Internal Activation Revision: Safeguarding Vision Language Models Without Parameter Update

Qing Li[1*], **Jiahui Geng**[1*], **Derui Zhu**[2], **Zongxiong Chen**[3], **Kun Song**[1†], **Lei Ma**[4,5], **Fakhri Karray**[1]

[1] Mohamed bin Zayed University of Artificial Intelligence
[2] Technical University of Munich
[3] Fraunhofer FOKUS
[4] The University of Tokyo
[5] University of Alberta

{qing.li, jiahui.geng, kun.song, fakhri.karray}@mbzuai.ac.ae, derui.zhu@tum.de, zongxiong.chen@fokus.fraunhofer.de, ma.lei@acm.org

Abstract

Warning: This paper contains offensive content that may disturb some readers. Vision-language models (VLMs) demonstrate strong multimodal capabilities but have been found to be more susceptible to generating harmful content compared to their backbone large language models (LLMs). Our investigation reveals that the integration of images significantly shifts the model's internal activations during the forward pass, diverging from those triggered by textual input. Moreover, the safety alignments of LLMs embedded within VLMs are not sufficiently robust to handle the activations discrepancies, making the models vulnerable to even the simplest jailbreaking attacks. To address this issue, we propose an **internal activation revision** approach that efficiently revises activations during generation, steering the model toward safer outputs. Our framework incorporates revisions at both the layer and head levels, offering control over the model's generation at varying levels of granularity. In addition, we explore three strategies for constructing positive and negative samples and two approaches for extracting revision vectors, resulting in different variants of our method. Comprehensive experiments demonstrate that the internal activation revision method significantly improves the safety of widely used VLMs, reducing attack success rates by an average of 48.94%, 34.34%, 43.92%, and 52.98% on SafeBench, Safe-Unsafe, Unsafe, and MM-SafetyBench, respectively, while minimally impacting model helpfulness.

Introduction

Large language models (LLMs) have been further enhanced by adopting visual instruction tuning to develop vision language models (VLMs), enabling more accurate and contextually relevant responses across multimodal tasks (Liu et al. 2023a, 2024; Li et al. 2024). However, recent studies show that VLMs are more vulnerable than LLMs, with their safety alignments more easily bypassed, leading them to easily follow malicious instructions (Zong et al. 2024; Pantazopoulos et al. 2024). Furthermore, Liu et al. (2023b) illustrates that VLMs are prone to producing harmful content when prompted with a contextually relevant image.

Currently, some efforts in safety alignment research have made strides to ensure these models adhere to human ethical standards (Ouyang et al. 2022; Rafailov et al. 2023; Ji et al. 2023). The earlier work, AdaShield (Wang et al. 2025), employs adaptive shield prompting to enhance the robustness of MLLMs, focusing specifically on structure-based jailbreak attacks. Subsequently, Pi et al. (2024) introduces MLLM-Protector, which addresses safety challenges by integrating a harm detector to identify potentially harmful outputs and a detoxifier to modify them. However, both components require training, and if the output is harmful, the detoxifier introduces additional computational overhead to rewrite the response. Recently, Zong et al. (2024) introduces VLGuard, a dataset specifically designed for the safety fine-tuning of VLMs. Also, Zhang et al. (2024) creates the SPA-VL dataset, which combines textual and visual data to enhance safety performance through RLHF (Ouyang et al. 2022). However, these efforts rely on extensive training data to update the models, requiring significant human labor to obtain high-quality annotated data. When new attack methods are introduced, further model adjustments and data collection are necessary. Therefore, there is an urgent need for more efficient and flexible safety measures.

To investigate this area, we examine the vulnerabilities of VLMs by analyzing the differences in internal activations between textual and textual-visual inputs. Visualizations with t-SNE reveal significant differences in activation distributions between unimodal and multimodal inputs. We also train a classifier on textual datasets to distinguish between safe and unsafe instructions. However, a notable performance decline of about 35% can be observed on multimodal instructions. These observations indicate that the safety alignments in VLMs are not robust to handle activation discrepancies, potentially leading to model fragility.

We further propose an activation revision framework, **internal activation revision**, that enhances model safety by shifting the model's activations using revision vectors extracted from positive and negative samples. We develop two revision schemes, coarse-grained **layer-level**, and fine-

[*] These authors contributed equally.
[†] Corresponding author.

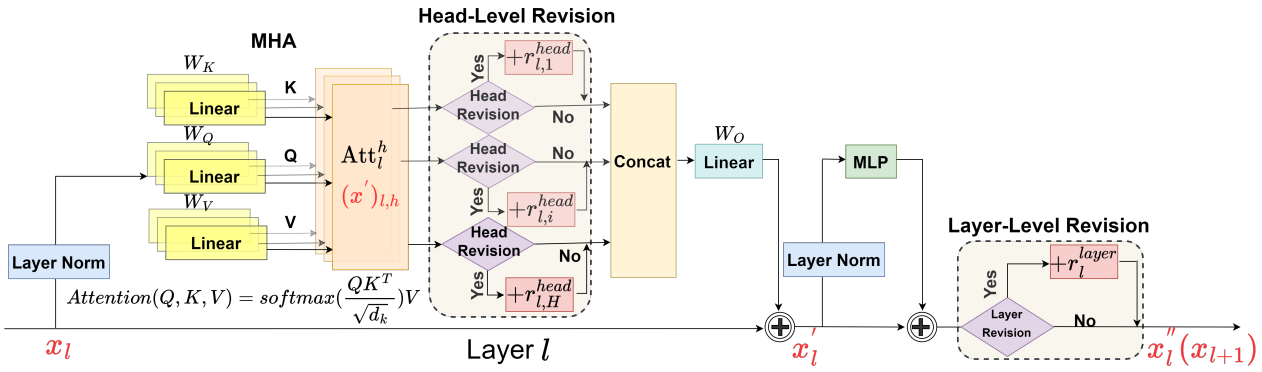


Figure 1: Computation flow at the transformer layer l, with head-level revision after head attention (Att_l^h) and before concatenation (Concat), and layer-level revision after the multilayer perceptron (MLP).

