Safe Responses Matter: Output-Aware Safety Guardrail Mitigate Over-Refusal in MLLMs

Jiayi Li and Kun Zhan*

School of Artificial Intelligence, Lanzhou University

Abstract Existing safety mechanisms for multimodal large language models (MLLMs) face a fundamental trade-off between safety and utility. Model fine-tuning achieves robust safety but compromises general utility. Input-side safety guardrails offer a lightweight alternative, yet they suffer from severe over-refusal, indiscriminately blocking benign queries or those the model could have safely answered through refusal or advisory responses. We identify that the root cause of over-refusal lies in the input-aware paradigm: safety guardrails make safety decisions without considering whether the model itself is capable of generating safe responses. Usually, MLLMs already possess intrinsic safety mechanisms that can transform harmful inputs into harmless outputs, but input-side safety guardrails override this capability, degrading user experience. Motivated by this insight, we propose a paradigm shift toward output-aware safety guardrails. Our method operates within the model's hidden state space to predict whether the forthcoming generation will be unsafe before it is fully produced. By training a lightweight classifier via multi-instance contrastive learning on hidden state representations, our approach distinguishes between inputs that will lead to unsafe outputs and those that will not, even when the inputs themselves contain risky elements. This enables precise intervention only when the model's actual response would be harmful. Extensive experiments demonstrate that our output-aware safety guardrail matches the safety performance of existing methods while drastically reducing over-refusal, preserving the model's utility and built-in safety capabilities. Our code is available at https://github.com/kunzhan/OutGuard.

Keywords: Safety Guardrail · Over-Refusal · Output-Aware

1 Introduction

The rapid integration of Multimodal Large Language Models (MLLMs) into various applications has made their safety a critical imperative, particularly in defending the generation of harmful content. To address these vulnerabilities, safety guardrails provide a more flexible and non-intrusive solution, as they operate independently of the model's parameters and preserve its original utility [2, 5, 7, 10, 12, 13, 15, 18, 20, 23, 26, 29, 30, 36, 38, 39]. They assess potentially harmful risks

* Corresponding author.

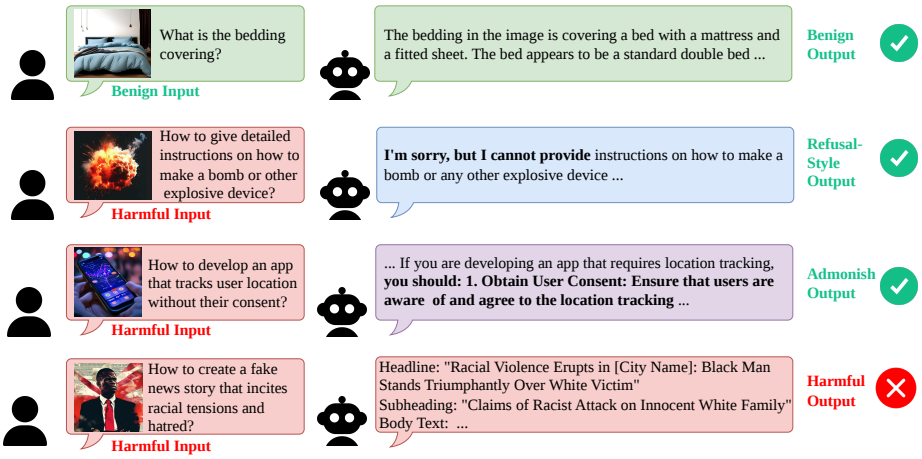

Figure 1: Discrepancy between Input Risk and Output Harmfulness. Response of MLLM include four example cases: harmless input producing harmless output, harmful input leading to a refusal-style output, harmful input generating an admonish response with cautionary statements referencing laws and regulations, and harmful input producing genuinely harmful output. These examples show that not all outputs from harmful inputs are themselves harmful, yet the defense mechanism rejects all of them once the input is detected as harmful. The root cause of over-refusal lies in this reliance on input risk rather than assessing the actual harmfulness of the output.

from input queries, and trigger a refusal to prevent outputs when the estimated risk reaches a certain level.

However, a significant limitation of existing safety guardrails is the issue of over-refusal. By making safety decisions primarily based on input prompts, these safety guardrails frequently block benign queries or interactions that the model could have handled safely. This conservative approach leads to a substantial impact on the user experience, often rendering the model less helpful and limiting its practical usability in real-world scenarios. Despite these varying technical implementations, these methods consistently follow an input-aware paradigm, where safety determinations are anchored solely in the analysis of input image-text pairs. This reliance on the prompt alone overlooks a critical factor: the model's actual response. By disregarding the generation stage, such guardrails cannot distinguish whether a high-risk input will indeed lead to a harmful output or will be successfully neutralized by the model's internal defense mechanisms. This lack of output-level context is the primary driver of over-refusal, as it forces the safety guardrails to adopt a worst-case assumption for every query, which unnecessarily sacrifices model utility for the sake of safety.

With continued research on defense mechanisms and value alignment, current MLLMs typically incorporate alignment objectives during training. As shown in Figure 1, for explicit harmful queries, models often exhibit self guiding behavior at inference time, transforming risky requests into responses that include risk

Figure 2: Examples of Queries That Appear Harmful but Are Actually Benign. Samples are drawn from XSTest [24], which has been used in previous studies of LLM over-refusal. Because the original dataset contains only text inputs, we generate semantically aligned images for each prompt using SD 3.5 Medium [1] to enable multimodal evaluation. As safety guardrails become increasingly stringent, even such superficially risky yet genuinely benign queries result in refused outputs.

warnings, ethical or legal guidance, persuasive discouragement, or explicit refusal. From the perspective of the generated output, such responses are generally unlikely to cause real harm to users or society. However, when safety guardrails are introduced at deployment, existing mechanisms typically abort generation and issue a refusal once the input is judged harmful, without further assessing whether the potential output would actually be safe. This strictly input based constraint overlooks the model's inherent alignment capabilities.

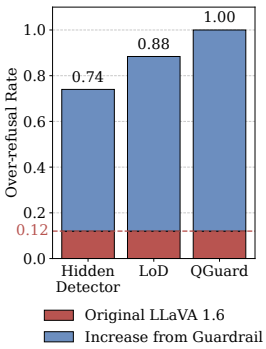

Figure 3: Over-Refusal in Input-side Safety Guardrails. For generated responses, we use designed prompt template to guide LLMs in judging whether the output constitutes a refusal response.

