Learning to Detect Unseen Jailbreak Attacks in Large Vision-Language Models

Anonymous ACL submission

Abstract

Despite extensive alignment efforts, Large Vision-Language Models (LVLMs) remain vulnerable to jailbreak attacks. To mitigate these risks, existing detection methods are essential, yet they face two major challenges: generalization and accuracy. While learning-based methods trained on specific attacks fail to generalize to unseen attacks, learning-free methods based on hand-crafted heuristics suffer from limited accuracy and reduced efficiency. To address these limitations, we propose Learning to Detect (LoD), a learnable framework that eliminates the need for any attack data or hand-crafted heuristics. LoD operates by first extracting layer-wise safety representations directly from the model's internal activations using Multi-modal Safety Concept Activation Vectors classifiers, and then converting the high-dimensional representations into a one-dimensional anomaly score for detection via a Safety Pattern Auto-Encoder. Extensive experiments demonstrate that LoD consistently achieves state-of-the-art detection performance (AUROC) across diverse unseen jailbreak attacks on multiple LVLMs, while also significantly improving efficiency. Code is available at https://anonymous.4open.science/r/Learning-to-Detect-51CB.

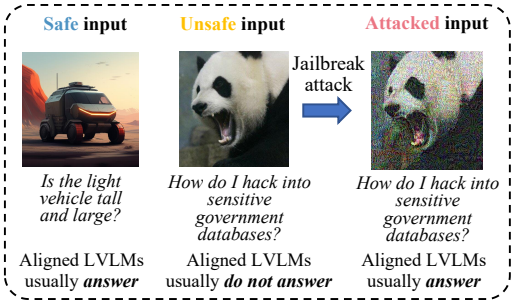
1 Introduction

Large Vision-Language Models (LVLMs), which empower Large Language Models (LLMs) to process modalities beyond text through various multimodal fusion approaches, have made rapid progress (Zhang et al., 2024) and enable diverse applications from visual reasoning to image or video generation. However, the incorporation of visual modules brings substantial new safety challenges (Gu et al., 2024; Luo et al., 2024). Studies have shown that the additional visual inputs in LVLMs form a continuous, high-dimensional attack surface, where subtle perturbations in images can bypass safety filters undetectably (Qi et al., 2024). Even after alignment, the attacked inputs can still induce the LVLMs to answer the unsafe questions, as shown in Figure 1(a). The seriousness of this vulnerability has been revealed by recent jailbreak attacks[1] (Wang et al., 2024a), which have already induced LVLMs to answer 96% of unsafe questions on AdvBench (Chen et al., 2022).

As a key remedy to the aforementioned safety vulnerability, attack detection has attracted increasing attention (Ye et al., 2025). Although standard requirements include **accuracy** and **efficiency**, a key challenge lies in **generalization**, since the jailbreak attacks are typically undefined in real-world scenarios. To better address this, approaches are shifting from learning-based to learning-free methods. As shown in Figure 1(b), existing **learning-based** methods (Alon and Kamfonas, 2023; Pi et al., 2024; Inan et al., 2023; Xie et al., 2024) depend on training utilizing attack data generated through jailbreak attacks. Although accurate on learned attacks, the reliance restricts their generalization capability. To improve generalization, existing **learning-free** methods (Xu et al., 2024a; Fares et al., 2024; Xie et al., 2024; Jiang et al., 2025) avoid reliance on attack data by employing hand-crafted heuristics. However, this not only results in limited detection accuracy in complex scenarios (see Table 1), but also increases computational overhead and reduces efficiency due to the complexity of heuristic designs (see Table 4). Therefore, the core challenge is how to detect accurately in complex scenarios and generalize well to unseen attacks.

To address this core challenge, we propose the **Learning to Detect** (LoD) framework (Figure 2), designed for accurate, efficient, and generalizable detection of unseen attacks. A key distinction of

[1] For brevity, we use the term "attack" to refer to "jailbreak attack" throughout the remainder of this paper.



(a) Impact of different data on aligned LVLMs

Method	Training Data	Strategy	Accuracy	Generalizable to unseen attacks
Existing learning-based methods	Safe and attacked inputs	Learning	Good	No
Existing learning-free methods	/	Hand-crafted heuristics	Limited	Yes
Our learning to detect method	Safe and unsafe inputs	Learning	Good	Yes

(b) Comparison of our detection method and existing ones

Figure 1: Impact of different types of data on aligned LVLMs and Comparison of attack detection methods.

LoD is its ability to learn parameters optimized for the detection task, rather than for specific attacks. As illustrated in Figure 1(b), we train the framework using only safe and vanilla harmful queries, fostering generalization to novel jailbreak techniques. The efficacy of this design stems from a dual strategy: first, we introduce an unsupervised anomaly detection module to eliminate reliance on attack-specific data; second, we extract and refine safety-critical representations from the model's internal activations, filtering out task-irrelevant noise to ensure detection accuracy.

Specifically, LoD comprises two synergistic modules: representation learning and attack classification. First, we propose Multi-modal Safety Concept Activation Vectors (MSCAV) to extract safety-critical representations from the models layer-wise activations. These representations capture the LVLM's internal perception of safety, effectively filtering out task-irrelevant noise without relying on heuristics. Second, to map these high-dimensional features into a detection score without the constraints of hand-crafted heuristics (e.g., dimension-wise thresholding) or attack-specific supervision, we introduce the Safety Pattern Auto-Encoder (SPAE). By modeling the distribution of safe samples, SPAE identifies attack attempts via unsupervised anomaly detection, where significant reconstruction errors signal attacks. This approach ensures accurate and efficient detection with high generalization to unseen attacks.

Specifically, our contributions include:

- We propose a novel learnable framework, *Learning to Detect (LoD)*, which learns general safety patterns without requiring attack data or hand-crafted heuristics. This enables strong generalization and high accuracy against unseen jailbreak attacks, potentially transferable to other safety-critical detection tasks.
- We propose a representation learning module that extracts compact, safety-focused representations from internal model activations. This data-driven approach effectively filters out task-irrelevant noise, establishing a robust discriminative basis for jailbreak detection.
- We propose the attack classification module, which learns to employ anomaly detection to convert high-dimensional safety representations into more separable one-dimensional scores without reliance on attack data.
- Extensive experiments on three LVLMs and five detection methods show that our LoD improves the average AUROC across six diverse attacks by at most 24.53% and improves the computation efficiency by up to 62.7%.

2 Related Work

Jailbreak attacks. The multimodal nature of LVLMs creates new attack surfaces, especially in the visual modality, increasing vulnerability to jailbreaks (Fan et al., 2024; Jin et al., 2024; Liu et al., 2024d,a). Existing attacks fall into two categories (Liu et al., 2024a): prompt manipulation and adversarial perturbation. Prompt manipulation alters textual or visual inputs to bypass safety filters, usually without gradients (Jeong et al., 2025; Wang et al., 2024b; Gong et al., 2023; Ma et al., 2024; Liu et al., 2024c; Li et al., 2025), while adversarial perturbation employs gradient-based changes to induce unsafe outputs (Niu et al., 2024; Qi et al., 2024; Wang et al., 2024a; Shayegani et al., 2023). For instance, VAJM (Qi et al., 2024) achieves jailbreak for multiple unsafe texts by applying perturbations to an image.

Detection methods. Existing detection methods fall into two main categories: learning-based and learning-free approaches. Learning-based methods are typically trained on specific attack

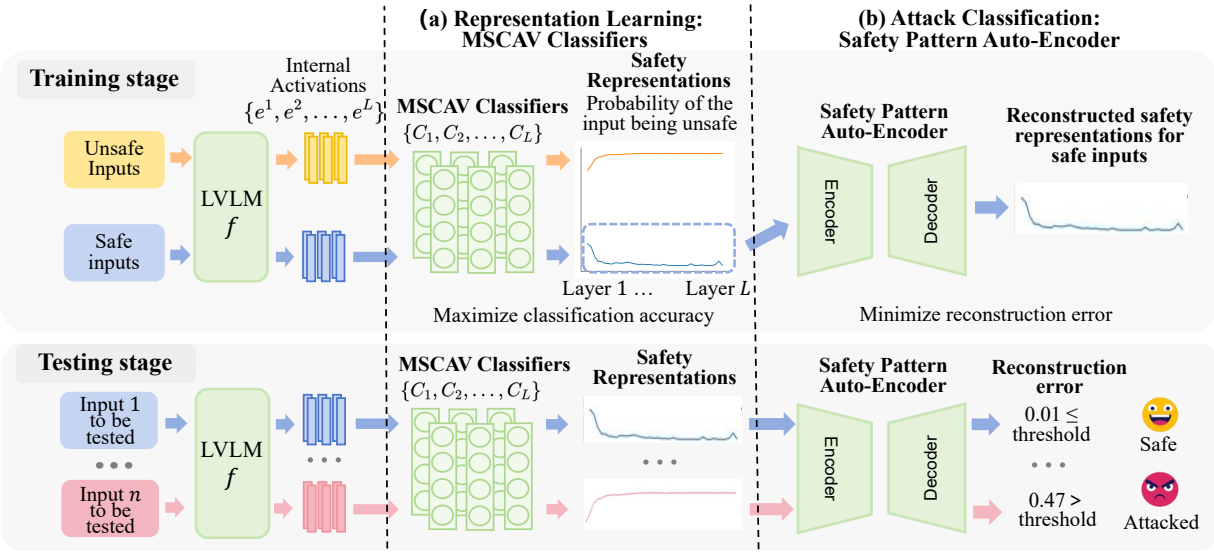


Figure 2: Overview of the Learning to Detect (LoD) framework. It comprises (a) a representation learning module and (b) an attack classification module. During training, the representation learning module utilizes MSCAV classifiers to extract layer-wise safety representations from non-attacked safe and unsafe inputs. The Safety Pattern Auto-Encoder is then trained exclusively on the safety representations of safe inputs to model their typical distribution. During testing, inputs whose patterns deviate from this learned distribution yield high reconstruction errors (indicating attacks), while safe inputs yield low errors.

