VCIFBench: Evaluating Complex Instruction Following for Video Understanding

Anonymous ACL submission

Abstract

Multimodal large language models (MLLMs) have shown strong capabilities in visual perception and reasoning, yet existing benchmarks typically test instruction following with relatively simple prompts. This falls short of practical settings, where users often impose explicit constraints on what the output should contain and how it should be organized. We present VCIFBench, a benchmark for evaluating complex instruction following in video understanding. VCIFBench is built through a dual-source construction pipeline that combines prompts adapted from existing video benchmarks with prompts generated directly from video content, composes them with structured constraints to form complex instructions, and evaluates the resulting outputs with a hybrid verification pipeline. The benchmark covers diverse task types, constraint dimensions, and composition patterns. Experiments on 11 state-of-the-art proprietary and open-source MLLMs show that reliable constraint satisfaction remains challenging across models and settings. We further derive a supervised fine-tuning dataset from the same construction pipeline and show that it improves overall instruction-following performance.

1 Introduction

Multimodal large language models (MLLMs) have recently shown strong capabilities in understanding and reasoning over heterogeneous inputs such as text, images, and videos (Dai et al., 2023; Comanici et al., 2025). In parallel, a growing ecosystem of benchmarks has emerged to evaluate different aspects of multimodal intelligence, including multimodal reasoning (Yue et al., 2024), long-video understanding (Zhou et al., 2025), and broad multimodal competence (Liu et al., 2024b). These benchmarks are valuable for probing model capabilities and failure modes, but they provide limited insight into whether MLLMs can reliably follow complex instructions under explicit constraints, especially in video-grounded settings.

This capability matters in practical use. In real applications, an MLLM must not only understand what happens in a video, but also satisfy user requirements on content, format, style, and structure. Such requirements become particularly important in agentic settings, where deviations from output schemas or tool-use protocols can trigger cascading failures (Zhang et al., 2025a). Although instruction following is widely recognized as a core competency for MLLMs (Ding et al., 2025; Li et al., 2025), the ability to satisfy complex instructions under explicit constraints remains much less studied in multimodal settings.

Existing work leaves two major gaps. First, multimodal instruction-following benchmarks remain limited in scope. In language-only settings, prior work has already studied complex instruction following with multiple constraints and composition patterns (He et al., 2024; Jiang et al., 2024; Wen et al., 2024). By contrast, current multimodal benchmarks often rely on short and relatively simple instructions (Bitton et al., 2023), focus primarily on image-based settings (Ding et al., 2025; Bitton et al., 2023), or restrict video evaluation to narrow task categories such as captioning (Li et al., 2025). However, instruction following in video should not be reduced to captioning alone. As task types shift from description to prediction, structured organization, or reasoning, the same constraint may become substantially harder to satisfy or verify. A model that appears compliant in captioning settings may still fail to follow comparable constraints under other task types.

Second, evaluation remains difficult. Rule-based verification is reliable for explicitly programmable requirements (Zhou et al., 2023a), but cannot cover many semantic constraints. LLM-based judging is more flexible, yet it can suffer from systematic

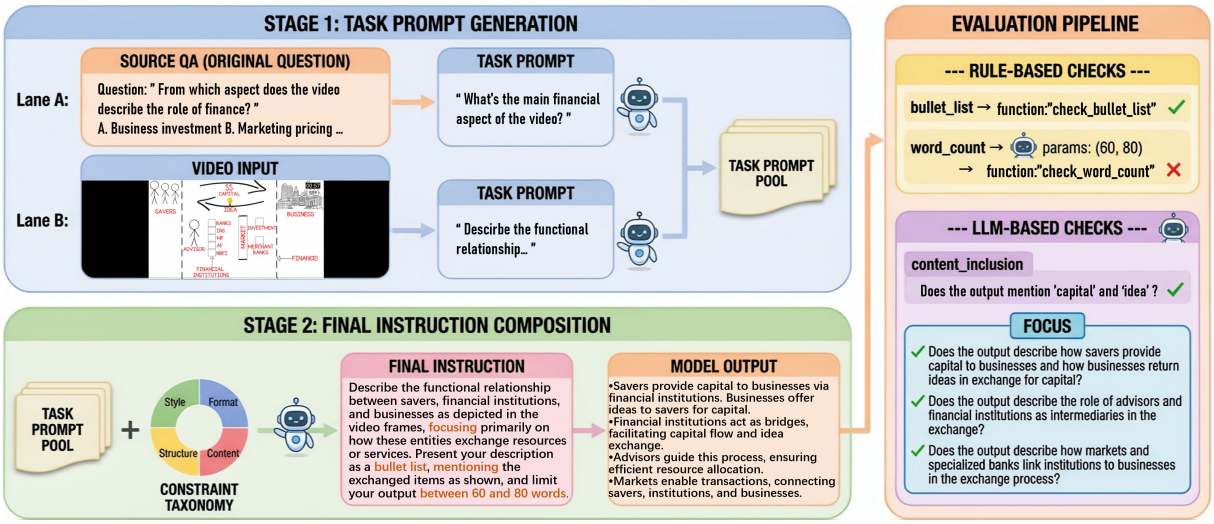


Figure 1: Overview of VCIFBench. In Stage 1, we derive task prompts from both original benchmark questions and direct video-grounded prompting. In Stage 2, each final instruction is constructed by combining a task prompt with sampled constraints from a structured constraint taxonomy. The resulting model output is then assessed using a hybrid evaluation pipeline.

biases such as length bias and positional bias (Fu et al., 2024b; Liu et al., 2024a). These limitations make it difficult to assess complex video instruction following in a way that is both broad in coverage and reasonably reliable.