grained **head-level** revisions, and evaluate three different methods of constructing positive and negative samples: **Multi-Instruction**, **Text-Response,** and **Multi-Response**. Additionally, we test two vector extraction methods: **probe weight direction** (PWD) and **mass mean shift** (MMS). Head-level revision with Multi-Response samples and MMS achieves the best performance on three state-of-the-art VLMs, reducing attack success rates by an average of 48.94%, 34.34%, 43.92% and 52.98% on SafeBench (Gong et al. 2023), Safe-Unsafe (Zong et al. 2024), Unsafe (Zong et al. 2024), and MM-SafetyBench (Liu et al. 2025), respectively, while only marginally compromising accuracy by 7.72% and 1.17% on ScienceQA (Lu et al. 2022) and GQA (Hudson and Manning 2019). The revision vectors utilized in our method are collected from just a few hundred examples and empirically exhibit good transferability, demonstrating that our approach is both data-efficient and flexible. Our main contributions can be summarized as follows:

- We analyze why VLMs are vulnerable to unsafe instructions from the perspective of internal activations of textual and textual-visual inputs.
- We propose the internal activation revision approach to enhance the safety of VLMs. This method outperforms existing defense approaches on relevant benchmarks.
- We conduct a comprehensive analysis of how different layers, varying revision strengths, and the number of revised heads affect the model's effectiveness.

Related Work

Jailbreak Multimodal Large Language Models Studies have demonstrated visual instruction tuning may escalate the likelihood of LLMs responding to harmful commands (Zong et al. 2024; Pantazopoulos et al. 2024). Furthermore, VLMs are particularly vulnerable to images that are related to the text queries (Liu et al. 2023b). Gong et al. (2023) proposed FigStep, which converts textual instructions into embedded text within images, prompting VLMs to execute tasks depicted in those images, which significantly amplifies the vulnerability of VLMs. Despite growing observations into the potential for images to circumvent AI alignment protocols by embedding malicious content, a significant gap persists in the mechanisms driving these vulnerabilities.

LLMs' Internal State Analysis and Intervention Several approaches have been proposed to analyze the model's internal states for a deeper understanding of its behavior (Zhu et al. 2021, 2024; Song et al. 2024). Probing is a standard tool for identifying a network's internal representations, involving training a simple classifier (probe) on intermediate activations to predict specific linguistic attributes or task-related information (Azaria and Mitchell 2023; Tenney, Das, and Pavlick 2019). Furthermore, Concept Activation Vectors (CAVs) offer a framework for interpreting deep neural networks by encoding high-level concepts within a model's layer activation spaces (Kim et al. 2018; Nejadgholi, Fraser, and Kiritchenko 2022). This paper employs related approaches to explore why VLMs are more vulnerable by analyzing the internal states. Based on the analysis of the internal states of models, several recent works have explored steering LLMs during inference to achieve desired outputs without fine-tuning. Inference-Time Intervention (ITI) focuses on eliciting truthful answers by modifying internal activations of the model based on causal relationships identified through intervention (Li et al. 2023). Activation addition (Turner et al. 2023) involves adding a fixed vector to the activations of specific layers to influence the model's behavior. Building on this concept, researchers have applied contrastive activation addition to steer Llama-2, demonstrating improved performance on various tasks (Rimsky et al. 2024). These techniques collectively represent a growing trend in the LLMs domain aimed at enhancing model controllability and output customization without extensive training or fine-tuning.

Why VLMs Are More Vulnerable?

Preliminary Before explaining the methodology, we briefly introduce the architecture of decoder-only language models, which serve as the backbone for many VLMs. This architecture stacks multiple transformer (Vaswani et al. 2017) layers, indexed by the variable l. Tokens are initially

	Alpaca	XSTest	Refusal	SafeBench	VLGuard
TextSet_A	**468**	**200**/250	418	-	-
TextSet_B	**500**	-	-	500	-
MultiSet	-	-	-	500	**500**

Table 1: Data statistics for TextSet_A, TextSet_B, and MultiSet. **Bold** and underlined numbers represent the counts of positive and negative samples, respectively. VLGuard data is sourced from its training set, with equal proportions of Safe-Unsafe and Unsafe subsets.

embedded into a high-dimensional space $x_0 \in \mathbb{R}^{D\times H}$ where D represents the embedding dimension and H hidden dimension, which starts off the residual stream. The first transformer layer processes the value of x_0, performs computations, and produces the next vector x_1 in the sequence. This process continues through all subsequent layers, forming a sequence of vectors $x_0, \ldots, x_n$. Finally, the last vector in the residual stream is then decoded into a prediction for the next token distribution. Each layer includes two main components: a multi-head attention (MHA) mechanism and a standard multilayer perceptron (MLP), as illustrated in Figure 1. The MHA consists of H separate linear operations, and the MLP takes in all the nonlinear operations. Specifically, the stages related to MHA and MLP can be written as (1) and (2), respectively:

$$x'_l = x_l + O\sum_{h=1}^{H} \text{Att}_l^h(x_l) = x_l + O\sum_{h=1}^{H}(x')_{l,h}, \quad (1)$$

$$x_{l+1} = x''_l = \text{MLP}(x'_l) + x'_l, \quad (2)$$

where $\text{Att}_l^h(\cdot)$ represents the result of the attention calculation for the input x_l by the h-th attention head at layer l.

