers suggests that certain semantic transformations (e.g., summarization or paraphrasing) can mitigate attacks without significant computational penalties.

A.2.2 Project Impact

By optimizing SemanticSmooth, this project would lead to a reduction in computational costs of LLM defenses. This optimized algorithm would be a system that is robust against adversarial attacks but does not compromise the performance of benign inputs. This can open the door for these defenses to be scalable and deployable across diverse environments. Additionally, this project could lead to more trust and safety in LLMs in sensitive domains, enabling the broader adoption of trusted LLMs in fields like healthcare, education and finance.

A.2.3 Methods and Feasibility

With regards to data, we will be using jailbreak prompts from the PAIR and SemanticSmooth experiments, including the AdvBench and InstructionFollow datasets. (**?**) Using this data, we will design a rule-based detection mechanism informed by attack patterns observed in PAIR (e.g., role-playing narratives) and transformations used in SemanticSmooth (e.g., summarization, paraphrasing) to achieve our goals.

We will implement an attack detection framework, which will be made up of a lightweight algorithmic detector that can classify inputs as benign or adversarial based on semantic cues. This algorithm will be assisted by signal processing techniques that will be used to identify patterns typical of common attacks like PAIR and GCG. This will be the framework of a dynamic transformation application that will apply targeted semantic transformations *only* to inputs flagged as adversarial. The selection of transformations will be optimized

using a heuristic approach to minimize runtime impact.

To evaluate our model, we will measure robustness against state-of-the-art attacks, such as PAIR, GCG and AutoDAN, using metrics such as attack success rate. We will also measure the runtime of our model with both benign and adversarial prompts. These statistics will be used to compare our model against the reinforcement learning-based model, Semantic-Smooth, to demonstrate improvements in efficiency and robustness.

A.2.4 Expected Output

The primary deliverable will be a technical report detailing the design and implementation of the proposed algorithmic framework, along with the quantitative results comparing the enhanced SemanticSmooth with its reinforcement learning-based predecessor. An analysis will be conducted of the trade-offs between the two models with regards to robustness and efficiency. Alongside our report will be a prototype implementation of the algorithm with an open-source codebase for reproducibility and further development.

A.2.5 Data Accessibility and Quality

The datasets required for this project are publicly available and have been validated for quality in prior research. No additional data collection is necessary, ensuring the feasibility of completing the project within the 10-week timeframe.