We present **VCIFBench**, a **V**ideo **C**omplex **I**nstruction **F**ollowing benchmark for evaluating whether MLLMs can satisfy complex, video-grounded instructions under diverse constraints. Instead of constructing instructions independently from task sources, VCIFBench is built through a dual-source construction pipeline that combines prompts adapted from existing video benchmark tasks with prompts generated directly from video content, and then composes them with structured constraints to form complex instructions.VCIFBench builds on existing video benchmark tasks and adapts them into constraint-rich instructions. Using videos curated from prior datasets such as TempCompass (Liu et al., 2024c), MMWorld (He et al., 2025), and MME (Fu et al., 2025), We build a benchmark that spans 19 task types and 40 constraint types across a wide range of scenarios, including science, sports, and arts. To better capture realistic instruction demands, we organize these constraints under multiple constraint dimensions and support diverse composition patterns.We further develop a hybrid evaluation pipeline comprising three complementary verification modes: rule-based verification, hybrid verification, and LLM-based judging, and improve diagnostic granularity by decomposing high-level semantic constraints into sub-constraints for finer-grained analysis.

Based on VCIFBench, we show that reliable complex instruction following remains far from solved in video understanding. Strong task performance does not consistently translate into strong constraint satisfaction. We further show that data derived from our construction pipeline can support supervised fine-tuning and improve overall instruction-following performance. Anonymous code and supplementary materials are available at `https://anonymous.4open.science/r/annoym0`.

Our contributions are as follows:

- We introduce VCIFBench, a benchmark for complex instruction following in video understanding, together with a dual-source construction pipeline that combines benchmark-adapted prompts and directly video-grounded prompts to build constraint-rich instructions.
- We develop a hybrid evaluation pipeline with three complementary verification modes—rule-based verification, hybrid verification, and LLM-based judging—and further improve diagnostic granularity through sub-constraint decomposition.
- We conduct a systematic empirical study of complex video instruction following across 11 proprietary and open-source MLLMs, and

show that data derived from our construction pipeline can also be used for supervised fine-tuning to improve overall instruction-following performance.

2 Related Work

2.1 Multimodal Large Language Models and Benchmarks

Recent multimodal large language models (MLLMs) have demonstrated strong capabilities in understanding and reasoning over images, videos, and other heterogeneous inputs (Dai et al., 2023; Guo et al., 2025; Comanici et al., 2025). To evaluate these capabilities, a wide range of multimodal benchmarks has been proposed, covering settings such as multimodal reasoning (Yue et al., 2024), temporal perception (Piryani et al., 2025), and video summarization (Jung and Kim, 2025). These benchmarks have been valuable for tracking model progress and exposing failure modes across tasks and modalities.

However, many existing multimodal benchmarks still emphasize task-level performance, such as overall accuracy, under relatively simple prompts (Fu et al., 2024a). As a result, they provide limited insight into whether models can satisfy fine-grained instruction requirements on output content, format, style, and structure. This limitation becomes especially important in practical deployment, where task completion alone is often insufficient if a model cannot reliably satisfy explicit user constraints. Our work is motivated by this gap and focuses on complex instruction following in video settings under diverse constraints.

2.2 Instruction Following

Instruction following has been studied extensively in language models. Early evaluation frameworks mainly emphasized general helpfulness under relatively simple prompts, while later work increasingly focused on explicit and verifiable requirements. For example, IFEval (Zhou et al., 2023a) evaluates programmatically checkable format constraints, while more recent benchmarks study complex instruction following across diverse task types and composition patterns, including combinations of semantic and format constraints (He et al., 2024; Wen et al., 2024). EifBench further considers settings with multiple instructions and multiple constraints (Zou et al., 2025). Prior work has also explored methods for improving compliance in such scenarios, including in-context learning (Zhou et al., 2023b), supervised fine-tuning (Zhang et al., 2025b), and preference optimization methods such as DPO (Ding et al., 2025).

In contrast, instruction following for MLLMs remains less developed, especially in video settings. Existing multimodal studies mainly focus on image-based instruction following (Ding et al., 2025; Bitton et al., 2023). In video settings, constraint satisfaction is complicated by temporally evolving evidence and by task types that go beyond captioning. IF-VidCap (Li et al., 2025) is a representative prior effort, but it remains centered on a relatively narrow, caption-oriented task. Broader forms of complex video instruction following therefore remain underexplored. Moreover, the same constraint may vary in both difficulty and failure mode across different video-grounded task types, such as prediction, structured organization, and reasoning. Our work addresses this gap by studying complex video instruction following across diverse constraint types and task types, and by introducing a hybrid evaluation pipeline that combines rule-based verification, hybrid verification, and LLM-based judging.

3 VCIFBench

3.1 Data Collection

We construct VCIFBench from established public datasets, including TempCompass (Liu et al., 2024c), MMWorld (He et al., 2025), and MME (Fu et al., 2025). Together, these sources cover a broad range of real-world scenarios, including outdoor sports, classroom instruction, and short-form web videos. The distribution of video durations is shown in Figure 2.

A key advantage of these datasets is that they are paired with carefully designed questions probing diverse aspects of video understanding. TempCompass and MME include tasks involving temporal reasoning and state or attribute change, while MME also covers counterfactual and predictive reasoning. These sources therefore provide a strong foundation for benchmark construction. We build part of VCIFBench by adapting such existing task prompts under additional explicit constraints, while also constructing complementary instructions directly from video content. The overall construction and evaluation pipeline is shown in Figure 1.