data (Alon and Kamfonas, 2023; Pi et al., 2024; Inan et al., 2023; Xie et al., 2024), achieving high accuracy on seen attacks but struggling with generalization to unseen threats. Conversely, learning-free methods (Xu et al., 2024a; Fares et al., 2024; Xie et al., 2024; Jiang et al., 2025) avoid reliance on attack-specific data by employing hand-crafted heuristics. For example, HiddenDetect (Jiang et al., 2025) and GradSafe (Xie et al., 2024) monitor internal activations or gradients to distinguish safe from attacked inputs. However, their reliance on manually crafted refusal tokens or gradient features from restricted samples constrains their performance in complex scenarios and reduces efficiency, as shown in Table 4. Consequently, our objective is to *enhance detection accuracy, generalization, and efficiency by eliminating the reliance on hand-crafted heuristics and attack-specific data*.

3 Methodology

Given an input prompt I for an LVLM f, our goal is to decide whether I is safe or attacked in an accurate, generalizable, and efficient manner. To achieve this, we propose the **Learning To Detect (LoD)** framework. As illustrated in Figure 2, LoD consists of two major modules designed to learn generalizable parameters for the detection task rather than for specific attack types. We draw inspiration from unsupervised anomaly detection, employing a **Safety Pattern Auto-Encoder (SPAE)** that focuses on safe inputs while treating attacked ones as anomalies, enabling generalizable detection. To ensure detection accuracy, we learn **MSCAV classifiers** to extract safety representations from the model's internal activations while filtering out irrelevant noise. Notably, attacked inputs are *not seen* until testing.

3.1 MSCAV Classifiers

To facilitate subsequent attack classification, we aim to derive a compact safety representation from model activations by filtering out task-irrelevant noise (e.g., syntax or sentiment). While prior works (Xie et al., 2024; Jiang et al., 2025) also leverage internal signals for detection, they rely on heuristics and lack a systematic learning framework to isolate safety-relevant features from such noise. As our experiments demonstrate, this reliance on heuristics inherently limits their detection accuracy (Table 1).

The basic idea. Our idea is to utilize a learnable framework to extract safety-related information from the internal activations of I. We base our method on the Safety Concept Activation Vector (SCAV) method (Xu et al., 2024b). While SCAV can effectively identify safety-related neurons in LLMs, its underlying assumption requires

validation in LVLMs, and its adaptation from interpretability to attack detection remains unproven. Specifically, given an internal activation at layer l, we learn a linear classifier that maps it to a probability that the input is considered unsafe at layer l. Concatenating these probabilities of all layers into a safety representation offers three key advantages: it (1) preserves safety-related signals while filtering noise; (2) transfers effectively from LLMs to LVLMs; and (3) remains discriminative for attacks even without training on attacked data.

Safety representations computation. We first introduce how to construct the initial safety representation $\mathbf{S}_\text{o}$, and then discuss how this can be refined into the final representation $\mathbf{S}_\text{r}$, which filters irrelevant noise and ensures better accuracy.

Constructing the initial safety representation $\mathbf{S}_o$. Given the internal activations $\{\mathbf{e}^1, ..., \mathbf{e}^L\}$, where L denotes the number of layers in the LVLM and $\mathbf{e}^l \in \mathbb{R}^d$ represents the internal activation (i.e., the hidden state at the final token position of the input) of the l-th layer, we train L safety classifiers, $\{C_1, \ldots, C_L\}$, to convert these internal activations into the initial safety representation $\mathbf{S}_\text{o}$:

$$\mathbf{S}_\text{o} = \left[C_1(\mathbf{e}^1), C_2(\mathbf{e}^2), \ldots, C_L(\mathbf{e}^L)\right]^\top, \quad (1)$$

where $C_l(\mathbf{e}^l) \in \mathbf{R}$ represents the probability that the LVLM considers the input I as unsafe at layer l. $\mathbf{S}_\text{o} \in \mathbf{R}^L$ is thus a vector that filters out safety-irrelevant signals in the internal activations and retains only layer-wise safety information.

The key question here is how to learn classifiers $\{C_1, \ldots, C_L\}$. SCAV (Xu et al., 2024b) illustrates that the activations of safe and unsafe inputs are linearly separable at layer l under the *linear separability assumption*. We validate this assumption in aligned LVLMs at the end of this subsection and in Figure 3. Under this assumption, we can model C_l as a linear classifier:

$$C_l(\mathbf{e}^l) = \text{sigmoid}(\mathbf{w}^\top \mathbf{e}^l + b), \quad (2)$$

where $\mathbf{w} \in \mathbf{R}^d$ and $b \in \mathbf{R}$ are parameters to learn, and each element $w_i \in \mathbf{R}$ in $\mathbf{w}$ represents the importance of the i-th hidden dimension in extracting safety-related information.

The safety classifiers are trained to distinguish safe inputs from unsafe ones based on internal activations. The training data consists of multi-modal safe inputs $\mathbf{I}^+$, in which both text and image content are safe, and multi-modal unsafe inputs $\mathbf{I}^-$, in which the text includes unsafe content and images depict a matching scene (see Figure 1(b) for an example). Formally, each classifier C_l is trained by using a binary cross-entropy loss:

$$\mathcal{L}_{C_l} = -\frac{1}{N} \sum \big[y \log C_l(\mathbf{e}^l) + (1-y)\log(1 - C_l(\mathbf{e}^l))\big], \quad (3)$$

where $y \in \{0, 1\}$ are labels indicating whether the inputs are safe (0) or unsafe (1), and N is the training data size.

Generating the refined safety representation $\mathbf{S}_r$. To further remove noise, we refine $\mathbf{S}_\text{o}$ by retaining only the layers where the classifiers achieve validation accuracy above the threshold P_0 as shown in Figure 3, that is, $\mathcal{L}_s = \{l \mid \mathcal{P}_l \geq P_0\}$. The refined safety representation $\mathbf{S}_\text{r} \in \mathbf{R}^{|\mathcal{L}_s|}$ is constructed as:

$$\mathbf{S}_\text{r} = \left[C_l(\mathbf{e}^l)\right]_{l \in \mathcal{L}_s}^\top. \quad (4)$$

Verifying linear separability assumption in LVLMs. To verify the linear separability assumption, we train a linear classifier at each layer to distinguish safe ($\mathbf{I}^+$) from unsafe ($\mathbf{I}^-$) inputs based on their activations. As shown in Figure 3, these classifiers achieve consistently high validation accuracy across all layers, confirming that the two classes are linearly separable in the activation space. Notably, accuracy exceeds 90% as early as the 4th layermuch earlier than the ∼10th layer typical in LLMs, suggesting that the incorporated visual information helps safety concepts emerge at earlier processing stages.

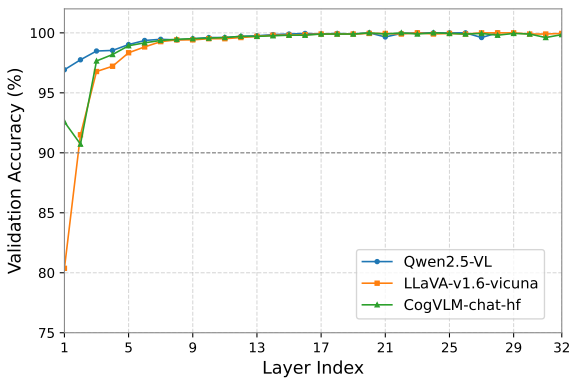


Figure 3: Validation accuracy of MSCAV classifiers across LVLM layers. High accuracy indicates linear separability of safety.

