

VC-IF: Introducing Visual-Constraint Instruction Following for Multi-Turn, Multi-Constraint Vision-Language Models

Jixiang Hong^{1*} Pi Bu^{2*} Yingyao Wang² Bo Peng² Peijie Wang³ Chen Wang²
Tengtao Song² Yang Yao² Jun Song² Rui Yan^{4†} Cheng Yu² Bo Zheng²

¹Gaoling School of Artificial Intelligence, Renmin University of China

²Future Living Lab, Alibaba Group ³Institute of Automation, Chinese Academy of Sciences

⁴School of Artificial Intelligence, Wuhan University

jxhong@ruc.edu.cn bupi.wj@alibaba-inc.com

wangpeijie2023@ia.ac.cn rui.yan@whu.edu.cn

Abstract

Large vision-language models (LVLMs) have made rapid advancements; however, existing benchmarks for evaluating their instruction-following abilities focus on purely textual constraints, e.g., format constraints and role-playing, which assess only language-level adherence and are not related to vision. To bridge this gap, we introduce VC-IF, a benchmark that involves **visual constraints** connecting instructions to explicit operations over image content, thereby directly probing visual spatial understanding and compositional compliance. VC-IF defines atomic constraints, systematically combining mixed settings, scaling from single- to multi-constraint scenarios, and single- and multi-turn scenarios. We evaluate 16 popular LVLMs and find that models performing well under text-only constraints may struggle when faced with visual constraints, highlighting significant gaps in their capabilities. Further, we conduct scaling analyses across varying numbers of constraints and dialogue turns, revealing performance trends as constraint complexity increases along different dimensions. Finally, we synthesize a large training dataset and propose VC-PO, a reinforcement learning strategy that uses a hybrid verifier mechanism to compute reward signals for multiple constraints. Applying VC-PO to Qwen3-VL-8B-Instruct improves its overall VC-IF score from 43.74 to 55.18 (+11.44). The VC-PO-trained model remains stable as the number of turns and constraints increases. Our data and code are available at [VC-IF](#).

1 Introduction

Large Vision-Language Models (LVLMs) have been evaluated for various capabilities (Bai et al., 2025a), such as general (Fu et al., 2025; Liu et al.,

2024a), knowledge (Gu et al., 2025; He et al., 2024), document (Mathew et al., 2021; Deng et al., 2025), reasoning (Chen et al., 2024; Wang et al., 2024; Zhang et al., 2024) (including spatial reasoning (Li et al., 2026; Huang et al., 2025)), and agent (Xu et al., 2025; Chen et al., 2025a). Model alignment (Rafailov et al., 2023; Hong et al., 2024; Song et al., 2026; Hong et al., 2026) and instruction following, as evaluated by benchmarks such as IFEval (Zhou et al., 2023) and MIA-Bench (Qian et al., 2025), have received considerable attention.

However, existing benchmarks (Ding et al., 2025) for evaluating “instruction following” typically construct hard cases using purely textual constraints. While these designs can reveal a model’s ability to generate text that satisfies linguistic constraints (Luo et al., 2024) (e.g., “produce at least 200 words”), they are essentially decoupled from visual understanding: a model can meet the constraint in text space without demonstrating perception, localization, or reasoning over the visual content. However, many real-world applications demand visually grounded constraints, such as content moderation (e.g., “ignore watermarks when describing the image”), accessibility assistance (e.g., “focus only on text for visually impaired users”), and privacy-preserving analysis (e.g., “describe while ignoring faces”). Figure 1a illustrates the distinction between predominantly textual instruction-following constraints and visually grounded constraints over image content.

To fill this research gap, we introduce **VC-IF** (Visual-Constraint Instruction-Following), a benchmark involving “visual constraints”. The core idea of VC-IF is to extend constraints from the textual domain into the visual domain, so that constraints include not only linguistic requirements but also explicit references to and operations on targets within images (e.g., “describe only the

*Equal contribution.

†Corresponding author.

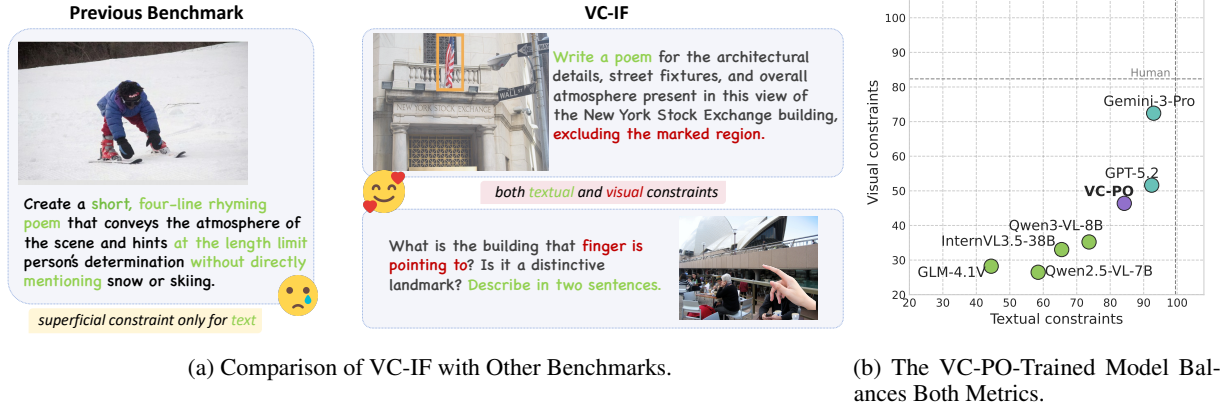


Figure 1: (a) Existing benchmarks mostly evaluate textual constraints, whereas VC-IF focuses on visual constraints for a more comprehensive evaluation of multimodal instruction following. (b) Performance on textual constraints does not necessarily align with performance on visual constraints, while the VC-PO-trained model achieves a more balanced profile across both dimensions.

content within the circled region”). Specifically, our VC-IF benchmark design emphasizes visual constraints and offers a detailed scaling analysis through extensive experiments with multiple constraints. **i) Atomic Constraints:** defines 36 types of atomic constraints, comprising 32 textual constraints (e.g., style, length, required mentions) and 4 visual constraints (e.g., spatial region, symbolic reference, spatial order, and referential language phrase). **ii) Construction Pipeline:** develops a pipeline to create visual-constraint data for scaling analysis. After quality filtering, we obtained a benchmark of 1,200 samples in three task categories: visual perception, compositional reasoning, and content generation, with single-constraint, multi-constraint, and multi-turn modes. **iii) Evaluation Protocol:** evaluates cases from single constraints to combinations (text-only, visual-only, cross-modal mixed) and tests scalability with up to 8 turns or 15 constraints. Also examines the impact of reasoning on instruction-following under complex constraint compositions.

Our evaluation of 16 popular LVLMs reveals that strong performance on text-only instruction-following does not necessarily transfer to visually constrained tasks. Many models struggle with visual elements, especially as the number of constraints and dialogue turns increases. This highlights the importance of considering visual constraints to improve LVLMs’ multimodal abilities.

Finally, we propose **VC-PO**, a GRPO-based reinforcement learning strategy that uses constraint-specific verifiers to provide reward signals for both textual and visual instruction following. Apply-

ing VC-PO to Qwen3-VL-8B-Instruct improves its overall VC-IF score from 43.74 to 55.18, an absolute gain of 11.44 points. As shown in Figure 1b, the VC-PO-trained model achieves a more balanced profile across textual and visual constraints. Our main contributions are:

- **VC-IF Benchmark.** We introduce VC-IF, a multi-turn, multi-constraint instruction-following benchmark and evaluation protocol focused on visual constraints.
- **Scaling Analysis of Constraints.** We explore a scaling analysis that evaluates scalability from one to multiple constraints (text-only, visual-only, mixed) and scalability with up to 8 turns or 15 constraints.
- **VC-PO.** We build a pipeline for large-scale data synthesis and construct a training dataset of 9.5K samples. We then optimize Qwen3-VL-8B-Instruct with VC-PO using constraint-wise verifier rewards, improving its overall VC-IF score from 43.74 to 55.18, an 11.44 points gain.

2 Related Work

Instruction Following Benchmarks for LLMs. Numerous benchmarks have been developed to evaluate the instruction-following capabilities of Large Language Models (LLMs). FollowBench (Jiang et al., 2024) introduces a detailed framework focusing on five constraint types, including content, situation, style, format, and example, to evaluate models on complex, real-world instructions. ComplexBench (Wen et al., 2024)

presents a hierarchical model of instructions (four constraint types, nineteen dimensions, and four composition types) to test LLMs under varied constraints. IFEval (Zhou et al., 2023) highlights strict format and structure compliance, revealing that even large LLMs often struggle with simple instructions in zero-shot scenarios. Recent work further studies complex instruction following from both training and evaluation perspectives: Constraint Back-Translation (Qi et al., 2025b) and P-Aligner (Song et al., 2026) develop instruction-synthesis strategies for improving complex instruction following and preference alignment, respectively, while AGENTIF (Qi et al., 2025a) benchmarks instruction-following ability in agentic scenarios.

Instruction Following Benchmarks for LVLMS.