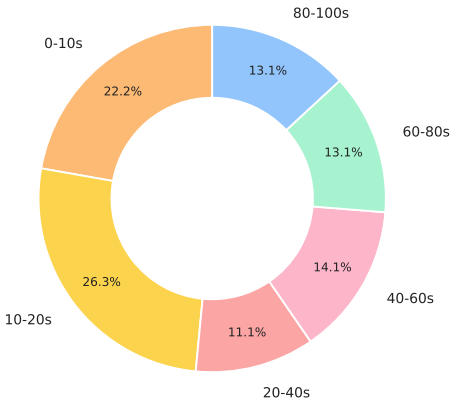


Figure 2: Distribution of video durations in VCIFBench. Most samples are concentrated in short and medium-length clips.

Figure 3: Hierarchical taxonomy of constraints in VCIFBench. The taxonomy is organized into four major constraint dimensions: format, content, style, and structure.

3.2 Instruction Generation

Task taxonomy. We derive a diverse set of task types from widely used MLLM benchmarks (Fu et al., 2024a), covering three broad categories: *perception*, *reasoning*, and *expressive generation*. Unlike prior benchmarks that mainly emphasize task completion, VCIFBench focuses on whether models can satisfy explicit constraints while performing the task. A more detailed task taxonomy and representative examples are provided in Appendix A.5.

Constraint taxonomy. We build on prior taxonomies of practical instruction constraints (Jiang et al., 2024; Zou et al., 2025; Li et al., 2025) and adapt them to video instruction following. We focus on constraints that arise naturally in realistic video-grounded use cases, including structured outputs, evidence-faithful descriptions, audience-aware responses, and prescribed organization patterns. As shown in Figure 3, we organize 40 constraint types into four constraint dimensions. **Format constraints** test compliance with explicit output specifications, such as serialization format, length bounds, and template-like patterns. **Content constraints** test whether models select, include, exclude, or emphasize video-supported information in a way that is faithful to the instruction. **Style constraints** examine whether the same content can be expressed under different communicative demands, such as tone, target audience, persona, and rhetoric. **Structure constraints** capture organization requirements that are especially relevant in video settings, such as chronological, causal, and hierarchical ordering. Importantly, although content-related constraints depend on video understanding, faithfulness to the video content is itself a core aspect of instruction following, as a helpful response must remain grounded in the video rather than rely on unsupported assumptions.

Compositional instruction types. Following prior work (Wen et al., 2024), we construct instructions with several common composition patterns: **And** for jointly satisfying multiple constraints; **Chain** for combining multiple subtasks; **Selection** for conditioning the response on explicit alternatives; and **Nested** for hierarchically combining multiple composition patterns. These patterns improve coverage and better reflect realistic user requests. We do not report a separate analysis for each composition pattern, since these categories are not mutually exclusive. In particular, **And** often appears within **Chain** and **Selection**, making isolated attribution ambiguous.

Instruction construction. As illustrated in Figure 1, we construct instructions from two complementary sources, using GPT-5.2 (Singh et al., 2025) as the primary generator.

(1) Benchmark-derived prompts. All selected source datasets provide QA-style annotations, which offer a natural starting point for instruction construction.We apply GPT-5.2 to rewrite suitable QA pairs into open-ended task prompts and discard cases that cannot be meaningfully transformed (e.g., questions explicitly asking about content absent from the video) and to select compatible constraints, composing them together to form the

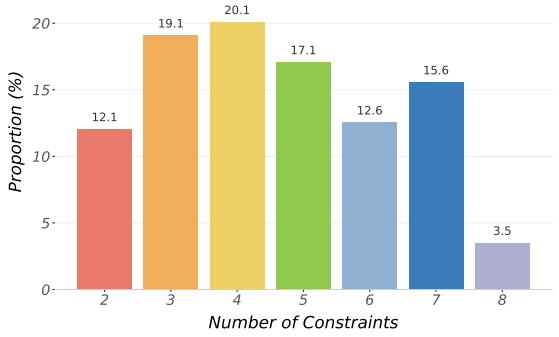


Figure 4: Distribution of the number of constraints per instruction in VCIFBench. Most instructions contain 3–7 constraints, with 4 being the most common setting.

final instruction. TempCompass (Liu et al., 2024c) also contains paired original and reversed videos, which naturally support binary **Selection**-style instructions, such as determining whether a building is collapsing or being constructed. These cases are useful for testing whether models follow instructions consistently across explicit alternatives, rather than appearing correct by chance on a single branch.

(2) Direct video-grounded prompts. For videos without suitable QA annotations, we construct each instruction by first selecting a task that is compatible with the observable video evidence and then combining it with candidate constraints. We use GPT-5.2 to generate the instruction. GPT-5.2 then reviews the generated instruction, detects potential conflicts among constraints, and rewrites it to improve linguistic fluency while preserving the intended formal requirements. Since different models may exhibit distinct hallucination patterns and verification biases, we further employ Gemini-2.5 Pro to examine whether content-related constraints are supported by, or potentially conflict with, the visible video content. After automated verification, we further conduct manual revision to correct residual inconsistencies.

3.3 Evaluation

Some constraints, such as output length bounds or required surface forms, are naturally suited to rule-based verification. However, purely rule-based methods cannot reliably assess open-ended constraints that depend on interpretation or video-grounded reasoning.

Motivated by prior work on LLM-as-a-Judge (Karpinska et al., 2021; Yu et al., 2024), we adopt a hybrid evaluation pipeline that combines rule-based verification with LLM-based judging.

Rule-based verification. For deterministically verifiable constraints, we directly implement checking functions and compute binary pass/fail outcomes programmatically.

Hybrid verification. Some constraints, such as word count, require parameters to be extracted from instructions before the checker can be executed. Following prior work on LLM tool use (Zhong et al., 2025; Wen et al., 2024), we provide the model with the checker signature and usage guidelines, and ask it to extract the required parameters. We then execute the checker using the extracted parameters.