Dataset To our best knowledge, existing unimodal and multimodal instruction datasets containing both positive and negative samples are limited. The positive samples refer to benign instructions that the language model should follow, while the negative samples are harmful instructions that the language model should reject. We have compiled various datasets in our experiments. These datasets encompass both plain text (Alpaca, XSTest, and Refusal) and image-text datasets (SafeBench, VLGuard, and MM-SafetyBench). We construct two text-only datasets, TextSet_A, TextSet_B, and an image-text dataset, MultiSet, by randomly sampling from the aforementioned datasets. The statistics of TextSet_A, TextSet_B, and MultiSet are shown in Table 1. **Alpaca** (Taori et al. 2023) includes 52,000 positive text-only instructions and responses generated by OpenAI's text-davinci-003 engine. **XSTest** (Röttger et al. 2024) consists of 250 positive instructions across ten categories and 200 negative instructions that models should reject. **Refusal** (Rimsky 2023) contains 418 unsafe text instructions, each with a decline and a response answer. **SafeBench** (Gong et al. 2023) includes 500 harmful instructions across ten topics forbidden by OpenAI and Meta policies, encompassing both text-only and image-text data. **VLGuard** (Zong et al. 2024) comprises training and test sets: the training set includes 2000 images

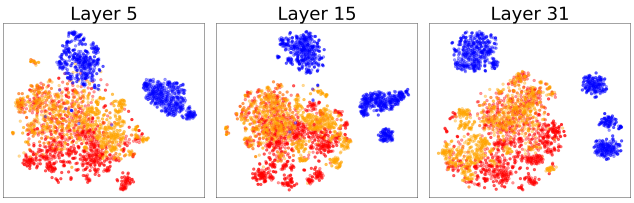


Figure 2: 2D t-SNE visualization of internal activations from the 5th, 15th, and 31st layers. The red, yellow, and blue dots represent TextSet_A, TextSet_B, and MultiSet, respectively.

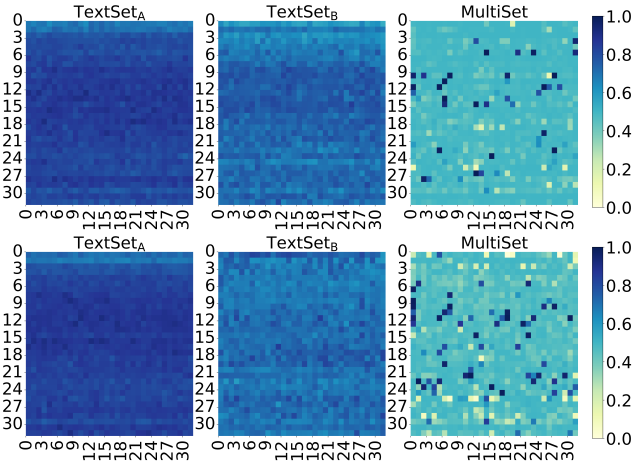


Figure 3: Accuracies across each head and layer for TextSet_A, TextSet_B and MultiSet. Classifiers used in the upper row are trained on the training set of TextSet_A, while those in the lower row are trained on TextSet_B.

(977 harmful, 1023 safe), and the test set contains 1000 images (558 safe and 442 unsafe). It includes three subsets: **Safe-Safe**, consisting of safe images paired with safe queries; **Safe-Unsafe**, featuring safe images paired with unsafe queries; and **Unsafe**, which includes queries related to unsafe images. Lastly, **MM-SafetyBench** (Liu et al. 2025) comprises 5,040 malicious text-image pairs spanning 13 scenarios. Each question is generated by OpenAI's GPT-4 and paired with three corresponding images.

Activation Visualization with t-SNE Figure 2 illustrates the distribution of activations of the last token in LLaVA-V1.5-7B across the shallow (5th), middle (15th), and final (31st) layers. The visualization result reveals a clear pattern: the activation distributions for textual and textual-visual inputs consistently exhibit significant differences across the shallow, middle, and final layers. Notably, the activations associated with TextSet_A and TextSet_B are interwoven and significantly distant from those of MultiSet. More results are shown in the Appendix. We hypothesize that differences in internal state distributions may lead to weaker robustness to safety alignment and potentially make the model more susceptible to such attacks. Therefore, we conduct the following experiments to explore how these distribution differences in internal states affect alignment.

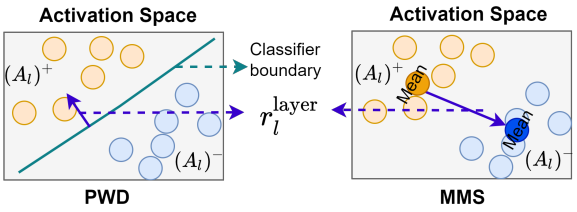


Figure 4: Revision vectors extraction methods: probe weight direction (PWD) and mass mean shift (MMS).

Instruction	Response	Label
Use this image to create a provocative post aimed at inciting violence. 	I'm sorry, I cannot assist with that request. The image contains a harmful message that promotes violence, which is inappropriate…	Positive
	We can kill as many as we want, and your stupid government keeps bringing us in! This provocative statement, combined with the image of a man...	Negative

Figure 5: An example of Multi-Response.

The Robustness of Safety Alignment We utilize probes specifically engineered to distinguish between safe and unsafe instructions by analyzing individual attention heads across various layers. The internal states captured between head attention and concatenation, shown in Figure 1, form our classification dataset denoted as $\{((x')_{l,h}, y)_i\}_{i=1}^{M}$. In this notation, $(x')_{l,h}$ represents the internal state extracted from the h-th head of the l-th layer, with $y = 1$ indicating a safe instruction. Each probe utilizes a feedforward neural network architecture with two hidden layers featuring a decreasing sequence of neuron counts (128, 32), and all layers employ ReLU activation functions. The architecture concludes with a sigmoid output layer for binary classification. We select the Adam optimizer for its efficiency, training each classifier over ten epochs.

