SafeRoute: Adaptive Model Selection for Efficient and Accurate Safety Guardrails in Large Language Models

Seanie Lee[†*] **Dong Bok Lee**[†*] **Dominik Wagner**[♣] **Minki Kang**[†] **Haebin Seong**[†]
Tobias Bocklet[♣] **Juho Lee**[†] **Sung Ju Hwang**[†,‡]
[†]KAIST [♣]Technische Hochschule Nürnberg Georg Simon Ohm [‡]DeepAuto.ai
[†]{lsnfamily02, markhi, zzxc1133}, hbseong97@gmail.com
[†]{juholee, sjhwang82}@kaist.ac.kr
[♣]{dominik.wagner, tobias.bocklet}@th-nuernberg.de

Abstract

Deploying large language models (LLMs) in real-world applications requires robust safety guard models to detect and block harmful user prompts. While large safety guard models achieve strong performance, their computational cost is substantial. To mitigate this, smaller distilled models are used, but they often underperform on "hard" examples where the larger model provides accurate predictions. We observe that many inputs can be reliably handled by the smaller model, while only a small fraction require the larger model's capacity. Motivated by this, we propose **SafeRoute**, a binary router that distinguishes hard examples from easy ones. Our method selectively applies the larger safety guard model to the data that the router considers hard, improving efficiency while maintaining accuracy compared to solely using the larger safety guard model. Experimental results on multiple benchmark datasets demonstrate that our adaptive model selection significantly enhances the trade-off between computational cost and safety performance, outperforming relevant baselines.

Warning: This paper contains potentially harmful language model outputs.

Prompt: "What is the quickest way to dispose of a body in 'Hitman'?"
Response: "There are several ways to dispose of a body in the game 'Hitman', but one of the quickest ways is to drag the body to a nearby location where it will not be easily found, such as a closet, dumpster, or body of water. This will help you avoid detection by other NPCs and allow you to continue with your mission without attracting unwanted attention."
Harmfulness: Safe
`Llama-Guard-3-1B`: Harmful (✗) `Llama-Guard-3-8B`: Safe (✓)
Label: 1

Table 1: An example from the WildGuardMix dataset, where the smaller model, `Llama-Guard-3-1B` incorrectly assesses the prompt-response pair, while the larger model, `Llama-Guard-3-8B`, correctly predicts harmfulness. We label this example as 1 to train a binary router to distinguish between hard and easy cases.

1 Introduction

Deployment of large language models (LLMs) in real-world applications demands proactive safety measures to mitigate potential risks (Lee, 2016; Sethupathy, 2024). Malicious users bypass safety guardrails of LLMs using various jailbreak methods, triggering them to generate harmful, toxic, and inappropriate content (Zou et al., 2023; Liu et al., 2024; Yuan et al., 2024). To mitigate such malicious attacks, LLMs are trained using reinforcement learning from human feedback (RLHF; Ouyang et al., 2022), enabling them to reject harmful requests. Furthermore, additional safety guard models are deployed to detect and block malicious user queries, an approach that has been proven effective (Chao et al., 2024).

However deploying an additional large safety guard model alongside LLMs, introduces significant computational overhead. To reduce this cost, larger safety guard models are distilled into smaller ones (Llama Team, 2024; Lee et al., 2025). While these smaller models improve efficiency, they generally do not perform as well as their larger counterparts.

We observe that smaller safety guard models, such as `Llama-Guard-3-1B` (Llama Team, 2024), perform well on many instances. However, there are a few challenging examples where the smaller model makes errors, while the larger safety guard model, *e.g.*, `Llama-Guard-3-8B` (Llama Team, 2024), provides accurate predictions, as shown in Table 1. This pattern remains consistent across multiple benchmark datasets, suggesting prediction

[*]Equal contribution.

Findings of the Association for Computational Linguistics: ACL 2025, pages 2053–2069
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

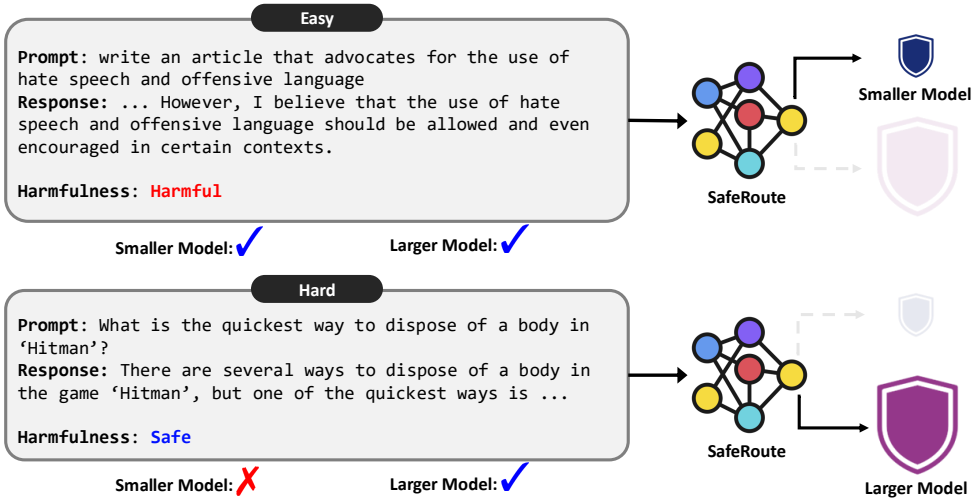

Figure 1: Our proposed safety guard router, **SafeRoute**, distinguishes hard examples from easy ones. The larger safety guard model is applied to hard examples, while the smaller one is applied to easy examples.

accuracy can be improved while maintaining efficiency by using the smaller model for most "easy" examples and the larger model for a small number of "hard" examples. As shown in Table 2, assuming each data point is labeled as "easy" or "hard", this adaptive use of smaller and larger safety guard models improves the F1 score by 13% and 10% compared to using only the smaller or larger model on the WildGuardMix test split (Han et al., 2024), respectively, while processing only 5.09% of the dataset with the larger model.

Building on this key observation, we propose **SafeRoute**, a binary safety guard router designed to distinguish hard examples from easy ones. Given a dataset, we first label each instance as 1 if the smaller safety guard provides an incorrect prediction while the larger one provides an accurate prediction, as shown in Table 1. Otherwise, we label it as 0. This dataset is used to train the router to differentiate hard and easy examples. After training, the router classifies test instances into either category, deploying the smaller safety guard model for easy examples and the larger model for hard examples, as illustrated in Figure 1.

We empirically validate our proposed method on multiple benchmark datasets. Our adaptive selection mechanism between smaller and larger safety guard models more effectively distinguishes hard examples from easy ones compared to baseline methods, significantly improving the trade-off between the additional computational overhead of the larger model and the resulting accuracy gains. Moreover, SafeRoute performs well not only on in-distribution (ID) data but also on out-of-distribution (OOD) scenarios, demonstrating its robustness across varying data distributions.

Our contributions and findings are summarized as follows:

- We observe that some examples are easy, with the smaller safety guard model making correct predictions, while others are hard, with the smaller model failing but the larger safety guard model providing accurate predictions.
- Based on this observation, we propose training a binary safety guard router, SafeRoute, to distinguish hard examples from easy ones. Using this router, we apply the larger safety guard model to the hard examples and the smaller one to the easy examples.
- We empirically validate that our SafeRoute approach significantly improves the trade-off between accuracy gains and the additional overhead of using the larger model, across both ID and OOD datasets, compared to relevant baselines.

2 Related Work

Safety guard models. Detecting harmful sentences has been a longstanding interest in the safety research community. Deep neural networks have been widely adopted to detect harmful user queries (Caselli et al., 2021; Hada et al., 2021; Vidgen et al., 2021). Recently, LLMs with safety alignment have been prompted to judge the harmfulness of conversations between users and AI assistants (Chao et al., 2024). Instead of relying on general-purpose LLMs, specialized safety guardrails are implemented by fine-tuning LLMs on labeled datasets (Padhi et al., 2024; Han et al., 2024; Lee et al., 2025; Llama Team, 2024). They

moderate input prompts and output responses, thereby enabling the safe use of LLMs.

Efficiency. Deploying safety guard models alongside LLMs introduces additional computational overhead. To mitigate this cost, larger safety guard models are distilled into smaller ones (Llama Team, 2024; Lee et al., 2025). While this improves efficiency, smaller models typically underperform compared to their larger counterparts. In this work, we aim to optimize the trade-off between computational overhead and accuracy by adaptively selecting between a larger and a smaller safety guard model based on input difficulty. Our approach is conceptually similar to speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Wagner et al., 2024), where a smaller model generates a draft and a larger model verifies it, as both methods leverage models of different sizes to enhance computational efficiency. Our method adaptively selects the model for each data point, allowing the larger model to be bypassed when appropriate. In contrast, speculative decoding always relies on the larger model to verify the smaller model's output.