With the rise of Large Vision–Language Models (LVLMS) (Bai et al., 2025b; Chen et al., 2025b), several benchmarks have been proposed to assess their ability to follow instructions with a multimodal context. MIA-Bench (Qian et al., 2025) provides 400 image–prompt pairs designed to probe models’ ability to follow layered, fine-grained instructions and to quantify execution accuracy and consistency. MM-IFEval (Ding et al., 2025) is a challenging and diverse multimodal instruction-following benchmark that encompasses two levels of constraints: compositional-level constraints on the output response and perceptual-level constraints related to the input image. Additionally, it features a comprehensive evaluation process that combines rule-based evaluation with judgment models. MCIF (Papi et al., 2025) is a multilingual, human-annotated benchmark based on scientific talks that spans speech, vision, and text across four languages to evaluate crosslingual and multimodal instruction following. Collectively, these efforts reveal persistent gaps in instruction adherence, contextual alignment, and multimodal fusion when models face diverse and compositional directives.

Benchmark Comparison. VC-IF extends constraints into the visual domain to require not only linguistic requirements but also explicit references to and operations on targets within images. The differences between our approach and other benchmarks are presented in Table 1. VC-IF features the highest number of evaluations (i.e., 1,200) and the most diverse set of constraints (i.e., 36), covering single-constraint, multi-constraint, and multi-turn scenarios under mixed textual and visual con-

Table 1: Comparison of our VC-IF benchmark with existing instruction-following benchmarks for LVLMS.

Attribute	MIA-Bench	MM-IF	VC-IF
# Sample	400	400	1,200
# Constr. Types	8	32	36
Text Constr.	✓	✓	✓
Visual Constr.	✗	✗	✓
Single-Turn	✓	✓	✓
Multi-Turn	✗	✗	✓
Bilingual	EN	EN & ZH	EN & ZH

straints. Furthermore, we analyze scaling performance with varying constraints, and conversation turns to reveal how performance trends shift with increased constraint complexity.

We further distinguish VC-IF from spatial grounding and region-understanding benchmarks such as GAR-Bench (Wang et al., 2026) and RefSpatialBench (Zhou et al., 2025). These benchmarks evaluate important capabilities such as region-conditioned understanding, multi-region reasoning, and spatial referring. In contrast, VC-IF focuses on visually grounded requirements as explicit instruction-following constraints that interact with textual constraints. By systematically defining visual constraints, composing them with textual constraints, and scaling evaluation to multi-constraint and multi-turn settings, VC-IF complements prior grounding benchmarks with a focus on complex multimodal instruction following. A feature-level comparison is provided in Appendix B.2.

3 VC-IF Construction

The construction pipeline of our VC-IF benchmark is illustrated in Figure 2. The pipeline includes: *task instantiation*, *atomic constraint integration*, and *constraints composition*.

Formally, we define instructions as consisting of a general question and one or more constraints, i.e., $I_t = f(Q; A_t)$, where t denotes the round number, ranging from one to many. A is a combination of multiple atomic constraints, defined as $A_t = f(a_1, a_2, \dots)$, including textual constraints a^t and visual constraints a^v . In summary, our instructions are a set of multi-turn, multi-constraint elements to thoroughly explore the capabilities of existing models.

3.1 Task Instantiation

Source Collection We source images from the publicly available dataset, MAMmoTH-VL (Guo et al., 2025), which collects from 118 open-source

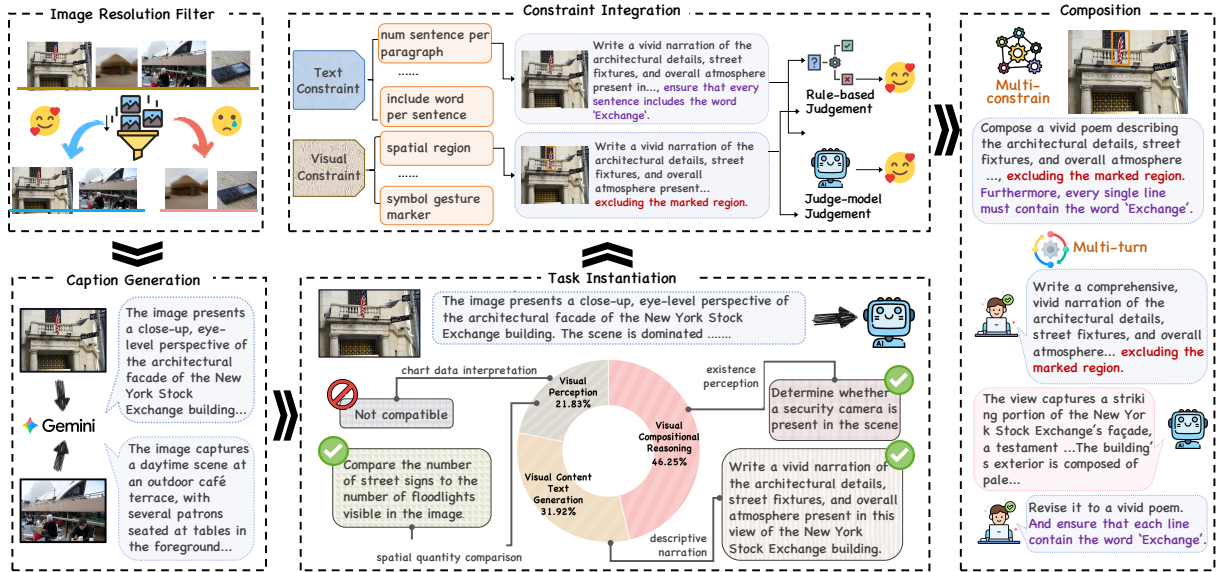


Figure 2: Overview of the VC-IF benchmark construction pipeline. The construction process consists of three main steps: *task instantiation*, *atomic constraint integration*, and *constraints composition*.

datasets across various domains. We first collect images from the dataset, filter out low-resolution ones, and further remove visually simple images based on image-complexity scores estimated by IC9600 (Feng et al., 2022), so as to retain images with sufficient visual content for constructing meaningful visual constraints. This yielded a high-quality image set, from which we randomly sampled 10,000 images as the starting point for our data construction. This source-image pool was used to construct both the VC-IF benchmark and the VC-PO training data; after finalizing the benchmark, we excluded all images used by VC-IF before constructing the training set, ensuring no image overlap between them. Then, we utilize Gemini-2.5-Pro to generate a detailed description for each image.

Task Definition We define three categories of vision-language instruction types:

- **Visual Perception:** focusing on direct interpretation of visual content, such as object existence, attribute, object counting, spatial relationship, OCR, etc.
- **Compositional Reasoning:** requiring multi-hop reasoning over visual elements, e.g., deducing spatial relations between objects, interpreting occlusions, or combining OCR text with scene content.
- **Vision-based Text Generation:** where the model produces textual output conditioned on

visual information, such as detailed captioning, dialogue completion, or poetry writing.

Within each category, the specific types of tasks are carefully designed to cover a wide range of challenges and complexities. The detailed taxonomy of task types and their specific forms are provided in the Appendix B.

3.2 Atomic Constraint Integration

After instantiating the base tasks, we increase complexity by integrating *atomic constraints* into the instructions. Similar to MM-IFEval (Ding et al., 2025) and MIA-Bench (Qian et al., 2025), we establish textual constraints for answer formats, content limitations, and stylistic guidelines. However, textual constraints alone are insufficient for thoroughly evaluating the model’s capabilities in visual perception and reasoning. To address this, we introduce *visual constraints*, requiring the model to focus on or disregard specific image regions, objects, or attributes during its responses.

Atomic Textual Constraint. The atomic textual constraints are categorized into several groups, including: Length, Format, Content, Style, Rhetorical, and Language.

Atomic Visual Constraint. We define four types of atomic visual constraints: spatial region, symbolic reference, spatial order, and referential language phrase. For the former two types, we further divide them into two modes: focus and exclusion,

where the model is instructed to either concentrate on or disregard specific visual content. The examples can be found in the Figure 2 and more details are in the Appendix D.

Constraint Verifier. For each atomic constraint type, we design a corresponding verification function to assess whether a response adheres to the given constraint. 1) The rule-based functions are primarily used for straightforward constraints, such as length and format constraints, while 2) the LVLM is employed as a judge for more complex constraints that require nuanced understanding, such as content and visual constraints. The detailed designs and mapping between atomic constraints and verification functions can be found in Appendix D. Table 14 lists the constraints handled by deterministic verifier functions; the remaining semantic and visually grounded constraints are evaluated by the LVLM judge.

Simultaneously, we specify the arguments for the verification function associated with each atomic constraint to facilitate subsequent evaluation. After this stage, each base task is augmented with textual and visual atomic constraints, resulting in a set of constrained instructions.

3.3 Constraints Composition

To further enhance the diversity and complexity of the visual instructions, we proceed to compose and combine multiple atomic constraints into more intricate tasks. This composition process involves two strategies: *multiple constraints merging* and *multi-turn constraints combination*.

For multiple constraints merging, we begin with a single-constraint instruction and iteratively integrate additional atomic constraints. At each iteration, we assess the compatibility of the new constraint with the existing instruction and ensure that the merged instruction remains coherent and solvable. This iterative merging continues until no further compatible constraints can be added.