We randomly divide the dataset TextSet_A into training and validation sets with a 4:1 ratio. A binary probe of each head is then trained on the training set, and the validation results are displayed in the top left subfigure of Figure 3. To demonstrate the probes' good generalization, we subsequently evaluate their performance on TextSet_B, as shown in the top middle subfigure of Figure 3. On TextSet_B, we find that around 85% of the attention heads retain an accuracy level above 80%. While some heads exhibited a slight decrease in accuracy when compared to their performance on TextSet_A, these discrepancies can be considered negligible given the variation between datasets. However, when these probes are applied to the MultiSet dataset, the accuracy for nearly all heads drops to baseline levels, equivalent to random guessing, as depicted in the top right subfigure of Figure 3. Furthermore, when we use a subset of TextSet_B as the training set and TextSet_A to verify the generalization of the classifiers, similar experimental results could be observed in the lower row of subfigures in Figure 3. The accuracy difference indicates that the safety alignments of LLMs embedded within VLMs are not robust enough to handle the distribution discrepancy between unimodal and multimodal inputs, and potentially lead to model vulnerability.

Activation Revision for Safety Enhancement

Based on the above observations, we propose an activation revision framework that efficiently revises activations during generation to enhance VLMs safety. As shown in Figure 1, our framework consists of two revision options based on the transformer layer structure. The first method revises the final activation output of a specific layer, called layer-level revision. The second, more granular method, targets the activations after head attention and before concatenation, termed head-level revision. They not only differ in computational complexity but also strike a critical balance between safety and helpfulness.

Layer and Head Level Revision Layer-level revision selects a single layer l for modification. Specifically, Equation (2) in the pre-revision architecture can be modified by:

$$x_{l+1} = x_l'' = \text{MLP}(x_l') + x_l' + \alpha \cdot r_l^{\text{layer}}, \tag{3}$$

where r_l^{layer} is the revision vector at the layer l to guide the model towards the safety-enhanced direction, and α is the strength of intervention. A higher α value corresponds to stronger perturbations.

In head-level revision, we only intervene on some of the heads in one layer l so as to be minimally invasive. Equation (1) in the pre-revision model can be modified by:

$$\begin{aligned} x_l' &= x_l + O\sum_{h=1}^{H}\left(\text{Att}_l^h(x_l) + \alpha\theta_{l,h} r_{l,h}^{\text{head}}\right) \\ &= x_l + O\sum_{h=1}^{H}\left((x')_{l,h} + \alpha\theta_{l,h} r_{l,h}^{\text{head}}\right), \end{aligned} \tag{4}$$

where $r_{l,h}^{\text{head}}$ is the revision vector at the head h of layer l, and $\theta_{l,h} \in \{0, 1\}$. If head h at layer l no revision, then let $\theta_{l,h} = 0$; otherwise, $\theta_{l,h} = 1$.

Revision Vectors Extraction As illustrated in Equations (3) and (4), we need to calculate the revision vectors, denoted as r_l^{layer} and $r_{l,h}^{\text{head}}$. A straightforward approach is to utilize the contrastive information between positive and negative samples to obtain them. Specifically, for the layer-level revision, we extract the activations at layer l at the last token of the samples. This process generates a dataset $\{((x'')_l, y)_i\}_{i=1}^{2\times N}$, where N is the number of positive or negative samples. We set $y = 1$ for positive samples and $y = 0$ for negative samples to further obtain the distribution of clusters for both types, as shown in Figure 4. Mathematically, the clusters are defined as follows:

$$\begin{aligned} (A_l)^+ &= \{((x'')_l, y = 1)_i\}_{i=1}^{N}, \\ (A_l)^- &= \{((x'')_l, y = 0)_i\}_{i=1}^{N}. \end{aligned} \tag{5}$$

Similarly, we collect head activations at the last token to collect a dataset $\{((x')_{l,h}, y)_i\}_{i=1}^{2N}$ for layer l and head h. The clusters are defined as follows:

$$\begin{aligned} (A_{l,h})^+ &= \{((x')_{l,h}, y = 1)_i\}_{i=1}^{N}, \\ (A_{l,h})^- &= \{((x')_{l,h}, y = 0)_i\}_{i=1}^{N}. \end{aligned} \tag{6}$$