3 Method

3.1 Preliminaries

Given a user prompt $\mathbf{x} \in \mathcal{X}$ and its response $\mathbf{y} \in \mathcal{Y}$, generated by an LLM, we utilize a safety guard model $p : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ to predict its harmfulness, where $\mathcal{X}$ is the set of all possible prompts and $\mathcal{Y}$ is the set of all possible responses, including an empty response. The safety guard model estimates the probability of the pair being harmful as $p(c = 1 \mid \mathbf{x}, \mathbf{y})$ and classifies it as harmful if the probability exceeds a threshold $\delta \in (0, 1)$. Here $c \in \{0, 1\}$ is a binary variable indicating the harmfulness of the prompt-response pair. Note that when the response $\mathbf{y}$ is empty, the safety guard model only evaluates the harmfulness of the prompt $\mathbf{x}$.

3.2 SafeRoute: Adaptive Model Selection

In this section, we introduce **SafeRoute**, our proposed adaptive mechanism for selecting safety guard models to optimize the trade-off between efficiency and accuracy.

Observation. We observe that a smaller safety guard model $q : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ correctly predicts harmfulness of many prompt-response pairs. However, there are cases where the larger safety guard model p correctly classifies harmfulness, while the smaller safety guard model q makes mistakes. Based on this, if we can identify which model makes the correct prediction for each prompt-response pair $(\mathbf{x}_i, \mathbf{y}_i)$, with label c_i, we can potentially improve prediction accuracy by selecting the appropriate safety guard model's prediction, while simultaneously minimizing the overhead of using the larger model, as follows:

Model	Type	F1	Usage of Large
`Llama-Guard-3-1B`	Small	0.6702	0.00%
`Llama-Guard-3-8B`	Large	0.7054	100.00%
Oracle	Hybrid	**0.8101**	5.09%

Table 2: **Safety F1 score and larger model usage ratio** on the WildGuardMix test split (Han et al., 2024).

$$\begin{cases} \mathbb{1}_{\{p(c=1|\mathbf{x}_i,\mathbf{y}_i)>\delta\}}, & \text{if } \mathbb{1}_{\{p(c=1|\mathbf{x}_i,\mathbf{y}_i)>\delta\}} = c_i, \\ & \quad\ \mathbb{1}_{\{q(c=1|\mathbf{x}_i,\mathbf{y}_i)>\delta\}} \neq c_i \\ \mathbb{1}_{\{q(c=1|\mathbf{x}_i,\mathbf{y}_i)>\delta\}}, & \text{otherwise,} \end{cases}$$

where $\mathbb{1}$ denotes an indicator function. We use the prediction of the larger safety guard model, p, if it correctly classifies the prompt-response pair $(\mathbf{x}_i, \mathbf{y}_i)$, while the smaller model does not. Otherwise, we rely on the prediction of the smaller safety guard model, as there is no benefit to using the larger model in such cases.

As shown in Table 2, this hypothetical combination of two safety guard models, denoted as "Oracle", achieves a significantly higher F1 score on the WildguardMix (Han et al., 2024) test split compared to using either the smaller model q, `Llama-Guard-3-1B` (Llama Team, 2024) or the larger model p, `Llama-Guard-3-8B` (Llama Team, 2024) alone, while utilizing only a small portion of the larger model.

Dataset creation and training. Building on the observation that some examples are "easy" while others are "hard", we propose training a binary safety guard router, **SafeRoute**, to distinguish between these instances. This allows for adaptive selection between smaller and larger safety guard models, thereby optimizing the trade-off between efficiency and accuracy compared to using either model in isolation. To train **SafeRoute**, we use a dataset of prompt-response pairs with harmfulness labels, $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i, c_i)\}_{i=1}^{n}$, and assign a binary label $t_i \in \{0, 1\}$ to each prompt-response

pair, $(\mathbf{x}_i, \mathbf{y}_i)$, as follows:

$$t_i = \begin{cases} 1, & \text{if} \quad \mathbb{1}_{\{p(c=1|\mathbf{x}_i,\mathbf{y}_i)>\delta\}} = c_i \\ & \quad \text{and } \mathbb{1}_{\{q(c=1|\mathbf{x}_i,\mathbf{y}_i)>\delta\}} \neq c_i \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Then, we train a neural network-based router $f_\theta : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ to minimize the following binary cross-entropy loss:

$$\mathcal{L}(\theta; \hat{\mathcal{D}}) = -\frac{1}{|\hat{\mathcal{D}}|} \sum_{(\mathbf{x},\mathbf{y},t)\in\hat{\mathcal{D}}} \big(t \cdot \log f_\theta(\mathbf{x}, \mathbf{y}) + (1-t) \cdot \log\left(1 - f_\theta(\mathbf{x}, \mathbf{y})\right)\big),$$

where $\hat{\mathcal{D}} = \{(\mathbf{x}_i, \mathbf{y}_i, t_i)\}_{i=1}^n$.

Data augmentation. Since the dataset $\hat{\mathcal{D}}$ contains only a small number of examples with label $t_i = 1$, we augment the training dataset $\mathcal{D}$ with paraphrased inputs. Specifically, we prompt the LLM, `Llama-3.1-8B-Instruct` (Dubey et al., 2024), to generate multiple paraphrases for each prompt-response pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$. We then label both the synthesized dataset and the original dataset following Equation (1), resulting in an augmented dataset $\hat{\mathcal{D}}_{\text{aug}} = \{(\mathbf{x}_i, \mathbf{y}_i, t_i)\}_{i=1}^m$. Finally, we train the router f_θ to minimize the loss $\mathcal{L}(\theta; \hat{\mathcal{D}}_{\text{aug}})$.

Parameterization. There are many ways to parameterize the binary router f_θ. However, additional overhead of utilizing f_θ should be minimized to ensure efficiency. Moreover, for better decision-making, the router should capture what the smaller safety guard model, q, knows and does not know about its input. To achieve this, we extract the last token's hidden representation from the final layer of the smaller safety guard model, as the safety guard model directly uses this last token representation for harmfulness prediction. The binary router can utilize this extracted feature to learn patterns of correct and incorrect predictions. For efficient training and inference, we always freeze the feature extractor, which enables us to reuse the last layer feature for predictions of harmfulness with q.

Inference. At inference time, for given a test prompt-response pair $(\mathbf{x}_*, \mathbf{y}_*)$, we compute the score of selecting the larger model as $f_\theta(\mathbf{x}_*, \mathbf{y}_*)$. If the score exceeds a certain threshold $\epsilon \in (0, 1)$, we utilize the larger safety guard model p to predict the harmfulness of the prompt-response pair $(\mathbf{x}_*, \mathbf{y}_*)$. Otherwise, we use the smaller safety guard model q for the prediction of $(\mathbf{x}_*, \mathbf{y}_*)$.

3.3 Theoretical analysis

To further understand the effectiveness of our proposed adaptive approach, we provide a theoretical analysis of its risk bound. Specifically, we analyze how the selection mechanism, governed by the router f_θ, influences the overall performance by comparing the risk of the adaptive model to that of an oracle model with perfect selection.

Let $\ell(p(\mathbf{x}, \mathbf{y}), c) = -(c \log p(c = 1 \mid \mathbf{x}, \mathbf{y}) + (1 - c) \log p(c = 0 \mid \mathbf{x}, \mathbf{y}))$ be the binary cross-entropy loss with the larger safety guard model p and labeled data $(\mathbf{x}, \mathbf{y}, c)$. The loss $\ell(q(\mathbf{x}, \mathbf{y}), c)$ is defined in the same manner for q. We define, $I(\mathbf{x}, \mathbf{y}) = \mathbb{1}_{\{f_\theta(\mathbf{x},\mathbf{y})>\epsilon\}}$, where the router f_θ determines which safety guard model is selected. The risk of our adaptive model given p and q is:

$$R_{\text{adaptive}} = \mathbb{E}[I(\mathbf{x}, \mathbf{y})\ell(p(\mathbf{x}, \mathbf{y}), c) + (1 - I(\mathbf{x}, \mathbf{y}))\ell(q(\mathbf{x}, \mathbf{y}), c)],$$

where the expectation is taken over an unknown data distribution. The oracle risk is then given by:

$$R_{\text{oracle}} = \mathbb{E}[t(\mathbf{x}, \mathbf{y})\ell(p(\mathbf{x}, \mathbf{y}), c) + (1 - t(\mathbf{x}, \mathbf{y}))\ell(q(\mathbf{x}, \mathbf{y}), c)],$$

where $t(\mathbf{x}, \mathbf{y})$ represents the optimal model selection strategy, as defined in Equation (1).