For multi-turn constraints combination, we design a sequence of instructions where each turn introduces one or two new atomic constraints. The multi-turn setup allows us to simulate more realistic interaction scenarios, where users may progressively refine or complicate their requests. At each turn, we format the newly introduced constraints a_k as a follow-up instruction, building upon the previous instruction A_{t-1} .

$$A_t = f(A_{t-1}; a_k, \dots), k \in \{a_t, a_v, a_m\}, \quad (1)$$

where t denotes the round number. A_t is a combination of multiple atomic constraints, including textual constraints a_t , visual constraints a_v , and mixed constraints a_m .

Through these composition strategies, we synthesize a rich set of complex visual instructions that challenge models to handle multiple constraints simultaneously and adapt to evolving instruction contexts.

3.4 Quality Control

At each stage of the construction pipeline, we implement rigorous quality control mechanisms to ensure the high quality and diversity of the resulting visual instructions. We engaged 20 college-educated annotators to refine our benchmark by assessing: 1) solvability of problems; 2) potential conflicts across multiple constraints and rounds; 3) accuracy of constraint verifier functions; and 4) typographical errors. All annotators underwent three trial rounds to familiarize themselves with the task, only proceeding to official annotations upon approval. Each candidate was independently checked by at least two annotators, with disagreements resolved by a third adjudicator before retention or refinement.

3.5 Dataset Statistics

After applying the rigorous quality control, we ultimately obtained 1,200 samples to evaluate instruction compliance under visual constraints. Table 2 illustrates the distribution of our benchmark.

Task Category	#Samples	Proportion
Visual Perception	262	21.83%
Compositional Reasoning	555	46.25%
Vision-based Text Generation	383	31.92%
Composition Mode		
Single-Constraint	628	52.33%
Multi-Constraint	359	29.92%
Multi-Turn	213	17.75%
Total	1,200	100%

Table 2: Benchmark statistics of our VC-IF.

4 VC-PO Method

We introduce **VC-PO**, a reinforcement learning method designed to enhance a model’s ability to follow instructions that include both textual and visual constraints by utilizing **verifiable reward signals**.

In our pipeline, each constraint in the instruction is associated with a corresponding verifier, which provides a fine-grained reward signal for each response. Under conditions with multiple constraints, the model’s responses are evaluated by all verifiers to determine the final reward. This design encourages the model to gradually explore and improve its ability to satisfy all constraints simultaneously in pursuit of higher scores.

Specifically, we adopt the GRPO (Shao et al., 2024; DeepSeek-AI, 2025) algorithm and sample 16 rollouts as the group per prompt for advantage estimation. We conduct the training with ROLL (Wang et al., 2025a) framework, where megatron is used for optimization and vLLM (Kwon et al., 2023) is used for efficient sampling. Training starts directly from Qwen3-VL-8B-Instruct without a preceding SFT stage; the same-data SFT baseline is trained separately for comparison. The overall reward R for each response is computed as the average score from all constraint verifiers of atomic constraints A_t , including Python functions and LVLM-as-a-judge¹:

$$R = \frac{1}{|A_t|} \sum_{a_i \in A_t} r_i, \quad r_i \in \{0, 1\}, \quad (2)$$

5 Experiment

5.1 Evaluation Setup

Baselines. We evaluate a diverse set of representative LVLMs, including both open-source and proprietary models. **GPT family**²: GPT-5.2, GPT-5-Chat, GPT-5-Mini, GPT-4o. **Gemini family** (Gemini Team et al., 2024): Gemini-3-Pro, Gemini-3-Flash, Gemini-2.5-Pro, Gemini-2.5-Flash. **Qwen family**³: Qwen3-VL-235B-A22B-Instruct, Qwen3-VL-32B-Instruct, Qwen3-VL-8B-Instruct; Qwen-2.5-VL-72B-Instruct, Qwen-2.5-VL-7B-Instruct. **Other open-source models**: GLM-4.1V-9B-Base⁴, InternVL3.5 series (Wang et al., 2025b).

Evaluation Metrics. We use instruction-following rate to assess model performance with a hybrid evaluator. Deterministic Python functions handle formally checkable constraints, whereas Gemini-3-Flash is the default LVLM judge for

semantic or visually grounded verifiers; all verifier descriptions are manually checked and revised before evaluation. Appendix B.6 reports the human–judge consistency protocol and results, and Appendix C.2 reports robustness results obtained by replacing the default judge with GPT-5.2.

5.2 Benchmark Results

We present the evaluation results of various open-source and proprietary LVLMs on the VC-IF benchmark in Table 3, and have the following findings:

- Closed-source models generally surpass open-source ones. Gemini-3-Pro achieves the best overall performance. Except for GPT-4o, all proprietary models outperform or match the best open-source model, Qwen3-VL-235B-A22B.
- Within the same model family or series, larger models typically yielded better performance. For instance, in the Qwen3-VL series, the overall performance improved from 43.74% to 54.91% as the model size increased from 8B to 235B.
- Text-only instruction-following does not necessarily transfer to visually constrained tasks, and performance degrades further as constraint count and dialogue turns increase.
- Multi-turn settings pose an additional challenge: models generally score lower than in single-turn multi-constraint settings, indicating difficulty in maintaining accumulated constraints across turns.
- Qwen3-VL-8B + VC-PO improves over the same Qwen3-VL-8B-Instruct backbone from 43.74 to 55.18 overall, a gain of 11.44 points. This improvement demonstrates the effectiveness of constraint-wise verifiable rewards without implying universal superiority over stronger models.

5.3 Scaling Analysis of Constraints

We further investigate the model performance under scaling of constraints, including multi-constraint and multi-turn scenarios.

Multi-constraint scaling. We evaluate the models on varying numbers of constraints within a single turn, investigating how the models’ performance changes as the number of constraints increases. The results can be found in Figure 3. 1) We observe a clear trend of decreasing performance as

¹The detailed definitions of all the verifier functions are provided in the Appendix D.

²<https://openai.com/index/>

³<https://help.aliyun.com/zh/model-studio/developer-reference/use-qwen-by-calling-api>

⁴<https://huggingface.co/zai-org/GLM-4.1V-9B-Base>

Table 3: Leaderboard of LVLMs on VC-IF benchmark. Both representative open-source and proprietary models are evaluated. The best results in each group are highlighted in **bold**.

Models	#Param.	Single Constr.		Multiple Constr.		Multi-turn Constr.		Overall
		Textual	Visual	Mixed	Textual	Mixed	Textual	
Human	–	99.50	82.35	78.95	84.21	89.47	94.74	88.20
<i>Proprietary Models</i>								
Gemini-3-Pro	–	92.98	72.43	80.34	86.79	69.09	76.96	79.77
Gemini-3-Flash	–	95.91	70.02	79.73	80.69	67.23	77.92	78.58
Gemini-2.5-Pro	–	94.74	66.96	71.56	72.76	56.00	61.47	70.58
Gemini-2.5-Flash	–	85.96	60.61	64.56	68.19	54.56	62.88	66.13
GPT-5.2	–	92.40	51.64	72.13	74.70	58.91	60.32	68.35
GPT-5-Chat	–	90.06	50.33	59.77	62.60	43.08	46.38	58.70
GPT-5-Mini	–	92.98	50.77	79.79	83.84	50.47	47.68	67.59
GPT-4o	–	85.38	37.32	42.76	44.21	42.21	45.76	49.61
<i>Open-source Models</i>								
GLM-4.1V-Base	9B	44.44	28.23	30.28	29.98	13.04	13.28	26.54
InternVL3.5-8B	8B	60.23	25.82	32.47	32.32	28.38	28.48	34.62
InternVL3.5-38B	38B	65.50	33.04	34.28	36.18	34.02	34.72	39.62
Qwen2.5-VL-72B-Instruct	72B	81.07	40.80	52.90	53.15	35.71	35.80	49.91
Qwen2.5-VL-7B-Instruct	7B	58.48	26.48	32.48	33.84	25.94	21.45	33.11
Qwen3-VL-235B-A22B-Instruct	235B	82.25	52.11	55.18	56.30	42.19	41.44	54.91
Qwen3-VL-32B-Instruct	32B	78.36	51.20	45.78	48.98	33.09	32.64	48.34
Qwen3-VL-8B-Instruct	8B	73.68	35.23	43.67	45.73	32.46	31.68	43.74
+ VC-PO (Ours)	8B	84.21	46.39	54.11	56.71	42.46	47.20	55.18

Table 4: General utility and safety evaluation of Qwen3-VL-8B before and after VC-PO training.

Model	MMMU	MMBench	MMStar	MMT-Bench	AI2D	OCRBench	MMAlign	POPE	MLLMGuard -ASD↓
Qwen3-VL-8B	57.44	85.48	66.07	65.97	81.88	895	74.21	88.07	27.24
+ VC-PO	59.67	86.25	69.40	67.25	80.83	894	83.33	88.03	27.21

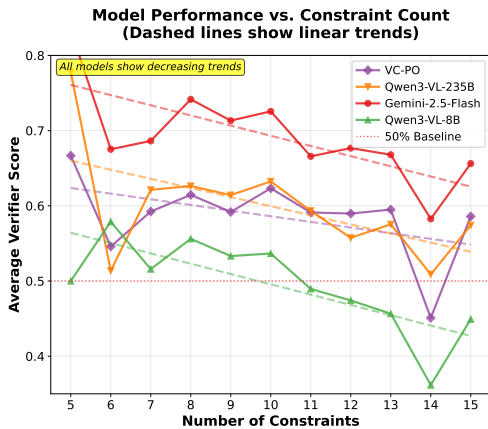


Figure 3: Performance with multi-constraint scaling.

the number of constraints increases, indicating the challenge of accommodating all constraints. 2) The VC-PO-trained Qwen3-VL-8B consistently outperforms its Qwen3-VL-8B-Instruct backbone across different numbers of constraints and approaches the performance of the much larger Qwen3-VL-235B-A22B model in this analysis.