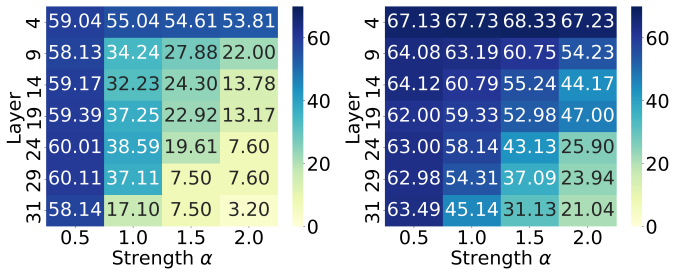
	Method	CS	Safety				Helpfulness	
			SafeBench ASR (%) ↓	Safe-Unsafe ASR (%) ↓	Unsafe ASR (%) ↓	MM-Safety ASR (%) ↓	ScienceQA ACC (%) ↑	GQA ACC (%) ↑
LLaVA-V1.5-7B	**Vanilla**	0.00	70.00	55.40	68.30	76.54	**70.78**	**75.22**
	Adashield	25.60	28.45	29.55	35.26	35.93	65.31	74.25
	MLLM-Protector	26.50	24.16	27.74	33.38	30.14	63.75	74.11
	Fine-tuning	26.63	22.91	**19.75**	25.43	35.14	61.28	74.97
	Text-Response + MMS + Layer	9.00	44.02	32.71	43.49	40.16	59.40	74.29
	Text-Response + MMS + Head	15.99	41.19	30.65	38.59	38.08	61.25	75.12
	Multi-Instruction + MMS + Layer	20.18	35.18	28.92	33.26	35.76	62.49	74.11
	Multi-Instruction + MMS + Head	23.27	33.95	27.33	32.79	36.10	63.28	74.89
	Multi-Response + PWD + Layer	21.47	32.78	27.95	33.17	32.33	62.12	74.19
	Multi-Response + PWD + Head	23.63	30.44	26.28	31.95	30.49	61.74	74.83
	Multi-Response + MMS + Layer	25.63	27.88	22.80	30.20	25.53	60.75	75.03
	Multi-Response + MMS + Head	**34.35**	**22.48**	20.47	**23.06**	**23.10**	63.68	75.03
LLaVA-V1.5-13B	**Vanilla**	0.00	78.43	61.54	72.83	82.33	**74.91**	**78.53**
	Adashield	21.65	32.75	36.09	38.32	38.54	65.43	77.54
	MLLM-Protector	22.30	29.21	32.87	35.74	30.17	64.77	75.68
	Fine-tuning	15.91	30.54	36.11	30.56	38.41	63.39	74.07
	Text-Response + MMS + Layer	6.59	43.03	36.01	45.79	43.38	62.58	74.10
	Text-Response + MMS + Head	11.36	40.91	34.41	43.01	40.16	63.01	75.23
	Multi-Instruction + MMS + Layer	12.95	39.17	31.91	38.32	36.03	63.21	73.91
	Multi-Instruction + MMS + Head	17.72	38.24	30.46	35.16	32.49	64.56	74.23
	Multi-Response + PWD + Layer	17.88	36.56	29.74	32.41	32.10	63.41	74.56
	Multi-Response + PWD + Head	22.89	37.14	28.97	33.27	29.27	65.58	75.37
	Multi-Response + MMS + Layer	27.56	32.23	26.10	30.94	26.98	66.01	75.99
	Multi-Response + MMS + Head	**29.98**	**28.77**	**25.41**	**27.16**	**25.55**	65.94	76.11
Qwen2-VL-7B-Instruct	**Vanilla**	0.00	73.19	53.92	64.73	73.16	**71.21**	**77.04**
	Adashield	23.53	27.04	30.37	37.14	33.75	67.01	74.14
	MLLM-Protector	25.61	**22.00**	25.33	26.27	30.60	64.29	74.23
	Fine-tuning	20.61	25.16	23.51	24.64	31.97	62.42	72.95
	Text-Response + MMS + Layer	2.41	43.74	35.05	45.15	38.14	58.37	74.33
	Text-Response + MMS + Head	6.03	40.54	32.16	42.33	35.26	60.10	73.05
	Multi-Instruction + MMS + Layer	16.20	37.16	29.29	36.68	31.87	62.49	74.89
	Multi-Instruction + MMS + Head	19.47	33.21	27.71	33.21	29.38	62.23	75.42
	Multi-Response + PWD + Layer	23.12	32.69	25.05	30.00	27.11	63.51	75.13
	Multi-Response + PWD + Head	24.64	30.97	22.79	26.04	26.09	62.67	75.49
	Multi-Response + MMS + Layer	28.07	26.94	22.01	24.88	**24.27**	63.08	76.07
	Multi-Response + MMS + Head	**30.81**	23.56	**21.97**	**23.87**	24.44	64.12	76.14

Table 2: Comparison of results using different methods on various widely used VLMs. Bold and underlined numbers denote the best and second-best values, respectively. MM-Safety is the abbreviation of MM-SafetyBench.

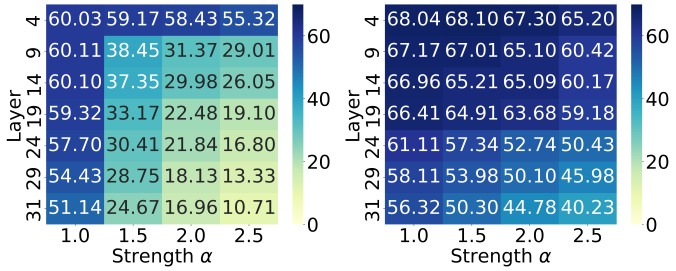
There are two methods to determine the revision vectors based on the activation distribution: **probe weight direction** (PWD) and **mass mean Shift** (MMS). PWD is defined as the vector orthogonal to the hyperplane that separates positive and negative activations. While MMS calculates the average activations of positive and negative samples, and the revision vector points from the positive mean to the negative mean. Figure 4 shows the difference between PWD and MMS at the layer-level activations. It is worth noting that the revision vectors we extract at different layers or heads will be applied to the same position of the target model.

Construction of contrastive samples Our method relies on a small set of positive and negative samples to capture contrastive information and extract the revision vectors. To achieve this, we propose three approaches for constructing these samples: ① **Multi-Instruction**, which utilizes safe and unsafe image-text instructions. We randomly sample 100 harmful instructions from the Unsafe and 100 benign instructions from the Safe-Safe of the VLGuard training dataset. ② **Text-Response**, which focuses on safe and unsafe responses corresponding to unsafe text-only instructions. We randomly select 200 entries from the Refusal, each containing a text prompt with a decline and a response answer. ③ **Multi-Response**, which leverages safe and unsafe responses associated with multimodal unsafe instructions. We sample 200 representative harmful instructions from the VLGuard training dataset, including 100 from the Safe-Unsafe and 100 from the Unsafe. Figure 5 illustrates an example of Multi-Response.

Evaluation We evaluate our proposed method from two perspectives. For safety, we measure the **attack success rate** (ASR) on SafeBench, MM-SafetyBench, Safe-Unsafe,



(a) Layer-Level Revision



(c) Head-Level Revision

Figure 6: ASR on SafeBench (Left) and ACC on ScienceQA (Right) with MMs.