LLM-based judging. For inherently subjective or semantic constraints, such as discourse structure and style, we adopt GPT-4.1 (Achiam et al., 2024) as the judge model and prompt it to produce a binary decision for each constraint, following prior work (Zou et al., 2025; Wen et al., 2024).

For content-related constraints, including summarization, focus, tracking, and content inclusion, a single holistic judgment is often too coarse to evaluate faithfulness to the video. Inspired by fine-grained and evidence-aware evaluation (Wen et al., 2024; Qin et al., 2024), we instead adopt a decomposition-based strategy. Specifically, we manually design targeted sub-constraints that break each high-level constraint into more explicit checks, as illustrated in Figure 1, and ask GPT-4.1 to judge each sub-constraint in a binary manner. This design provides finer-grained diagnosis of semantic failures than a single top-level judgment alone.

Metrics. VCIFBench is constraint-centric: an instruction is counted as satisfied if and only if all annotated constraints are satisfied. We therefore report both constraint-level and instruction-level metrics. CPR measures performance on annotated constraints and can be aggregated by constraint type or by constraint dimension, while IPR measures whether a model satisfies the full set of annotated constraints for an instruction. IPR better reflects practical deployment settings, where violating any critical constraint may invalidate the output.

Let $\mathcal{S}$ be the set of samples. For each sample $i \in \mathcal{S}$, let $\mathcal{C}_i$ denote its annotated constraint set, and let $p_{i,c} \in \{0, 1\}$ indicate whether the model satisfies constraint $c \in \mathcal{C}_i$.

CPR (Constraint Pass Rate). For a set of annotated constraints $\mathcal{A}$, we define

$$\mathrm{CPR}(\mathcal{A}) = \frac{1}{|\mathcal{A}|} \sum_{(i,c)\in\mathcal{A}} p_{i,c}.$$

In practice, $\mathcal{A}$ can correspond to all instances of a particular constraint type or to all constraints within a given dimension. In the main paper, we primarily report dimension-level aggregates.

IPR (Instruction Pass Rate). While CPR evaluates satisfaction at the level of individual annotated constraints, IPR measures whether the full constraint set of an instruction is satisfied jointly. We define instruction-level success as satisfying *all* annotated constraints for a sample:

$$\mathrm{IPR} = \frac{1}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} \mathbf{1}[\forall c \in \mathcal{C}_i,\ p_{i,c} = 1].$$

3.4 SFT Dataset Construction

We further construct an SFT dataset of 400 samples using the same instruction construction pipeline. For each sample, we first prompt GPT-5 to generate a candidate response, and then use GPT-4.1 (Singh et al., 2025) to identify potentially unsatisfied constraints. We finally perform manual verification to correct rule-based errors and inspect issues identified during LLM-based judging.

4 Experiments

4.1 Experimental Setup

Our final test set contains 200 samples. We evaluate 11 multimodal large language models (MLLMs), including proprietary models such as Gemini-3-Pro (Gemini Team, Google DeepMind, 2025), Gemini-2.5-Flash (Comanici et al., 2025), GPT-5 (Singh et al., 2025), and Qwen3-VL-Plus (Bai et al., 2025), as well as open-source models such as InternVL3.5-8B (Wang et al., 2025), InternVL3.5-38B (Wang et al., 2025), Qwen3-VL-30B-A3B-Instruct (Bai et al., 2025), Qwen3-VL-8B-Instruct (Bai et al., 2025), MiMo-VL-7B-RL (Xiaomi et al., 2025), MiniCPM-o-4.5 (Yao et al., 2024), and GLM-4.1V (Hong et al., 2026). Table 1 reports instruction pass rate (IPR) together with dimension-level constraint pass rates (CPR) for Format, Content, Style, and Structure.

For all models, we set the temperature to 0.1. Due to differences in native video input interfaces and context-length constraints, we directly use raw video as input for Gemini models. For the remaining models, we uniformly sample video frames at 0.5 frames per second (FPS), cap the maximum spatial side length at 1080 pixels, and resize the other side proportionally to preserve the original aspect ratio. For evaluation, we use GPT-4.1 (Achiam et al., 2024) as the judge model for LLM-based judging and hybrid verification.

4.2 Main Results

Table 1: Comparison of different models on instruction pass rate and dimension-level constraint pass rates (%) across the four constraint dimensions.

Model	IPR	Format	Content	Style	Structure
Proprietary					
GPT-5	67.2	89.5	84.3	98.7	97.6
Gemini-3-pro	68.7	92.3	81.5	96.8	94.4
Gemini-2.5-flash	49.7	84.3	70.0	93.8	77.5
Qwen3-vl-plus	63.9	87.0	85.8	98.0	91.3
Open Source					
GLM-4.1V-9B-Thinking	15.8	46.4	44.1	46.3	41.7
InternVL3.5-8B	21.2	61.3	65.2	69.9	83.3
InternVL3.5-38B	24.6	62.9	69.3	81.7	88.1
MiMo-VL-7B-RL	39.4	79.6	66.7	87.6	85.7
MiniCPM-o-4.5	44.9	80.5	73.0	92.8	86.5
Qwen3-VL-30B-A3B-Instruct	47.0	81.8	75.3	90.8	81.0
Qwen3-VL-8B-Instruct	52.6	79.0	77.7	87.5	83.1

Table 1 reports instruction pass rate (IPR) together with dimension-level constraint pass rates (CPR) across the four constraint dimensions. Overall, proprietary models achieve the strongest performance: GPT-5 and Gemini-3-Pro obtain the highest IPR, while open-source models remain clearly behind on full instruction satisfaction. This gap suggests that reliable compliance under complex, video-grounded instructions remains challenging, especially in fully open-source settings.