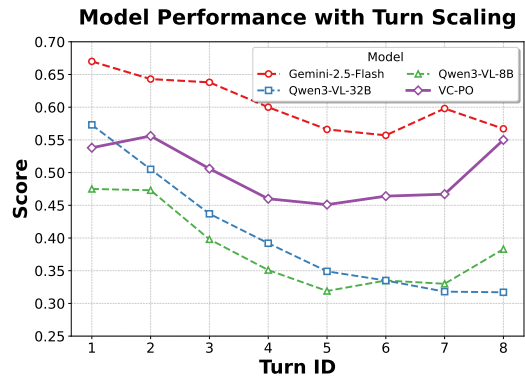


Figure 4: Performance with multi-turn scaling.

Multi-turn scaling. We assess the models’ ability to handle constraints that are introduced progressively over multiple turns of interaction, analyzing how well the models can maintain and integrate constraints across turns. Specifically, for each task, we design a sequence of turns in a conversation, where each turn introduces two additional constraints. We then evaluate the models’ performance

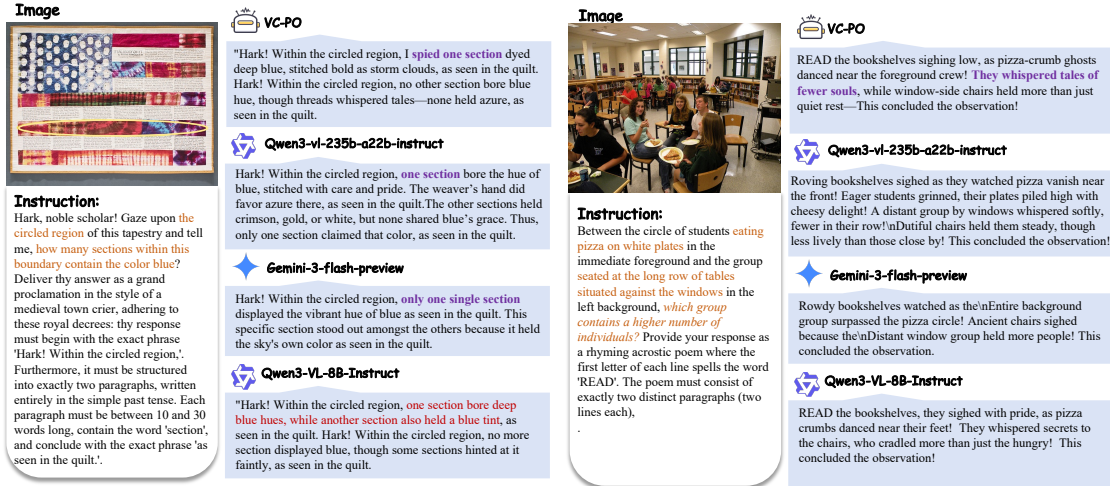


Figure 5: Case study of the models’ responses to the visual-constrained instruction in the VC-IF benchmark.

at each turn to observe how effectively they can manage the increasing complexity of constraints over the course of the interaction. The results are shown in Figure 4. We observe that 1) the model’s performance gradually decreases with more rounds, indicating that maintaining accuracy across multiple rounds is challenging due to the impact of growing context length on performance. 2) the VC-PO-trained Qwen3-VL-8B outperforms its base checkpoint and continues to maintain strong performance as the level of difficulty and context length increase. It also remains relatively stable as the number of turns scales up, showing a trend similar to that of Gemini-2.5-Flash.

5.4 Discussion on Reasoning

We analyze the impact of reasoning before answering on multi-constraint instruction-following performance by comparing direct answering versus reasoning-before-answering across three model families. As shown in Figure 6, reasoning consistently yields substantial improvements (around 10%), demonstrating its effectiveness in enhancing constraint adherence under complex compositions.

5.5 Results of VC-PO Training

Instruction-following benchmarks. We evaluate the VC-PO-trained Qwen3-VL-8B on VC-IF and existing instruction-following benchmarks using the publicly available VLMEvalKit tool⁵. As shown in Table 5, the trained model improves over the same Qwen3-VL-8B-Instruct backbone on MIA-Bench, MM-IFEval, and IFEval, indicating that training with complex visual constraints

also benefits simpler instruction-following tasks.

General utility and safety. We further evaluate whether VC-PO training harms general multimodal capabilities. Table 4 reports results on general utility benchmarks, including MMMU (Yue et al., 2024), MMBench (Liu et al., 2024a), MM-Star (Chen et al., 2024), MMT-Bench (Ying et al., 2024), and AI2D (Kembhavi et al., 2016), as well as document understanding, preference alignment, hallucination, and safety benchmarks, including OCRBench (Liu et al., 2024b), MMA-alignBench (Zhao et al., 2025), POPE (Li et al., 2023), and MLLMGuard-ASD (Gu et al., 2024). The VC-PO-trained model improves performance on most general benchmarks and substantially improves MMA-alignBench. Meanwhile, OCRBench and POPE remain stable, and MLLMGuard-ASD does not increase, suggesting that the VC-IF gains do not come at the cost of document understanding, hallucination, or safety.

5.6 Case Study

We present several cases to illustrate the strengths and weaknesses of different models, as shown in Figure 5. In the left case, with a circled region constraint, there are two “blue” sections in the image, but only one is circled. The VC-PO-trained model accurately identifies and counts the circled blue section, showing its capability in handling visual constraints. The Qwen3-VL-8B baseline, however, mistakenly counts both blue sections. In the right case, the model must first recognize the two groups of people by visual features to answer the related question. The VC-PO-trained model correctly identifies both groups and provides the

⁵<https://github.com/open-compass/VLMEvalKit>

right answer.

Additional ablation results on SFT, GSPO, and frozen-vision-model variants are provided in Appendix C.1.

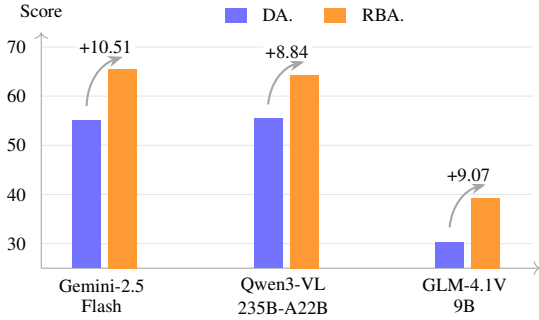


Figure 6: Impact of reasoning before answering on multi-constraint instruction-following performance. “DA.” indicates direct answering; “RBA.” refers to reasoning before answering.

Table 5: Evaluation Results on MIA-Bench, MM-IFEval and IFEval. “Pr.” and “Ins.” represent prompt-level accuracy and instruction-level accuracy.

Model	MIA	MM-IFEval	IFEval	
			Pr.	Ins.
Qwen3-VL-8B	94.59	68.12	87.25	91.37
+ VC-PO	95.47	70.71	89.65	93.17

6 Conclusion

In this paper, we introduced VC-IF, a benchmark that defines visual atomic constraints and composes them with textual constraints across single-constraint, multi-constraint, and multi-turn settings. Evaluations of 16 LVLMs show that text-only instruction-following does not reliably transfer to visually grounded constraints. We further proposed VC-PO, a reinforcement learning strategy that leverages constraint-wise verifier scores as reward signals, substantially improving visual instruction-following performance while preserving general utility and safety. We hope that VC-IF and VC-PO can facilitate future research on systematic evaluation, data construction, and optimization for multimodal instruction following.

Limitations

While VC-IF fills an important gap by introducing visual constraints into complex instruction-following tasks, it has several limitations. First, VC-IF covers a limited set of visual atomic constraints over static images and does not address video, 3D scenes, or temporal reasoning. Second, its images and constraint compositions can-

not cover the full diversity of real-world domains and may introduce biases that affect generalization. Third, the hybrid automatic-human evaluation pipeline remains subject to metric and judgment noise. Fourth, the focus on explicit, operation-style instructions may underrepresent open-ended or creative multimodal tasks.

VC-PO also has limitations: its binary, equally averaged constraint rewards do not model differences in constraint difficulty or interactions, and the main training study uses only one 8B backbone. Moreover, despite disjoint training and evaluation images and positive results on external benchmarks, using the same instruction-generation framework may still introduce template-specific adaptation. Future work should explore adaptive rewards, additional backbones, and more diverse generation pipelines.

Ethical Considerations

All authors confirm that this work complies with the ACL Code of Ethics. The authors declare no conflicts of interest throughout the research process. In addition, we disclose any remaining limitations, selected prompt templates and excerpts, and evaluation details in the Appendix to enhance transparency and reproducibility.