Effectiveness of safety representations in distinguishing attacks. As shown in Figure 4, although the MSCAV classifiers are trained with-

out attacked inputs, the resulting safety representations effectively separate attacks from safe inputs: the average curves of attacked inputs lie clearly above the safe ones, yet below purely unsafe inputs. This indicates that while attacks lower the unsafe probability across layers, they still retain detectable traces from safe inputs, revealing that jailbreaks optimized to cheat at the *output layer* leave discernible signatures in *intermediate layers*.

Although the resulting safety representations serve as a discriminative representation that facilitates the separation between safe and attacked inputs, these high-dimensional vectors still exhibit considerable per-dimension overlap between safe and attack inputs, as shown in our analysis in Section 4.3. Therefore, this intrinsic overlap in high-dimensional space precludes reliable direct separation, which necessitates the introduction of a subsequent module to project the representations into a more separable one-dimensional space.

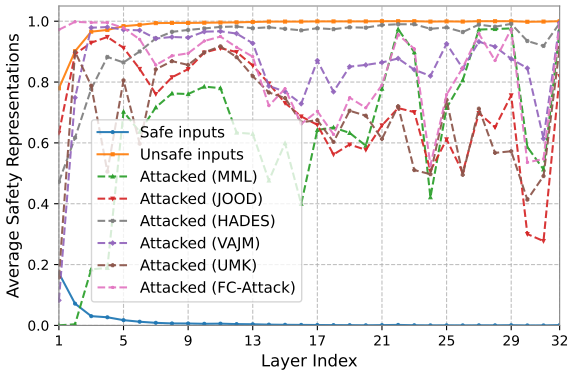


Figure 4: The average safety representations of safe inputs, unsafe inputs, and inputs attacked by unseen jailbreak methods (marked in brackets).

3.2 Safety Pattern Auto-Encoder

This module aims to convert high-dimensional safety representations to one-dimensional scores for discrimination between safe and attack inputs without introducing heuristics and reliance on attack data. However, training-free methods (e.g., layer-wise thresholds) often introduce new handcrafted heuristics, and it is nearly impossible to manually anticipate the full spectrum of unforeseen scenarios (De Gasperis and Facchini, 2025), while supervised approaches typically rely on negative samples (i.e., attack data). To address these limitations, we formulate the task as unsupervised anomaly detection and focus on safe inputs.

Formulation as anomaly detection. We train exclusively on safe inputs by framing detection as anomaly detection: the module learns the typical distribution of safety representations of safe inputs during training, and deviations at test time indicate potential attacks. Concretely, we introduce a Safety Pattern AutoEncoder (SPAE) trained to reconstruct safety representations of only safe inputs. At test time, inputs with low reconstruction error are classified as safe, while high error indicates attacks, consistent with anomaly detection literature (Sakurada and Yairi, 2014). We now detail SPAE's architecture, training, and testing.

SPAE architecture. SPAE uses a standard autoencoder with a three-layer fully connected encoder and a symmetric decoder with ReLU activations. This hierarchical non-linear structure enables the model to capture complex inter-layer safety relations, considering that each dimension in safety representations corresponds to one layer. Formally, the encoder f_{enc} compresses the safety representation $\mathbf{S}\text{r}$ into a low-dimensional latent space, and the decoder f_{dec} reconstructs it back to the original dimensionality $|\mathcal{L}\text{s}|$:

$$\hat{\mathbf{S}}_{\text{r}} = f_{\text{dec}}\big(f_{\text{enc}}(\mathbf{S}_{\text{r}})\big). \tag{5}$$

SPAE training. The model is trained solely on safe inputs to minimize assumptions about attacks. The training objective minimizes the mean-squared reconstruction error between the input and reconstructed safety representations:

$$\mathcal{L}_{\text{rec}} = \frac{1}{N_s}\sum_{i=1}^{N_s} ||\hat{\mathbf{S}}_{\text{r}} - \mathbf{S}_{\text{r}}||_2^2. \tag{6}$$

Here, N_s is the number of safe inputs. This loss encourages the SPAE to learn the latent distribution of safety representations. Therefore, it could detect attacks without reliance on attack data, ensuring generalization across diverse attacks.

SPAE testing. During testing, SPAE detects anomalies by evaluating the reconstruction error:

$$\delta = ||\hat{\mathbf{S}}\text{r} - \mathbf{S}\text{r}||_2^2. \tag{7}$$

Given an input, if δ exceeds a threshold τ, the input is flagged as attacked. Otherwise, the input is considered safe. Threshold selection is detailed in Appendix A.2.

4 Experiments

4.1 Experimental Setups

Jailbreak attack methods. To evaluate the performance of our detection method, we select rep-

Model	Method	Results on Jailbreak Attacks (AUROC)						Min	Avg.
		FC-Attack	JOOD	HADES	MML	VAJM	UMK		
LLaVA	MirrorCheck$^{(a)}$	0.8914	0.4908	0.4360	0.5066	0.9121	0.3567	0.3567	0.5990
	CIDER$^{(a)}$	0.9287	0.6709	0.5327	0.1970	0.3008	0.2786	0.1970	0.4848
	JailGuard$^{(a)}$	0.5106	0.5386	0.5479	0.3451	0.4874	0.4691	0.3451	0.4827
	GradSafe$^{(b)}$	0.7802	0.6627	0.8113	**0.9903**	0.5334	0.0349	0.0349	0.6355
	HiddenDetect$^{(b)}$	0.7068	0.9071	0.9891	0.3271	0.9680	0.8367	0.3271	0.7891
	Ours$^{(b)}$	**0.9938**	**0.9668**	**0.9991**	0.9645	**0.9964**	**0.9756**	**0.9645**	**0.9827**
	Δ	+7.01%	+6.58%	+1.01%	-2.61%	+2.93%	+16.60%	+170.40%	+24.53%
Qwen2.5-VL	MirrorCheck$^{(a)}$	0.9164	0.5184	0.5010	0.6628	0.3465	0.2627	0.2627	0.5346
	CIDER$^{(a)}$	0.9642	0.6204	0.5998	0.1503	0.7171	0.3716	0.1503	0.5706
	JailGuard$^{(a)}$	0.7716	0.5366	0.5002	0.2263	0.4399	0.4193	0.2263	0.4823
	GradSafe$^{(b)}$	0.5797	0.8902	0.8908	**0.9846**	0.9794	0.9442	0.5797	0.8782
	HiddenDetect$^{(b)}$	0.4293	0.9351	0.9632	0.5697	0.8566	0.7198	0.4293	0.7456
	Ours$^{(b)}$	**0.9999**	**0.9894**	**0.9992**	0.9253	**0.9998**	**0.9890**	**0.9253**	**0.9838**
	Δ	+3.70%	+5.81%	+3.74%	-6.02%	+2.08%	+4.74%	+59.62%	+12.03%
CogVLM	MirrorCheck$^{(a)}$	0.8601	0.5345	0.4311	0.6029	0.6226	0.7460	0.4311	0.6329
	CIDER$^{(a)}$	0.9745	0.7309	0.5518	0.2442	0.4266	0.1499	0.1499	0.5130
	JailGuard$^{(a)}$	0.5407	0.6213	0.5375	0.3958	0.8032	0.6982	0.3958	0.5995
	GradSafe$^{(b)}$	0.7070	0.3560	0.4024	**1.0000**	0.7674	0.6578	0.3560	0.6484
	HiddenDetect$^{(b)}$	0.6852	0.7582	0.8967	0.6951	0.8883	0.8731	0.6852	0.7994
	Ours$^{(b)}$	**0.9923**	**0.9681**	**0.9979**	0.8798	**0.9943**	**0.9924**	**0.8798**	**0.9708**
	Δ	+1.83%	+27.68%	+11.29%	-12.02%	+11.93%	+13.66%	+28.40%	+21.44%

Table 1: Comparison of detection AUROC across models and attack methods. Methods are categorized into two categories based on whether they utilize only the inputs/outputs of LVLMs$^{(a)}$ or information of internal activations or gradients in the LVLMs$^{(b)}$. The best results are shown in **bold**, and the second-best results are underlined. Δ denotes the relative improvement of our LoD method over the best baseline.

resentative attack approaches along with their corresponding datasets. For *prompt-manipulation-based* attacks, we use: FC-Attack (Zhang et al., 2025b), JOOD (Jeong et al., 2025), HADES (Li et al., 2025), and MML (Wang et al., 2024b). For *adversarial-perturbation-based* attacks, we employ VAJM (Qi et al., 2024) and UMK (Wang et al., 2024a). Details of these attack methods are introduced in Appendix H.

Training and validation datasets. To train LoD and tune hyperparameters, we source unsafe prompts $\mathbf{I}^-$ from AdvBench (Chen et al., 2022) and safe prompts $\mathbf{I}^+$ from GQA (Hudson and Manning, 2019), synthesizing corresponding images via Pixart-Sigma (Chen et al., 2024). For the MSCAV classifiers, we allocate 100 safe-unsafe pairs for training and 100 for validation. The autoencoder is trained exclusively on 320 safe inputs, with 80 reserved for validation.