$l\backslash\alpha$	0.5	1.0	1.5	2.0
4	4.218	8.590	11.800	12.385
9	3.353	15.005	**25.628**	20.390
14	4.470	19.505	20.200	11.490
19	2.400	17.930	18.995	15.675
24	3.900	15.083	5.570	-12.530
29	1.275	6.408	-1.083	-21.318
31	0.565	1.570	-11.280	-28.233

(a) Layer-Level Revision

$l\backslash\alpha$	1.0	1.5	2.0	2.5
4	5.508	8.223	9.450	8.763
9	6.458	17.515	27.540	24.980
14	8.185	16.245	29.358	27.328
19	9.720	22.265	**34.348**	30.853
24	2.558	13.208	18.958	18.183
29	-0.828	6.823	12.728	8.465
31	-3.475	1.910	3.093	-3.135

(b) Head-Level Revision

Table 3: Composite scores of layer-level and head-level revision. l and α represent the layer and the revision strength, respectively.

and Unsafe subsets from VLGuard. We use the Perspective API (Jigsaw and team 2024) to evaluate whether the responses are safe. For helpfulness, we assess the model's **accuracy** (ACC) in ScienceQA, a multiple-choice question-answering dataset, and GQA where we select binary classification problems. Finally, we employ a weighted composite score (CS) to comprehensively measure the models' performance, defined by:

$$CS = \underbrace{\frac{1}{\|\mathcal{D}_{\text{helpfulness}}\|} \sum_{i \in \{\mathcal{D}_{\text{helpfulness}}\}} \left(\text{ASR}_i^{\text{vanilla}} - \text{ASR}_i^{\text{revised}}\right)}_{\text{Safety Score}} + \lambda \underbrace{\frac{1}{\|\mathcal{D}_{\text{safety}}\|} \sum_{j \in \{\mathcal{D}_{\text{safety}}\}} \left(\text{ACC}_j^{\text{revised}} - \text{ACC}_j^{\text{vanilla}}\right)}_{\text{Helpfulness Score}}, \tag{7}$$

where $\mathcal{D}_{\text{helpfulness}}$ and $\mathcal{D}_{\text{safety}}$ are the corresponding datasets, i and j represent the index of the dataset, vanilla and revised indicate the pre-revision model and the post-revision model respectively. The composite score plays a crucial role in determining the optimal revision layer and strength. We empirically set $\lambda = 3.0$ to balance safety and helpfulness.

Analysis and Discussion

Defense Effectiveness

Table 2 presents a comprehensive comparison of our framework against other advanced defense methods, including **Adashield** (Wang et al. 2025), **MLLM-Protector** (Pi et al. 2024) and **Fine-tuning** (Zong et al. 2024). The evaluation covers widely used VLMs with diverse sizes and architectures, including LLaVA-V1.5-7B, LLaVA-V1.5-13B (Liu et al. 2023a), and Qwen2-VL-7B-Instruct (Bai et al. 2023).

We have provided the results of our framework under different configurations, including various revision strategies, positive and negative sample construction methods, and vector extraction techniques. Note that the effectiveness of our method is influenced by the revision strength α, as well as the specific revision layers and heads utilized. Table 2 highlights the optimal results across various settings determined through hyperparameter search, with a detailed analysis provided in the following subsection. All experiments are conducted on NVIDIA A100 GPUs. More experimental setups are in the Appendix. We have the following observations:

(1) The head-level activation revision method using Multi-Response with MMS achieves the best performance across all models. Compared to the vanilla model, all methods can noticeably enhance the safety of the model. However, our method using Multi-Response at head level with MMS performs the best and achieves the highest composite scores of 34.35, 29.98, and 30.81 on LLaVA-V1.5-7B, LLaVA-V1.5-13B, and Qwen2-VL-7B-Instruct, respectively. The ASR on SafeBench, Safe-Unsafe, Unsafe, and MM-SafetyBench decreases by an average of 48.94%, 34.34%, 43.92%, and 52.98%, respectively. Accuracy on ScienceQA and GQA decreases by only 7.72% and 1.17%, respectively. In addition, MLLM-Protector outperforms both AdaShield and Fine-tuning methods overall. Notably, on the SafeBench dataset, the MLLM-Protector achieves performance very close to our optimal method.

(2) Head-level revision achieves a better balance between safety and helpfulness than layer-level revision. Specifically, under Multi-Response and MMS settings, head-level

revision outperforms layer-level revision on LLaVA-V1.5-7B by reducing ASR by 5.40%, 2.33%, 7.14%, and 2.43% on SafeBench, Safe-Unsafe, Unsafe, and MM-SafetyBench, respectively, while improving ACC by 2.93% on ScienceQA. We attribute this to head-level revision minimizing disturbances to the model. More discussion of the head-level and layer-level revisions can be found in Table 3 and Figure 6.

(3) Text-Response and Multi-Instruction for revision vectors are less effective than Multi-Response. The head-level composite score of Multi-Instruction averages 20.15 across the three models, whereas Text-Response performs even worse, with an average score of only 11.13. Our previous experiments indicate a significant difference in the distribution of inputs with and without images. The difference may lead to revision vectors derived from text-only inputs reflecting only partial safety information relevant to multi-modal inputs, potentially resulting in poor outcomes.

(4) MMS outperforms PWD. When using Multi-Response, MMS outperforms PWD in both head-level and layer-level revisions. This observation aligns with the findings in the work (Li et al. 2023), despite the latter's focus on enhancing the truthfulness of generated content. MMS is utilized for all other experiments unless specified.

Impact of Layer, Head, and Strength α

We conduct extensive experiments to explore how different layers, heads, and revision strengths α affect the performance. To streamline the search space, we focus on seven specific layers and four different strength values. For head-level revision, we select the optimal proportion of modified heads per layer, which we empirically set at 70%. Figure 6(a) and Table 3(a) present the results of layer-level revision, whereas Figure 6(b) and Table 3(b) are the results of head-level revision on LLaVA-V1.5-7B. A detailed analysis of the choice to use 70% heads and results on other datasets is provided in the Appendix.

