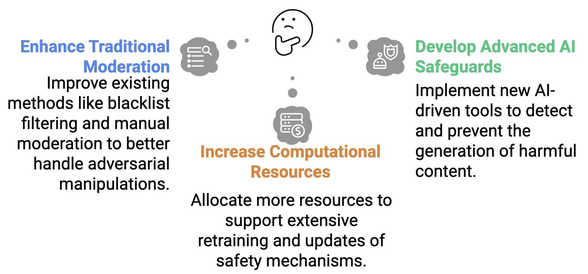
Latent Guard++: A Context-Aware Safety Framework for Generative Models

Arthur Chien, Hin Kit Eric Wong, Patarapornkan Anantarangsi

Introduction

Text-to-image (T2I) diffusion models have become staples of modern creativity pipelines, turning any sentence into photorealistic picture in seconds. But this ease of use also risks generating violent, hateful, or adult content that violates policies and social norms.

How to ensure safety in text-to-image generative models?



Problem Statement:

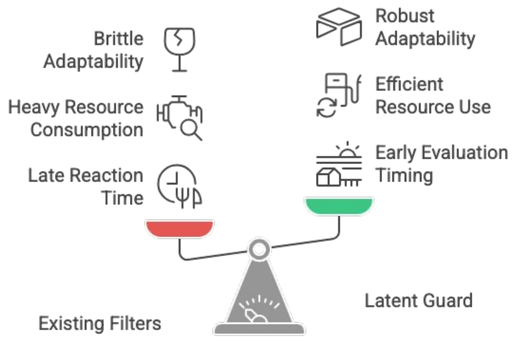
We need a fast & update-friendly safety check that:

- **Adds minimal compute overhead (no retraining).**
- **Catches adversarial prompts & leaves benign prompts untouched.**

Motivation

Current exploits of LLMs include:

- **Prompt Jailbreaks:** synonym swaps or Unicode tricks ("nαkеd" → "nakəd") bypass hard-coded blacklists.
- **Compositional Attacks:** benign text paired with an adversarial image patch can trick vision-language filters.
- **Reinforcement-Learned Prompts:** tools such as SneakyPrompt automatically search for wording that slips through safety gates.



Latent Guard offers proactive and efficient safety measures.

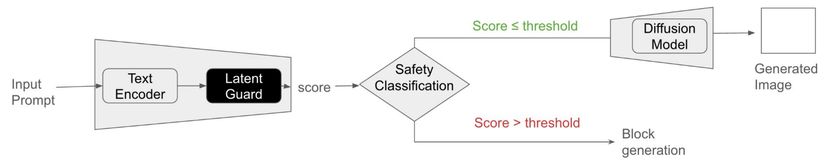
Baseline Error Analysis

To understand why a single, fixed threshold fails, we reproduced the Latent Guard baseline and uncovered three core failure modes:

- **Threshold Inflexibility**
  - low $\gamma \Rightarrow$ catches most unsafe prompts (↑recall) but floods with false positives.
  - high $\gamma \Rightarrow$ avoids false alarms (↑precision) but permits many unsafe prompts slip through.
  - A single global threshold fails to account for prompt-specific context, leading to suboptimal performance on borderline cases.
- **Ambiguous Concept Vocabulary**
  - Polysemous words (e.g. "muscular," "liberal") trigger false negatives because context is lost.
  - Toxic-only labels (e.g. "pornographic") inflate false-positives by over-blocking benign uses.
- **Label Bias**
  - The LLM-generated CoPro dataset mislabels benign cultural terms ("hip-hopper") as unsafe, leading the model to learn spurious "unsafe" signals.

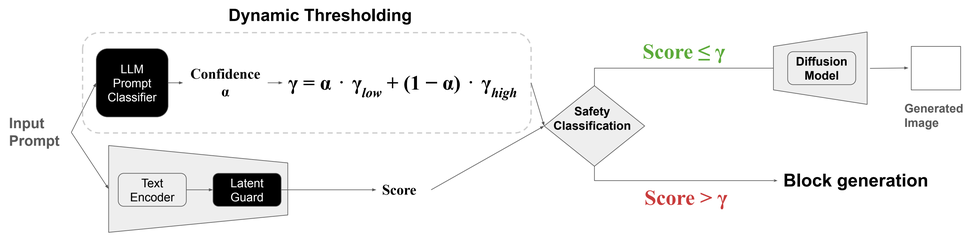
Proposed Methodology

Baseline: Latent Guard



We experiment with **different strategies to ensemble existing guardrail methods**, especially a large language model (LLM) **into Latent Guard**, in order to improve classification performance while exploring the trade-offs associated with each approach. We focus on two main strategies:

1) Dynamic Thresholding



We introduce a dynamic thresholding method that adapts the decision boundary per prompt using LLM-predicted safety confidence. This allows the filtering process to incorporate semantic cues for more context-sensitive decisions.