Across dimensions, **Style** and **Structure** are generally easier than **Format** and **Content**. In particular, **Content** remains a persistent bottleneck: even strong proprietary models do not uniformly excel on this dimension, and several open-source models lag substantially behind. **Format** is also far from solved, suggesting that satisfying explicit output constraints—such as precise counts, fixed patterns, or constrained formatting—remains a practical source of failure.

Among open-source models, Qwen3-VL-8B-Instruct achieves the highest IPR, outperforming several larger open-source baselines. This suggests that instruction-following performance is not determined by scale alone.

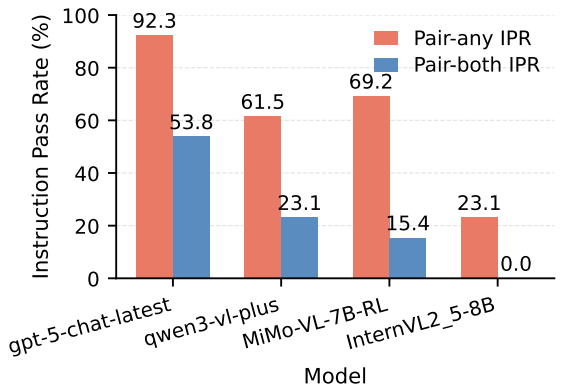


Figure 5: Pair-level instruction pass rates on the coherent selection test.

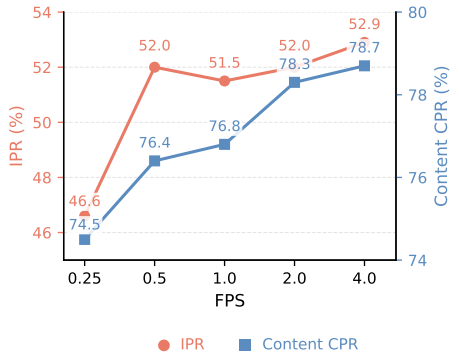


Figure 6: IPR and Content CPR across frame sampling rates at fixed 480p.

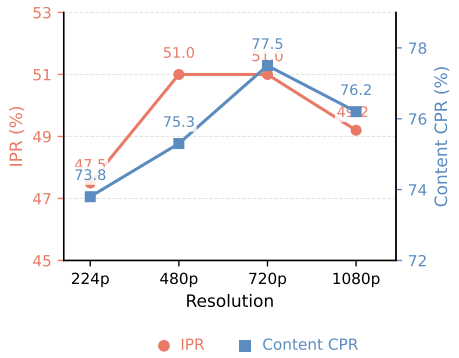


Figure 7: IPR and Content CPR across input resolutions at fixed 0.5 fps.

4.3 Consistency in Selection

We further examine whether models remain consistent across both branches of *Selection*-type instructions.

We construct this test set from TempCompass (Liu et al., 2024c) by converting multiple-choice questions into original–reverse pairs with the same instruction template but opposite correct branches. A representative example is provided in Appendix 8.

We evaluate 13 paired instances with two pair-level metrics: **Pair-any IPR**, the proportion of pairs passed on at least one branch, and **Pair-both IPR**, the proportion passed on both branches.

As shown in Fig. 5, proprietary models exhibit smaller drops from Pair-any IPR to Pair-both IPR, indicating stronger cross-branch consistency. In contrast, several open-source models degrade substantially under the stricter metric, suggesting that their apparent success is often branch-sensitive.

These results highlight a limitation of conventional evaluation: passing one branch of a paired *Selection* setup does not necessarily indicate robust understanding of the underlying decision structure or stable compliance with the corresponding constraints.Future benchmarks should therefore place greater emphasis on paired consistency tests and other controlled perturbations.

4.4 Effects of Frame Sampling and Resolution

We further analyze how temporal sampling density and spatial resolution affect performance. For the frame-sampling study, resolution is fixed at 480p; for the resolution study, frame rate is fixed at 0.5 fps. In addition to overall instruction pass rate (IPR), we report Content CPR because frame sampling and resolution most directly affect content-level grounding, both theoretically and in practice.

As shown in Fig. 6 and Fig. 7, the two scaling factors exhibit different behaviors. Increasing resolution from 224p to 720p improves both IPR and Content CPR, but performance drops at 1080p, indicating that higher spatial resolution is not uniformly beneficial. One possible explanation is that additional resolution provides finer detail but also introduces redundant visual information and greater processing burden.

By contrast, increasing fps yields a more consistent improvement in Content CPR, while IPR also improves overall despite minor fluctuations. This suggests that denser temporal sampling contributes useful new evidence, especially for motion-dependent content, whereas stronger content grounding does not automatically translate into perfect end-to-end instruction compliance.

Overall, higher-quality visual input is helpful, but the gains are non-monotonic. Temporal scaling appears more consistently beneficial than spatial scaling, while excessive resolution may reduce efficiency by introducing detail that is not relevant to

the instruction.

4.5 Error Analysis

We observe several recurring failure modes in complex video instruction following.

First, even non-conflicting constraints can interfere with one another. For example, when an instruction requires exactly five sentences together with a fixed closing pattern, a model may append an extra sentence to satisfy the ending, thereby violating the sentence-count constraint. A similar issue appears in `Selection`-type instructions: a model may first state the selected branch and then generate the required two paragraphs, resulting in an extra paragraph and thus violating the paragraph-count constraint. These cases show that failures can arise from constraint interaction itself, even when each individual requirement is relatively simple.

Second, content-related failures vary substantially across task types. In prediction or hypothetical tasks, models may rely on textual priors and generate plausible continuations that are not grounded in the video, even when the instruction explicitly requires inclusion of specific video-supported content. Likewise, under causal or chronological constraints, a model may explain the development of environmental pollution using general world knowledge rather than the events shown in the video. These failures highlight the value of our manually designed sub-constraints, which make such content deviations more visible, and they further support the need for task diversity in VCIFBench.