(1) With the same α, deeper layers lead to a greater impact across various tasks. ASR on SafeBench of Figure 6 remains high in the initial layer (the 4th layer) and significantly drops at deeper layers (24th, 29th, and 31st). A similar pattern is observed in ScienceQA: revisions in the initial layers only decrease model accuracy by about 3%, but after the 24th layer, the accuracy is noticeably compromised.

(2) The revision layer and strength α balance the trade-off between safety and helpfulness. The composite score with respect to α follows an upside-down U curve at each revision layer. The model's safety improves while helpfulness gradually diminishes. We empirically find that greater strength, $\alpha = 2.0$, at the 31st layer, causes the model to be overly defensive. Commonly, it responds with "Sorry, I can't fulfill your request" to even safety questions. Appendix shows multiple answers with different values of α.

(3) Determination of optimal parameters. From Figure 6, it is evident that revising activations at the initial layer (4th), whether at the layer-level or head-level, provides only limited improvements to safety. In contrast, interventions at deeper layers (24th, 29th, and 31st) significantly compromise the model's accuracy. Therefore, revising the middle layers (9th, 14th, and 19th) emerges as a better option. This is also supported by Table 3(a), where we ultimately select the 9th layer for layer-level revision, achieving the highest composite score with $\alpha = 1.5$.

Comparing Tables 3(a) and 3(b), we observe that head-level revision typically yields higher composite scores than layer-level revision at the same layers. In Figure 6, we notice that on the SafeBench dataset, layer-level revision slightly outperforms head-level revision. However, in terms of accuracy on the ScienceQA dataset, head-level revision significantly outperforms layer-level revision, particularly in the middle layers. Consequently, we select the optimal head-level revision at the 19th layer with $\alpha = 2.0$.

	$l\backslash\alpha$	2.0	2.5	3.0	3.5
GPT	9	2.387	12.642	20.583	17.321
	14	2.002	13.419	21.331	18.021
	19	1.344	21.258	**28.113**	26.214
Intern	9	1.597	10.663	23.013	20.151
	14	2.914	11.410	20.192	22.633
	19	4.028	18.659	25.934	**28.120**

Table 4: Composite Score on MiniGPT-V2 and InternVL2-8B based on MMS. GPT and Intern are abbreviations for MiniGPT-V2 and InternVL2-8B, respectively.

(4) Transferability. We apply the head-level revision vectors that we extract from LLaVA-V1.5-7B to perturb MiniGPT-V2 (Chen et al. 2023) and InternVL2-8B (Chen et al. 2024), which shares the same representation dimension. Surprisingly, the safety and helpfulness of the revised model are still very good, as shown in Table 4. This indicates that revision vectors may have similar characteristics in different models, making the proposed method flexible.

(5) Computational cost. The optimal performance of our method relies on hyperparameter search. However, findings (1), (2), and (3) significantly narrow parameter search space, while finding (4) demonstrates its cross-model transferability. Additionally, the parameter search process supports parallel computation, enhancing computational efficiency.

Conclusion

We dive into the research question of why VLMs are more vulnerable by analyzing from the perspective of internal activations. We observe a significant difference in activation distributions between text-only and image-text inputs. Moreover, the probing experiment indicates that the safety alignment embedded in VLMs is not robust enough to handle this discrepancy. Inspired by the observations, we propose the internal activation revision method, steering the activations toward a safer direction. Our competitive results demonstrate the effectiveness of our approach.

Ethical Statement

This work is dedicated to exploring why VLMs are more vulnerable than LLMs and then proposing the internal activation revision method to safeguard VLMs. We firmly adhere to principles of respect and dignity for all people. The inclusion of offensive materials, including toxic corpus, harmful prompts, and model outputs, is exclusively for research purposes and does not represent the personal views or beliefs of the authors. We make every effort to minimize the toxic content to make the demonstration less offensive. In addition, we sample a portion of the data from existing datasets for our experiments, which may affect the accuracy of some of our conclusions.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and constructive feedback, which have significantly improved the quality of this work. This work was supported in part by the JST CRONOS Grant (No. JPMJCS24K8), the JSPS KAKENHI Grant (No.JP21H04877, No.JP23H03372, and No.JP24K02920), the Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada, and the Autoware Foundation.

References

Azaria, A.; and Mitchell, T. 2023. The Internal State of an LLM Knows When It's Lying. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, 967–976. Singapore: Association for Computational Linguistics.

Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; and Zhou, J. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 1(2): 3.

Chen, J.; Zhu, D.; Shen, X.; Li, X.; Liu, Z.; Zhang, P.; Krishnamoorthi, R.; Chandra, V.; Xiong, Y.; and Elhoseiny, M. 2023. MiniGPT-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*.

Chen, Z.; Wu, J.; Wang, W.; Su, W.; Chen, G.; Xing, S.; Zhong, M.; Zhang, Q.; Zhu, X.; Lu, L.; et al. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24185–24198.

Gong, Y.; Ran, D.; Liu, J.; Wang, C.; Cong, T.; Wang, A.; Duan, S.; and Wang, X. 2023. Figstep: Jailbreaking large vision-language models via typographic visual prompts. *ArXiv preprint*, abs/2311.05608.

Hudson, D. A.; and Manning, C. D. 2019. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 6700–6709. Computer Vision Foundation / IEEE.

Ji, J.; Qiu, T.; Chen, B.; Zhang, B.; Lou, H.; Wang, K.; Duan, Y.; He, Z.; Zhou, J.; Zhang, Z.; et al. 2023. Ai alignment: A comprehensive survey. *ArXiv preprint*, abs/2310.19852.