Acknowledgments

This work was supported in part by the Key-Area Research and Development Program of Guangdong Province (2024B0101040008).

References

- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. [Qwen3-vl technical report](#). *Preprint*, arXiv:2511.21631.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. [Qwen2.5-vl technical report](#). *Preprint*, arXiv:2502.13923.
- Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao

- Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. 2025a. [Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding](#). *Preprint*, arXiv:2406.10819.
- Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. 2024. [Are we on the right way for evaluating large vision-language models?](#) *Preprint*, arXiv:2403.20330.
- Peng Chen, Pi Bu, Yingyao Wang, Xinyi Wang, Ziming Wang, Jie Guo, Yingxiu Zhao, Qi Zhu, Jun Song, Siran Yang, Jiamang Wang, and Bo Zheng. 2025b. [Combatvla: An efficient vision-language-action model for combat tasks in 3d action role-playing games](#). *Preprint*, arXiv:2503.09527.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and Cheng-Lin Liu. 2025. [Longdocurl: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating](#). *Preprint*, arXiv:2412.18424.
- Shengyuan Ding, Shenxi Wu, Xiangyu Zhao, Yuhang Zang, Haodong Duan, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. 2025. [Mm-ifengine: Towards multimodal instruction following](#). *Preprint*, arXiv:2504.07957.
- Tinglei Feng, Yingjie Zhai, Jufeng Yang, Jie Liang, Deng-Ping Fan, Jing Zhang, Ling Shao, and Dacheng Tao. 2022. Ic9600: A benchmark dataset for automatic image complexity assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):8577–8593.
- Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiaowu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. 2025. [Mme: A comprehensive evaluation benchmark for multimodal large language models](#). *Preprint*, arXiv:2306.13394.
- Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.
- Jihao Gu, Yingyao Wang, Pi Bu, Chen Wang, Ziming Wang, Tengtao Song, Donglai Wei, Jiale Yuan, Yingxiu Zhao, Yancheng He, Shilong Li, Jiaheng Liu, Meng Cao, Jun Song, Yingshui Tan, Xiang Li, Wenbo Su, Zhicheng Zheng, Xiaoyong Zhu, and Bo Zheng. 2025. ["see the world, discover knowledge": A chinese factuality evaluation for large vision language models](#). *Preprint*, arXiv:2502.11718.
- Tianle Gu, Zeyang Zhou, Kexin Huang, Dandan Liang, Yixu Wang, Haiquan Zhao, Yuanqi Yao, Xingge Qiao, Keqing Wang, Yujiu Yang, Yan Teng, Yu Qiao, and Yingchun Wang. 2024. Mllmgaurd: a multi-dimensional safety evaluation suite for multimodal large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 7256–7295.
- Jiawei Guo, Tianyu Zheng, Yizhi Li, Yuelin Bai, Bo Li, Yubo Wang, King Zhu, Graham Neubig, Wenhu Chen, and Xiang Yue. 2025. Mammoth-vl: Eliciting multimodal reasoning with instruction tuning at scale. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13869–13920.
- Yancheng He, Shilong Li, Jiaheng Liu, Yingshui Tan, Weixun Wang, Hui Huang, Xingyuan Bu, Hangyu Guo, Chengwei Hu, Boren Zheng, Zhuoran Lin, Xuepeng Liu, Dekai Sun, Shirong Lin, Zhicheng Zheng, Xiaoyong Zhu, Wenbo Su, and Bo Zheng. 2024. [Chinese simpleqa: A chinese factuality evaluation for large language models](#). *Preprint*, arXiv:2411.07140.
- Jixiang Hong, Quan Tu, Changyu Chen, Gao Xing, Ji Zhang, and Rui Yan. 2024. CycleAlign: Iterative distillation from black-box LLM to white-box models for better human alignment. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14596–14609.
- Jixiang Hong, Yiran Zhang, Guanzhong Wang, Yi Liu, Ji-Rong Wen, and Rui Yan. 2026. [SUDER: Self-improving unified large multimodal models for understanding and generation with dual self-rewards](#). In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 1686–1697. Association for Computing Machinery.
- Yibin Huang, Wang Xu, Wanyue Zhang, Helu Zhi, Jingjing Huang, Yangbin Xu, Yangang Sun, Conghui Zhu, and Tiejun Zhao. 2025. [Video2Layout: Recall and reconstruct metric-grounded cognitive map for spatial reasoning](#). *Preprint*, arXiv:2511.16160.
- Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjuan Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. [Follow-bench: A multi-level fine-grained constraints following benchmark for large language models](#). *Preprint*, arXiv:2310.20410.
- Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *European conference on computer vision*, pages 235–251. Springer.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with pagedattention](#). *Preprint*, arXiv:2309.06180.

- Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 292–305. Association for Computational Linguistics.
- Zongzhao Li, Zongyang Ma, Mingze Li, Songyou Li, Yu Rong, Tingyang Xu, Ziqi Zhang, Deli Zhao, and Wenbing Huang. 2026. **STAR-R1: Multi-view spatial trAnsformation reasoning by reinforcing multi-modal LLMs**. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12041–12051.
- Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2024a. **Mmbench: Is your multi-modal model an all-around player?** *Preprint*, arXiv:2307.06281.
- Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024b. Ocr-bench: on the hidden mystery of ocr in large multi-modal models. *Science China Information Sciences*, 67(12):220102.
- Run Luo, Haonan Zhang, Longze Chen, Ting-En Lin, Xiong Liu, Yuchuan Wu, Min Yang, Minzheng Wang, Pengpeng Zeng, Lianli Gao, Heng Tao Shen, Yunshui Li, Xiaobo Xia, Fei Huang, Jingkuan Song, and Yongbin Li. 2024. **Mmevol: Empowering multimodal large language models with evol-instruct**. *Preprint*, arXiv:2409.05840.
- Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. **Docvqa: A dataset for vqa on document images**. *Preprint*, arXiv:2007.00398.
- Sara Papi, Maike Züfle, Marco Gaido, Beatrice Savoldi, Danni Liu, Ioannis Douros, Luisa Bentivogli, and Jan Niehues. 2025. **Mcif: Multimodal crosslingual instruction-following benchmark from scientific talks**. *Preprint*, arXiv:2507.19634.
- Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. 2025a. **AGENT-TIF: Benchmarking large language models instruction following ability in agentic scenarios**. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc.
- Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025b. Constraint back-translation improves complex instruction following of large language models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 2388–2398.
- Yusu Qian, Hanrong Ye, Jean-Philippe Fauconnier, Peter Gräsch, Yinfei Yang, and Zhe Gan. 2025. **Mia-bench: Towards better instruction following evaluation of multimodal llms**. *Preprint*, arXiv:2407.01509.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. **Direct preference optimization: Your language model is secretly a reward model**. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. **Deepseekmath: Pushing the limits of mathematical reasoning in open language models**. *Preprint*, arXiv:2402.03300.
- Feifan Song, Bofei Gao, Yifan Song, Yi Liu, Weimin Xiong, Yuyang Song, Tianyu Liu, Guoyin Wang, and Houfeng Wang. 2026. **P-Aligner: Pre-aligning LLMs via principled instruction synthesis**. In *Proceedings of the ACM Web Conference 2026*, WWW ’26, pages 8477–8480, New York, NY, USA. Association for Computing Machinery.
- Haochen Wang, Yuhao Wang, Tao Zhang, Yikang Zhou, Yanwei Li, Jiacong Wang, Jiani zheng, Ye Tian, Jiahao Meng, Zilong Huang, Guangcan Mai, Anran Wang, Yunhai Tong, Zhuochen Wang, Xiangtai Li, and Zhaoxiang Zhang. 2026. **Grasp any region: Towards precise, contextual pixel understanding for multimodal LLMs**. In *The Fourteenth International Conference on Learning Representations*.
- Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. 2024. **Measuring multimodal mathematical reasoning with math-vision dataset**. *Preprint*, arXiv:2402.14804.
- Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, Zichen Liu, Haizhou Zhao, Dakai An, Lunxi Cao, Qiyang Cao, Wanxi Deng, Feilei Du, Yiliang Gu, Jiahe Li, and 22 others. 2025a. **Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library**. *Preprint*, arXiv:2506.06122.
- Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, and 56 others. 2025b. **InternV3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency**. *Preprint*, arXiv:2508.18265.
- Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuan Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. **Benchmarking complex instruction-following with multiple constraints composition**. *Preprint*, arXiv:2407.03978.
- Xinrun Xu, Pi Bu, Ye Wang, Börje F. Karlsson, Ziming Wang, Tengtao Song, Qi Zhu, Jun Song, Zhiming Ding, and Bo Zheng. 2025. **Deepphy: Benchmarking agentic vlms on physical reasoning**. *Preprint*, arXiv:2508.05405.

Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, Jiayi Lei, Quanfeng Lu, Runjian Chen, Peng Xu, Renrui Zhang, Haozhe Zhang, Peng Gao, Yali Wang, Yu Qiao, and 3 others. 2024. [Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi](#). *Preprint*, arXiv:2404.16006.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, and 3 others. 2024. [Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi](#). *Preprint*, arXiv:2311.16502.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. 2024. [Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems?](#) *Preprint*, arXiv:2403.14624.