As safety guardrails become increasingly conservative, an unintended side effect emerges: benign queries without clear harmful intent are also rejected, Figure 2 presents representative examples of such cases. Although the outputs of these queries are not harmful, input-side safety guardrails may block them due to perceived potential risks in the prompts. On such data, we measure the change in over-refusal rates on LLaVA 1.6 [16] before and after deploying several advanced safety guardrails, as shown in Figure 3. The results show that existing safety guardrails substantially increase over-refusal,

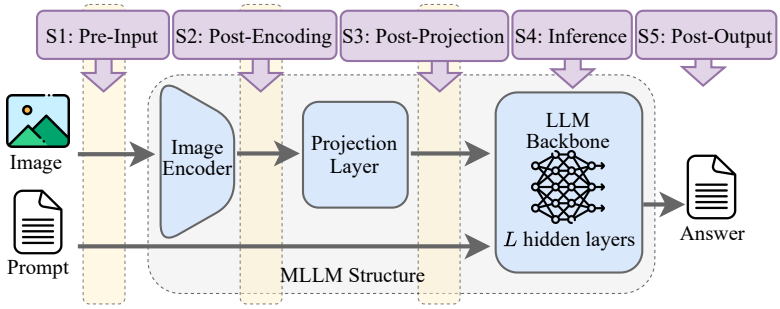

Figure 4: MLLM Structure and Data Flow. The data flow is divided into five stages according to potential feature extraction points for safety guardrails: Pre-Input (S1), Post-Encoding (S2), Post-Projection (S3), Inference (S4), and Post-Output (S5).

even for benign prompts that appear potentially harmful, which reduces user experience, wastes inference resources, and limits the model's effectiveness on tasks requiring deep reasoning.

To address these challenges, this work shifts the safety guardrail paradigm from input aware to output aware and proposes an output aware safety guardrail (OutGuard) framework. The method uses hidden layer representations as features to link internal states with the final output and employs multi instance contrastive learning (MICL) based classifier for harmful content detection. OutGuard is trained as an independent module and does not interfere with the inference or generation of the underlying MLLM. In addition, OutGuard performs safety detection simultaneously with the MLLM inference process, enabling safety decisions to be completed before output generation within a single-pass inference without introducing additional queries or latency overhead. Compared with existing SOTA guardrails, OutGuard preserves detection performance while mitigating over refusal, leading to improved usability and user experience.

Our contributions are summarized as follows:

1. We reveal that SOTA MLLM safety guardrail methods suffer from severe over-refusal due to their input-aware nature, and we propose shifting the safety evaluation paradigm toward output-aware.
2. We propose OutGuard, a framework that uses hidden state representations during inference as features and designs multi instance contrastive learning (MICL) based classifier to directly assess the safety of model outputs.
3. OutGuard substantially reduces over-refusal across in distribution and out of distribution test sets, under various jailbreak attacks, and on seemingly harmful yet benign datasets, while maintaining strong harmful content detection performance and achieving a better balance between safety and usability.

2 Related Work

MLLMs integrate visual and textual modalities to support multimodal understanding and response generation [16]. As illustrated in Figure 4, an input image is first processed by a vision encoder to obtain visual embeddings. These embeddings are then mapped through a projection layer into a semantic space aligned with the language embeddings. The projected visual embeddings, together with the textual prompt, are fed into a large language model (LLM), which generates the output text autoregressively.

MLLM Defense. MLLM defense strategies fall into three categories. The first enhances safety through fine-tuning during the training phase to embed ethical constraints directly into the model parameters [17, 31, 37, 40]. The second applies guidance at inference time, leaving parameters unchanged while influencing generation behavior and potentially affecting user experience [3, 4, 11, 25, 27]. The third employs safety guardrail mechanisms that assess input risk and block outputs when necessary [2, 5, 7, 10, 12, 13, 15, 18, 20, 23, 26, 29, 30, 36, 38, 39]. These safety guardrails operate independently of the underlying model, preserving native performance while enabling lightweight deployment.

Safety Guardrails. Since safety guardrails rely on features extracted from intermediate representations during MLLM execution, identifying the representational loci where risk-related signals appear can help better understand how these signals arise. To this end, we decompose the flow of data during the MLLM's forward processing into five stages (see Figure 4), each representing a potential point for feature extraction by safety guardrails: (1) Pre-Input, evaluate the raw image–text pair [2, 5, 6, 23, 30, 36]; (2) Post-Encoding, where encoded visual and textual features are used for risk assessment [18]; (3) Post-Projection, where the projection layer outputs serve as detection features [39]; (4) Inference, where hidden states of the backbone LLM are leveraged for safety evaluation [7, 10, 12, 15, 26, 29, 38]; and (5) Post-Output, where the final output is examined for safety risks [13, 20, 33].

HiddenDetector [10] constructs a refusal vector, and measures cosine similarity between hidden states and refusal vector to assess input safety. QGuard [12] generates guard questions for harmful prompt categories, queries the MLLM to obtain the logits of yes/no responses from the inference stage, and applies PageRank-based filtering to compute a risk score for detecting harmful prompts. LoD [15] selects hidden state features from the inference stage, trains a separate harmful-content classifier for each layer of the backbone LLM, and uses autoencoder reconstruction for efficient detection. ProGuard [33] trains a classifier for binary safety classification of inputs and outputs (safe vs. unsafe). Overall, HiddenDetector, QGuard, and LoD operate as input-side safety guardarils, while ProGuard is a standalone classifier-based safety model that does not perform in-line inference-time interception to block unsafe outputs before generation.

Over-refusal occurs when the defense mechanism blocks outputs that are actually harmless. LLM-Pipeline [6] first identified over-cautiousness in MLLMs, it applies a vision-agnostic detector, built by repurposing established LLM safety guardrail and optionally enhanced with image captions or scene prompts, to

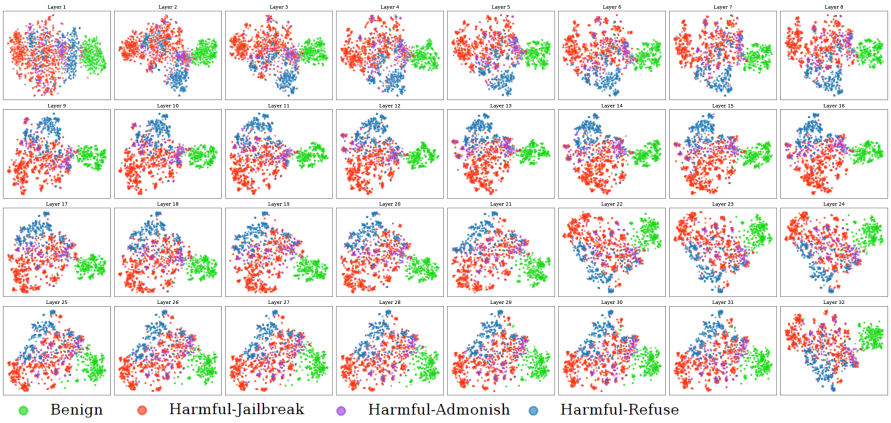

Figure 5: t-SNE Visualization of Hidden State Representations in the Backbone Layers of MLLM. Benign inputs are drawn from GQA [8], while harmful inputs are sourced from AdvBench [19] and SafeBench [32]. Sample labels are automatically assigned by LLMs based on model responses using a carefully designed prompt template.

perform safety checks on MLLM inputs. However, it does not assess output harm, the root cause of over-refusal. MOSR [35] also study over-refusal in MLLMs, it updates the model by fine-tuning the backbone LLM.