Third, repetition remains a noticeable issue in some open-source models, particularly the InternVL family. Even with a repetition penalty of 1.1, this behavior often further harms format compliance.

These patterns show that failures in VCIFBench are not limited to semantic misunderstanding. Models also struggle with constraint interaction, content compliance, and stable generation when many explicit constraints must be satisfied jointly.

4.6 Effect of Supervised Fine-Tuning

Table 2: Effect of supervised fine-tuning (SFT) on Qwen3-VL-8B.

Model	IPR	Format	Content	Style	Structure
Qwen3-VL-8B	50.5	81.8	77.5	90.8	**85.7**
Qwen3-VL-8B + SFT	**59.8**	**85.9**	**82.9**	**91.6**	79.4

For SFT, we fine-tune Qwen3-VL-8B-Instruct on our training set using LoRA. The model is trained for 3 epochs with a learning rate of 1×10^{-5}, a per-device batch size of 1, and gradient accumulation over 4 steps. After training, we merge the LoRA adapter into the base model and evaluate it using the same full evaluation pipeline. During evaluation, we set the input resolution to 720p and the frame sampling rate to 0.5 fps.

Table 2 shows that SFT substantially improves overall instruction-following performance, raising IPR by 9.3 points. The clearest gains appear in **Format** (+4.1) and **Content** (+5.4), suggesting improved compliance with explicit output constraints as well as stronger adherence to video-grounded content under instruction constraints. By contrast, **Structure** drops by 6.3 points. One possible explanation is that SFT more readily improves local, directly supervised behaviors, whereas structural constraints depend more on global organization patterns. If the organization patterns in the reference answers do not align well with the base model's native generation tendencies, imitation may not consistently improve structure-sensitive constraints and may even introduce mild interference.

5 Conclusion

We present **VCIFBench**, a benchmark for evaluating complex instruction following in video understanding, together with a construction pipeline and a hybrid evaluation pipeline. Experiments on 11 MLLMs show that strong task performance does not necessarily translate into reliable constraint satisfaction: substantial weaknesses remain in fine-grained format control, adherence to video-grounded content, and robustness under instructions that combine multiple explicit constraints. We further show that supervised fine-tuning on data derived from our pipeline can improve overall instruction-following performance. We hope VCIFBench will serve as a useful step toward more reliable, controllable, and practically deployable multimodal systems.

6 Limitations

First, our current benchmark primarily centers on English instructions and outputs. Broader multilingual coverage would be a useful extension for studying complex video instruction following across languages and linguistic conventions.

Second, VCIFBench is designed to evaluate con-

straint satisfaction under complex video-grounded instructions. Its results should therefore be interpreted as complementary to broader measures of general video understanding ability, rather than as a replacement for them.

Finally, our evaluation pipeline combines rule-based verification, hybrid verification, and LLM-based judging to balance coverage and reliability. Although this design improves practical coverage, variations in judge behavior and interpretation cannot be fully ruled out, especially for more semantic or subjective constraints.

Potential Risks. As a benchmark and controlled evaluation framework, VCIFBench is not intended as a direct standard for deployment. However, it may still introduce risks if used without care. In particular, models may be optimized to perform well on benchmark-specific constraints or evaluator preferences without achieving comparable robustness in open-ended real-world use. In addition, data derived from our construction pipeline may encourage optimization toward narrowly measurable forms of instruction compliance, which could introduce trade-offs in other capabilities when used for supervised fine-tuning.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Yonatan Bitton, Hritik Bansal, Jack Hessel, Rulin Shao, Wanrong Zhu, Anas Awadalla, Josh Gardner, Rohan Taori, and Ludwig Schmidt. 2023. Visit-bench: A dynamic benchmark for evaluating instruction-following vision-and-language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 26898–26922. Curran Associates, Inc.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Wenliang Dai, Junnan Li, DONGXU LI, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 49250–49267. Curran Associates, Inc.

Shengyuan Ding, Shenxi Wu, Xiangyu Zhao, Yuhang Zang, Haodong Duan, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. 2025. Mmifengine: Towards multimodal instruction following. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1099–1109.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu 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.

Chaoyou Fu, Yi-Fan Zhang, Shukang Yin, Bo Li, Xinyu Fang, Sirui Zhao, Haodong Duan, Xing Sun, Ziwei Liu, Liang Wang, Caifeng Shan, and Ran He. 2024a. Mme-survey: A comprehensive survey on evaluation of multimodal llms. *Preprint*, arXiv:2411.15296.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2024b. GPTScore: Evaluate as you desire. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6556–6576, Mexico City, Mexico. Association for Computational Linguistics.

Gemini Team, Google DeepMind. 2025. Gemini 3 pro model card.

Xianda Guo, Ruijun Zhang, Yiqun Duan, Yuhang He, Dujun Nie, Wenke Huang, Chenming Zhang, Shuai Liu, Hao Zhao, and Long Chen. 2025. Surds: Benchmarking spatial understanding and reasoning in driving scenarios with vision language models. *Preprint*, arXiv:2411.13112.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196.

Xuehai He, Weixi Feng, Kaizhi Zheng, Yujie Lu, Wanrong Zhu, Jiachen Li, Yue Fan, Jianfeng Wang, Linjie Li, Zhengyuan Yang, Kevin Lin, William Wang, Lijuan Wang, and Xin Wang. 2025. Mmworld: Towards multi-discipline multi-faceted world model evaluation in videos. In *International Conference on Learning Representations*, volume 2025, pages 26931–26964.

Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi,

Junhui Ji, Lihang Pan, and 1 others. 2026. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *Preprint*, arXiv:2507.01006.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.