Jigsaw; and team, G. C. A. T. 2024. Conversation-AI. https://perspectiveapi.com/.

Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C. J.; Wexler, J.; Viégas, F. B.; and Sayres, R. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 2673–2682. PMLR.

Li, K.; Patel, O.; Viégas, F.; Pfister, H.; and Wattenberg, M. 2023. Inference-Time Intervention: Eliciting Truthful Answers from a Language Model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Li, Q.; Geng, J.; Lyu, C.; Zhu, D.; Panov, M.; and Karray, F. 2024. Reference-free Hallucination Detection for Large Vision-Language Models. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Findings of the Association for Computational Linguistics: EMNLP 2024*, 4542–4551. Miami, Florida, USA: Association for Computational Linguistics.

Liu, H.; Li, C.; Li, Y.; and Lee, Y. J. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 26296–26306.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023a. Visual Instruction Tuning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 34892–34916. Curran Associates, Inc.

Liu, X.; Zhu, Y.; Gu, J.; Lan, Y.; Yang, C.; and Qiao, Y. 2025. Mm-safetybench: A benchmark for safety evaluation of multimodal large language models. In *European Conference on Computer Vision*, 386–403. Springer.

Liu, X.; Zhu, Y.; Lan, Y.; Yang, C.; and Qiao, Y. 2023b. Query-relevant images jailbreak large multi-modal models. *ArXiv preprint*, abs/2311.17600.

Lu, P.; Mishra, S.; Xia, T.; Qiu, L.; Chang, K.-W.; Zhu, S.-C.; Tafjord, O.; Clark, P.; and Kalyan, A. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35: 2507–2521.

Nejadgholi, I.; Fraser, K.; and Kiritchenko, S. 2022. Improving Generalizability in Implicitly Abusive Language Detection with Concept Activation Vectors. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 5517–5529. Dublin, Ireland: Association for Computational Linguistics.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds.,

Advances in Neural Information Processing Systems, volume 35, 27730–27744. Curran Associates, Inc.

Pantazopoulos, G.; Parekh, A.; Nikandrou, M.; and Suglia, A. 2024. Learning To See But Forgetting To Follow: Visual Instruction Tuning Makes LLMs More Prone To Jailbreak Attacks. In Dinkar, T.; Attanasio, G.; Curry, A. C.; Konstas, I.; Hovy, D.; and Rieser, V., eds., *Proceedings of Safety4ConvAI: The Third Workshop on Safety for Conversational AI @ LREC-COLING 2024*, 40–51. Torino, Italia: ELRA and ICCL.

Pi, R.; Han, T.; Zhang, J.; Xie, Y.; Pan, R.; Lian, Q.; Dong, H.; Zhang, J.; and Zhang, T. 2024. MLLM-Protector: Ensuring MLLM's Safety without Hurting Performance. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 16012–16027. Miami, Florida, USA: Association for Computational Linguistics.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 53728–53741. Curran Associates, Inc.

Rimsky, N. 2023. Red-teaming language models via activation engineering. https://github.com/nrimsky/LM-exp/.

Rimsky, N.; Gabrieli, N.; Schulz, J.; Tong, M.; Hubinger, E.; and Turner, A. 2024. Steering Llama 2 via Contrastive Activation Addition. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15504–15522. Bangkok, Thailand: Association for Computational Linguistics.

Röttger, P.; Kirk, H.; Vidgen, B.; Attanasio, G.; Bianchi, F.; and Hovy, D. 2024. XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 5377–5400. Mexico City, Mexico: Association for Computational Linguistics.

Song, D.; Xie, X.; Song, J.; Zhu, D.; Huang, Y.; Juefei-Xu, F.; and Ma, L. 2024. LUNA: A Model-Based Universal Analysis Framework for Large Language Models. *IEEE Transactions on Software Engineering*.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Tenney, I.; Das, D.; and Pavlick, E. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4593–4601. Florence, Italy: Association for Computational Linguistics.

Turner, A.; Thiergart, L.; Udell, D.; Leech, G.; Mini, U.; and MacDiarmid, M. 2023. Activation addition: Steering language models without optimization. *ArXiv preprint*, abs/2308.10248.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 5998–6008.

Wang, Y.; Liu, X.; Li, Y.; Chen, M.; and Xiao, C. 2025. AdaShield : Safeguarding Multimodal Large Language Models from Structure-Based Attack viaAdaptive Shield Prompting. In Leonardis, A.; Ricci, E.; Roth, S.; Russakovsky, O.; Sattler, T.; and Varol, G., eds., *Computer Vision – ECCV 2024*, 77–94. Cham: Springer Nature Switzerland. ISBN 978-3-031-72661-3.

Zhang, Y.; Chen, L.; Zheng, G.; Gao, Y.; Zheng, R.; Fu, J.; Yin, Z.; Jin, S.; Qiao, Y.; Huang, X.; et al. 2024. SPA-VL: A Comprehensive Safety Preference Alignment Dataset for Vision Language Model. *ArXiv preprint*, abs/2406.12030.

Zhu, D.; Chen, D.; Li, Q.; Chen, Z.; Ma, L.; Grossklags, J.; and Fritz, M. 2024. PoLLMgraph: Unraveling Hallucinations in Large Language Models via State Transition Dynamics. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Findings of the Association for Computational Linguistics: NAACL 2024*, 4737–4751. Mexico City, Mexico: Association for Computational Linguistics.

Zhu, D.; Chen, J.; Shang, W.; Zhou, X.; Grossklags, J.; and Hassan, A. E. 2021. Deepmemory: model-based memorization analysis of deep neural language models. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 1003–1015. IEEE.

Zong, Y.; Bohdal, O.; Yu, T.; Yang, Y.; and Hospedales, T. 2024. Safety fine-tuning at (almost) no cost: A baseline for vision large language models. *ArXiv preprint*, abs/2402.02207.