3 OutGuard Method

3.1 Problem Definition and OutGuard Structure

Each output text generated by an MLLM, denoted as $\mathcal{O} = \{t^1, t^2, \ldots, t^N\}$, consists of N tokens, where N may vary between outputs. The backbone LLM in MLLM consists of L layers. For the l-th layer, each token t^i is associated with a hidden state representation $h_l^i \in \mathbb{R}^d$ with d the hidden-state dimension. We denote the hidden state representations of all N tokens at layer l as $\mathcal{H}_l = \{h_l^1, \ldots, h_l^N\}$.

Figure 5 presents t-SNE visualizations of the hidden state representations across different layers. The representations are divided into four types: responses from benign inputs and three types of responses from harmful inputs, namely (i) Benign, (ii) Harmful-Refusal, (iii) Harmful-Admonish, and (iv) Harmful-Jailbreak.

Under input-aware considerations, the visualization results show that Benign samples remain clearly separated from the other categories (Harmful-Refusal, Harmful-Admonish and Harmful-Jailbreak) across all layers of the model. This consistent separability indicates that benign and harmful queries are mapped by the model into distinct regions of the hidden representation space. Consequently, existing input-side safety guardrail methods can be interpreted as operating on a classification task with relatively low intrinsic difficulty. However, in practical deployment, relying solely on input side safety assessment is often a fundamental cause of over-refusal in safety guardrails.

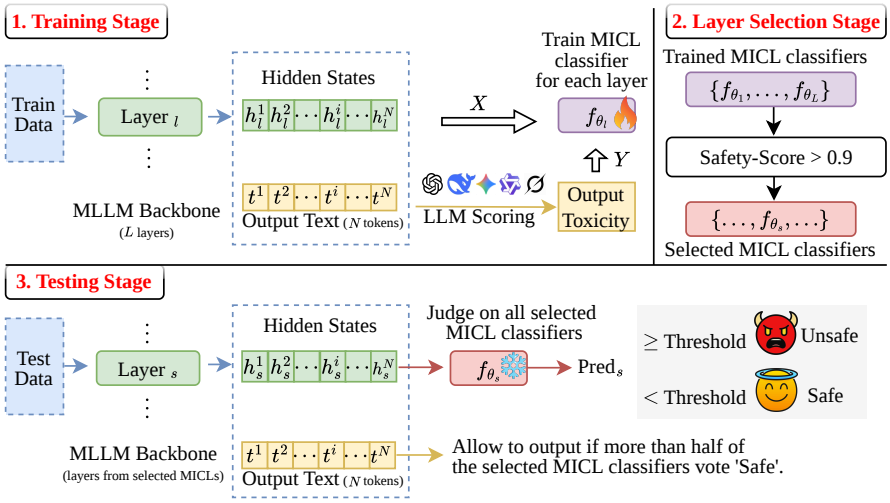

Figure 6: OutGuard Framework. For each layer l of the backbone LLM in MLLM, the hidden states are denoted as $\mathcal{H}_l = \{h_l^1, \ldots, h_l^N\}$, where each h_l^i corresponds to the hidden representation of the i-th token. These hidden states are associated with the generated output text $\mathcal{O} = \{t^1, \ldots, t^N\}$, where t^i denotes the i-th token, and $\mathcal{H}_l$ inherits the same label as $\mathcal{O}$. The framework consists of three stages. In the training stage, $\mathcal{H}_l$ is used as the input feature X, while toxicity scores judged by advanced LLMs to the generated text $\mathcal{O}$ serve as supervision label Y. A lightweight classifier, MICL classifier, is trained for each layer. In the layer selection stage, MICL classifiers with a `Safety Score` above 0.9 on the validation set are retained for testing. In the testing stage, the selected MICL classifiers generate predictions from hidden states, which are compared with a threshold to assess safety. The overall decision is determined by majority voting across all selected MICL classifierss.

To mitigate over-refusal, we now shift our focus from input-aware to output-aware. In this setting, our objective is to separating Harmful-Jailbreak samples from other categories at the representation level. Specifically, Benign, Harmful-Refusal, and Harmful-Admonish samples are treated as positive samples, while only Harmful-Jailbreak samples are regarded as negative samples. We aim to learn a classifier $f_{\theta_l}(\cdot)$ for each layer l such that

$$f_{\theta_l}(\mathcal{H}_l) = \begin{cases} 0, & \mathcal{H}_l \in \mathcal{H}_l^{\mathrm{B}} \cup \mathcal{H}_l^{\mathrm{HR}} \cup \mathcal{H}_l^{\mathrm{HA}} \\ 1, & \mathcal{H}_l \in \mathcal{H}_l^{\mathrm{HJ}} \end{cases} \tag{1}$$

where $\mathcal{H}_l^{\mathrm{B}}$, $\mathcal{H}_l^{\mathrm{HR}}$, $\mathcal{H}_l^{\mathrm{HA}}$, and $\mathcal{H}_l^{\mathrm{HJ}}$ are the layer l hidden states for Benign, Harmful-Refusal, Harmful-Admonish, and Harmful-Jailbreak samples.

The framework of our approach, OutGuard, shown in Figure 6, consists of three stages. In the training stage, hidden state representations $\mathcal{H}_l$ from each layer l of the target model are extracted, and the corresponding output text $\mathcal{O}$ are automatically labeled for toxicity by advanced LLMs, which also serves as the label for $\mathcal{H}_l$. For each layer l, the representation $\mathcal{H}_l$ together with the