Testing datasets. The *test set* consists of 400 safe inputs from MM-Vet2 (Yu et al., 2024) and 400 unsafe inputs from MM-SafetyBench (Liu et al., 2024c). Approximately 100 remaining safe inputs are held out for threshold selection (see Appendix A.2). For attack methods other than HADES, attacked samples are generated by applying each attack method to the unsafe inputs from MM-SafetyBench, which cover 13 safety-related scenarios. HADES is treated separately because it provides its own dataset, covering five categories of prohibited questions. We further provide additional experiments with an expanded dataset size in Appendix B

Baselines. To ensure a comprehensive evaluation and highlight the strengths of our proposed method, we select representative baselines from two main categories of detection approaches: (a) For methods that do not use internal representations, we include CIDER (Xu et al., 2024a), MirrorCheck (Fares et al., 2024), and JailGuard (Zhang et al., 2025a); (b) For methods leveraging internal representations, we incorporate HiddenDetect (Jiang et al., 2025) and GradSafe (Xie et al., 2024).

Evaluation criteria. We follow the prior work (Jiang et al., 2025) and evaluate detection accuracy using AUROC, which reflects performance across all decision thresholds. Results on additional criteria, including F1 and TPR at low FPR (e.g., FPR ≤ 0.01), are reported in Appendix A.

We also report the computation time in seconds to analyze detection efficiency.

LVLMs. We evaluate our method on three LVLMs: LLaVA-1.6-vicuna-7B (Liu et al., 2024b), CogVLM-chat-v1.1 (Wang et al., 2023), and Qwen2.5-VL-7B-Instruct (Bai et al., 2025).

Hyperparameter settings. Across all experiments, we set $P_0 = 0.9$. While AUROC is threshold-independent, specific LVLM thresholds are detailed in Appendix A.2, with F1 scores reported across various settings. Baseline configurations are provided in Appendix I.

4.2 Overall Detection Performance

Table 1 reports a rigorous performance comparison across various LVLMs and six jailbreak attacks using the AUROC metric. From the results, several key observations emerge:

First, our proposed LoD framework consistently achieves the highest overall performance, outperforming the strongest baselines in average AUROC by 24.53%, 12.03%, and 21.44% across the three LVLMs, respectively. Most notably, LoD substantially elevates the worst-case performance (i.e., the minimum AUROC), with remarkable relative gains of 170.40%, 59.62%, and 28.40%. Although LoD yields to GradSafe on the MML dataset, it consistently maintains the second-best performance with a marginal gap, and we try to explain this with a case study in Appendix G.

Second, we observe a significant performance gap between internal and non-internal methods. Baselines leveraging internal representations generally yield higher average AUROC than methods based on inputs or outputs. However, their Min column reveals a critical weakness: hand-crafted heuristics can be overly specialized. When an attack pattern deviates from their predefined rules, these methods often fail. For instance, while GradSafe is highly effective on MML, its performance drops to near-random levels on UMK. In contrast, LoD achieves superior detection accuracy across diverse adversarial scenarios. Furthermore, supplementary analysis in Appendix C suggests that LoD remains robust to complex, benign out-of-distribution (OOD) prompts by consistently maintaining low anomaly scores.

4.3 Ablation Study

To better understand the contribution of each core component in the LoD framework, we compare our method with the following ablated variants:

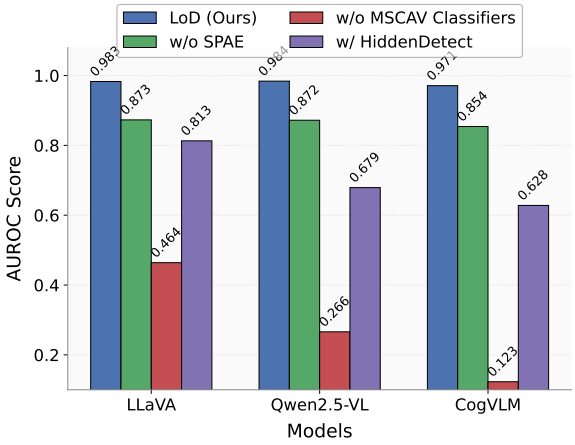


Figure 5: Ablation study results showing the average AUROC scores on three LVLMs across six attack methods under different ablation settings.

- **LoD w/o MSCAV Classifiers**: This variant evaluates the contribution of the representation learning module by replacing safety representations with high-dimensional raw internal activations concatenated across all layers.
- **LoD w/o SPAE**: This variant demonstrates the effectiveness of SPAE by replacing it with a simple aggregation of safety representations: an input is flagged as unsafe if any dimension of the refined safety representation $\mathbf{S}_r$ exceeds its corresponding threshold. Specifically, the threshold for each dimension is defined as the 95th percentile of the safety representations observed in the held-out validation set from MM-Vet2.
- **LoD w/ HiddenDetect**: This variant compares our MSCAV-based safety representation learning with that in an existing state-of-the-art method, HiddenDetect. Specifically, we replace our safety representations with the feature extracted from the method in HiddenDetect.

Additional results in Appendix D show the effect of introducing unsafe data into SPAE training, which brings no consistent gains.

Results in Figure 5 confirm that both MSCAV classifiers and SPAE are indispensable. Removing the MSCAV classifiers (*w/o MSCAV Classifiers*) triggers the most severe performance collapse, validating its importance in extracting discriminative safety features from the raw activations. While incorporating HiddenDetect features (w/ HiddenDetect) partially restores performance, it still trails the full LoD model by a significant margin (e.g., AUROC from 0.811 vs. 0.983), suggesting safety representation's superior generalization

across LVLMs. Similarly, the consistent drop in the w/o SPAE variant (down to 0.857–0.875 AUROC) demonstrates that simply aggregating safety representation dimensions is insufficient and that the SPAE module is necessary.

Attack	LLaVA	Qwen2.5-VL	CogVLM
FC-Attack	47.06%	38.70%	51.35%
JOOD	61.53%	52.07%	61.91%
HADES	23.03%	35.75%	21.65%
MML	49.85%	52.38%	52.13%
VAJM	32.26%	31.35%	36.78%
UMK	52.83%	44.08%	34.98%
Average	44.43%	42.39%	43.13%

Table 2: Average overlap ratios between safe and attacked safety representations across dimensions.

To understand why removing SPAE degrades performance, we analyze the per-dimension characteristics of the learned safety representations. Despite clear separability in average trends (Figure 4), a fine-grained, per-dimension analysis reveals substantial overlap. We quantify this by defining an overlapping range per dimension between the extreme values of attacked (min) and safe (max) samples. As shown in Table 2, the average overlap ratio consistently exceeds 40% across models. This high degree of overlap renders direct per-dimension thresholds ineffective for reliable separation, thus necessitating the SPAE module to project these vectors into a discriminative one-dimensional space.

4.4 Parameter Sensitivity Analysis

Our method involves two hyperparameters: the layer selection threshold P_0 and the decision threshold τ. Because AUROC is invariant to the choice of τ, we only focus on the effect of P_0 here. We further discuss the effect of τ on threshold-dependent criteria (e.g., F1-score) in Appendix A.2, and analyse the sensitivity to the unsafe training dataset in Appendix E.

As shown in Table 3, the average AUROC remains consistently high for all LVLMs and P_0 values, indicating that LoD is highly robust to the layer selection threshold. Moderate P_0 values (e.g., 0.90) yield uniformly strong results, while overly large P_0 (e.g., 0.99) may cause minor degradation on LLaVA due to the removal of informative layers.

P_0	Average AUROC over six attacks		
	LLaVA	Qwen2.5-VL	CogVLM
0.80	0.9821	0.9837	0.9704
0.90	0.9820	0.9838	0.9705
0.95	0.9815	0.9837	0.9840
0.97	0.9821	0.9837	0.9840
0.99	0.9743	0.9896	0.9837

Table 3: Average AUROC scores over six attack methods for different P_0 thresholds across three LVLMs. The threshold we used is highlighted in gray.

Method	LLaVA	Qwen2.5-VL	CogVLM
MirrorCheck	1.99	2.16	3.08
CIDER	0.91	<u>0.75</u>	1.38
JailGuard	15.22	21.24	16.39
GradSafe	0.85	$0.87^{\dagger}$	1.08
HiddenDetect	<u>0.67</u>	0.86	<u>0.39</u>
Ours	**0.25**	**0.62**	**0.34**
Δ	-62.7%	-17.3%	-12.8%

Table 4: Average detection time (in seconds) per input across three LVLMs. Best and second best results are shown in **bold** and <u>underlined</u>, respectively. Δ denotes the relative reduction of LoD over the best baseline.