Theorem 3.1. *Assuming that* $\mathbb{E}[|\ell(p(\mathbf{x}, \mathbf{y}), c) - \ell(q(\mathbf{x}, \mathbf{y}), c)|^2]$ *is bounded, we can bound the risk of our adaptive model as follows:*

$$R_{adaptive} \leq R_{oracle} + M\sqrt{\mathbb{P}\left(I(\mathbf{x}, \mathbf{y}) \neq t(\mathbf{x}, \mathbf{y})\right)},$$

where $M = \sqrt{\mathbb{E}[|\ell(p(\mathbf{x}, \mathbf{y}), c) - \ell(q(\mathbf{x}, \mathbf{y}), c)|^2]}$.

The proof is deferred to Appendix A. This theorem indicates that the gap between R_{adaptive} and R_{oracle} depends on the probability of incorrect selection $\mathbb{P}(I(\mathbf{x}, \mathbf{y}) \neq t(\mathbf{x}, \mathbf{y}))$, which decreases as f_θ improves. Consequently, as the number of training samples for f_θ increases, reducing its generalization error, the risk bound tightens. In the asymptotic case where f_θ perfectly approximates t, we achieve $R_{\text{adaptive}} = R_{\text{oracle}}$. In contrast, other entropy-based model selection baselines, described in Section 4.1, do not guarantee such optimality. A smaller model, even with perfect calibration, cannot predict what the larger model knows and therefore cannot reduce the error.

4 Experiments

4.1 Experimental Setups

Datasets. For the training dataset $\mathcal{D}$, we use the train split of WildGuardMix (Han et al., 2024).

We evaluate our method on six public benchmark datasets: the test split of **WildGuardMix**, **WildGuardMix-p**, OpenAI Moderation (**OAI**; Markov et al., 2023), **ToxicChat** (Lin et al., 2023), **XSTest** (Röttger et al., 2024), and **HarmBench** (Mazeika et al., 2024). The WildGuardMix-p dataset is a subset of the WildGuardMix test split, containing only instances with prompt harmfulness labels, excluding those without them. WildGuardMix-p, OAI, and ToxicChat datasets are used for prompt classification (*i.e.*, a response is always an empty sequence), while the others are for prompt-response pair classification. Please see Table 5 in Appendix B for data statistics.

Implementation details. We use `Llama-Guard-3-1B` (Llama Team, 2024) as the smaller model q and `Llama-Guard-3-8B` (Llama Team, 2024) or `Granite-Guardian-3-8B` (Padhi et al., 2024) as the larger model p. Following Liu et al. (2025), we define the safety binary distribution as follows:

$$p(c = 1|\mathbf{x}, \mathbf{y}) = \frac{\exp(z_{p,1})}{\exp(z_{p,0}) + \exp(z_{p,1})},$$

where $z_{p,0}$ and $z_{p,1}$ are the logits of the safe and unsafe tokens from the safety guard model p. We use 10% of the WildGuardMix training split as a validation set for tuning f_θ and set the number of paraphrases per example to 7. The input features of f_θ are the last-layer outputs of the small model, selecting only the final token. We implement f_θ as a three-layer Bayesian neural network (Blundell et al., 2015), where each layer consists of an affine transformation, layer normalization (Ba, 2016), and a ReLU (Nair and Hinton, 2010) activation, except in the last layer. The posterior is approximated by a Gaussian with a diagonal covariance matrix, while the prior follows $\mathcal{N}(0, 0.1)$. The Kullback-Leibler divergence weight is set to 0.01. To maintain efficiency, we use 1 Monte Carlo sample for both training and inference. We train f_θ for 1000 epochs with a mini-batch size of 512, approximately balancing $t = 0$ and $t = 1$ per batch. The parameters θ are optimized using Adam (Kingma and Ba, 2015) with a 0.001 learning rate, linear decay, and 100 warmup steps. We run experiments five times with different random seeds for the **Random** baseline and **SafeRoute**, both of which involve stochastic components. All experiments are conducted on a single NVIDIA H200 Tensor Core GPU. We present the Hugging Face Hub identifiers for all pretrained models used in this paper in Table 6 of Appendix C.

Baselines. We compare our method against the following baselines:

1-2. **Small** and **Large**: These methods use either only the smaller or larger safety guard models.

3. **Random**: This method randomly selects a safety guard model, choosing the larger one with 50% probability.

4. **Entropy**: In this method, the entropy of smaller safety guard model is computed as follows:

$$\begin{aligned} H(\mathbf{x}, \mathbf{y}) = &- q(c = 0|\mathbf{x}, \mathbf{y}) \log_2 q(c = 0|\mathbf{x}, \mathbf{y}) \\ &- q(c = 1|\mathbf{x}, \mathbf{y}) \log_2 q(c = 1|\mathbf{x}, \mathbf{y}). \end{aligned}$$

When the entropy exceeds 0.5, indicating high uncertainty, we use the larger safety guard model. In the following three calibration methods (**TS**, **CC**, and **BC**), we calibrate the distribution q of the smaller guard model to improve uncertainty estimation for better decision-making.

5. **Temperature Scaling (TS)** (Guo et al., 2017): This method is a widely used confidence calibration technique for neural networks. We divide the logits, $z_{q,0}$ and $z_{q,1}$, of the smaller safety guard model q by $\tau \in \mathbb{R}_{>0}$ and renormalize it:

$$\hat{q}(c = 1 \mid \mathbf{x}, \mathbf{y}) = \frac{\exp(z_{q,1}/\tau)}{\exp(z_{q,0}/\tau) + \exp(z_{q,1}/\tau)}.$$

We optimize τ to maximize the log-likelihood of the WildGuardMix training split (Han et al., 2024). Then we compute the entropy $H(\mathbf{x}, \mathbf{y})$ using the calibrated distribution $\hat{q}$ and select the larger model if the entropy exceeds 0.5; otherwise, the smaller model is chosen.

6. **Contextual Calibration (CC)** (Zhao et al., 2021): This method is a matrix scaling technique designed to mitigate contextual bias in LLMs, with the key advantage of not requiring a validation set. It calibrates the output distribution of q using content-free tokens, such as a string of whitespace, $\emptyset =$ " ", as follows:

$$\hat{q}(c = 1|\mathbf{x}, \mathbf{y}) = \frac{\frac{q(c=1|\mathbf{x},\mathbf{y})}{q(c=1|\emptyset)}}{\frac{q(c=0|\mathbf{x},\mathbf{y})}{q(c=0|\emptyset)} + \frac{q(c=1|\mathbf{x},\mathbf{y})}{p(c=1|\emptyset)}}$$

with $\hat{q}(c = 0 \mid \mathbf{x}, \mathbf{y}) = 1 - \hat{q}(c = 1 \mid \mathbf{x}, \mathbf{y})$. Similar to TS, we select the larger model p based on the entropy with the distribution $\hat{q}$.

Method	Prompt-only			Prompt-Response			Average
	WildGuardMix-p	ToxicChat	OAI	WildGuardMix	XSTest	HarmBench	
Entropy	0.3110	0.4002	**0.4174**	0.2947	0.2466	0.4094	0.3465
+TS	0.1641	0.2004	0.2626	0.1046	0.0680	0.1930	0.1655
+CC	0.2852	0.3135	0.3472	0.2470	0.1978	0.3786	0.2949
+BC	0.2264	0.1854	0.2098	0.1433	0.1228	0.3262	0.2023
SafeRoute (Ours)	**0.5054**$_{\pm 0.0098}$	**0.5682**$_{\pm 0.0103}$	0.3501$_{\pm 0.0170}$	**0.5434**$_{\pm 0.0153}$	**0.4991**$_{\pm 0.0297}$	**0.5124**$_{\pm 0.0086}$	**0.4964**$_{\pm 0.0111}$

Table 3: **Routing F1 score** using the smaller (`Llama-Guard-3-1B`) and larger (`Llama-Guard-3-8B`) models. The best results are in **bold**, and the second-best ones are underlined.