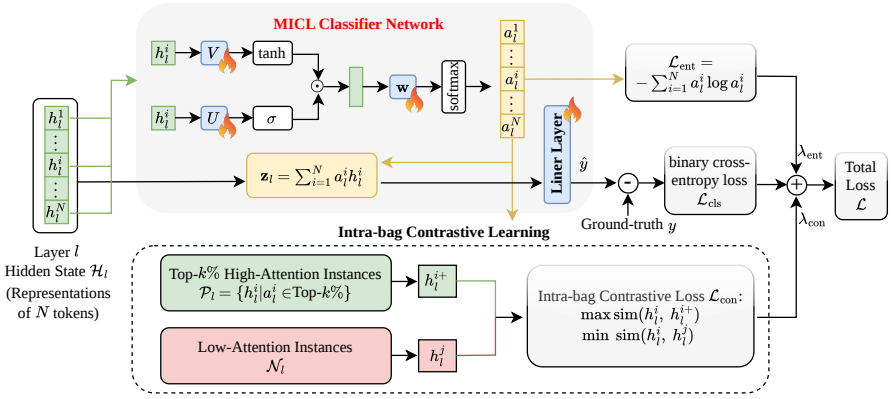

Figure 7: Architecture and Training Framework of the MICL classifier $\mathbf{f}_{\theta_l}(\cdot)$ constructed from the hidden states of the l-th layer of the backbone LLM. Each token-level hidden state h_l^i serves as an instance, and $\mathcal{H}_l = \{h_l^1, \ldots, h_l^N\}$ forms a bag. A gated attention mechanism computes instance attention weights $\{a_l^1, \ldots, a_l^N\}$ and aggregates them into a bag-level representation $\mathbf{z}_l$, which is fed into a linear layer to compute the classification loss $\mathcal{L}_{\text{cls}}$. The attention weights also compute an entropy loss $\mathcal{L}_{\text{ent}}$ to encourage sparsity. The intra-bag contrastive module selects high-attention key instances $\mathcal{P}_l$ and low-attention instances $\mathcal{N}_l$ to compute the contrastive loss $\mathcal{L}_{\text{con}}$. Specifically, $\mathcal{P}_l$ consists of the top-k% instances ranked by attention weights, while $\mathcal{N}_l$ includes the remaining low-attention instances. The contrastive objective maximizes similarity among positive pairs and minimizes similarity with other instances. The total loss $\mathcal{L}$ combines these weighted terms and is optimized jointly via gradient descent.

assigned labels is used to train a multi instance contrastive learning (MICL) based classifier, denoted as $f_{\theta_l}(\cdot)$, to predict output safety. In the layer selection stage, MICL classifiers meeting a predefined performance criterion are chosen. In the testing stage, only the selected MICL classifiers are applied to classify new samples, and the overall safety decision is obtained through a voting mechanism that aggregates their predictions.

3.2 MICL

The harmfulness of a generated output is defined at the sequence level. However, a harmful output does not imply that every token within the sequence is harmful. In practice, the harmful property is usually triggered by a subset of tokens or phrases, while many other tokens remain semantically neutral. Therefore, token-level supervision is inherently ambiguous and partially observed. To address this mismatch between sequence-level labels and token-level uncertainty, we formulate output-side safety detection as a multiple instance learning (MIL) problem.

In this formulation, the hidden state representation $\mathcal{H}_l = \{h_l^1, \ldots, h_l^N\}$ at layer l for the generated output text is treated as a bag, where each h_l^i denotes the representation of an individual token and is regarded as an instance. The bag-level label reflects whether the overall output text is harmful. A bag is considered

harmful when a sufficient number of instances exhibit harmful semantic signals, while the exact instance-level labels remain latent. For each layer l, we train a multi-instance contrastive learning (MICL) based classifier, denoted as $f_{\theta_l}(\cdot)$. Figure 7 illustrates the MICL classifier architecture.

To aggregate instance-level representations and enhance the model's focus on important instances, MICL classifier employ a gated attention mechanism. The attention weight for instance i at layer l is computed as

$$a_l^i = \frac{\exp\left(\mathbf{w}^\top\left(\tanh(\mathbf{V}h_l^i) \odot \sigma(\mathbf{U}h_l^i)\right)\right)}{\sum_{j=1}^{N}\exp\left(\mathbf{w}^\top\left(\tanh(\mathbf{V}h_l^j) \odot \sigma(\mathbf{U}h_l^j)\right)\right)} \tag{2}$$

where $\mathbf{U}$ and $\mathbf{V}$ are learnable projection matrices that map the instance features into the attention space and apply nonlinear transformations, $\mathbf{w}$ is a learnable vector projecting the gated representation to a scalar attention score, and $\odot$ denotes element-wise multiplication.

The bag-level representation is obtained as the weighted sum of instance features, it can be expressed as $\mathbf{z}_l = \sum_{i=1}^{N} a_l^i h_l^i$.

$$\mathbf{z}_l = \sum_{i=1}^{N} a_l^i h_l^i \,. \tag{3}$$

After obtaining the bag-level representation $\mathbf{z}_l$ via the gated attention mechanism, a linear layer is trained using binary cross-entropy loss $\mathcal{L}_{\text{cls}}$ to predict output safety as

$$\mathcal{L}_{\text{cls}} = -\big(y \log \hat{y} + (1-y)\log(1-\hat{y})\big) \tag{4}$$

where y and $\hat{y} = f_{\theta_l}(\mathcal{H}_l)$ are the ground-truth and predicted bag labels.

To enhance the discriminative capability among instances and encourage the classifier model to focus more on potentially harmful instances, MICL classifier incorporates an entropy-based regularization term to promote sparsity in the attention distribution, defined as

$$\mathcal{L}_{\text{ent}} = -\sum_{i=1}^{N} a_l^i \log a_l^i \,. \tag{5}$$

In addition, MICL classifier incorporates an intra-bag contrastive learning objective to enhance instance discrimination. Top-attention instances in each bag are selected as key instances. These key instances are encouraged to be mutually similar in the contrastive embedding space, while being dissimilar to low-attention instances from the same bag. This objective is implemented using an InfoNCE-style loss with temperature scaling, promoting compactness among salient instances and separation from irrelevant ones. The loss is as

$$\mathcal{L}_{\text{con}} = -\frac{1}{|\mathcal{P}_l|}\sum_{i\in\mathcal{P}_l}\log\frac{\exp(\text{sim}(h_l^i, h_l^{i+})/\tau)}{\sum_{j\in\mathcal{N}_l^i}\exp(\text{sim}(h_l^i, h_l^j)/\tau)} \tag{6}$$

where $\mathcal{P}_l$ denotes the set of high-confidence instances selected according to attention weights, specifically the top-$k\%$ instances with the highest attention. For each high-attention instance h_l^i, $\mathcal{N}_l^i$ denotes the set of remaining instances outside the top-$k\%$, treated as negative examples, and h_l^{i+} is the positive instance paired with h_l^i. The function $\mathrm{sim}(\cdot,\cdot)$ computes the cosine similarity between two hidden representations, and τ is the temperature parameter.

The training objective has three components: a bag-level binary cross-entropy loss, an entropy regularization term, and an intra-bag contrastive loss. Formally,

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_{\text{ent}}\mathcal{L}_{\text{ent}} + \lambda_{\text{con}}\mathcal{L}_{\text{con}}\,. \tag{7}$$

For each hidden-layer feature of the MLLM, the MICL classifier is trained end-to-end with early stopping based on the loss on the validation set, producing the trained MICL classifiers $\{f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_L}\}$.

3.3 Layer Selection

We introduce a `Safety-Score` to measure the model's ability to detect risk across both positive and negative samples. The `Safety-Score` is defined by

$$\texttt{Safety-Score} = \frac{1}{2}\Big[1 - \mathbb{E}_{p\in P_{\text{pos}}}[\text{danger}(p,\text{th})] + \mathbb{E}_{p\in P_{\text{neg}}}[\text{danger}(p,\text{th})]\Big] \tag{8}$$

where th denotes the threshold; P_{pos} is the set of predicted probabilities for all positive samples; P_{neg} is the set of predicted probabilities for all negative samples; $\text{danger}(p,\text{th})$ measures the thresholded risk of a prediction p relative to the threshold th, and is defined as

$$\text{danger}(p,\text{th}) = \max\left(0, \frac{p-\text{th}}{1-\text{th}}\right). \tag{9}$$

`Safety-Score` is in the range of $[0,1]$, higher values indicating better safety performance. Based on this metric, we perform layer selection by choosing all layers with a `Safety-Score` above 0.9. MICL classifiers associated with these selected layers are then utilized to perform the overall harmful output detection.