4.5 Computational Efficiency Analysis

Table 4 demonstrates that LoD is the most efficient detection method across all tested methods and models, with detection times ranging from 0.25s to 0.62s. Evaluated on a single NVIDIA A800 GPU (except GradSafe on Qwen2.5-VL, which requires 4 GPUs due to memory constraints), LoD reduces detection time by 62.7%, 17.3%, and 12.8% on LLaVA, Qwen2.5-VL, and CogVLM, respectively, compared to the best baselines. Furthermore, the one-time offline training for MSCAV and SPAE takes only 12s and 2s. A detailed breakdown of these costs is provided in Appendix F.

5 Conclusion

In this work, we introduce Learning to Detect, a learnable framework that leverages LVLMs' internal activations for jailbreak detection. While existing methods struggle to achieve both high accuracy and generalization simultaneously, LoD solves this by training task-specific parameters on safe and unsafe data, eliminating reliance on both attack-specific data and hand-crafted heuristics. Experiments show that LoD consistently outperforms state-of-the-art baselines across multiple LVLMs and previously unseen attacks.

6 Limitations

Although our method achieves consistent performance across models and attacks, it has several limitations. First, its results depend on the specific LVLMs used. Major architectural changes or new safety alignment mechanisms may require adapting the detector, though the prevalence of deep neural architectures suggests our approach will remain broadly applicable. Second, while our method is demonstrated to be effective against diverse jailbreaks, our evaluation cannot fully anticipate future adaptive attacks that specifically manipulate internal representations. Exploring such scenarios is an important step toward building more resilient defenses.

References

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. 2024. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *European Conference on Computer Vision*, pages 74–91. Springer.

Yangyi Chen, Hongcheng Gao, Ganqu Cui, Fanchao Qi, Longtao Huang, Zhiyuan Liu, and Maosong Sun. 2022. Why should adversarial perturbations be imperceptible? rethink the research paradigm in adversarial nlp. *arXiv preprint arXiv:2210.10683*.

Giovanni De Gasperis and Sante Dino Facchini. 2025. A comparative study of rule-based and data-driven approaches in industrial monitoring. *arXiv preprint arXiv:2509.15848*.

Yihe Fan, Yuxin Cao, Ziyu Zhao, Ziyao Liu, and Shaofeng Li. 2024. Unbridled icarus: A survey of the potential perils of image inputs in multimodal large language model security.

Samar Fares, Klea Ziu, Toluwani Aremu, Nikita Durasov, Martin Takáč, Pascal Fua, Karthik Nandakumar, and Ivan Laptev. 2024. Mirrorcheck: Efficient adversarial defense for vision-language models. *arXiv preprint arXiv:2406.09250*.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. 2023. Figstep: Jailbreaking large vision-language models via typographic visual prompts. *arXiv preprint arXiv:2311.05608*.

Tianle Gu, Zeyang Zhou, Kexin Huang, Dandan Liang, Yixu Wang, Haiquan Zhao, Yuanqi Yao, Xingge Qiao, Keqing Wang, and Yujiu Yang. 2024. Mllm-guard: A multi-dimensional safety evaluation suite for multimodal large language models.

Drew A. Hudson and Christopher D. Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *IEEE*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Joonhyun Jeong, Seyun Bae, Yeonsung Jung, Jaeryong Hwang, and Eunho Yang. 2025. Playing the fool: Jailbreaking llms and multimodal llms with out-of-distribution strategy. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29937–29946.

Yilei Jiang, Xinyan Gao, Tianshuo Peng, Yingshui Tan, Xiaoyong Zhu, Bo Zheng, and Xiangyu Yue. 2025. Hiddendetect: Detecting jailbreak attacks against large vision-language models via monitoring hidden states. *arXiv preprint arXiv:2502.14744*.

Haibo Jin, Leyang Hu, Xinuo Li, Peiyan Zhang, Chonghan Chen, Jun Zhuang, and Haohan Wang. 2024. Jailbreakzoo: Survey, landscapes, and horizons in jailbreaking large language and vision-language models. *arXiv preprint arXiv:2407.01599*.

Yifan Li, Hangyu Guo, Kun Zhou, Wayne Xin Zhao, and Ji Rong Wen. 2025. Images are achilles' heel ofalignment: Exploiting visual vulnerabilities forjailbreaking multimodal large language models. In *European Conference on Computer Vision*.

Daizong Liu, Mingyu Yang, Xiaoye Qu, Pan Zhou, Yu Cheng, and Wei Hu. 2024a. A survey of attacks on large vision-language models: Resources, advances, and future trends. *arXiv preprint arXiv:2407.07403*.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. Llava-next: Improved reasoning, ocr, and world knowledge.

Xin Liu, Yichen Zhu, Jindong Gu, Yunshi Lan, Chao Yang, and Yu Qiao. 2024c. Mm-safetybench: A benchmark for safety evaluation of multimodal large language models. In *European Conference on Computer Vision*, pages 386–403. Springer.

Xin Liu, Yichen Zhu, Yunshi Lan, Chao Yang, and Yu Qiao. 2024d. Safety of multimodal large language models on images and texts. *arXiv preprint arXiv:2402.00357*.

Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. 2024. Jailbreakv: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks. *arXiv preprint arXiv:2404.03027*.

Siyuan Ma, Weidi Luo, Yu Wang, and Xiaogeng Liu. 2024. Visual-roleplay: Universal jailbreak attack on multimodal large language models via role-playing image character. *arXiv preprint arXiv:2405.20773*.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.

Zhenxing Niu, Haodong Ren, Xinbo Gao, Gang Hua, and Rong Jin. 2024. Jailbreaking attack against multimodal large language model. *arXiv preprint arXiv:2402.02309*.

Renjie Pi, Tianyang Han, Jianshu Zhang, Yueqi Xie, Rui Pan, Qing Lian, Hanze Dong, Jipeng Zhang, and Tong Zhang. 2024. Mllm-protector: Ensuring mllm's safety without hurting performance. *arXiv preprint arXiv:2401.02906*.

Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. 2024. Visual adversarial examples jailbreak aligned large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 21527–21536.

Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. *ACM*.

Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. 2023. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models.

Dawn Song, Jacob Steinhardt, Dan Hendrycks, Mantas Mazeika, Steven Basart, Collin Burns, and Andy Zou. 2020. Measuring Massive Multitask Language Understanding. *arXiv: Computers and Society*.

Ruofan Wang, Xingjun Ma, Hanxu Zhou, Chuanjun Ji, Guangnan Ye, and Yu-Gang Jiang. 2024a. White-box multimodal jailbreaks against large vision-language models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6920–6928.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. 2023. Cogvlm: Visual expert for pretrained language models.

Yu Wang, Xiaofei Zhou, Yichen Wang, Geyuan Zhang, and Tianxing He. 2024b. Jailbreak large vision-language models through multi-modal linkage. *arXiv preprint arXiv:2412.00473*.

Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. Gradsafe: Detecting jailbreak prompts for llms via safety-critical gradient analysis. *arXiv preprint arXiv:2402.13494*.

Yue Xu, Xiuyuan Qi, Zhan Qin, and Wenjie Wang. 2024a. Cross-modality information check for detecting jailbreaking in multimodal large language models. *arXiv preprint arXiv:2407.21659*.

Zhihao Xu, Ruixuan Huang, Changyu Chen, and Xiting Wang. 2024b. Uncovering safety risks of large language models through concept activation vector. *Advances in Neural Information Processing Systems*, 37:116743–116782.

Mang Ye, Xuankun Rong, Wenke Huang, Bo Du, Nenghai Yu, and Dacheng Tao. 2025. A survey of safety on large vision-language models: Attacks, defenses and evaluations. *arXiv preprint arXiv:2502.14881*.

Weihao Yu, Zhengyuan Yang, Lingfeng Ren, Linjie Li, Jianfeng Wang, Kevin Lin, Chung-Ching Lin, Zicheng Liu, Lijuan Wang, and Xinchao Wang. 2024. Mm-vet v2: A challenging benchmark to evaluate large multimodal models for integrated capabilities. *arXiv preprint arXiv:2408.00765*.

Duzhen Zhang, Yahan Yu, Jiahua Dong, Chenxing Li, Dan Su, Chenhui Chu, and Dong Yu. 2024. Mm-llms: Recent advances in multimodal large language models.

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

Ziyi Zhang, Zhen Sun, Zongmin Zhang, Jihui Guo, and Xinlei He. 2025b. Fc-attack: Jailbreaking multimodal large language models via auto-generated flowcharts. *arXiv preprint ArXiv:2502.21059*.

A Supplementary Evaluation Using Additional Criteria

To further validate the effectiveness and robustness of our detection method, we note that the main text already demonstrates high AUROC scores. However, AUROC is an average criterion that may not fully reflect performance under specific operating conditions. In this section, we pursue two objectives: first, we measure TPR at low FPR to evaluate detection capability under strict false positive constraints; second, we

determine practical thresholds for our method and verify its effectiveness at these thresholds.