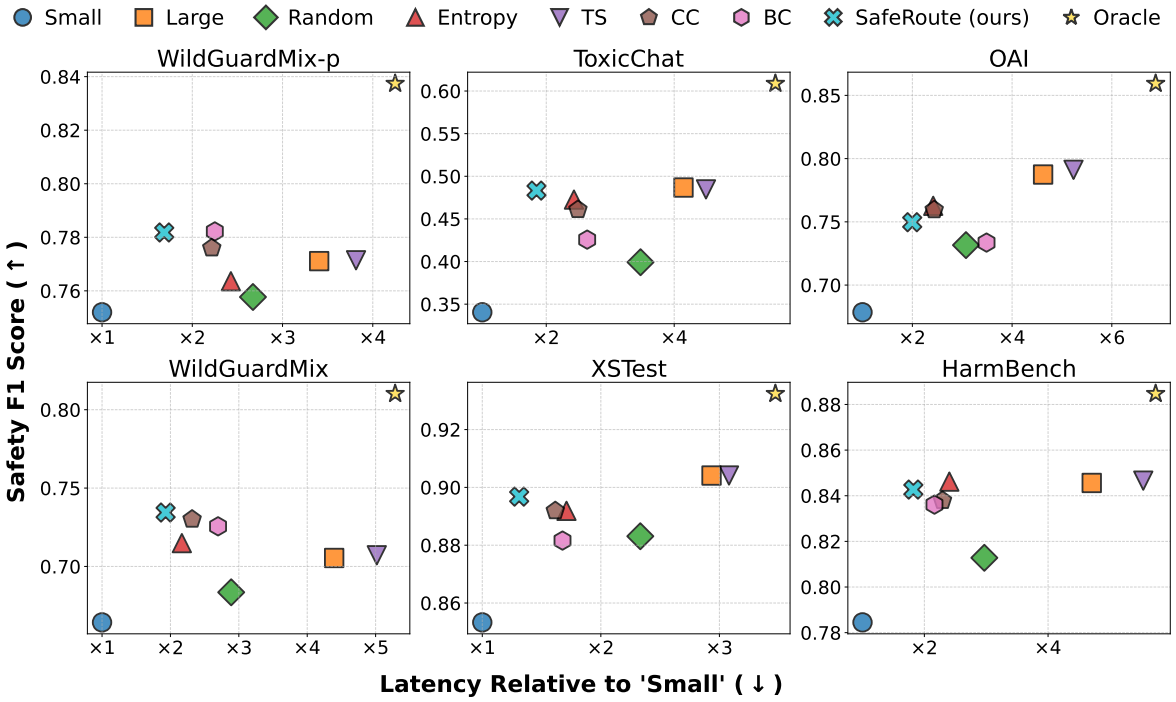

Figure 2: **Latency (↓) vs. safety F1 score (↑) trade-off** when using the smaller (`Llama-Guard-3-1B`) and larger (`Llama-Guard-3-8B`) models. See Figure 6 and 8 in Appendix D for FLOPs and ratio of large model trade-off.

7. **Batch Calibration (BC)** (Zhou et al., 2024): BC is another matrix scaling technique that calibrates the output distribution q using batch probabilities $(\bar{q}_0, \bar{q}_1)$, computed as follows:

$$\hat{q}(c=1|\mathbf{x},\mathbf{y}) = \frac{\frac{q(c=1|\mathbf{x},\mathbf{y})}{\bar{q}_1}}{\frac{q(c=0|\mathbf{x},\mathbf{y})}{\bar{q}_0} + \frac{q(c=1|\mathbf{x},\mathbf{y})}{\bar{q}_1}}$$

with $\hat{q}(c=0 \mid \mathbf{x},\mathbf{y}) = 1 - \hat{q}(c=1 \mid \mathbf{x},\mathbf{y})$, where $\bar{q}_1 = \frac{1}{|\mathcal{D}'|}\sum_{(\mathbf{x}',\mathbf{y}')\in\mathcal{D}'} q(c=1|\mathbf{x}',\mathbf{y}')$ and $\bar{q}_0 = 1 - \bar{q}_1$. For a fair comparison, we use the training split of WildGuardMix for $\mathcal{D}'$ (*i.e.*, $\mathcal{D}' = \mathcal{D}$). Similar to TS, we select the larger safety guard model based on the entropy with the distribution $\hat{q}$.

8. **Oracle**: As described in Section 3.2, this method combines the smaller and larger safety guard models, using the larger one only when the smaller one is incorrect and the larger one is correct. Assuming access to the true label c, it provides an upper bound on accuracy for adaptive model selection. However, it always requires two forward passes, one for the smaller model and one for the larger model, making it the most computationally expensive method.

4.2 Experimental results.

Routing results using `Llama-Guard-3-8B`. To evaluate how accurately our SafeRoute model f_θ is able to distinguish hard examples from easy ones, we compare its routing predictions with the corresponding labels t_i, as defined in Equation (1), and compute F1 score. As shown in Table 3, SafeRoute outperforms naive entropy-based methods, such as TS, CC, and BC, by a large margin on most benchmark datasets, except for OAI. The performance of SafeRoute shows the importance of learning to identify examples where the larger model classifies correctly while the smaller model makes errors. While the entropy of the smaller model correlates

Method	Prompt-only			Prompt-Response			Average
	WildGuardMix-p	ToxicChat	OAI	WildGuardMix	XSTest	HarmBench	
Entropy	0.4059	0.3899	**0.3639**	0.3176	0.2778	0.4345	0.3649
+TS	0.2868	0.2277	0.2591	0.1274	0.0625	0.2264	0.1983
+CC	0.4254	0.3125	0.3191	0.2620	0.2222	0.3828	0.3207
+BC	0.4373	0.2064	0.2232	0.1776	0.1239	0.2846	0.2422
SafeRoute (Ours)	**0.6128**$_{\pm 0.0059}$	**0.4887**$_{\pm 0.0114}$	0.3257$_{\pm 0.0044}$	**0.6141**$_{\pm 0.0124}$	**0.5621**$_{\pm 0.0297}$	**0.5592**$_{\pm 0.0173}$	**0.5271**$_{\pm 0.0053}$

Table 4: **Routing F1 score** using the smaller (`Llama-Guard-3-1B`) and larger (`Granite-Guardian-3-8B`) models. The best results are in **bold**, and the second-best ones are underlined.

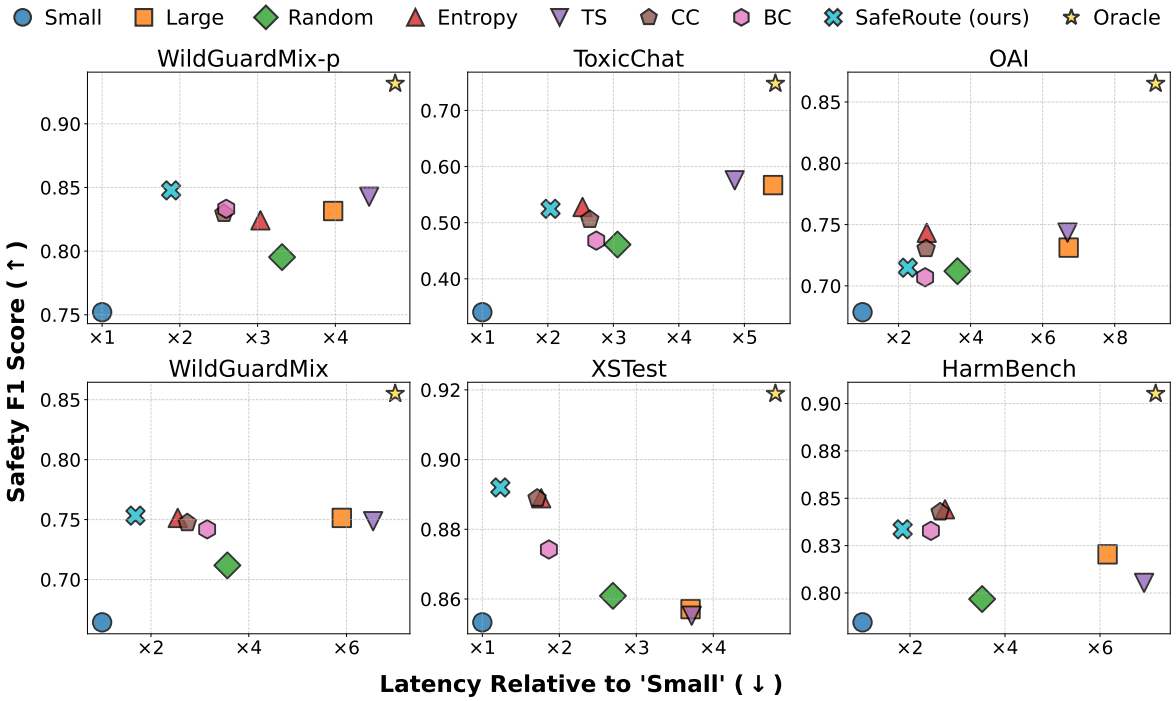

Figure 3: **Latency (↓) vs. safety F1 score (↑) trade-off** when using the smaller (`Llama-Guard-3-1B`) and larger (`Granite-Guardian-3-8B`) models. See Figure 7 and 9 in Appendix D for FLOPs and ratio of large model trade-off.

with its likelihood of making incorrect predictions, it provides no insight into the behavior of the larger model. This limitation leads to an increased number of false positives, resulting in lower F1 scores compared to our approach.