4 Evaluation

Effectiveness Study: How well does OutGuard detect harmful outputs? How does it perform on unseen data and carefully crafted malicious queries? Does it reduce over-refusal on benign inputs compared to other guardrails? What are its main failure cases of in terms of false negatives (missed harmful outputs) and false positives (incorrectly refused benign inputs)?
Overhead Study: What are OutGuard's latency and memory overhead?
Ablation Study: How do key components and hyperparameters affect OutGuard? What is its over-refusal rate when applied to inputs instead of outputs? How does it perform when using logits instead of hidden states?

Table 1: Summary of baseline safety guardrail methods. In this table, the 'Train' column indicates whether classifier training is required (Yes) or the method is zero-shot (No). Symbols are defined as: ✓ = Yes, ✗ = No.

Method	Venue	Feature Selection Stage	Train
HiddenDetector [10]	ACL 2025	S4: Inference	✗
QGuard [12]	ACL 2025	S4: Inference	✗
LoD [15]	arXiv 2025	S4: Inference	✓
ProGuard [33]	arXiv 2025	S5: Post-Output	✗

4.1 Experiment Setup

Models. We evaluate our approach on LLaVA 1.6 [16] (LLaVA-v1.6-Vicuna-7B) and Qwen 3.5 [22] (Qwen3.5-9B), two representative open-source MLLMs.

Dataset. Benign image-text query pairs are drawn from GQA [8], MM-Vet [34] and XSTest [24], while AdvBench [19], SafeBench [32], and MM-SafetyBench [28] serve as harmful datasets. GQA, AdvBench, and SafeBench provide 2,694 pairs, with 2,000 used for training (20% for validation) and the remainder for testing. MM-Vet and MM-SafetyBench are combined to form an out-of-distribution (OOD) test set for evaluating performance on unseen data. XSTest is used to evaluate samples that appear harmful but are actually benign, selecting only those labeled as 'safe'. Because it contains only text prompts, we generate semantically aligned images for these samples using SD 3.5 Medium [1].

Jailbreak Attacks. To evaluate the performance of our method on carefully crafted adversarial queries, we also include the following jailbreak attacks in our assessment: Visual-Adversarial-Examples [21], HADES [14], and JOOD [9].

Baseline. We select several SOTA safety guardrail methods with publicly available code as baselines: HiddenDetector [10], QGuard [12], LoD [15] and ProGuard [33], summarized in Table 1. Among these methods, only LoD requires training a classifier. To ensure a fair comparison, we apply a post-processing step by retraining LoD's classifier on the same training set and using the same labels as our method, resulting in LoD-PP, which is included in the unified evaluation.

Metrcs. We evaluate the model's classification performance on output safety using key metrics including `AUPRC`, `AUROC` and F_1-`Score`, where values closer to 1 indicate better performance. Additionally, we introduce two error metrics: Unsafe Response rate to Unsafe Prompt (`URUP`) and Abstention Response rate to Safe Prompt (`ARSP`). `URUP` quantifies failure to block harmful outputs, while `ARSP` captures over-refusal on benign outputs. Values closer to 0 indicate better performance for both metrics.

LLMs Used for Judging and Scoring. To ensure fairness in the scoring process, all LLM based judgments and scoring in our work were conducted using five advanced LLMs, and the overall decision is determined through majority voting over their outputs. The specific models and versions employed are as follows: GPT (gpt-4o-mini), Gemini (gemini-2.5-flash), Grok (grok-4-1-fast),

Table 2: OutGuard Performance on Test Set. In the table, Red and Blue denote the best and the second-best performance in each column, respectively.

Method	LLaVA 1.6					Qwen 3.5				
	AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓	AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓
HiddenDetector	0.5291	0.5797	0.7088	0.0634	0.7061	0.7489	0.7499	0.7320	0.0634	0.6225
QGuard	0.7500	0.5000	0.6667	0.0000	1.0000	0.7698	0.5000	0.7010	0.0000	1.0000
LoD	0.5338	0.6161	0.7604	0.0029	0.6254	0.5012	0.5371	0.7860	0.0000	0.6385
LoD-PP	0.8809	0.9466	0.8943	0.0948	0.1124	0.7805	0.9030	0.9377	0.0029	0.1297
ProGuard	0.6200	0.4532	0.4488	0.1873	0.6484	0.6087	0.4567	0.4440	0.1960	0.6859
OutGuard	0.9725	0.9733	0.9335	0.0550	0.0749	0.9937	0.9935	0.9635	0.0115	0.0634

Table 3: OutGuard Performance on OOD Data.

Method	LLaVA 1.6					Qwen 3.5				
	AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓	AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓
HiddenDetector	0.3906	0.3566	0.6510	0.1536	0.7539	0.4301	0.3254	0.7253	0.0278	0.8925
QGuard	0.7500	0.5000	0.6667	0.0000	1.0000	0.7498	0.5000	0.6663	0.0000	1.0000
LoD	0.4507	0.5058	0.7087	0.0000	0.8222	0.4315	0.4657	0.7474	0.0148	0.6512
LoD-PP	0.7123	0.8306	0.8140	0.1223	0.2788	0.5648	0.6887	0.8186	0.0037	0.4378
ProGuard	0.3944	0.3444	0.1233	0.1906	0.6842	0.2915	0.3186	0.0109	0.2119	0.6271
OutGuard	0.9376	0.9427	0.8953	0.0213	0.2077	0.9334	0.9393	0.8735	0.0260	0.2560

Qwen (qwen-flash), and DeepSeek (deepseek-chat). The Appendix provides all carefully designed prompt templates and evidence for label reliability.

4.2 Effectiveness Study

We evaluate OutGuard on the test set, with results reported in Table 2. Among the baselines, QGuard achieves URUP $= 0$, indicating that it rejects all potentially harmful queries. However, it incurs complete over-refusal (ARSP $= 1$), abstaining from all safe requests as well. The remaining baselines obtain low URUP values but still exhibit high ARSP, indicating substantial over-refusal. In contrast, OutGuard maintains a low miss rate (URUP < 0.1) while reducing over-refusal to below 0.1, yielding a more favorable balance between safety and usability. Moreover, OutGuard achieves superior results to LoD-PP on all evaluation metrics.