Woojun Jung and Junyeong Kim. 2025. QEVA: A reference-free evaluation metric for narrative video summarization with multimodal question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24632–24642, Suzhou, China. Association for Computational Linguistics.

Marzena Karpinska, Nader Akoury, and Mohit Iyyer. 2021. The perils of using Mechanical Turk to evaluate open-ended text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1265–1285, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shihao Li, Yuanxing Zhang, Jiangtao Wu, Zhide Lei, Yiwen He, Runzhe Wen, Chenxi Liao, Chengkang Jiang, An Ping, Shuo Gao, and 1 others. 2025. Ifvidcap: Can video caption models follow instructions? *arXiv preprint arXiv:2510.18726*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, and 1 others. 2024b. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer.

Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. 2024c. TempCompass: Do video LLMs really understand videos? In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8731–8772, Bangkok, Thailand. Association for Computational Linguistics.

Bhawna Piryani, Abdelrahman Abdallah, Jamshid Mozafari, Avishek Anand, and Adam Jatowt. 2025. It's high time: A survey of temporal question answering. *Preprint*, arXiv:2505.20243.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. InFoBench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048, Bangkok, Thailand. Association for Computational Linguistics.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, and 1 others. 2025. Internvl3. 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, Wenchuang 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. In *Advances in Neural Information Processing Systems*, volume 37, pages 137610–137645. Curran Associates, Inc.

LLM Xiaomi, Bingquan Xia, Bowen Shen, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang, Huaqiu Liu, Jiebao Xiao, Jinhao Dong, and 1 others. 2025. Mimo: Unlocking the reasoning potential of language model – from pretraining to posttraining. *Preprint*, arXiv:2505.07608.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, and 1 others. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *Preprint*, arXiv:2408.01800.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2024. MM-vet: Evaluating large multimodal models for integrated capabilities. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 57730–57754. PMLR.

Xiang Yue, Yuansheng Ni, Tianyu Zheng, Kai Zhang, 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. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025a. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25,

New York, NY, USA. Association for Computing Machinery.

Tao Zhang, ChengLIn Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. 2025b. CFBench: A comprehensive constraints-following benchmark for LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32926–32944, Vienna, Austria. Association for Computational Linguistics.

Lucen Zhong, Zhengxiao Du, Xiaohan Zhang, Haiyi Hu, and Jie Tang. 2025. Complexfuncbench: Exploring multi-step and constrained function calling under long-context scenario. *Preprint*, arXiv:2501.10132.

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

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. 2025. Mlvu: Benchmarking multi-task long video understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13691–13701.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023b. Controlled text generation with natural language instructions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42602–42613. PMLR.

Tao Zou, Xinghua Zhang, Haiyang Yu, Minzheng Wang, Fei Huang, and Yongbin Li. 2025. EIFBENCH: Extremely complex instruction following benchmark for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20930–20953, Suzhou, China. Association for Computational Linguistics.

A Additional Details

A.1 Additional Implementation Details

Qwen3-VL and MiniCPM experiments were conducted on three NVIDIA RTX PRO 6000 GPUs (96GB each), while all remaining experiments used a single GPU. Supervised fine-tuning of Qwen3-VL-8B on the training split for three epochs took approximately 2.5 hours. The total local inference time across all experiments was approximately 90 hours.

All datasets used in this work were drawn from publicly released or publicly accessible benchmark resources and were used in accordance with their stated licenses or access conditions. Specifically, TempCompass is released for academic research use under CC BY-NC 4.0. MMWorld provides an MIT-licensed repository, while explicitly stating that the benchmark videos are collected from the Internet and that copyright remains with the original video owners. MME is distributed for academic and non-commercial research use through its stated access procedure. We use these resources exclusively for research and benchmark construction.

A.2 Residual Failure Modes in Instruction Construction

After automated verification, we manually inspect residual inconsistencies. Despite multi-stage filtering, several recurring failure modes remain.

First, some instructions contain mismatches between content-related constraints and visual evidence, e.g., requiring details or events that do not appear in the video. We also observe cross-constraint conflicts, such as exact-word requirements that are incompatible with the required output language.

Second, some **Selection**-type instructions remain underspecified. In these cases, the unselected branch is not described clearly enough, or the branching condition is framed too subjectively (e.g., whether the response should be written "as a news report" or "as a tutorial"), rather than being tied to an objectively identifiable alternative in the video. This weakens the intended design of **Selection**, where the correct branch should be uniquely determined by visual evidence.

A.3 Diagnostic Test for Evaluation Functions

Before formal evaluation, we sample 20 instances per constraint type to stress-test the checking functions. This reveals several common failure modes. For example, some outputs contain valid JSON but prepend natural-language prefixes such as "Here is an example:" or "The answer in JSON format is:", which can trigger false negatives under naive parsing. We also observe variation in fixed-opening constraints, where responses begin with punctuation or ellipsis (e.g., "To begin with..."), causing overly rigid string matching to fail. Based on these observations, we iteratively refine the checking functions to better handle realistic output variation while preserving strictness.

For fixed-opening and fixed-closing constraints, some outputs contain extra wrapper text, such as leading guidance phrases or trailing questions. We therefore first use the judge model to decide

Verification Type	Agreement
Overall	0.929
Rule-based	0.917
Hybrid	0.974
LLM-based	0.912

Table 3: Agreement between the evaluation pipeline and a human annotator by verification type. Overall agreement across all aligned constraint decisions is 0.929, with Cohen's $\kappa = 0.8004$.

whether rule-based verification is suitable and, if so, to extract the required parameters before executing the checker; otherwise, we fall back to direct LLM-based judging. Since the corresponding rule functions are effective in most cases, we adopt this hybrid strategy in the final pipeline.