Xiangyu Zhao, Shengyuan Ding, Zicheng Zhang, Haian Huang, Maosong Cao, Weiyun Wang, Jiaqi Wang, Xinyu Fang, Wenhao Wang, Guangtao Zhai, Haodong Duan, Hua Yang, and Kai Chen. 2025. [Omnialign-v: Towards enhanced alignment of mllms with human preference](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18490–18515.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. [Group sequence policy optimization](#). *Preprint*, arXiv:2507.18071.

Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and Shanghang Zhang. 2025. [Roborefer: Towards spatial referring with reasoning in vision-language models for robotics](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 28404–28481. Curran Associates, Inc.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. [Instruction-following evaluation for large language models](#). *Preprint*, arXiv:2311.07911.

A Appendix Overview

This appendix provides comprehensive supplementary materials organized into several parts for the convenience of readers.

- More Details (§B)
- More Results (§C)
- Task Definition (§D)
- Prompt Template (§E)

B More Details

B.1 Quality Control

At each stage of the construction pipeline, we implement rigorous quality control mechanisms to ensure the high quality and diversity of the resulting visual instructions. We engaged 20 college-educated annotators to refine our benchmark by assessing: 1) solvability of problems; 2) potential conflicts across multiple constraints and rounds; 3) accuracy of constraint verifier functions; and 4) typographical errors. All annotators underwent three trial rounds to familiarize themselves with the task, only proceeding to official annotations upon approval. Each candidate was independently checked by at least two annotators, with disagreements resolved by a third adjudicator before retention or refinement.

During the source collection stage, we filter out images with inappropriate content or low quality using a combination of rule-based checks and LVLM-based assessments. In the task instantiation stage, we analyze the compatibility of each image–description pair with the defined task types, and then post-evaluate the instantiated tasks for their appropriateness, solvability, and difficulty level. In the atomic constraint integration stage, we assess the compatibility of each instantiated task with the defined atomic constraints, ensuring that the generated constrained instructions are coherent and solvable. Finally, in the constraints composition stage, we verify that the merged and multi-turn instructions maintain clarity and solvability after each addition of new constraints.

B.2 Comparison with Spatial and Grounding Benchmarks

We provide a more detailed comparison with spatial grounding and region-understanding benchmarks in Table 6. While these benchmarks contain image-grounded tasks, their primary focus is perception, region-conditioned understanding, or spatial referring. VC-IF instead formalizes visual requirements as explicit instruction-following constraints and studies how they interact with textual constraints under compositional and multi-turn settings.

B.3 Benchmark Constraint Modality

To further clarify the visual focus of VC-IF, we report the constraint-modality distribution within each composition mode in Table 7. Overall, 883

Table 6: Feature-level comparison between VC-IF and spatial or grounding-related benchmarks. “VC” denotes visual constraints.

Benchmark	Primary Focus	VC Definition	VC Composition	Construction Pipeline
MM-IFEval-P-level (Ding et al., 2025)	Perception and logic	✗	✗	✗
MM-IFEval-C-level (Ding et al., 2025)	Perception with textual constraints	✗	✗	✓
GAR-Bench (Wang et al., 2026)	Region-conditioned understanding	✗	✗	✓
RefSpatialBench (Zhou et al., 2025)	Spatial referring and grounding	✗	✗	✓
VC-IF (ours)	Visual-constraint following	✓	✓	✓

out of 1,200 samples (73.58%) involve at least one visual constraint.

Table 7: Constraint-modality distribution across composition modes in VC-IF. “w/ Visual Constr.” denotes samples involving at least one visual constraint.

Subset	Total	w/ Visual Constr.	Text-Only
Single	628	457 (72.77%)	171 (27.23%)
Multiple	359	268 (74.65%)	91 (25.35%)
Multiturn	213	158 (74.18%)	55 (25.82%)
Total	1,200	883 (73.58%)	317 (26.42%)

B.4 VC-PO Training Data

We construct 9,490 training samples for VC-PO using the same instruction-generation pipeline described in Section 3. These samples are automatically synthesized multi-constraint instructions and do not involve additional manual annotation or verification. The training set is used only for VC-PO optimization, while VC-IF is kept as the held-out evaluation benchmark. As summarized in Table 8, the training data contains both text-only and textual-visual mixed constrained instructions.

Table 8: Composition of the VC-PO training data.

Category	#Samples
Text-only constraints	3,170
Visual-text Mixed constraints	6,320
Total	9,490

To avoid benchmark leakage, we remove any training sample whose image overlaps with the VC-IF benchmark image set before training. We will release the VC-PO training data together with the benchmark and code to support reproducibility and further research.

B.5 Evaluation Protocol

Upon completing the previous construction stages, we obtain a diverse set of visual instruction tasks, including single-constraint, multi-constraint, and multi-turn instructions. Next, we construct our evaluation protocol to assess model performance on these instructions. Specifically, we evaluate on single-constraint instructions, multi-constraint instructions, and multi-turn instructions separately to gain a comprehensive understanding of model capabilities across varying levels of task complexity. For each instruction, we employ a two-stage evaluation process: *response generation* and *response evaluation*. In the response evaluation stage, we utilize the verification functions designed for each atomic constraint in the instructions. For multi-constraint and multi-turn instructions, we aggregate the results from all relevant verification functions to derive a final score.

B.6 Evaluation Consistency

We assess the consistency between human judgments and the default automatic evaluator. We sample 100 responses from Gemini-2.5-Flash and Qwen3-VL-235B-A22B, stratified across the Single Constr., Multiple Constr., and Multi-turn Constr. subsets in a 40/30/30 split. Two independent teams of 10 annotators judge verifier-level satisfaction using the image, instruction, response, and corresponding verifier criteria. Within each team, labels are aggregated by majority vote, with ties resolved by a designated adjudicator. Agreement is computed as the match rate between each team’s aggregated labels and the automatic evaluation. As shown in Table 9, both teams exhibit high agreement with the automatic evaluator.

Table 9: Agreement (%) between the automatic evaluator and two independent human teams.

Response source	Team 1	Team 2
Gemini-2.5-Flash	98.1	98.0
Qwen3-VL-235B-A22B	97.4	97.7

C More Results

C.1 VC-PO Ablation

To better understand the contribution of VC-PO and its key design choices, we compare it with an SFT baseline trained on the same 9.5K-sample dataset and ablate two factors: the choice of GRPO versus GSPO and whether to update or freeze the vision model during training. As shown in Table 10, VC-PO with GRPO outperforms SFT, suggesting that the improvement goes beyond supervised fitting to the constructed training distribution. GRPO also slightly outperforms GSPO, while freezing the vision model reduces the overall score, indicating that jointly optimizing the vision component is beneficial for visually grounded instruction following.

C.2 Evaluator-Robustness Validation

To assess whether VC-IF scores depend on the behavior of the default Gemini-3-Flash judge, we additionally re-evaluate all model responses using GPT-5.2 as the LVLM judge. As shown in Table 11, the results lead to conclusions consistent with the main evaluation: strong proprietary models remain competitive, open-source models still lag behind on visually constrained and multi-turn settings, and the VC-PO-trained Qwen3-VL-8B improves substantially over its base checkpoint. This supports that VC-IF measures visual- and textual-constraint following rather than imitation of a single evaluator.

D Task Definition

In this section, we will provide a detailed explanation of our definition of constraints, along with examples and the methods for validating these constraints.

D.1 Task Categories

VC-IF comprises three task categories: Visual Perception (262; 21.83%), Compositional Reasoning (555; 46.25%), and Vision-based Text Generation (383; 31.92%). It contains 628 single-constraint, 359 multi-constraint, and 213 multi-turn examples.

Table 12 presents the sub-tasks and their definitions under each major category.

D.2 Visual Constraint Types

As the core of VC-IF, visual constraints primarily consist of four important types: spatial region, symbolic reference, spatial order, and referential language phrase, as shown in Table 13.

D.3 Verifier Function Mapping

To facilitate the training of methods such as GRPO (Shao et al., 2024) and GSPO (Zheng et al., 2025), we provide a verifiable function for each constraint that delivers accurate reward signals. Table 14 illustrates all function mappings.

Table 10: Ablation results of VC-PO on VC-IF. “frozen VM” denotes freezing the vision model during training.

Model	Single Constr.		Multiple Constr.		Multi-turn Constr.		Overall
	Textual	Visual	Mixed	Textual	Mixed	Textual	
Qwen3-VL-8B-Instruct	73.68	35.23	43.67	45.73	32.46	31.68	43.74
SFT	78.36	43.54	50.64	53.66	44.57	41.12	51.98
VC-PO (GSPO)	82.61	44.79	53.87	55.86	42.26	46.20	54.27
VC-PO (GRPO, frozen VM)	81.29	43.33	54.34	56.61	42.06	45.30	53.82
VC-PO (GRPO)	84.21	46.39	54.11	56.71	42.46	47.20	55.18

Table 11: Evaluator-robustness validation: VC-IF leaderboard evaluated with an alternative judge model (GPT-5.2), confirming that results are not dependent on a specific evaluator. The table follows the same structure as Table 3.