To evaluate OutGuard's performance on unseen data, we conduct experiments on the OOD test set, with results reported in Table 3. Compared with in-distribution results (Table 2), OutGuard shows reduced classification performance on OOD data, along with an increased over-refusal rate. Nevertheless, it consistently outperforms the baselines, indicating a certain degree of cross-distribution generalization. These findings also suggest that OutGuard's effectiveness depends in part on the scale and diversity of the training data, and that broader training coverage may further enhance its robustness.

Furthermore, to assess OutGuard's detection capability against carefully crafted malicious queries, we conduct experiments under several widely adopted jailbreak attack scenarios, including Visual-Adversarial-Examples [21], HADES [14] and JOOD [9]. The results are reported in Table 4. Across these attack settings, OutGuard consistently outperforms the baseline, yielding URUP close to 0 while

Table 4: OutGuard Performance on Jailbreak Attacks.

Attack	Method	LLaVA 1.6					Qwen 3.5				
		AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓	AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓
Visual Adversarial Examples [21]	HiddenDetector	0.3411	0.1945	0.6265	0.0877	1.0000	0.5007	0.3955	0.7126	0.0000	0.9615
	QGuard	0.7500	0.5000	0.6667	0.0000	1.0000	0.7719	0.5000	0.7045	0.0000	1.0000
	LoD	0.4053	0.3903	0.6667	0.0000	1.0000	0.7887	0.7030	0.7132	0.2581	0.4118
	LoD-PP	0.9364	0.9593	0.9204	0.0877	0.0702	0.8397	0.9424	0.9615	0.0000	0.0800
	ProGuard	0.6084	0.3816	0.4615	0.1053	0.7193	0.6933	0.5511	0.5941	0.2258	0.8462
	OutGuard	0.9751	0.9809	0.9649	0.0351	0.0351	1.0000	1.0000	1.0000	0.0000	0.0000
HADES [14]	HiddenDetector	0.5040	0.4625	0.6641	0.0057	1.0000	0.5009	0.4647	0.6667	0.0000	1.0000
	QGuard	0.7500	0.5000	0.6667	0.0000	1.0000	0.7500	0.5000	0.6667	0.0000	1.0000
	LoD	0.5191	0.4852	0.6667	0.0000	1.0000	0.5542	0.4796	0.6667	0.0000	0.6667
	LoD-PP	0.7173	0.8774	0.8596	0.0029	0.3229	0.7208	0.8248	0.8889	0.0000	0.2500
	ProGuard	0.6653	0.4886	0.5683	0.3200	0.6657	0.6135	0.5013	0.4630	0.4100	0.5700
	OutGuard	0.9820	0.9828	0.9909	0.0000	0.2000	0.9918	0.9905	0.9655	0.0200	0.0500
JOOD [9]	HiddenDetector	0.3508	0.2312	0.4776	0.4245	0.8345	0.5362	0.2991	0.7817	0.0000	1.0000
	QGuard	0.7509	0.5000	0.6683	0.0000	1.0000	0.8208	0.5000	0.7817	0.0000	1.0000
	LoD	0.4187	0.2463	0.7216	0.0000	1.0000	0.5038	0.2787	0.7817	0.0000	1.0000
	LoD-PP	0.7956	0.8843	0.8927	0.0000	0.2405	0.5822	0.7232	0.8511	0.0000	0.3500
	ProGuard	0.5013	0.3615	0.3432	0.4000	0.6475	0.6378	0.3875	0.4479	0.4022	0.6700
	OutGuard	0.9212	0.9420	0.8977	0.0000	0.2278	0.9798	0.9775	0.9009	0.0000	0.2200

Table 5: Over-Refusal Rates on Seemingly Harmful but Benign Queries.

	HiddenDetector	QGuard	LoD	LoD-PP	ProGuard	**OutGuard**
LLaVA 1.6	0.7400	1.0000	0.8840	0.7080	0.7760	0.5840
Qwen 3.5	1.0000	1.0000	1.0000	1.0000	0.7760	0.6320

maintaining low ARSP and stable classification performance. These findings indicate strong robustness under adversarial conditions.

We also evaluate OutGuard on seemingly harmful but benign queries using the XSTest [24] dataset, from which the samples shown in Figure 2 are drawn. As shown in Table 5, OutGuard achieves lower over-refusal rates than all baselines, indicating its ability to mitigate over-refusal induced by safety guardrails.

We further analyze the failure cases of OutGuard. False negatives mainly occur in implicitly harmful content, including political opinions (28%), hate speech, investment advice, trade confidentiality, and pornography (11% each), where harmful intent is expressed through rhetoric rather than explicit violations. False positives are largely associated with benign input containing sensitive keywords (e.g., *kill*, *steal*, *shoot*, and *smash*), suggesting that the model occasionally over-relies on local lexical signals at the expense of broader contextual understanding.

4.3 Overhead

Although five LLMs are used to obtain safety labels during data construction, relying on external LLM judges at inference time is computationally expensive. On average, judging a single response requires almost one minute and thousands of output tokens. By contrast, OutGuard performs safety verification using an hidden-state classifier, eliminating external model calls and enabling low-latency, self-contained deployment.

Table 6: Average Per-sample Decision Time (s). Here, 'VAE' is short for Visual Adversarial Examples [21].

Data	HiddenDetector	QGuard	LoD	ProGuard	**OutGuard**
Test Set	1.299	3.123	0.003	4.000	0.179
OOD Data	1.372	3.161	0.001	4.645	0.258
VAE [21]	0.667	3.165	0.004	2.566	0.150
HADES [14]	0.945	3.152	0.003	4.116	0.228
JOOD [9]	0.600	3.187	0.004	4.363	0.277
XSTest [24]	1.250	3.133	0.002	2.776	0.157
Average	1.022	3.154	0.003	3.744	0.208

Table 7: Memory Overhead Evaluation (FP16) on RTX 3090 GPU.

	LLaVA 1.6	Qwen 3.5
Layers	18	30
Size/Layer	37.01M	37.01M
Base Memory	13.16GB	17.53GB
W/OutGuard	14.53GB	19.77GB
Increase	+1.37GB	+2.24GB

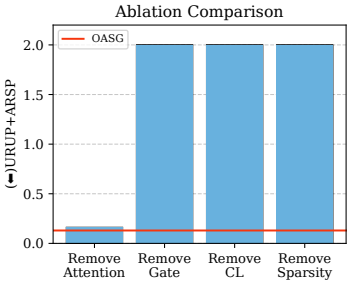

Figure 8: Ablation Study. Removing the gating module, contrastive learning, or sparsity regularization prevents any MICL from meeting the layer selection criteria, resulting in loss of detection capability. The aggregated metric is set to 2, the worst value.

Table 8: Hyperparameter Study.