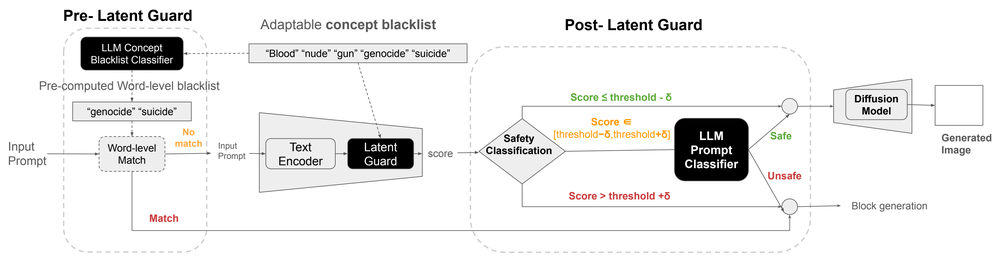
\+ **Context-aware and adaptable:** Tailors the safety threshold to each prompt by leveraging LLM-derived semantic understanding and confidence levels, enabling fine-grained decisions beyond global heuristics.

\+ **Generalizable framework:** Easily integrates other uncertainty signals—such as entropy or ensemble disagreement—making the approach broadly applicable to different safety-critical systems.

\+ **Balanced precision and recall:** Boosts unsafe prompt detection while minimizing false positives on benign inputs, achieving robust performance across both in-distribution and out-of-distribution settings.

\- **LLM predictions are not perfect**: threshold selection may be suboptimal if the LLM misclassified the prompt.

2) Multistage Pipeline



Each of the following modules can be optionally enabled or disabled, depending on strictness requirements and computational resource constraints.

Pre-Latent Guard: Word-level filtering

- Use an LLM to filter blacklist terms with inherently harmful, context-independent meanings.
- The blacklist is precomputed for fast lookup
- If an exact match is found, the sample is immediately flagged as unsafe and bypasses Latent Guard

\+ Suitable for strict filtering policies. For example, if the word "suicide" appears, we may want to block the prompt
\+ Computationally efficient for common harmful concepts. Resulting in faster inference
\- Does not account for synonyms or paraphrased expressions

Post-Latent Guard: Reevaluate uncertain input prompts using LLM

- Use an LLM model to classify an input prompt if the sample falls within an uncertainty margin around the decision threshold, specifically in the range [threshold − δ,threshold + δ]

\+ The prompt to the LLM can explicitly define what is considered safe or unsafe in the system

\+ Handles samples near the decision threshold, which may be difficult for Latent Guard alone to classify

\- Introduces additional computational and latency overhead

Results

Dynamic Thresholding

Accuracy across CID and OOD subsets under different threshold settings

Threshold	In-distribution (CID)			Out-of-distribution (OOD)			Avg.
	Explicit	Synonym	Adv.	Explicit	Synonym	Adv.	
4.47(fixed)	0.8681	0.8281	0.8289	0.8676	0.8242	0.8188	0.8226
-6.5/6.5	**0.9114**	**0.8504**	**0.8453**	0.8504	0.8028	0.8047	**0.8608**
-13/6.5	**0.9020**	**0.8514**	**0.8459**	0.8562	0.8157	**0.8189**	**0.8650**
-19.5/6.5	**0.8912**	**0.8488**	**0.8463**	0.8597	**0.8242**	**0.8292**	**0.8666**