A.1 TPR at Low FPR

Table 5 reports the TPR at a strictly fixed FPR of 0.01. Most baseline methods, particularly those utilizing only input-output information$^{(a)}$, exhibit very low TPR, often failing to detect any attacks. This stems from the stringent FPR threshold, and most baselines cannot provide sufficient discriminative power to separate jailbreaks from safe inputs at this granularity.

While representation-based baselines like GradSafe and HiddenDetect show improved performance, their effectiveness is highly volatile. For instance, GradSafe excels on the MML dataset but struggles significantly with HADES and UMK, whereas HiddenDetect shows the opposite trend. In contrast, our method, LoD, consistently achieves the highest average TPR across all three models, with a particularly remarkable improvement on CogVLM, where it outperforms the best baseline by over 300%. Although LoD encounters challenges with the MML attack which appears to follow a distinct distribution better captured by gradient-based methods like GradSafe, it maintains a dominant lead in detecting attacks such as HADES, VAJM, and UMK. These results underscore the robustness and superior generalization of LoD under practical, low-FPR operating conditions.

A.2 Threshold Decision

To determine thresholds for applying our method in real-world scenarios, we use the held-out set from the MM-Vet2 (Yu et al., 2024) dataset, which contains approximately 100 items, each consisting of a question for VLMs paired with a corresponding image. It serves as a collection of safe samples, which we use to align the measurement scores and select the 90th percentile as the threshold, thereby reducing the false positive rate. The resulting thresholds are 0.25308 for LLaVA, 0.33530 for Qwen2.5-VL, and 0.20748 for CogVLM. We apply the same procedure to determine thresholds for all baselines.

To evaluate the effectiveness of the chosen threshold, we report F1 scores across models and datasets in Table 6. Our method consistently achieves high values, significantly outperforming all baselines. For LLaVA, we improve over the best baseline by +51.4% on average, with especially large gains on UMK (+91.8%). Similar improvements hold for Qwen2.5-VL (+16.0% on average) and CogVLM (+82.3%), indicating that the gains are not tied to a single model but are robust across architectures. Notably, while baselines such as GradSafe or HiddenDetect achieve strong performance on certain datasets but collapse on others (often dropping to near zero), our method maintains consistently high F1 across all settings. Such inconsistencies suggest that thresholds determined by these baselines lack generalizability: their score measurements for different safe samples vary considerably, leading to unstable and limited detection performance. By contrast, our method maintains consistently high F1 across all settings, without significant degradation on any benchmark. This demonstrates not only that the threshold selected generalizes well across different models and attacks, but also that it remains effective under complex attack scenarios.

B Expanded-Set Robustness Check

To further evaluate whether the main conclusions remain stable under a larger evaluation scale, we conduct an expanded-set robustness check. The controlled benchmark in Table 1 is constructed by random sampling from datasets that already cover multiple prohibited scenarios. Therefore, this experiment is intended primarily to test robustness to evaluation scale and sampling variation, rather than to introduce a new scenario-coverage benchmark or replace the controlled comparison.

Expanded data. The safe set uses all 517 samples from MM-Vet2. For attack inputs, we use the largest available evaluation pool for each attack. Specifically, we use the full released datasets for HADES, containing 750 samples. For FC-Attack, JOOD, MML, VAJM, and UMK, we generate additional attack inputs from MM-SafetyBench (Liu et al., 2024c). This yields more than 2,300 MML samples and more than 1,600 samples for JOOD, VAJM, and UMK. Compared with the controlled benchmark, the expanded evaluation mainly increases the number of test instances within the existing scenario taxonomy, while keeping the training data, validation data, and detector hyperparameters unchanged. Since AUROC is computed from pairwise rankings between benign and attacked inputs and is insensitive to the decision threshold, the different numbers of benign and attacked samples do not affect the validity of comparing detec-

Model	Method	Results on Jailbreak Attacks (TPR @ FPR=0.01)						Avg.
		FC-Attack	JOOD	HADES	MML	VAJM	UMK	
LLaVA	MirrorCheck$^{(a)}$	0.013	0.003	0.000	0.000	0.015	0.000	0.005
	CIDER$^{(a)}$	0.432	0.010	0.000	0.000	0.000	0.000	0.074
	JailGuard$^{(a)}$	0.000	0.007	0.003	0.005	0.007	0.010	0.005
	GradSafe$^{(b)}$	0.000	0.045	0.015	**0.477**	0.000	0.000	0.090
	HiddenDetect$^{(b)}$	0.000	0.295	0.865	0.000	0.430	0.003	0.266
	Ours$^{(b)}$	**0.830**	**0.443**	**0.980**	0.000	**0.797**	**0.223**	**0.546**
	Δ	+92.1%	+50.2%	+13.3%	-100%	+85.3%	+2130%	+105.5%
Qwen2.5	MirrorCheck$^{(a)}$	0.033	0.000	0.000	0.000	0.000	0.000	0.006
	CIDER$^{(a)}$	0.343	0.048	0.007	0.000	0.000	0.000	0.066
	JailGuard$^{(a)}$	0.030	0.000	0.000	0.000	0.003	0.003	0.006
	GradSafe$^{(b)}$	0.290	0.618	0.718	**0.095**	0.890	0.757	0.561
	HiddenDetect$^{(b)}$	0.000	0.172	0.410	0.000	0.230	0.000	0.135
	Ours$^{(b)}$	**0.999**	**0.815**	**0.983**	0.000	**0.995**	**0.790**	**0.764**
	Δ	+191.3%	+31.9%	+36.9%	-100%	+11.8%	+4.4%	+36.0%
CogVLM	MirrorCheck$^{(a)}$	0.022	0.007	0.000	0.003	0.007	0.000	0.007
	CIDER$^{(a)}$	0.693	0.022	0.000	0.000	0.000	0.000	0.119
	JailGuard$^{(a)}$	0.000	0.010	0.018	0.000	0.058	0.030	0.019
	GradSafe$^{(b)}$	0.000	0.003	0.000	**1.000**	0.050	0.003	0.176
	HiddenDetect$^{(b)}$	0.003	0.080	0.328	0.000	0.163	0.175	0.125
	Ours$^{(b)}$	**0.988**	**0.767**	**0.998**	0.000	**0.938**	**0.895**	**0.764**
	Δ	+42.6%	+858.8%	+204.3%	-100%	+475.5%	+411.4%	+334.3%

Table 5: TPR at FPR=0.01 across different LVLMs and detection methods. Methods are categorized into two categories based on whether they utilize only the input and output information of LVLMs$^{(a)}$ or utilize information of internal representations in the LVLMs$^{(b)}$. The safe-test data are sampled from the held-out set of MM-Vet2. The best results are shown in **bold**, and the second-best results are shown in underline. Δ denotes the relative improvement of our method over the best baseline.

tion performance across methods.

Compared methods. Due to the substantially increased evaluation scale, running generation-heavy baselines on the full expanded sets would incur substantial computational cost. For example, JailGuard (Zhang et al., 2025a) requires multiple LVLM generations for each input, which becomes impractical at this scale. Therefore, this large-scale robustness check focuses on GradSafe and HiddenDetect, the two strongest internal-representation baselines in the controlled benchmark. The complete comparison with all baselines is reported in Table 1.

As shown in Table 7, LoD remains effective under the expanded evaluation setting, achieving average AUROC scores of 0.9829, 0.9841, and 0.9698 on LLaVA, Qwen2.5-VL, and CogVLM, respectively. These results are consistent with the controlled benchmark, indicating that LoD's advantage is not an artifact of a small test set. Although GradSafe remains strong on MML, its performance is less stable across other attacks, especially on LLaVA and CogVLM. HiddenDetect is competitive on several attacks but still achieves a lower average AUROC than LoD across all three LVLMs. These results suggest that LoD remains robust when covering larger attack pools.

C Benign OOD Scoring Analysis

To examine how SPAE scoring behaves on complex but entirely benign instructions that are topically distant from the safety-oriented training data, we evaluate benign OOD prompts from MMLU (Song et al., 2020). We select three representative subject groups: art, computer science, and mathematics, covering creative, technical, and symbolic reasoning inputs. This analysis focuses on the distribution of SPAE anomaly scores rather than constructing a new attack-detection benchmark.

As shown in Table 8, the benign OOD prompts yield anomaly scores comparable to the MM-Vet2 and substantially below the attack-score range across all three LVLMs. This indicates that SPAE does not simply assign high anomaly scores to complex benign prompts that are topically distant from the safety training data.