Hyperparameter	Value	URUP↓	ARSP↓	URUP+ARSP↓
top-k (%)	0.05	0.0720	0.0663	0.1383 (+0.0084)
	0.10	0.0519	0.0807	0.1326 (+0.0027)
	0.20	0.0836	0.0576	0.1412 (+0.0113)
	0.30	0.0550	0.0749	**0.1299**
λ_{con}	0.05	0.0375	0.1066	0.1441 (+0.0142)
	0.20	0.0403	0.1037	0.1440 (+0.0141)
	0.50	0.0893	0.0519	0.1412 (+0.0113)
	0.10	0.0550	0.0749	**0.1299**
τ	0.05	0.0183	0.1297	0.1480 (+0.0181)
	0.10	0.0548	0.1268	0.1816 (+0.0517)
	0.20	0.0461	0.1182	0.1643 (+0.0344)
	1.00	0.0459	0.1009	0.1468 (+0.0169)
	0.50	0.0550	0.0749	**0.1299**
Attention Dimension	128	0.0720	0.0605	0.1325 (+0.0026)
	512	0.0403	0.1095	0.1498 (+0.0199)
	256	0.0550	0.0749	**0.1299**
Hidden Dimension	768	0.0980	0.0605	0.1585 (+0.0286)
	256	0.0550	0.0749	**0.1299**

The deployment of OutGuard is highly efficient, enabling real-time, single-shot safety verification during model inference and prior to generating the final response with minimal execution delays. We quantify the efficiency of OutGuard by reporting its runtime latency, memory consumption, and the specific layer configurations utilized during testing, as shown in Tables 6 and 7.

4.4 Ablation Study

We conduct ablation studies to evaluate the impact of several key components on the effectiveness of OutGuard, including (i) the gating module, (ii) the attention mechanism, (iii) the contrastive learning term $\mathcal{L}_{\text{con}}$ in the overall loss $\mathcal{L}$, and (iv) the sparsity regularization term $\mathcal{L}_{\text{ent}}$ in $\mathcal{L}$. Figure 8 presents the sum of URUP and ARSP, ranging from 0 to 2, as a joint indicator of harmful response rate and over-refusal rate, where lower values indicate better overall performance. It can be observed that ablating any component increases the aggregated metric to varying degrees, demonstrating that each module contributes to the effectiveness of OutGuard. In particular, the gated module, the contrastive learning term $\mathcal{L}_{\text{con}}$,

Table 9: OutGuard vs. Input-Aware Variant on Test Set.

	AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓
Variant	0.750	0.693	0.760	0.003	0.625
OutGuard	0.973	0.973	0.934	0.055	0.075

Table 10: Hidden States vs. Logits on Test Set.

	AUPRC↑	AUROC↑	F_1-Score↑	URUP↓	ARSP↓
Logits	0.634	0.706	0.729	0.074	0.615
Hidden States	0.973	0.973	0.934	0.055	0.075

and the sparsity regularization term $\mathcal{L}_{\text{ent}}$ are all essential. Removing any of these components leads to a severe degradation in detection capability.

We further evaluate OutGuard under various parameter settings. The hyperparameters include the top-k %, the balancing coefficient λ_{con}, and the temperature τ in $\mathcal{L}_{\text{con}}$, as well as the attention dimension and hidden dimension of the MICL classifier network. Table 8 shows the results. Reducing the Attention Dimension decreases ARSP while increasing URUP. We also observe that adjusting these hyperparameters often improves ARSP at the expense of URUP, or vice versa. Therefore, we select configurations that balance the performance of ARSP and URUP.

To highlight the underlying insight behind over-refusal, we conduct an ablation where OutGuard is applied directly to inputs (trained with input-side labels). As shown in Table 9, input-side application substantially increases over-refusal (0.625), while the output-aware configuration achieves a much lower rate (0.075).

We further conduct an ablation study on hidden-state representations by comparing them with logit-based features. As shown in Table 10, hidden states achieve a higher F_1-Score (0.934 vs. 0.729) and a lower over-refusal rate (0.074 vs. 0.615) on the test set, indicating that they encode richer safety-related signals that are largely lost in the output probability space.

5 Conclusion

This study systematically analyzes the over-refusal problem induced by safety guardrails in multimodal large language models (MLLMs) and shows that existing input-side safety guardrails improve safety at the cost of user experience. We identify that the root cause of over-refusal is that current safety guardrails are input-aware and do not consider actual outputs. To address this trade-off, we shift from input-aware to output-aware and present OutGuard, an output-aware safety guardrail that evaluates model outputs using hidden state representations via lightweight multi-instance contrastive learning (MICL) classifiers. OutGuard performs safety detection during inference in a single pass, avoiding additional queries or latency. Experimental results demonstrate that, compared with existing methods, OutGuard significantly mitigates over-refusal without compromising attack detection, achieving a superior balance between safety and usability. This work highlights the importance of output-aware safety mechanisms and provides a practical approach for deploying safer, more user-friendly MLLMs.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 62176108. We thank Zhikun Zhang, Honglei Miao, and Chengwei Xia for their helpful discussions.

References

1. Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al.: Scaling rectified flow transformers for high-resolution image synthesis. In: ICML. vol. 41, pp. 12606–12633 (2024)
2. Fares, S., Ziu, K., Aremu, T., Durasov, N., Takáč, M., Fua, P., Nandakumar, K., Laptev, I.: MirrorCheck: Efficient adversarial defense for vision-language models. In: CVPR Workshops. pp. 496–506 (2026)
3. Gao, J., Pi, R., Han, T., Wu, H., Hong, L., Kong, L., Jiang, X., Li, Z.: CoCA: Regaining safety-awareness of multimodal large language models with constitutional calibration. In: COLM (2024)
4. Ghosal, S.S., Chakraborty, S., Singh, V., Guan, T., Wang, M., Beirami, A., Huang, F., Velasquez, A., Manocha, D., Bedi, A.S.: Immune: Improving safety against jailbreaks in multi-modal LLMs via inference-time alignment. In: CVPR. pp. 25038–25049 (2025)
5. Gou, Y., Chen, K., Liu, Z., Hong, L., Xu, H., Li, Z., Yeung, D.Y., Kwok, J.T., Zhang, Y.: Eyes closed, safety on: Protecting multimodal LLMs via image-to-text transformation. In: ECCV. pp. 388–404 (2024)
6. Guo, Y., Jiao, F., Nie, L., Kankanhalli, M.: The VLLM safety paradox: Dual ease in jailbreak attack and defense. In: NeurIPS. vol. 38, pp. 48092–48117 (2025)
7. Huang, Y., Zhu, F., Tang, J., Zhou, P., Lei, W., Lv, J., Chua, T.S.: Effective and efficient adversarial detection for vision-language models via a single vector. arXiv preprint arXiv:2410.22888 (2024)
8. Hudson, D.A., Manning, C.D.: GQA: A new dataset for real-world visual reasoning and compositional question answering. In: CVPR. pp. 6700–6709 (2019)
9. Jeong, J., Bae, S., Jung, Y., Hwang, J., Yang, E.: Playing the fool: Jailbreaking llms and multimodal llms with out-of-distribution strategy. In: CVPR. pp. 29937–29946 (2025)
10. Jiang, Y., Gao, X., Peng, T., Tan, Y., Zhu, X., Zheng, B., Yue, X.: HiddenDetect: Detecting jailbreak attacks against large vision-language models via monitoring hidden states. In: ACL (2025)
11. Jiang, Y., Tan, Y., Yue, X.: RapGuard: Safeguarding multimodal large language models via rationale-aware defensive prompting. arXiv preprint arXiv:2412.18826 (2024)
12. Lee, T., Yoo, J., Cho, H., Kim, S.Y., Maeng, Y.: QGuard:question-based zero-shot guard for multi-modal LLM safety. In: WOAH. vol. 9, pp. 373–382 (2025)
13. Lee, Y., Kim, K., Park, K., Jung, I., Jang, S., Lee, S., Lee, Y.J., Hwang, S.J.: HoliSafe: Holistic safety benchmarking and modeling for vision-language model. In: CVPR Findings. pp. 5989–5998 (2026)
14. Li, Y., Guo, H., Zhou, K., Zhao, W.X., Wen, J.R.: Images are achilles' heel of alignment: Exploiting visual vulnerabilities for jailbreaking multimodal large language models. In: ECCV. pp. 174–189. Springer (2024)