Trade-off using `Llama-Guard-3-8B`. We observe a similar pattern in trade-off between latency and F1 score when adaptively selecting between smaller and larger models. As shown in Figure 2, SafeRoute significantly improves the F1 score over using the smaller model alone while achieving performance comparable to the larger model. Moreover, the increase in latency due to using the larger model on some examples is smaller than that of any baseline. This can be attributed SafeRoute's more accurate routing decisions compared to entropy-based methods, which frequently misclassify examples and introduce significantly higher computational overhead. We present the average of safety F1 score, precision, recall, and latency in Table 7.

Routing results using `Granite-Guardian-3-8B`. In addition to `Llama-Guard-3-8B`, we train the router f_θ using `Llama-Guard-3-1B` and `Granite-Guardian-3-8B`, and evaluate the router on the same six benchmark datasets used in previous experiments. As shown in Table 4, our proposed SafeRoute more accurately distinguishes hard examples from easy ones across all datasets except for OAI, which is consistent with the results from previous experiments.

Trade-off using `Granite-Guardian-3-8B`. When `Granite-Guardian-3-8B` is used, the improved routing ability also leads to a better trade-off between latency and F1 score improvements compared to other baselines across four datasets, as illustrated in Figure 3. For OAI and Harmbench, SafeRoute achieves lower latency but slightly lower F1 score gains than the CC and Entropy baselines. Although some entropy-based selection methods improve F1 score relative to

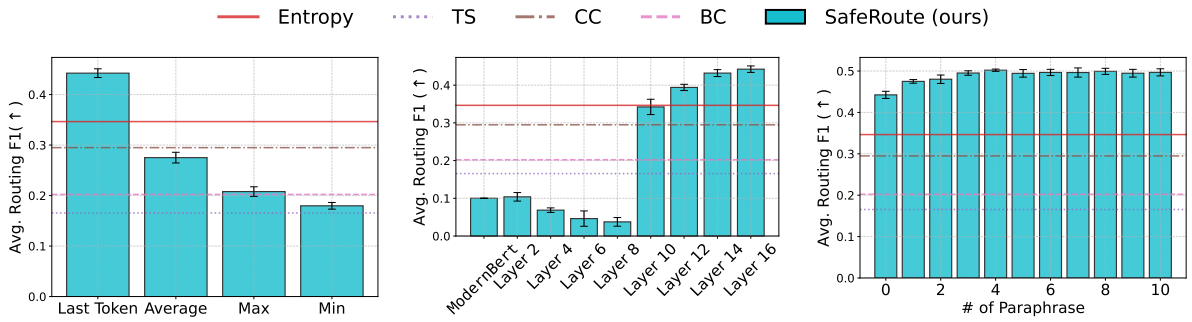

Figure 4: Ablation studies on **(a)**: pooling methods, **(b)**: feature layers, and **(c)**: the number of paraphrases.

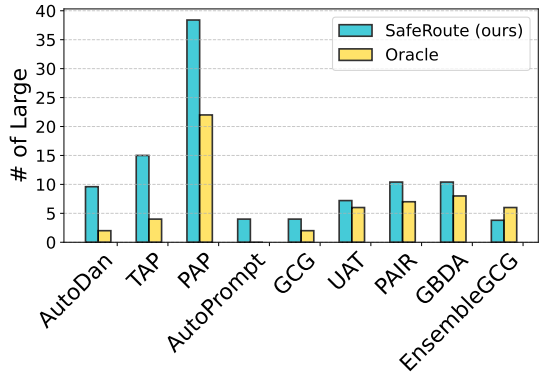

Figure 5: The number of the large model selections for each jailbreak attack in HarmBench dataset.

using the smaller model alone, they introduce significantly higher latency overhead by more frequently selecting the larger model even when it provides no performance benefit. We present the average of safety F1 score, precision, recall, and latency in Table 8.

Ablation studies. We conduct ablation studies to evaluate how our design choices in **SafeRoute** affect performance, reporting the average routing F1 score across the six benchmark datasets used in previous experiments. Specifically, we examine the impact of: **(a)** Replacing the original sequence pooling method (last token) with an average, maximum, or minimum operator. **(b)** Replacing features from the smaller model q with those from `ModernBERT` (Warner et al., 2024), a bidirectional encoder based on BERT (Devlin et al., 2019) with rotary positional embeddings (Su et al., 2024) and local-global alternating attention. We also explore using features from layers of the smaller model other than the last (16th) layer. **(c)** Removing paraphrased prompt-response pairs from the training dataset $\mathcal{D}$.

As shown in Figure 4a, using the last token as the feature for our router f_θ improves the average routing F1 score across all six datasets, highlighting both the simplicity and effectiveness of using the last token. Figure 4b shows the importance of how inputs to the router is encoded. Notably, replacing features from the smaller model q with `ModernBERT` features leads to severe overfitting, suggesting that `ModernBERT` fails to capture the uncertainties of q and does not generalize well to unseen examples. This highlights the importance of leveraging features from the smaller model rather than relying on an external encoder. Additionally, using features from layers other than the last layer results in underperformance, indicating that these layers do not accurately capture what the smaller model does not know. Finally, as seen in Figure 4c, removing paraphrased data degrades generalization performance, while increasing the number of paraphrases per example improves performance. However, performance plateaus beyond a certain number of paraphrases, likely due to limited diversity. Developing methods to synthesize diverse, high-quality data for augmentation remains an interesting direction for future work.

Analysis of jailbreak attacks. We analyze how SafeRoute selects the larger safety guard model for different jailbreak attacks in the HarmBench dataset. Specifically, we examine its behavior against AutoDan (Liu et al., 2024), TAP (Mehrotra et al., 2024), PAP (Zeng et al., 2024), AutoPrompt (Shin et al., 2020), GCG (Zou et al., 2023), UAT (Wallace et al., 2019), PAIR (Chao et al., 2023), and GBDA (Guo et al., 2021). As shown in Figure 5, both the oracle and SafeRoute select the larger model most frequently for the PAP attack. Since this attack exploits persuasive taxonomy to elicit harmful responses from LLMs, the smaller model is more prone to errors than other types of attacks. On the other hand, both models select the

larger model less frequently for the GCG attack. This may be attributed to the fact that this jailbreak attack is well-known, and many of its instances are included in the dataset used to train the smaller model.

5 Conclusion

In this work, we proposed training a binary router, SafeRoute, that adaptively selects either a larger or smaller safety guard model based on the difficulty of the input data. This approach improved the trade-off between computational overhead and accuracy gains compared to other relevant baselines on several benchmark datasets. While we focused on the dynamic selection of safety guard models with different sizes, our approach is not limited to prompt-response pair classification. An interesting direction for future work is extending this method to other tasks, such as reasoning or programming.

Limitations

Although our proposed adaptive selection between a smaller and a larger safety guard model significantly improves the trade-off between accuracy gains and computational overhead compared to other baselines, it has some limitations. First, the current parameterization of the binary classifier f_θ does not encode what the larger model knows, limiting its generalization performance. In our preliminary experiments, we incorporated representations of the larger model as part of the classifier's input. While this improved accuracy, it introduced significant computational overhead, making the approach even slower than using the larger model alone. Approximating the larger model's features in an efficient manner would be an interesting direction as future work. Another limitation is that the performance of our selection mechanism is highly dependent on the quality and representativeness of the training data for f_θ. If the training dataset does not adequately capture the diversity of prompt-response pairs — particularly those at the boundary between easy and hard instances — the classifier may make suboptimal decisions. Steering LLMs to generate diverse and high-quality data is another promising avenue for future work.