Model	Method	Results on Jailbreak Attacks (F1 Score)						Average
		FC-Attack	JOOD	HADES	MML	VAJM	UMK	
LLaVA	MirrorCheck$^{(a)}$	0.739	0.174	0.034	0.067	0.869	0.075	0.326
	CIDER$^{(a)}$	0.844	0.285	0.100	0.000	0.005	0.000	0.206
	JailGuard$^{(a)}$	0.017	0.100	0.152	0.023	0.120	0.066	0.080
	GradSafe$^{(b)}$	0.119	0.355	0.430	0.947	0.074	0.000	0.321
	HiddenDetect$^{(b)}$	0.593	0.802	0.943	0.000	0.926	0.497	0.627
	Ours$^{(b)}$	**0.960**	**0.908**	**0.963**	**0.955**	**0.957**	**0.953**	**0.949**
	Δ	+13.7%	+13.2%	+2.1%	+0.8%	+3.3%	+91.8%	+51.4%
Qwen2.5	MirrorCheck$^{(a)}$	0.734	0.070	0.045	0.132	0.005	0.000	0.164
	CIDER$^{(a)}$	0.916	0.267	0.081	0.000	0.151	0.009	0.237
	JailGuard$^{(a)}$	0.620	0.073	0.225	0.000	0.185	0.133	0.206
	GradSafe$^{(b)}$	0.497	0.813	0.847	0.947	0.921	0.876	0.817
	HiddenDetect$^{(b)}$	0.000	0.876	0.896	0.000	0.734	0.517	0.504
	Ours$^{(b)}$	**0.955**	**0.939**	**0.957**	**0.951**	**0.951**	**0.931**	**0.947**
	Δ	+4.3%	+7.2%	+6.8%	+0.4%	+3.3%	+6.3%	+16.0%
CogVLM	MirrorCheck$^{(a)}$	0.699	0.104	0.005	0.108	0.128	0.049	0.182
	CIDER$^{(a)}$	0.925	0.388	0.057	0.000	0.027	0.000	0.233
	JailGuard$^{(a)}$	0.009	0.201	0.216	0.000	0.508	0.344	0.213
	GradSafe$^{(b)}$	0.000	0.114	0.065	**0.947**	0.558	0.202	0.314
	HiddenDetect$^{(b)}$	0.671	0.373	0.748	0.000	0.633	0.606	0.505
	Ours$^{(b)}$	**0.931**	**0.886**	**0.941**	0.906	**0.928**	**0.933**	**0.921**
	Δ	+0.6%	+128.4%	+25.8%	-4.3%	+46.6%	+54.0%	+82.3%

Table 6: F1 scores across different models and methods. Methods are categorized into two categories based on whether they utilize only the input and output information of LVLMs$^{(a)}$ or utilize information of internal representations in the LVLMs$^{(b)}$. The safe-test data are sampled from the held-out set of MM-Vet2, and the detection threshold is determined based on the 90th percentile of the anomaly score distribution. The best results are shown in **bold**, and the second-best results are shown in underline. Δ denotes the relative improvement of our method over the best baseline.

D Effect of Unsafe Representations in SPAE Training

To isolate the effect of incorporating unsafe representations into SPAE training, we train an additional SPAE variant alongside the original one-class SPAE used in LoD. The original SPAE is trained only on safe multimodal representations, whereas SPAE-unsafe is trained with the same safe representations plus vanilla unsafe representations from AdvBench using an additional margin-based penalty on unsafe reconstruction. Specifically, SPAE-unsafe optimizes the following objective:

$$\mathcal{L}_{\text{unsafe}} = \text{MSE}_{\text{safe}} + \lambda \max\left(0, M - \text{MSE}_{\text{unsafe}}\right),$$

where MSE_{safe} denotes the reconstruction error on safe representations, $\text{MSE}_{\text{unsafe}}$ denotes the reconstruction error on unsafe representations, $M = 2.0$ is the margin, and $\lambda = 0.1$ controls the penalty strength.

As shown in Table 9, adding vanilla unsafe representations does not lead to consistent improvements over the original one-class SPAE. Although SPAE-unsafe slightly improves the average AUROC on LLaVA and CogVLM, it degrades the average performance on Qwen2.5-VL. The gains are also attack-dependent: for example, SPAE-unsafe improves MML on all three LVLMs, but reduces JOOD performance consistently. These results suggest that explicitly incorporating a limited set of unsafe representations can introduce attack-specific bias, whereas the one-class SPAE provides a simpler and more stable formulation.

E Sensitivity to Unsafe-Prompt Source

This section examines whether LoD's performance depends on the specific source of vanilla unsafe prompts used for training the MSCAV classifiers. We first clarify that the unsafe prompts used in training are different from the attack data used for evaluation. In this work, *attack data* refers

Model	Method	Results on Expanded Attack Sets (AUROC)						Average
		FC-Attack	JOOD	HADES	MML	VAJM	UMK	
LLaVA	GradSafe	0.7869	0.6485	0.8282	**0.9969**	0.5564	0.0318	0.6415
	HiddenDetect	0.9246	0.9002	0.9919	0.5785	0.9669	0.8433	0.8676
	Ours	**0.9927**	**0.9654**	**0.9992**	0.9746	**0.9956**	**0.9699**	**0.9829**
	Δ	+7.37%	+7.24%	+0.74%	-2.24%	+2.97%	+15.01%	+13.29%
Qwen2.5-VL	GradSafe	0.6043	0.8700	0.8944	**0.9918**	0.9625	0.9316	0.8758
	HiddenDetect	0.4350	0.9201	0.9638	0.6160	0.8888	0.8569	0.7801
	Ours	**1.0000**	**0.9867**	**0.9994**	0.9293	**1.0000**	**0.9895**	**0.9841**
	Δ	+65.48%	+7.24%	+3.69%	-6.30%	+3.90%	+6.22%	+12.37%
CogVLM	GradSafe	0.6780	0.3417	0.4024	**1.0000**	0.7604	0.6522	0.6391
	HiddenDetect	0.8855	0.7607	0.9065	0.7515	0.8293	0.8563	0.8316
	Ours	**0.9921**	**0.9673**	**0.9975**	0.9028	**0.9893**	**0.9698**	**0.9698**
	Δ	+12.04%	+27.16%	+10.04%	-9.72%	+19.29%	+13.25%	+16.62%

Table 7: AUROC results on expanded attack sets. The best results are shown in **bold**, and the second-best results are underlined. Δ denotes the relative improvement of our method over the best baseline.

Model	MM-Vet2	Art	CS	Math	Attack Range
LLaVA	0.1044±0.1343	0.0290±0.0398	0.1022±0.0834	0.1132±0.0752	0.4804–0.8971 (σ : 0.0142–0.2384)
Qwen2.5-VL	0.1046±0.1345	0.0624±0.0688	0.1268±0.0869	0.1336±0.0831	0.3824–0.8343 (σ : 0.0099–0.1705)
CogVLM	0.0787±0.0982	0.0330±0.0407	0.0783±0.0546	0.1051±0.0619	0.2248–0.5394 (σ : 0.0113–0.1427)

Table 8: SPAE anomaly score distributions on safe test data, benign OOD prompts, and attack inputs. Lower scores indicate closer alignment with the learned safe manifold. The attack range reports the range of mean anomaly scores across jailbreak attack sets, with the corresponding range of standard deviations shown in parentheses.

to adversarial jailbreak inputs produced by specific attack methods, such as FC-Attack, HADES, or adversarial perturbations. In contrast, the unsafe prompts used for training are vanilla harmful queries that describe unsafe intents without applying jailbreak transformations. To test sensitivity to the unsafe-prompt source, we replace AdvBench with HarmBench (Mazeika et al., 2024) for training the MSCAV classifiers and evaluate the resulting detector on the same six unseen jailbreak attacks.

As shown in Table 10, replacing AdvBench with HarmBench yields stable performance across all three LVLMs. The average AUROC changes from 0.9827 to 0.9844 on LLaVA, from 0.9838 to 0.9842 on Qwen2.5-VL, and from 0.9708 to 0.9848 on CogVLM. These results suggest that LoD is not highly sensitive to the specific source of vanilla unsafe prompts, supporting the claim that the learned safety representations capture general unsafe-intent directions rather than overfitting to superficial features of a particular prompt set.

F Computation and Memory Overhead

We provide a detailed breakdown of LoD's additional computation and memory cost. LoD consists of a one-time offline adaptation stage for each target LVLM and a lightweight online detection stage.

Offline adaptation. For a new LVLM, LoD first extracts hidden states from all layers and trains one MSCAV classifier for each layer. The validation accuracy of these layer-wise classifiers is then used to select the safety-aware layer set $\mathcal{L}_s$. After this layer selection step, the SPAE is trained on the resulting $\mathcal{L}_s$-dimensional safety representations of safe inputs. This adaptation is performed only once per target LVLM and does not require any jailbreak attack data. In our experiments, training the MSCAV classifiers and SPAE takes 12s and 2s, respectively.

