Visual Description Grounding Reduces Hallucinations and Boosts Reasoning in LVLMs

Sreyan Ghosh[♠◆*] Chandra Kiran Evuru[♠*] Sonal Kumar[♠*] Utkarsh Tyagi[♠] Oriol Nieto[◆] Zeyu Jin[◆] Dinesh Manocha[♠]

[♠]University of Maryland, College Park, [◆]Adobe, USA

Understanding hallucinations in LVLMs

- **What are LVLMs?** Large Vision Language Models (LVLMs) like Qwen2-VL, are Large Language Models (LLMs) fine-tuned on text-image instruction-response pairs that enables them to take text-image pairs as input.
- **What is a hallucination?** Hallucination refers to the mismatch between factual content and the model's generated responses. Hallucination mitigation aims to reduce these discrepancies for more accurate outputs and improve task performance.

Visual Recognition $\xrightarrow[\text{Perception}]{}$ Knowledge Extraction (optional) → Reasoning (optional)

Hallucination mitigation techniques struggle to generalize beyond real-world scenes

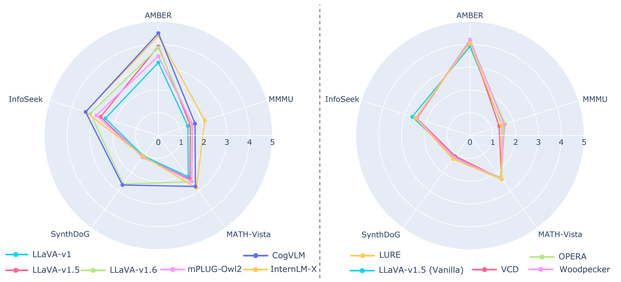


Figure 1. **(Left)** Performance comparison of LVLMs on common benchmarks. **(Right)** Performance comparison of mitigation techniques applied to LLaVA-1.5.

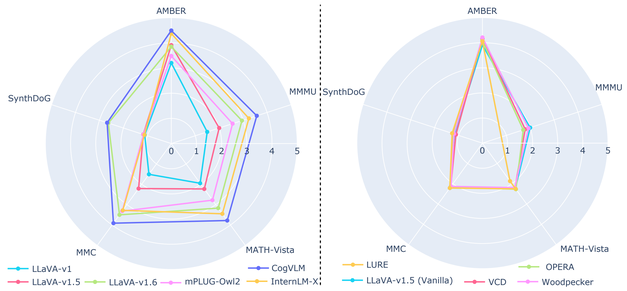


Figure 2. **(Left)** Performance comparison of LVLMs on benchmarks prompted for image descriptions. **(Right)** Performance comparison of mitigation techniques applied to LLaVA-1.5.

Types of Visual Understanding Hallucination

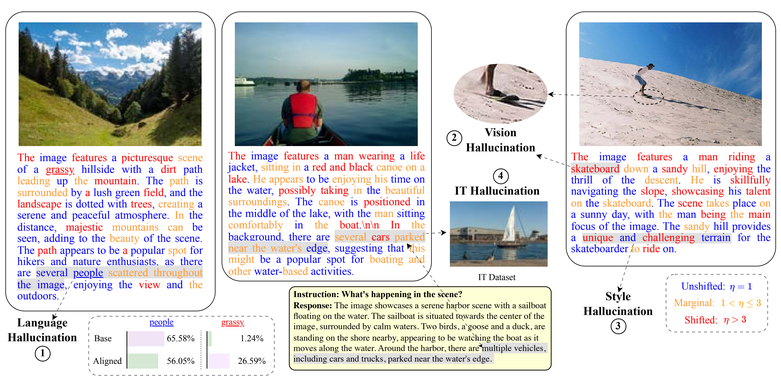


Figure 3. We propose an algorithm to categorize hallucinations into 4 different types.

The Visual Perception Gap: LVLMs can see but not perceive.

LVLMs often depend on language priors instead of fully attending to input images for reasoning tasks, leading to a critical issue: the **visual perception gap**. While they recognize visual elements and have the knowledge to respond factually, they struggle to interpret these elements in context, causing hallucinations and incorrect responses.

Figure 4. The Base Rank for AMBER (cap) is higher than Math-Vision (reason).