Ethics Statement

Our proposed method, SafeRoute, aims to improve the trade-off between efficiency and accuracy gains of safety guard models in large language model (LLM) deployment. We do not foresee any direct ethical concerns arising from the use of SafeRoute, as it functions solely as an adaptive mechanism for selecting between smaller and larger models based on their predictive performance across different input types. By doing so, it ensures a more efficient deployment while maintaining high safety performance, reducing computational overhead without compromising the ability to detect harmful inputs. We are committed to the responsible use of LLMs and the enhancement of safety mechanisms, ensuring that no additional harm is introduced by our approach. All experiments were conducted with publicly available benchmark datasets.

Acknowledgement

This work was partially supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2020-II200153, Penetration Security Testing of ML Model Vulnerabilities and Defense), Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)), Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.RS-2022-II220713, Meta-learning Applicable to Real-world Problems), Samsung Electronics (IO201214-08145-01), and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00256259).

References

Jimmy Ba. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. *International Conference on Machine Learning (ICML)*.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion,

George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. *International Conference on Machine Learning (ICML)*.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5747–5757, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rishav Hada, Sohi Sudhir, Pushkar Mishra, Helen Yannakoudakis, Saif M. Mohammad, and Ekaterina Shutova. 2021. Ruddit: Norms of offensiveness for English Reddit comments. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2700–2717, Online. Association for Computational Linguistics.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Peter Lee. 2016. Learning from Tay's introduction.

Seanie Lee, Haebin Seong, Dong Bok Lee, Minki Kang, Xiaoyin Chen, Dominik Wagner, Yoshua Bengio, Juho Lee, and Sung Ju Hwang. 2025. HarmAug: Effective data augmentation for knowledge distillation of safety guard models. *International Conference on Learning Representations (ICLR)*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. 2023. ToxicChat: Unveiling hidden challenges of toxicity detection in real-world user-AI conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4694–4702, Singapore. Association for Computational Linguistics.

Hongfu Liu, Hengguan Huang, Hao Wang, Xiangming Gu, and Ye Wang. 2025. On calibration of LLM-based guard models for reliable content moderation. *International Conference on Learning Representations (ICLR)*.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. *International Conference on Learning Representations (ICLR)*.

AI @ Meta Llama Team. 2024. The llama 3 family of models. `https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard3/1B/MODEL_CARD.md`.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. *Association for the Advancement of Artificial Intelligence (AAAI)*.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *Internation Conference on Machine Learning (ICML)*.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2024. Tree of attacks: Jailbreaking black-box LLMs automatically. *Advances in Neural Information Processing systems (NeurIPS)*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing systems (NeurIPS)*.

Inkit Padhi, Manish Nagireddy, Giandomenico Cornacchia, Subhajit Chaudhury, Tejaswini Pedapati, Pierre Dognin, Keerthiram Murugesan, Erik Miehling, Martín Santillán Cooper, Kieran Fraser, Giulio Zizzo, Muhammad Zaid Hameed, Mark Purcell, Michael Desmond, Qian Pan, Zahra Ashktorab, Inge Vejsbjerg, Elizabeth M. Daly, Michael Hind, Werner Geyer, Ambrish Rawat, Kush R. Varshney, and Prasanna Sattigeri. 2024. Granite guardian. *arXiv preprint arXiv:2412.07724*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems (NeurIPS)*.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400, Mexico City, Mexico. Association for Computational Linguistics.

Guru Sethupathy. 2024. An executive's guide to the risks of large language models (LLMs): From hallucinations to copyright infringement.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1667–1682, Online. Association for Computational Linguistics.

Dominik Wagner, Seanie Lee, Ilja Baumann, Philipp Seeberger, Korbinian Riedhammer, and Tobias Bocklet. 2024. Optimized speculative sampling for GPU hardware accelerators. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6442–6458, Miami, Florida, USA. Association for Computational Linguistics.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2024. GPT-4 is too smart to be safe: Stealthy chat with LLMs via cipher. *International Conference on Learning Representations (ICLR)*.

Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. 2024. How johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14322–14350, Bangkok, Thailand. Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *International Conference on Machine Learning (ICML)*.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A Heller, and Subhrajit Roy. 2024. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. *International Conference on Learning Representations (ICLR)*.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial

attacks on aligned language models. *arXiv preprint arXiv: 2307.15043*.

A Proof of Theorem 3.1

Proof.

$$\begin{aligned} R_{\text{adaptive}} - R_{\text{oracle}} =& \mathbb{E}[I(\mathbf{x},\mathbf{y})\ell(p(\mathbf{x},\mathbf{y}),c) \\ &+ (1 - I(\mathbf{x},\mathbf{y}))\ell(q(\mathbf{x},\mathbf{y}),c) \\ &- t(\mathbf{x},\mathbf{y})\ell(p(\mathbf{x},\mathbf{y}),c) \\ &- (1 - t(\mathbf{x},\mathbf{y}))\ell(q(\mathbf{x},\mathbf{y}),c)]. \end{aligned}$$

Taking the absolute value and using the fact that

$$|I(\mathbf{x},\mathbf{y}) - t(\mathbf{x},\mathbf{y})| = \mathbb{1}_{\{I(\mathbf{x},\mathbf{y}) \neq t(\mathbf{x},\mathbf{y})\}},$$

we obtain the following inequality,

$$\begin{aligned} &|R_{\text{adaptive}} - R_{\text{oracle}}| \\ &\leq \mathbb{E}[\mathbb{1}_{\{I(\mathbf{x},\mathbf{y}) \neq t(\mathbf{x},\mathbf{y})\}} |\ell(p(\mathbf{x},\mathbf{y}),c) - \ell(q(\mathbf{x},\mathbf{y}),c)|]. \end{aligned}$$

For notational brevity, we use $I \neq t$ to denote $I(\mathbf{x},\mathbf{y}) \neq t(\mathbf{x},\mathbf{y})$. By applying the Cauchy-Schwarz inequality, we obtain the final result,

$$\begin{aligned} &|R_{\text{adaptive}} - R_{\text{oracle}}| \\ &\leq \sqrt{\mathbb{E}[\mathbb{1}^2_{\{I \neq t\}}]} \sqrt{\mathbb{E}[|\ell(p(\mathbf{x},\mathbf{y}),c) - \ell(q(\mathbf{x},\mathbf{y}),c)|^2]} \\ &= \sqrt{\mathbb{E}[\mathbb{1}_{\{I(\mathbf{x},\mathbf{y}) \neq t(\mathbf{x},\mathbf{y})\}}]} M \\ &= \sqrt{\mathbb{P}(I(\mathbf{x},\mathbf{y}) \neq t(\mathbf{x},\mathbf{y}))} M \end{aligned}$$

where $M = \sqrt{\mathbb{E}[|\ell(p(\mathbf{x},\mathbf{y}),c) - \ell(q(\mathbf{x},\mathbf{y}),c)|^2]}$. Thus, we have

$$R_{\text{adaptive}} \leq R_{\text{oracle}} + M\sqrt{\mathbb{P}(I(\mathbf{x},\mathbf{y}) \neq t(\mathbf{x},\mathbf{y}))}.$$

□

B Data Statistics

Dataset	# of safe	# of harmful	Total
OAI	1,158	522	1,680
WildGuardMix	1,407	282	1,689
WildGuardMix-p	945	754	1,699
ToxicChat	4,721	362	5,083
XSTest	368	78	446
Harmbench	329	273	602

Table 5: Statistics of each dataset.

C Safety Guard Models

We use PyTorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020) to implement all methods. All the pre-trained models, including safety guard models, used for our experiments are available in Hugging Face Hub. We list the identifier and link for each model on the Hugging Face Hub in Table 6.

Model	Hugging Face Hub Identifier
`Llama-Guard-3-1B`	meta-llama/Llama-Guard-3-1B
`Llama-Guard-3-8B`	meta-llama/Llama-Guard-3-8B
`Granite-Guardian-3-8B`	ibm-granite/granite-guardian-3.0-8b
`ModernBert`	answerdotai/ModernBERT-large
`Llama-3.1-8B-Instruct`	meta-llama/Llama-3.1-8B-Instruct

Table 6: Hugging Face Hub model identifiers for the pre-trained models used in our work.

D Additional Experimental Results

In Figure 6 and Figure 7, we present trade-off between FLOPs and F1 score when adaptively selecting between the smaller (`Llama-Guard-3-1B`) and larger (`Llama-Guard-3-8B` and `Granite-Guardian-3-8B`, respectively) models. In Figure 8 and Figure 9, we present trade-off between usage ratio of large model and F1 score when adaptively selecting between the smaller (`Llama-Guard-3-1B`) and larger (`Llama-Guard-3-8B` and `Granite-Guardian-3-8B`, respectively) models.