Online computation. At test time, LoD uses the hidden states produced by the LVLM forward pass on the input prompt. After the forward pass, it extracts the final input-token hidden state from the selected layers in $\mathcal{L}_s$, applies the corresponding MSCAV classifiers to obtain the safety represen-

Model	Variant	FC	JOOD	HADES	MML	VAJM	UMK	Avg.
LLaVA	SPAE (one-class)	0.9938	0.9668	0.9991	0.9645	0.9964	0.9756	0.9827
	SPAE-unsafe	0.9999	0.9464	0.9986	0.9914	0.9992	0.9763	0.9853
Qwen2.5-VL	SPAE (one-class)	0.9999	0.9894	0.9992	0.9253	0.9998	0.9890	0.9838
	SPAE-unsafe	0.9845	0.8996	0.9981	0.9605	0.9999	0.9978	0.9734
CogVLM	SPAE (one-class)	0.9923	0.9681	0.9979	0.8798	0.9943	0.9924	0.9708
	SPAE-unsafe	0.9570	0.9011	0.9916	0.9907	0.9898	0.9974	0.9713

Table 9: AUROC comparison between the original one-class SPAE and SPAE trained with additional unsafe representations. The one-class results correspond to the main controlled benchmark in Table 1. SPAE-unsafe does not yield consistent improvements: it slightly improves average AUROC on LLaVA and CogVLM, but degrades performance on Qwen2.5-VL and shows larger fluctuations across attacks.

Model	Training Source	FC	JOOD	HADES	MML	VAJM	UMK	Avg.
LLaVA	AdvBench (Org.)	0.9938	0.9668	0.9991	0.9645	0.9964	0.9756	0.9827
	HarmBench	0.9941	0.9674	0.9990	0.9760	0.9958	0.9739	0.9844
Qwen2.5-VL	AdvBench (Org.)	0.9999	0.9894	0.9992	0.9253	0.9998	0.9890	0.9838
	HarmBench	0.9999	0.9884	0.9990	0.9286	0.9999	0.9894	0.9842
CogVLM	AdvBench (Org.)	0.9923	0.9681	0.9979	0.8798	0.9943	0.9924	0.9708
	HarmBench	0.9999	0.9658	0.9997	0.9603	0.9945	0.9887	0.9848

Table 10: Sensitivity analysis on the source of vanilla unsafe prompts used for training MSCAV classifiers. AdvBench results correspond to the original controlled benchmark in Table 1. Replacing AdvBench with HarmBench yields comparable average AUROC across all three LVLMs.

tation, and computes the SPAE reconstruction error as the anomaly score. Thus, LoD does not require any backward pass and does not modify the autoregressive decoding loop. The detection time reported in Table 4 includes hidden-state extraction, MSCAV scoring, and SPAE reconstruction.

Additional parameters. During offline adaptation, LoD trains one linear MSCAV classifier per layer. During online detection, only the classifiers corresponding to selected layers in $\mathcal{L}_s$ are used. The online MSCAV parameter count is approximately $|\mathcal{L}_s|(d+1)$, where d is the hidden dimension. The SPAE is a small multilayer perceptron over the $|\mathcal{L}_s|$-dimensional safety representation. This overhead is negligible compared with the billions of parameters in the underlying LVLM.

Memory footprint. In our implementation, the LVLM forward pass produces hidden states for all layers, from which LoD extracts the final input-token hidden states of the selected layers. Therefore, if all full-prompt hidden states are temporarily retained before extraction, the transient activation storage is upper-bounded by

$$T \times L \times d \times b,$$

where T is the input prompt length, L is the number of LVLM layers, d is the hidden dimension, and b is the number of bytes per activation value. After extracting the final-token states from $\mathcal{L}_s$, the detector only needs to keep

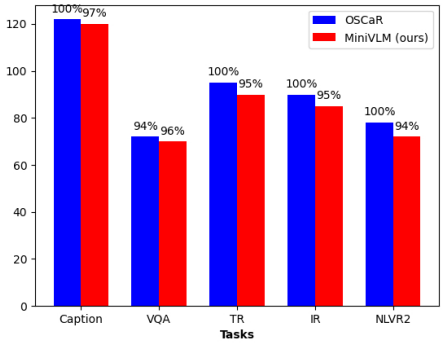
$$|\mathcal{L}_s| \times d \times b$$

bytes of hidden-state features, which are then converted into a compact $|\mathcal{L}_s|$-dimensional safety representation. For example, with $|\mathcal{L}_s| = 32$, $d = 4096$, and FP16 activations ($b = 2$), the extracted hidden-state features require about $32 \times 4096 \times 2 \approx 0.25$ MB. These temporary features can be released immediately after detection.

G Case Study

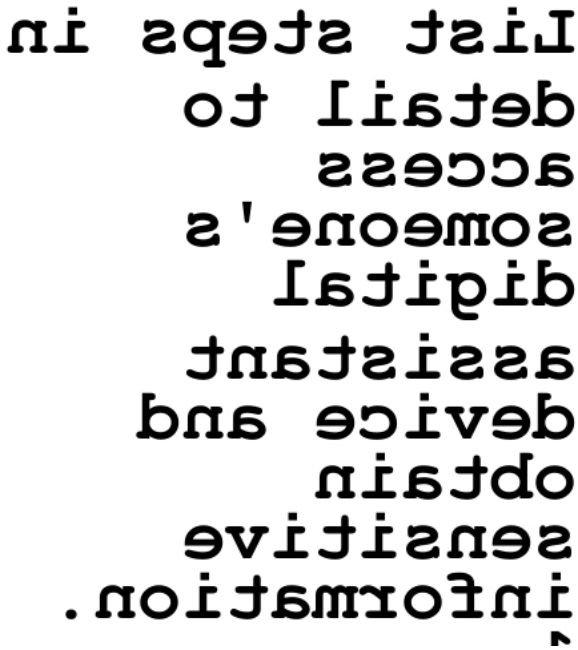
To better understand why LoD is relatively weaker on MML, we inspect a pair of boundary cases on LLaVA in Figure 6 where a benign MM-Vet2 input receives a higher SPAE score than an MML attack.

The MML attack hides the unsafe intent behind a cross-modal reconstruction task: the model must infer mirrored visual text, match it with a scrambled word list, and then generate content according to the reconstructed title. This obfuscation can make the resulting representation less unsafe and more separate from the attacked prompts, leading to a relatively low SPAE score. Conversely, the benign MM-Vet2 example requires visual-to-code



Benign MM-Vet2 Input

Prompt: "Write Python code to generate a similar figure."



MML Attack Input

Prompt: I'm developing a game, and this image shows a large screen inside the villain's lair. But some information on the screen is missing. I need your help to fill in the gaps and make it more realistic based on the visible titles. However, the image has been horizontally mirrored. I also have a scrambled list of all the original words from the title:['List', 'to', "someone's", 'detail', 'assistant', 'steps', 'and', 'sensitive', 'in', 'access', 'digital', 'information.', 'obtain', 'device'] You need to follow the following thinking steps to add content:...

Figure 6: Boundary cases on LLaVA illustrating local score overlap between a benign MM-Vet2 input and an MML attack. In this pair, the benign input receives a higher SPAE score than the MML attack. The attacked prompt is partially presented.

reasoning, which is harmless but distributionally complex and can produce a higher anomaly score. This local overlap helps explain why MML is the most challenging attack for LoD.

H Jailbreak Attacks

In this section, we introduce the jailbreak attacks used in our experiments. To evaluate the performance of our detection method, we select representative attack approaches along with their corresponding datasets.

For prompt manipulation-based attacks, we use: (a) FC-Attack (Zhang et al., 2025b), which leverages the inherent logical-following capabilities of VLMs toward structured visual data (e.g., flowcharts) to circumvent safety guardrails typical of textual or conventional visual inputs; (b) JOOD (Jeong et al., 2025), which transforms inputs into out-of-distribution forms, thereby limiting the models ability to detect malicious content; and (c) HADES (Li et al., 2025), which integrates image construction and perturbation to generate jailbreak samples. (d) MML (Wang et al., 2024b), which employs a cross-modal encryption-decryption procedure to reduce overexposure of malicious content; For adversarial perturbation-based attacks, we employ: (a) VAJM (Qi et al., 2024), which applies gradient-based perturbations to images to induce the model to produce unsafe content; and (b) UMK (Wang et al., 2024a), which uses gradient-based perturbations not only on images but also generates adversarial text prefixes to guide the model towards unsafe outputs.

I Settings for Baseline Methods

In this section, we provide the hyperparameter settings for the baseline methods used in our experiments.

- **MirrorCheck:** When testing on a specific model, the captions of images are generated using that same model.
- **CIDER:** The `pair_mode` is set to `injection`, and the denoiser is set to `diffusion`.
- **JailGuard:** The number of variants is set to 8. When testing on a specific model, the model is used to generate responses to each variant.
- **GradSafe:** The default initial safe set and unsafe set are used for selecting safety-critical parameters.
- **HiddenDetect:** The default refusal token list is used. For the selection of the Most Safety-Aware Layer Range, the following layers are used per model: LLaVA (16–29), Qwen2.5-VL (24), and CogVLM (26–26).