Models	Dataset	k	Variance	Range
LLaVA-v1	AMBER	1.1 / 4.2	0.0 / 0.1	0.0 / 0.3
LLaVA-v1.5	AMBER	1.0 / 3.9	0.0 / 0.1	0.0 / 0.4
CogVLM	AMBER	1.0 / 3.4	0.0 / 0.1	0.0 / 0.5
LLaVA-v1	MMMU	1.2 / 3.7	0.0 / 0.1	0.0 / 0.4
LLaVA-v1.5	MMMU	1.2 / 3.5	0.0 / 0.2	0.0 / 0.4
CogVLM	MMMU	1.1 / 3.2	0.0 / 0.2	0.0 / 0.5

Table 1. The probability space for hallucinated tokens is dominated by a set of low and equally confident tokens (non-/hallucinated format).

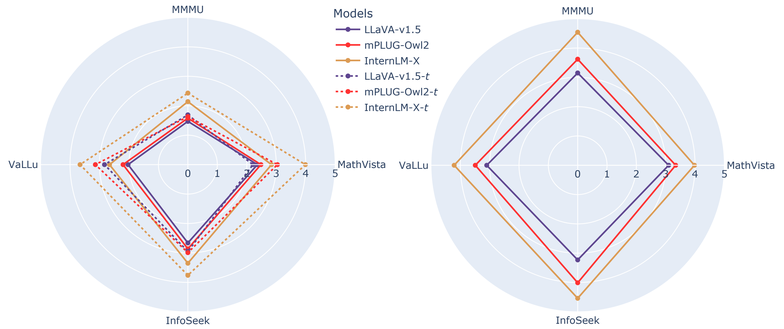
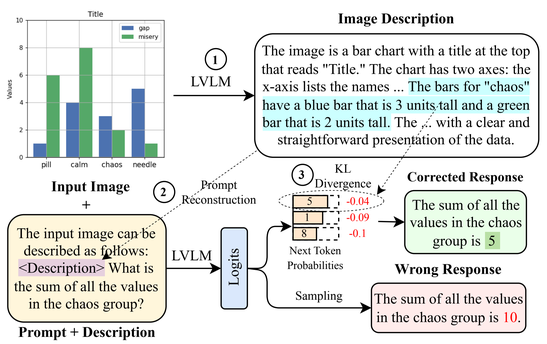


Figure 5. **(Left)** Performance comparison of LVLMs when prompted w/ original prompt vs rephrased prompts w/o image. **(Right)** Performance comparison of LVLMs for their ability to generate a faithful image description.

Visual Description Grounding Decoding (VDGD)

We propose **Visual Description Grounding Decoding (VDGD)**, a *training-free* method to reduce hallucinations for cognitive prompts. We prepend an image description to the prompt and, at each decoding step, select the token with the lowest KL divergence from the description.



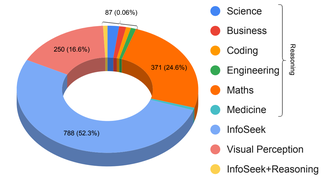
For each token in the top-K plausible tokens, we calculate the deviation with the prompt:

$$\mathrm{KL}_{w_k}^{x_i} = \min_{1 \le j \le n} \mathrm{KL}\left(\text{one-hot}(w_k) \| p_{\text{VLM}}(\cdot \mid x_{<j})\right).$$

We replace the logit of each token with its negative KL divergence from the input prompt, then apply softmax to obtain a probability distribution. This makes tokens with high deviation (large KL) less likely to be sampled.

VaLLu Benchmark

- We built the VaLLu by amalgamating **1500 non-noisy instances** from existing benchmarks. These benchmarks include Oven, MMMU, MMC, MathVista, HallusionBench, MATH-Vision, and MME.
- We focus on QAs that require **open-ended responses** (as opposed to MCQ) to help us in robust hallucination evaluation.



Performance comparison of VDGD with various baselines.