Models	#Param.	Single Constr.		Multiple Constr.		Multi-turn Constr.		Overall
		Textual	Visual	Mixed	Textual	Mixed	Textual	
Proprietary Models								
Gemini-3-Pro	–	90.06	70.90	71.88	77.79	61.70	63.84	72.69
Gemini-3-Flash	–	90.64	67.61	72.66	73.58	61.51	70.24	72.71
Gemini-2.5-Pro	–	90.64	65.86	66.68	68.50	54.56	55.84	67.01
Gemini-2.5-Flash	–	84.21	59.96	61.15	63.82	53.01	56.80	63.16
GPT-5.2	–	92.98	52.52	70.81	72.97	60.15	59.68	68.19
GPT-5-Chat	–	90.06	50.33	59.50	59.96	45.56	45.92	58.56
GPT-5-Mini	–	94.74	52.74	76.28	79.17	49.53	47.68	66.69
GPT-4o	–	83.63	40.18	42.38	42.48	43.89	43.84	49.40
Open-source Models								
GLM-4.1V-Base	9B	46.78	32.39	34.73	33.13	21.10	18.40	31.09
InternVL3.5-8B	8B	59.06	28.23	36.05	33.94	34.64	32.48	37.40
InternVL3.5-38B	38B	63.16	35.89	38.91	37.60	38.05	35.68	41.55
Qwen2.5-VL-72B-Instruct	72B	82.25	43.02	53.58	53.35	38.73	37.44	51.40
Qwen2.5-VL-7B-Instruct	7B	63.16	29.76	37.86	36.99	31.28	25.12	37.36
Qwen3-VL-235B-A22B-Instruct	235B	82.25	54.99	56.17	54.57	45.51	43.68	56.20
Qwen3-VL-32B-Instruct	32B	77.78	53.17	48.76	48.58	38.67	34.24	50.20
Qwen3-VL-8B-Instruct	8B	76.02	38.07	46.76	47.87	37.80	33.92	46.74
+ VC-PO (Ours)	8B	86.55	47.05	55.62	55.79	46.87	45.44	56.22

Table 12: Task categories of VC-IF: Visual Perception (262; 21.83%), Compositional Reasoning (555; 46.25%), and Vision-based Text Generation (383; 31.92%).

Task Category	Sub-task Option	Definition
Visual Perception	existence	Determine if an object exists in the image.
	attribute_recognition	Recognize the color/material/category/... of an object.
	counting	Count the number of specific objects in an image.
	position_orientation	Describe the position (e.g., top-left) or orientation (e.g., facing forward) of an object.
	ocr_reading	Recognize and extract text content from an image.
	ui_component_recognition	Identify user interface components (e.g., buttons, menus).
	layout_understanding_document	Understand the structural layout of a document/table/chart/other visual content.
Compositional Reasoning	chart_data_interpretation	Interpret trends/extremum values/changes in chart data (e.g., increasing, decreasing).
	spatial_relative_position	Reason about the relative spatial positions between objects (e.g., A is to the right of B).
	spatial_occlusion	Understand situations where objects are partially hidden or occluded.
	spatial_topological_containment	Understand containment relationships between objects.
	spatial_topological_adjacency	Understand adjacency relationships between objects.
	spatial_topological_touching	Understand touching relationships between objects.
	spatial_quantity_comparison	Compare the quantities of different objects or groups.
Vision-based Text Generation	multimodal_ocr_scene_inference	Perform joint inference combining OCR text with scenes/actions.
	multimodal_action_label_consistency	Determine if OCR labels (e.g., product tags) are consistent with recognized actions.
	descriptive_narration	Generate a detailed description of a scene.
	descriptive_summary	Generate a concise summary of what the image mainly shows.
	dialogue_completion	Based on visual content and past dialogue, generate the next turn in the conversation.

Table 13: Visual Constraint Types of VC-IF, which primarily consist of four important types: spatial region, symbolic reference, spatial order, and referential language phrase.

Constraint Type	Definition
spatial_region	Introduce a marked region into the image as a constraint, e.g., describe the person in the green bounding box, write the characters in the drawn circle, etc.
symbolic_reference	Introduce a symbol/gesture marker into the image as a constraint, e.g., add an arrow to a specific object/concept, or add a gesture icon pointing to a specific object/concept in the image.
spatial_order	Require the response to describe or analyze visual content following a specific spatial order, e.g., "from right to left", "from bottom to top", "clockwise", "from background to foreground", "far-to-near", etc.
referential_language_phrase	Use referential or indirect language in the instruction to implicitly indicate an object or concept in the visual input. Require the model to first infer and identify the object or concept, and then perform a subsequent task. For example, "Describe the person wearing the red hat" or "What is the color of the vehicle closest to the camera?"

Table 14: Verifier function mapping used to compute rewards for VC-PO training.

Constraint Type	Constraint Name	Verifier Function
Number/Length Constraints	num_char	check_num_char_within_range
	num_word	check_num_word_within_range
	num_sentence	check_num_sentence_within_range
	num_paragraph	check_num_paragraph_within_range
	num_word_per_sentence	check_num_word_per_sentence_within_range
	num_word_per_paragraph	check_num_word_per_paragraph_within_range
	num_sentence_per_paragraph	check_num_sentence_per_paragraph_within_range
Text Boundary Constraints	suffix	check_ends_with
	prefix	check_starts_with
	prefix_each_paragraph	check_starts_with_each_paragraph
	suffix_each_paragraph	check_ends_with_each_paragraph
Structure Constraints	structure_json	check_is_json
Word Inclusion & Exclusion Constraints	include_word	check_include_word
	exclude_word	check_exclude_word
	include_word_per_sentence	check_include_word_per_sentence
	include_word_per_paragraph	check_include_word_per_paragraph

E Prompt Template

We provide selected prompt templates and excerpts from the main construction stages in this appendix; some excerpts are abbreviated for space.

Prompt Template for Task Construction

You are an expert in generating precise, actionable, and solvable instructions for various tasks based on visual content.

You will be given:

- Visual Content: An image and its detailed description
- Task Types: A list of task types

Your Task: Given the image with its detailed description, you should generate an appropriate instruction for each task type. The instruction should be concise, clear, and solvable.

Description of the Image:
{description}

Task Types:
{task_types}

Output Format:

Your output should be a JSON list where:

- Each item in the list corresponds to one task type.
- Each item should include the task type and its corresponding instruction in the following structure:

```
{{
  "Task_Type": "<task_type>",
  "Reasoning": "Carefully and detailedly reason on whether the task type is appropriate for the
given image; how to instantiate the abstract task type into a natural and solvable instruction
according to the content of the image and the task type; how to make the instruction have a
certain level of difficulty.",
  "Instruction": "<instruction content>"
}},
```

Important Rules:

- 1. Before generating an instruction, assess whether the task type is appropriate for the given image and its description. If not, write 'Not Compatible' in the instruction field for that task type.
- 2. The instruction must be grounded in the visual evidence and solvable based on the task's nature:
 - For Fact-based Tasks (e.g., OCR, counting, attribute recognition): The specific information required (like text, numbers, or small details) must be fully visible and unambiguous. Do not generate instructions for details that are occluded, or partially cut off (e.g., a price on the menu board blocked by a person).
 - For Contextual/Generative Tasks (e.g., dialogue completion, summary, creative reasoning): While these tasks are open-ended, they must be logically anchored in the visible scene. The instruction should be based on clear contextual cues rather than requiring the user to guess non-existent or invisible specific facts.
 - Feasibility Check: Before finalizing, verify that the instruction does not force the solver to "hallucinate" information that is physically blocked or missing from the description.
- 3. The instruction can be in the form of a question or a command, depending on what fits the task type best.
- 4. Ensure that the generated instruction has a certain level of difficulty. Avoid overly simple instructions and aim for instructions that require reasoning, analysis and fine-grained perception, while maintaining solvability based on the provided image.

Figure 7: Prompt template for task construction.

Prompt Template for Single Constraint (Plain)

You are an expert at integrating constraints into the instruction for visual tasks. Your task is to analyze the provided inputs and incorporate the given constraint into the base instruction. You will also provide a verification function that helps check whether the model output meets the constraint.

You will be given:

- Image
- Base Instruction: The initial instruction/question.
- Constraint Type: A constraint type that should be applied to the base instruction.
- Verifier Function: Docstring of a Python function used for verifying the model output against the constraint.

Your Goal:

With reasoning, analyze the compatibility of the constraint type with the base instruction. If compatible, concretize the constraint and modify the base instruction to integrate the constraint in a clear, precise, and feasible manner. If the constraint is incompatible with the base instruction, clearly state that it is "Not Compatible."