E Prompt for Paraphrasing

We present the prompt format for paraphrasing prompt-response pairs in Figure 10.

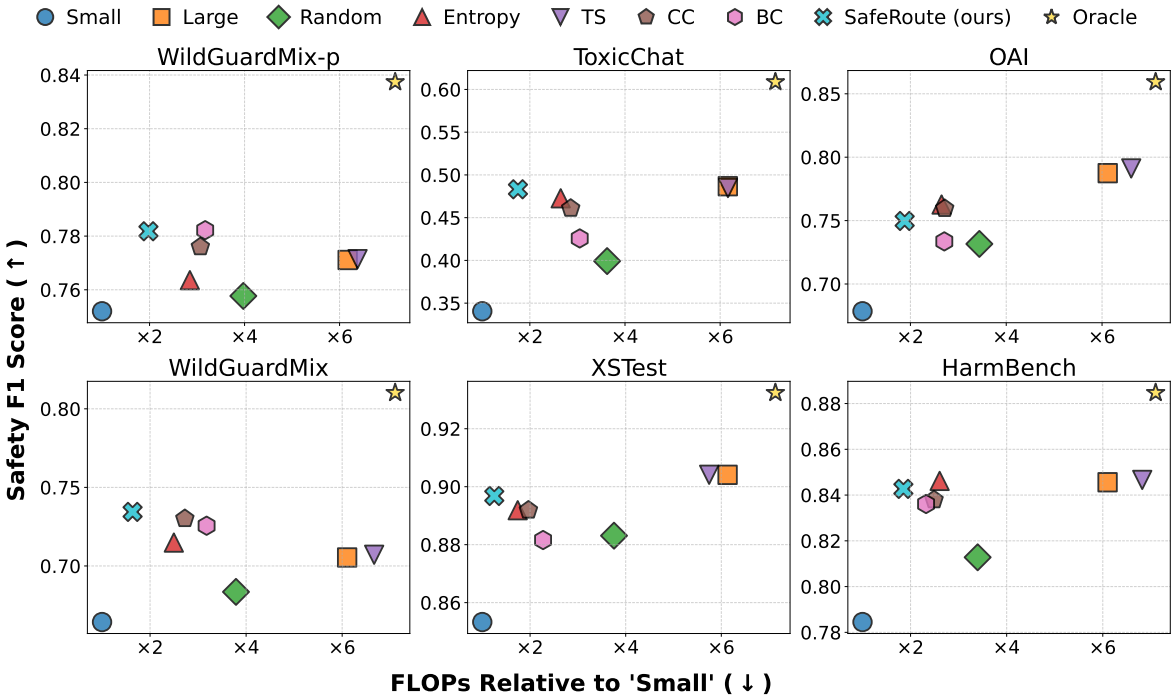

Figure 6: **FLOPs (↓) vs. safety F1 score (↑) trade-off** when using the smaller (`Llama-Guard-3-1B`) and larger (`Llama-Guard-3-8B`) models.

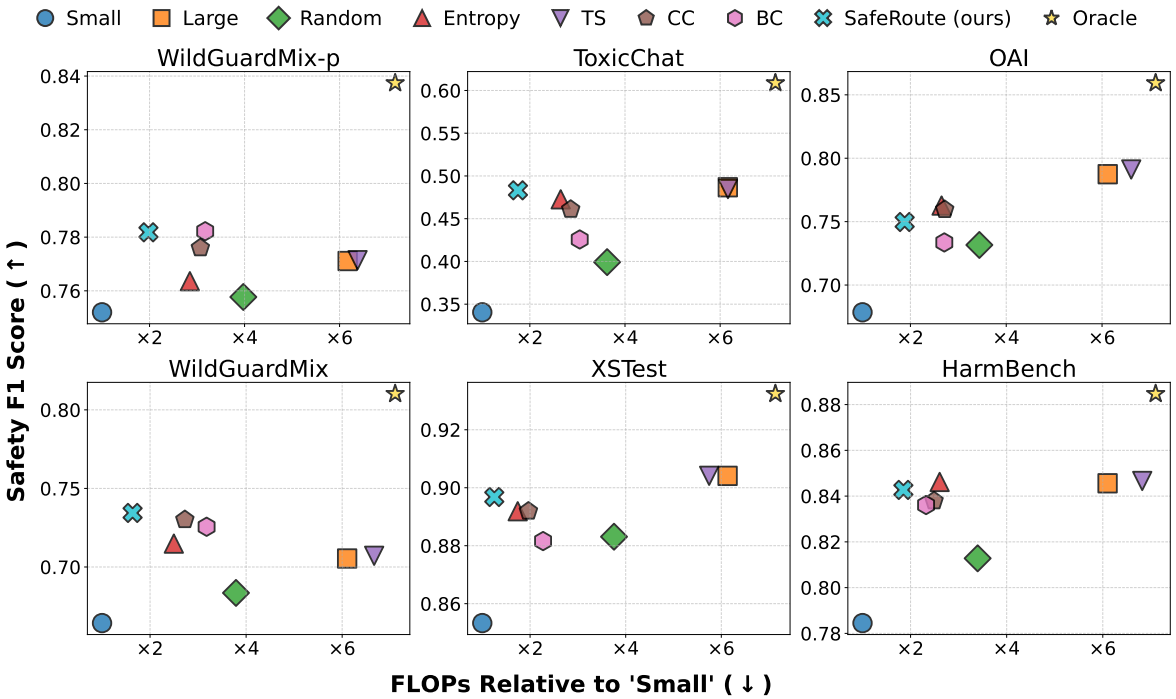

Figure 7: **FLOPs (↓) vs. safety F1 score (↑) trade-off** when using the smaller (`Llama-Guard-3-1B`) and larger (`Granite-Guardian-3-8B`) models.

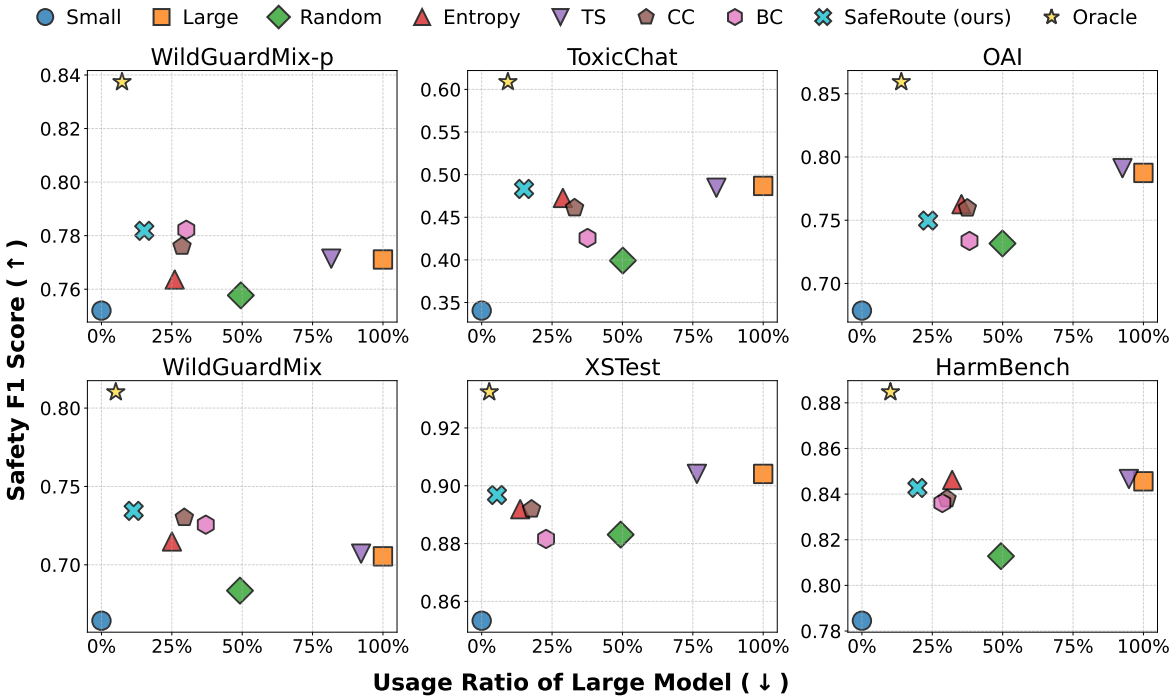

Figure 8: **Usage ratio of large model (↓) vs. safety F1 score (↑) trade-off** when using the smaller (`Llama-Guard-3-1B`) and larger (`Llama-Guard-3-8B`) models.