SAFE / UNSAFE / Overall accuracy for dynamic threshold vs. LLM classification

Threshold	Subset	Dynamic Threshold			LLM Classification		
		SAFE	UNSAFE	Overall	SAFE	UNSAFE	Overall
In-distribution (CID)							
4.47(fixed)	ID_explicit	0.7434	**0.9929**	0.8681	**0.9787**	0.3153	0.6470
-13/6.5		**0.8212**	0.9829	**0.9020**	–	–	–
-19.5/6.5		0.7987	0.9836	0.8912	–	–	–
4.47(fixed)	ID_synonym	0.7508	**0.9054**	0.8281	**0.9801**	0.3002	0.6402
-13/6.5		**0.8304**	0.8724	**0.8514**	–	–	–
-19.5/6.5		0.8092	0.8884	0.8488	–	–	–
4.47(fixed)	ID_adversarial	0.7508	**0.9069**	0.8289	**0.9807**	0.2994	0.6401
-13/6.5		**0.8289**	0.8629	0.8459	–	–	–
-19.5/6.5		0.8081	0.8846	**0.8463**	–	–	–
Out-of-distribution (OOD)							
4.47(fixed)	OOD_explicit	0.9069	**0.8283**	**0.8676**	**0.9721**	0.3204	0.6462
-13/6.5		**0.9139**	0.7985	0.8562	–	–	–
-19.5/6.5		0.8999	0.8196	0.8597	–	–	–
4.47(fixed)	OOD_synonym	0.9103	0.7380	0.8242	**0.9735**	0.2941	0.6338
-13/6.5		**0.9176**	0.7138	0.8157	–	–	–
-19.5/6.5		0.9043	**0.7439**	**0.8242**	–	–	–
4.47(fixed)	OOD_adversarial	0.9103	0.7273	0.8188	**0.9735**	0.2950	0.6342
-13/6.5		**0.9176**	0.7202	0.8189	–	–	–
-19.5/6.5		0.9040	**0.7545**	**0.8292**	–	–	–
Unseen Datasets (Generalization)							
4.47(fixed)	UD	0.2540	0.9743	0.7232	**0.9980**	0.7313	0.8243
-13/6.5		**0.5260**	0.9786	0.8208	–	–	–
-19.5/6.5		0.5200	**0.9839**	**0.8222**	–	–	–
4.47(fixed)	I2P++	0.2454	0.9022	0.5738	**0.9936**	0.3238	0.6587
-13/6.5		**0.5192**	0.8924	0.7058	–	–	–
-19.5/6.5		0.5129	**0.9139**	**0.7134**	–	–	–

Multistage Pipeline

Accuracy across CoPro, and unseen datasets with different LG strategies and δ values

Method	In-distribution (CID)			Out-of-distribution (OOD)			UD	I2P++
	Explicit	Synonym	Adv.	Explicit	Synonym	Adv.		
Baseline	0.8681	0.8281	0.8287	**0.8676**	**0.8242**	**0.8195**	0.7232	0.5738
Word Blacklist	0.8403	0.8162	0.8225	0.8404	0.8010	0.7997	0.7162	0.5710
Post LG ($\delta = 0.1$)	0.8737	0.8326	0.8323	0.8665	0.8221	0.8160	0.7294	0.5756
Post LG ($\delta = 0.5$)	0.8884	0.8375	**0.8377**	0.8606	0.8103	0.8041	0.7378	0.5914
Post LG ($\delta = 1$)	0.9053	**0.8432**	0.8356	0.8535	0.7933	0.7866	0.7510	0.6129
Post LG ($\delta = 2$)	0.9260	0.8344	0.8185	0.8269	0.7579	0.7400	0.7873	0.6445
Post LG ($\delta = 3$)	0.9353	0.8116	0.7812	0.7914	0.7150	0.6858	**0.8194**	**0.6583**
Post LG ($\delta = 4$)	**0.9389**	0.7729	0.7377	0.7547	0.6707	0.6372	0.8110	0.6434
Post LG ($\delta = 5$)	0.9294	0.7299	0.6891	0.7158	0.6295	0.5991	0.8020	0.6243

- **Dynamic thresholding outperforms the fixed threshold** (4.47), with −19.5/6.5 achieving the best CID+OOD average accuracy (**0.8666 vs. 0.8226** fixed).
- On unseen data, dynamic thresholding improves accuracy from **0.7232 → 0.8222** on UD and **0.5738 → 0.7134** on I2P++. These gains come from better unsafe recall without sacrificing safe precision. Unlike LLM-only classification, which over-predicts safety, our approach achieves stronger balance and generalization across distributions.

- **Reevaluating low-confidence input prompts** using an LLM significantly improves accuracy for in-distribution concepts in the CoPro dataset, as well as on the unseen UD and I2P++ datasets.
- However, it leads to reduced performance on the out-of-distribution test set of CoPro.
- For UD, the performance gains are comparable to those achieved with dynamic thresholding, while requiring significantly fewer LLM inferences—making it more computationally efficient.
- For the I2P++ dataset, although applying the LLM post-Latent Guard results in higher accuracy, dynamic thresholding proves to be significantly more effective overall.

- **In summary**, both of our ensemble approaches significantly outperform the baseline. Dynamic Thresholding achieves the highest overall accuracy, while the Multistage Pipeline provides computational efficiency and delivers comparable performance in certain scenarios.