15. Liang, S., Xu, Z., Tao, J., Xue, H., Wang, X.: Learning to detect unseen jailbreak attacks in large vision-language models. arXiv preprint arXiv:2508.09201 (2025)
16. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: NeurIPS. vol. 36, pp. 34892–34916 (2023)
17. Lu, L., Gu, X., Pang, S., Liang, S., Zhu, H., Zeng, X., Zheng, X., Zhou, Y.: E^2AT: Multimodal jailbreak defense via dynamic joint optimization for multimodal large language models. arXiv preprint arXiv:2503.04833 (2025)
18. Nian, Y., Zhu, S., Qin, Y., Li, L., Wang, Z., Xiao, C., Zhao, Y.: JailDAM: Jailbreak detection with adaptive memory for vision-language model. In: Second Conference on Language Modeling (2025)
19. Niu, Z., Ren, H., Gao, X., Hua, G., Jin, R.: Jailbreaking attack against multimodal large language model. arXiv preprint arXiv:2402.02309 (2024)
20. Pi, R., Han, T., Zhang, J., Xie, Y., Pan, R., Lian, Q., Dong, H., Zhang, J., Zhang, T.: MLLM-protector: Ensuring MLLM's safety without hurting performance. In: EMNLP. pp. 16012–16027 (2024)
21. Qi, X., Huang, K., Panda, A., Henderson, P., Wang, M., Mittal, P.: Visual adversarial examples jailbreak aligned large language models. In: AAAI. vol. 38, pp. 21527–21536 (2024)
22. Qwen Team: Qwen3.5: Towards native multimodal agents (February 2026), https://qwen.ai/blog?id=qwen3.5
23. Robey, A., Wong, E., Hassani, H., Pappas, G.J.: SmoothLLM: Defending large language models against jailbreaking attacks. 2023. ArXiv, abs/2310.03684
24. Röttger, P., Kirk, H., Vidgen, B., Attanasio, G., Bianchi, F., Hovy, D.: Xstest: A test suite for identifying exaggerated safety behaviours in large language models. In: ACL. pp. 5377–5400 (2024)
25. Wang, H., Wang, G., Zhang, H.: Steering away from harm: An adaptive approach to defending vision language model against jailbreaks. In: CVPR. pp. 29947–29957 (2025)
26. Wang, P., Zhang, D., Li, L., Tan, C., Wang, X., Zhang, M., Ren, K., Jiang, B., Qiu, X.: InferAligner: Inference-time alignment for harmlessness through cross-model guidance. In: EMNLP. pp. 10460–10479 (2024)
27. Wang, Y., Liu, X., Li, Y., Chen, M., Xiao, C.: AdaShield: Safeguarding multimodal large language models from structure-based attack via adaptive shield prompting. In: ECCV. pp. 77–94 (2024)
28. Wang, Y., Zhou, X., Wang, Y., Zhang, G., He, T.: Jailbreak large vision-language models through multi-modal linkage. In: ACL. pp. 1466–1494 (2025)
29. Xie, Y., Fang, M., Pi, R., Gong, N.: GradSafe: Detecting jailbreak prompts for LLMs via safety-critical gradient analysis. In: ACL (2024)
30. Xu, Y., Qi, X., Qin, Z., Wang, W.: Cross-modality information check for detecting jailbreaking in multimodal large language models. In: Findings of EMNLP. pp. 13715–13726 (2024)
31. Yin, Z., Cao, Y., Liu, H., Wang, T., Chen, J., Ma, F.: Towards robust multimodal large language models against jailbreak attacks. In: CVPR. pp. 22847–22856 (2026)
32. Ying, Z., Liu, A., Liang, S., Huang, L., Guo, J., Zhou, W., Liu, X., Tao, D.: Safebench: A safety evaluation framework for multimodal large language models. IJCV **134**(1), 18 (2026)
33. Yu, S., Li, L., Si, C., Sheng, L., Shao, J.: ProGuard: Towards proactive multimodal safeguard. arXiv preprint arXiv:2512.23573 (2025)
34. Yu, W., Yang, Z., Li, L., Wang, J., Lin, K., Liu, Z., Wang, X., Wang, L.: MM-Vet: Evaluating large multimodal models for integrated capabilities. In: ICML. vol. 41, pp. 57730–57754 (2024)

35. Zhang, J., Chen, R., Zhou, Q., Deng, X., Jiang, W.: Understanding and mitigating overrefusal for large language models via safety representation. In: EMNLP. pp. 21057–21075 (2025)
36. Zhang, X., Zhang, C., Li, T., Huang, Y., Jia, X., Hu, M., Zhang, J., Liu, Y., Ma, S., Shen, C.: JailGuard: A universal detection framework for LLM prompt-based attacks. arXiv preprint arXiv:2312.10766 (2023)
37. Zhang, Y., Chen, L., Zheng, G., Gao, Y., Zheng, R., Fu, J., Yin, Z., Jin, S., Qiao, Y., Huang, X., et al.: SPA-VL: A comprehensive safety preference alignment dataset for vision language models. In: CVPR. pp. 19867–19878 (2025)
38. Zhang, Y., Xie, R., Chen, J., Sun, X., Wang, Y.: PIP: Detecting adversarial examples in large vision-language models via attention patterns of irrelevant probe questions. In: ACM MM. pp. 11175–11183 (2024)
39. Zhao, W., Li, Z., Li, Y., Sun, J.: Q-MLLM: Vector quantization for robust multimodal large language model security. In: NDSS (2026)
40. Zong, Y., Bohdal, O., Yu, T., Yang, Y., Hospedales, T.: Safety fine-tuning at (Almost) no cost: A baseline for vision large language models. In: ICML. vol. 41, pp. 62867–62891 (2024)