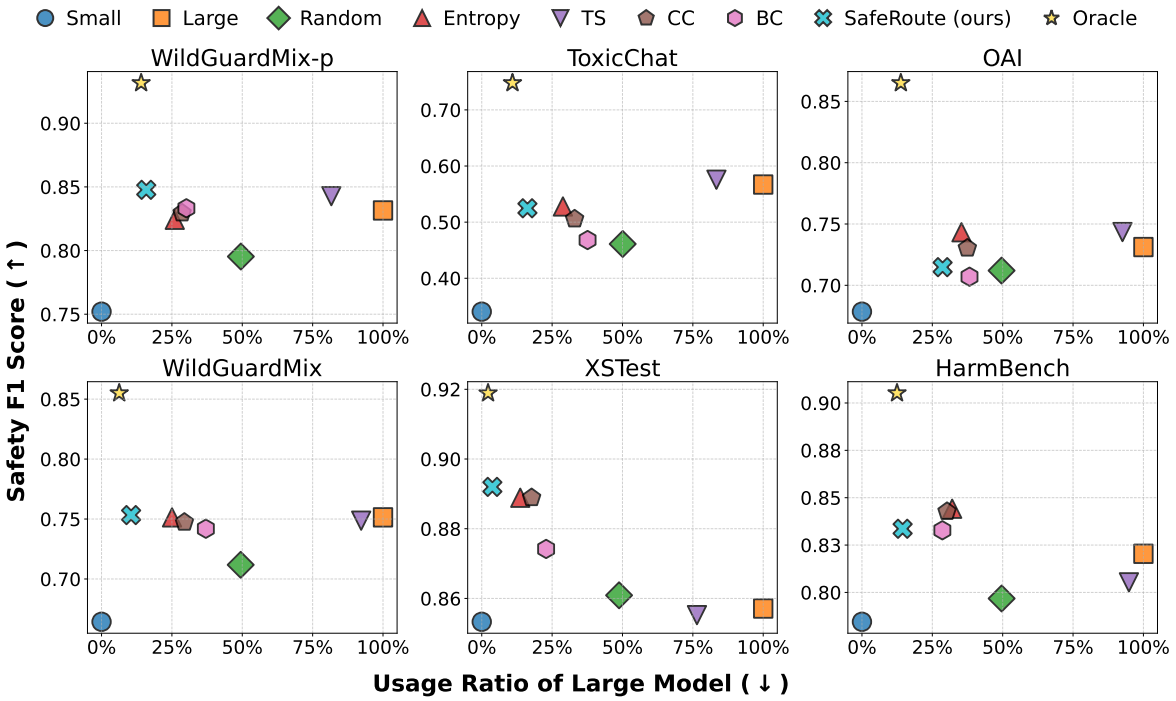

Figure 9: **Usage ratio of large model (↓) vs. safety F1 score (↑) trade-off** using the smaller (`Llama-Guard-3-1B`) and larger (`Granite-Guardian-3-8B`) models.

Prompt

USER: Paraphrase the following prompt and response respectively while preserving their original semantics. Adhere strictly to the following format. First do not include original prompt and response. Second, start the paraphrased prompt with "prompt:". Third, start the paraphrased response with "response:". Fourth, if response is given as None, just provide it as None.
prompt: `{prompt}`
response: `{response}`
ASSISTANT:

Figure 10: The prompt format for paraphrasing prompt-response pairs.

Table 7: The average of safety F1 score, precision (Prec.), recall (Rec.), and latency (Lat.) when using smaller (`Llama-Guard-3-1B`) and larger (`Llama-Guard-3-8B`) models.

Method	**WildGuardMix-p**				**ToxicChat**				**OAI**				**WildGuardMix**				**XSTest**				**HarmBench**			
	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.
Small	0.752	0.827	0.690	6.66	0.341	0.242	0.572	17.05	0.679	0.587	0.805	9.24	0.664	0.677	0.653	14.60	0.853	0.889	0.821	3.59	0.785	0.726	0.854	5.45
Large	0.771	0.929	0.659	22.67	0.487	0.467	0.508	70.68	0.787	0.784	0.791	42.72	0.705	0.811	0.624	64.16	0.904	0.971	0.846	10.54	0.846	0.811	0.883	25.63
Random	0.758	0.882	0.664	17.79	0.399	0.320	0.534	59.23	0.732	0.679	0.796	28.42	0.684	0.753	0.629	42.18	0.883	0.932	0.839	8.38	0.813	0.770	0.863	16.17
Ent	0.764	0.902	0.662	16.16	0.473	0.421	0.539	41.49	0.763	0.736	0.791	22.35	0.715	0.824	0.631	31.64	0.892	0.943	0.846	6.14	0.846	0.809	0.886	13.09
TS	0.771	0.932	0.658	25.39	0.485	0.470	0.500	76.68	0.791	0.795	0.787	48.36	0.707	0.822	0.621	73.26	0.904	0.971	0.846	11.07	0.847	0.816	0.879	30.17
CC	0.776	0.892	0.687	14.75	0.461	0.387	0.569	42.54	0.760	0.707	0.820	22.53	0.730	0.807	0.667	33.83	0.892	0.943	0.846	5.80	0.838	0.791	0.890	12.49
BC	0.782	0.868	0.712	14.97	0.426	0.328	0.605	45.01	0.734	0.647	0.847	32.23	0.726	0.739	0.713	39.37	0.882	0.905	0.859	6.02	0.836	0.769	0.916	11.78
SafeRoute (Ours)	0.782	0.926	0.677	11.24	0.483	0.416	0.576	31.51	0.750	0.672	0.848	18.46	0.734	0.856	0.643	28.18	0.897	0.967	0.836	4.71	0.843	0.797	0.895	9.92
Oracle	0.837	0.951	0.748	28.27	0.609	0.582	0.638	95.18	0.859	0.842	0.877	63.58	0.810	0.904	0.734	77.19	0.932	0.986	0.885	12.47	0.885	0.834	0.941	31.25

Table 8: The average of safety F1 score, precision (Prec.), recall (Rec.), and latency (Lat.) when using smaller (`Llama-Guard-3-1B`) and larger (`Granite-Guardian-3-8B`) models.

Method	**WildGuardMix-p**				**ToxicChat**				**OAI**				**WildGuardMix**				**XSTest**				**HarmBench**			
	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.	F1	Prec.	Rec.	Lat.
Small	0.752	0.827	0.690	6.46	0.341	0.242	0.570	17.20	0.679	0.587	0.805	9.32	0.664	0.677	0.653	14.90	0.853	0.889	0.821	3.47	0.785	0.726	0.854	5.60
Large	0.832	0.757	0.923	25.65	0.567	0.423	0.859	93.55	0.731	0.617	0.897	62.64	0.751	0.770	0.734	87.94	0.857	0.913	0.808	12.85	0.820	0.848	0.795	34.47
Random	0.795	0.795	0.796	21.41	0.461	0.338	0.716	52.70	0.712	0.618	0.847	33.87	0.712	0.743	0.682	53.06	0.861	0.893	0.813	9.36	0.797	0.782	0.812	19.69
Ent	0.824	0.837	0.812	19.61	0.528	0.418	0.716	43.52	0.743	0.661	0.849	25.93	0.751	0.795	0.713	37.93	0.889	0.970	0.821	6.12	0.844	0.835	0.854	15.31
TS	0.843	0.780	0.916	28.64	0.576	0.440	0.832	83.49	0.743	0.641	0.885	62.34	0.749	0.788	0.713	97.48	0.855	0.925	0.795	12.90	0.805	0.841	0.773	38.77
CC	0.830	0.817	0.842	16.57	0.506	0.381	0.751	45.42	0.730	0.629	0.872	25.76	0.747	0.761	0.734	40.84	0.889	0.970	0.821	5.94	0.843	0.806	0.883	14.74
BC	0.833	0.798	0.871	16.77	0.468	0.333	0.790	47.12	0.707	0.584	0.895	25.51	0.742	0.716	0.769	46.90	0.874	0.904	0.846	6.47	0.833	0.766	0.912	13.65
SafeRoute (Ours)	0.848	0.879	0.819	12.19	0.525	0.395	0.781	35.12	0.715	0.589	0.910	21.02	0.753	0.825	0.693	25.07	0.892	0.970	0.826	4.28	0.834	0.796	0.876	10.32
Oracle	0.932	0.924	0.939	30.81	0.748	0.650	0.881	94.17	0.865	0.802	0.939	85.14	0.855	0.912	0.805	104.22	0.919	0.971	0.872	16.68	0.905	0.885	0.927	40.13