Benchmark	Baseline	LLaVA-v1	LLaVA-1.5	LLaVA-1.6	mPLUG-Owl2	InternLM-X	CogVLM	CogVLM2	Qwen2-VL
MMMU	Vanilla-greedy	1.26	1.35	1.42	1.40	2.01	1.66	2.08	2.39
	Vanilla-sampling	1.27	1.44	1.40	1.41	2.05	1.64	2.10	2.35
	VCD	1.34	1.52	1.44	1.53	2.22	1.68	2.14	2.64
	OPERA	1.30	1.43	1.57	1.64	2.25	1.62	2.21	2.44
	Woodpecker	1.32	1.44	1.63	1.61	2.12	1.65	2.14	2.56
	LRV	1.29	1.49	1.61	1.58	2.08	1.59	2.18	2.48
	LURE	1.31	1.47	1.60	1.64	2.27	1.66	2.27	2.71
	VDGD (ours)	1.49 (+18%)	1.62 (+20%)	1.75 (+23%)	1.72 (+23%)	2.39 (+19%)	1.71 (+3%)	2.47 (+19%)	2.91 (+22%)
MathVista	Vanilla-greedy	1.56	1.65	2.00	1.92	2.56	2.15	2.45	2.97
	Vanilla-sampling	1.54	1.68	1.91	1.94	2.52	2.19	2.46	2.95
	VCD	1.67	1.68	2.07	2.11	2.59	2.53	2.81	3.19
	OPERA	1.64	1.83	2.04	2.04	2.62	2.30	2.53	3.22
	Woodpecker	1.72	2.10	2.19	2.13	2.43	2.43	2.59	3.13
	LRV	1.68	1.68	2.17	1.99	2.61	2.20	2.62	3.30
	LURE	1.59	2.03	2.21	2.07	2.37	2.45	2.49	3.13
	VDGD (ours)	1.84 (+18%)	2.19 (+33%)	2.44 (+22%)	2.24 (+17%)	2.88 (+13%)	2.62 (+22%)	2.93 (+20%)	3.59 (+21%)
MME	Vanilla-greedy	3.32	3.54	3.65	3.49	3.75	3.57	3.64	3.86
	Vanilla-sampling	3.34	3.53	3.62	3.48	3.77	3.58	3.66	3.85
	VCD	3.46	3.61	3.77	3.59	3.96	3.67	3.95	4.21
	OPERA	3.42	3.59	3.82	3.54	3.93	3.60	3.88	4.09
	Woodpecker	3.37	3.55	3.76	3.63	3.82	3.59	3.92	4.15
	LRV	3.43	3.63	3.78	3.52	3.95	3.61	3.96	3.99
	LURE	3.42	3.62	3.87	3.67	3.93	3.72	3.77	4.10
	VDGD (ours)	3.59 (+8%)	3.70 (+5%)	3.99 (+9%)	3.82 (+9%)	4.12 (+10%)	3.79 (+6%)	4.09 (+12%)	4.34 (+12%)
VaLLu	Vanilla-greedy	1.95	2.03	2.63	2.20	2.66	2.82	3.23	3.45
	Vanilla-sampling	1.86	2.01	1.64	2.18	2.70	2.83	3.21	3.40
	VCD	1.47	1.55	1.80	1.80	2.32	2.63	3.19	3.42
	OPERA	2.04	2.05	2.62	2.26	2.65	2.85	3.22	3.47
	Woodpecker	2.01	2.05	2.67	2.23	2.60	2.91	3.28	3.52
	LRV	1.98	2.10	2.65	2.19	2.59	2.88	3.32	3.51
	LURE	2.03	2.03	2.64	2.24	2.64	2.78	3.28	3.48
	VDGD (ours)	2.16 (+11%)	2.64 (+30%)	3.16 (+20%)	2.72 (+24%)	3.45 (+30%)	3.01 (+7%)	3.48 (+8%)	3.67 (+6%)

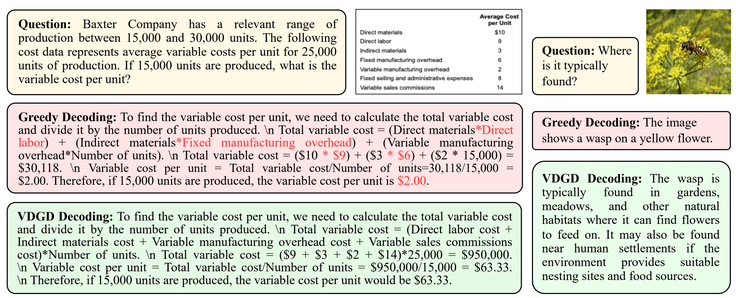


Figure 6. Comparison of responses from vanilla greedy decoding with VDGD.

Future Work

- **Handling for inaccurate descriptions:** We will handle the error accumulation in VDGD that comes from inaccurate image descriptions in the prefix.
- **Compute efficiency:** We also want to make VDGD more efficient as it requires two passes currently.
- **Application to reasoning models:** VDGD will be extremely useful to o1-style models that think before responding. We would like to explore effective ways to integrate the same.

Figure 7. Project page.