Output Format:

Your output must strictly follow this JSON structure:

```
{
  "reasoning": "Provide your logical reasoning process to determine whether the constraint is compatible with the base instruction. Detail how the constraint can be concretized and integrated, or explain why the constraint is not compatible.",
  "specific_constraint": "The concretized and clear constraint content to be integrated.",
  "integrated_instruction": "The revised instruction with the constraint integrated. It requires the model to follow the instruction while meeting the constraint. Or 'Not Compatible'.",
  "verifier": {
    "name": "the name of the given verification function",
    "args": {
      "arg_name": arg_value / null,
      "arg_name": arg_value / null,
      ... } }
},
```

Inputs

Base Instruction:

{instruction}

Constraint Type:

{constraint}

Verifier Function:

{verifier_docstring}

Important Requirements:

1. Perform reasoning or thinking to determine compatibility and integration feasibility. If the given constraint is not compatible with the current instruction, return 'Not Compatible' in the 'integrated_instruction' field, like:

```
{
  "reasoning": "xxx",
  "integrated_instruction": "Not Compatible",
  "verifier": "Not Compatible"
}
```
2. Ensure the constraint is clearly and specifically defined.
3. Ensure the constraint and integrated instruction non-trivial; do not provide any hint of answer for the instruction, especially not in acrostic key, suffix, prefix, or required inclusion words.
4. Keep the modified instruction concise, actionable, and aligned with the visual task.
5. For the "text" arg in the verifier function, you should set as "[PLACEHOLDER]", which will be replaced with the model output of the instruction. For other args, you should specify the arg value.
6. The verifier for constraints like "around [NUM] words/characters" should be set with a reasonable range for the numeric args.
7. When specifying suffix or prefix, do not add needless '...' before the suffix or after the prefix.

Figure 8: Prompt template for single-constraint integration with plain text constraints.

Prompt Template for Single Constraint (Region Mark)

You are an expert at integrating constraints into the visual instruction. Your task is to integrate a spatial region constraint into the base instruction by concretizing the constraint. Then, create a verifier, which will be used to verify the responses from other models to the integrated instruction.

You will be given:

- Image and its Description
- Base Instruction
- Mark Type: A mark type used to mark the region of interest.

Constraint Type:

spatial_region: Introduce a marked region into the image as a constraint.

The constraint can work in TWO ways:

- FOCUS mode: Task focuses ON the marked/circled/boxed region (e.g., "describe the person in the bounding box")
- EXCLUDE mode: Task explicitly EXCLUDES the marked/circled/boxed region (e.g., "describe the room excluding the marked object")

Your Task:

Your task is to concretize the constraint and integrate the constraint into the base instruction/question. The integration mainly includes two parts: 1) adding the mark to the image, and 2) modifying the base instruction accordingly. Both parts should carefully consider the image and the base instruction. If the given constraint type is not compatible with the current instruction, you should return 'Not Compatible'.

Output Format:

Your output must strictly follow this JSON structure:

...

Inputs Description:

{description}

Base Instruction:

{instruction}

Mark Type:

{mark_type}

Important Requirements:

- 1. Compatibility: If the given constraint is not compatible with the base instruction, return 'Not Compatible' in the 'integrated_instruction' field, like:

```
{ "reasoning": "xxx",  
  "integrated_instruction": "Not Compatible",  
  "verifier": "Not Compatible" },
```
- 2. If there is already a circle or bounding box in the image, return 'Not Compatible'. This is to avoid confusion and ambiguity.
- 3. Mark Content Scope: The 'mark_content' should describe a specific, relevant part of the image, not almost the entire image.
- 4. Wording: Avoid redundant supplementary words referring to the marked region in the final 'instruction' (e.g., do not add parenthetical clarifications like "(2012 to 2013)").
- 5. Mark-dependence: Design the integrated 'instruction' so the answer requires referencing the marked region. - Without the mark, the question should be ambiguous or allow multiple plausible answers, making it impossible to uniquely answer using only the text and the unmarked image. - Do not leak key information or alternative clues in the text that reveal what the mark points to; the mark must be the necessary and sufficient disambiguating evidence. - Remove the supplementary description of the marked content in the integrated 'instruction'.
- 6. Increase difficulty: Try to make the instruction more difficult with the integrated constraint. Both FOCUS and EXCLUDE modes can increase difficulty in different ways.
- 7. Verifier: The verifier is used to verify the responses from other models to the integrated instruction. It is NOT used to verify the integrated instruction.

...

Figure 9: Prompt template for single-constraint integration with spatial region marks.

Prompt Template for Single Constraint (Symbolic Reference)

You are an expert at integrating constraints into the instruction for visual tasks. Your task is to integrate a symbolic reference constraint into the base instruction by concretizing the constraint. Then, create a verifier, which will be used to verify the responses from other models to the integrated instruction.

You will be given:

- Image and its Description
- Base Instruction
- Mark Type: A mark type suggested to be used in the constraint.

Constraint Type:

symbolic_reference: Introduce a symbol/gesture marker into the image as a constraint, e.g., add an arrow to a specific object/concept, or add a hand icon pointing to a specific object/concept in the image.

The constraint can work in TWO ways:

- FOCUS mode: Task focuses ON the marked/pointed/arrow-indicated region (e.g., "describe the person in the bounding box")
- EXCLUDE mode: Task explicitly EXCLUDES the marked/pointed/arrow-indicated region (e.g., "describe the room excluding the marked object")

Your Task:

Your task is to concretize the constraint and integrate the constraint into the base instruction. The integration mainly includes two parts: 1) locate an appropriate coordinate [x, y] in the image to add the mark; 2) modify the base instruction accordingly. Both parts should carefully consider the image and the base instruction. If the given constraint type is not compatible with the current instruction, you should return 'Not Compatible'.

Important Requirements:

1. Compatibility: If the given constraint is not compatible with the base instruction, return 'Not Compatible' in the 'integrated_instruction' field, like:

```
{{"reasoning": "xxx",  
  "integrated_instruction": "Not Compatible",  
  "verifier": "Not Compatible" }},
```
2. If there is already an arrow or hand mark in the image, you should return 'Not Compatible'.
3. Mark Type: There are some circumstances where the mark type may be specially handled. - Use 'arrow' if hands or human body parts are visible in the image - Use 'hand' if arrows or directional symbols are visible in the image - Otherwise, use the mark type suggested in 'Mark Type' input
4. Mark content: Select mark_content that can be pointed by a single mark (e.g., finger, arrow).
5. Wording: Avoid redundant supplementary words referring to the marked content in 'integrated_instruction' (e.g., do not add parenthetical clarifications like "the leftmost one").
6. Mark-dependence: Design the integrated 'instruction' so the answer requires referencing the mark. - Without the mark, the question should be ambiguous or allow multiple plausible answers, making it impossible to uniquely answer using only the text and the unmarked image. - Do not leak key information or alternative clues in the text that reveal what the mark points to; the mark must be the necessary and sufficient disambiguating evidence. - Prefer objects/regions requiring precise localization so the answer strongly depends on the exact marked content. - Remove the supplementary description of the marked content in the integrated 'instruction', e.g., you should remove 'thematic phrase in the footer' in '... the pointed thematic phrase in the footer...' and just write 'the pointed text', when using 'thematic phrase in the footer' can locate the marked content without the mark.
7. Increase difficulty: Try to make the instruction more difficult with the integrated constraint. Both FOCUS and EXCLUDE modes can increase difficulty in different ways.
8. Verifier: The verifier is used to verify the responses from other models to the integrated instruction. It is NOT used to verify the integrated instruction.
9. Verifier Constraint Consistency: The "constraint" for "verifier" should be consistent with the 'mark_position' in 'integrated_instruction'.
10. If the above requirements can not be satisfied, you should return 'Not Compatible' in 'integrated_instruction' and 'verifier' fields.

Figure 10: Prompt template for single-constraint integration with symbolic references.

Prompt Template for Compose Multiple Constraints

You are an "Intelligent Instruction Composer". Your goal is to create the most complex, coherent, and useful multi-constraint instruction possible by merging the **largest compatible subset** of instructions from a given list.

You will be given:

1. An image: The visual content that all instructions are based on
2. A list of instruction objects, where each object contains:
 - 'instruction': A text description with a specific constraint
 - 'verifier': A validation mechanism with a name and arguments
 - 'constraint_type': The category of constraint

Your Task

1. Review all instructions and their constraints. Analyze the compatibility between the instructions and identify potential conflicts.
2. Identify the largest subset of instructions that can be merged without logical conflicts, prioritizing instructions that appear earlier in the instruction list. If some conflicts can be resolved by slightly modifying the constraint and corresponding verifier without changing the constraint type, you can do so.
3. Merge the largest compatible subset into a single complex, coherent and natural multi-constraint instruction.
4. Collect the **verifiers** from the instructions that are merged into the multi-constraint instruction. If necessary, you can adjust the individual verifiers to ensure they align with the multi-constraint instruction.

...

Important Rules:

- 1. Visual Constraints are of high priority to merge into the multi-constraint instruction.
- 2. Maximize the number of constraints merged.
- 3. Information Leakage Prevention:
 - When merging instructions, DO NOT include specific answer details from verifier criteria which are not in original instructions in the merged instruction.
 - For visual constraints, refer to marked regions generically (e.g., "the marked region", "the highlighted area", "the pointed content", etc.) without describing what specific features to look for.
 - The merged instruction should preserve the original difficulty level - if the original instruction requires inference, don't provide hints that make it trivial.
 - Keep verifier criteria detailed for validation purposes, but do not leak additional details into the instruction.
- 4. Do not combine individual judge_model_check verifiers, leave them as separate verifiers.
- 5. You can slightly modify the original constraints in order to merge them into the multi-constraint instruction, while adjusting the corresponding verifiers to ensure they align with the new instruction.

Input Data:

Instructions:

{instructions}

Figure 11: Prompt template for composing multiple constraints.