A.4 Alignment and Robustness of the Evaluation Pipeline

We further assess the evaluation pipeline from two perspectives: alignment with human judgment and robustness across judge models.

Alignment with human judgment. We compare binary constraint decisions from the pipeline against those from a human annotator on the annotated subset. Table 3 reports agreement by verification type. Across all 295 aligned constraint decisions, the overall agreement is 0.929, with Cohen's $\kappa = 0.8004$, indicating strong agreement beyond chance.

Agreement is highest for *hybrid verification*, followed by *rule-based verification*, and slightly lower for *LLM-based judging*. This is consistent with the fact that fully judge-based decisions tend to involve more interpretation-sensitive constraints. Overall, these results suggest that the pipeline provides a reasonable approximation to human judgments.

Robustness across judge models. We also test whether the evaluation results remain stable under a different judge model. Specifically, we repeat the same constraint-level scoring procedure using Gemini 2.5 Flash as an alternative judge, while keeping the prompts, evaluation steps, and decision rules unchanged.

As shown in Table 4, GPT-4o mini and Gemini 2.5 Flash achieve high raw agreement (0.91) and substantial agreement in Cohen's κ (0.63), suggesting that the pipeline is reasonably robust across judge models.

Manual inspection shows that most disagreements arise from a small number of boundary cases. For some *Selection* examples, responses partially discuss both candidate branches, leading judges to differ on whether the correct branch was selected. Some outputs are primarily written in the target language but contain short spans in another language, yielding different judgments on language-related constraints.

Judge Pair	Agreement	Cohen's κ
GPT-4.1 vs. Gemini 2.5 Flash	0.91	0.63

Table 4: Constraint-level agreement between two judge models under the same evaluation pipeline. Gemini 2.5 Flash is used as an alternative judge with the same prompts and scoring procedure.

A.5 Detailed Task Taxonomy

Table 5 summarizes the task taxonomy used in VCIFBench. The definitions are taken from the task descriptions used in prompts during instruction construction.

A.6 Detailed Constraint Taxonomy

Table 6 summarizes the constraint taxonomy used in VCIFBench. The definitions are derived from the constraint descriptions used in prompts during instruction construction.

The evaluation column follows Section 3.3: *Rule-based* constraints are checked by deterministic functions; *LLM-based* constraints are judged by the model when satisfaction depends on semantic interpretation or video-grounded reasoning; and *Hybrid* constraints first use an LLM to extract parameters from the natural-language constraint description, after which the checker is executed programmatically.

B Examples

In this section, we provide representative examples generated and evaluated by our pipeline, including instructions constructed by our generation pipeline, corresponding model outputs, evaluation results produced by our evaluation pipeline, and the prompts used in our implementation.

Table 5: Detailed glossary of task types in VCIFBench, including concise definitions.

Task Type	**Definition**
	Perception
emotion_recognition	Infer emotions only when they are clearly visible from facial expression or body language.
harmful_content_detection	Detect harmful or unsafe content only when supported by explicit visual evidence.
text_recognition	Extract readable on-screen text only when it is clearly visible.
	Reasoning
event_sequencing	Determine the temporal order of events in the video.
state_change_detection	Describe what changes from earlier to later stages based on visual evidence.
temporal_query	Answer before, during, or after questions with justification grounded in the frames.
spatial_arrangement_description	Describe the layout of objects or entities using relative spatial positions such as left, right, inside, or behind.
object_relationship	Describe physical or functional relationships among objects, such as support, attachment, containment, or blocking.
cause_effect_identification	Identify cause–effect relations only when they are visually supported, avoiding unsupported speculation.
predictive_reasoning	Predict likely outcomes only when the prediction is strongly grounded in the observed visual context.
	Expressive Generation
summarization	Produce a faithful high-level summary of the video rather than a transcript.
tutorial_generation	Convert what is shown in the video into step-by-step instructions.
professional_writing	Present the content in a formal style, such as a report, briefing, or meeting minutes.
practical_writing	Express the content in an actionable form, such as a checklist, plan, or troubleshooting guide.
platform_copywriting	Produce platform-oriented copy grounded in the facts shown in the video.
role_play_response	Respond consistently from a specified persona while remaining grounded in the video content.
creative_writing	Generate creative text that does not contradict the visible facts.
style_transfer_rewriting	Rewrite the content in a specified style while preserving factual consistency.
point_of_view_reframing	Reframe the same content from a different viewpoint without changing the underlying facts.

Table 6: Detailed glossary of constraint types in VCIFBench, including their definitions and evaluation types.

Constraint Type	Evaluation	Definition
		Format
json	Rule-based	Output must be valid JSON (object or array).
csv	Rule-based	Output must be valid CSV.
bullet_list	Rule-based	Use an unordered bullet list.
numbered_list	Rule-based	Use an ordered list.
markdown	Rule-based	Use markdown formatting.
plain_text	Rule-based	Plain text only.
word_count	Hybrid	Enforce word count bounds.
sentence_count	Hybrid	Enforce sentence count bounds.
paragraph_count	Hybrid	Enforce paragraph count bounds.
item_count	Hybrid	Enforce item or list entry count bounds.
fixed_opening	Hybrid	Start with the exact specified string.
fixed_closing	Hybrid	End with the exact specified string.
pattern_opening	LLM-based	The opening should follow a described pattern rather than an exact fixed string.
pattern_closing	LLM-based	The closing should follow a described pattern rather than an exact fixed string.
prefix_label	Hybrid	Each item should begin with the specified prefix or label format.
letter_case	Hybrid	Apply the required casing rule to the specified scope.
emphasis	LLM-based	Emphasize specified spans with the given markers.
tense	LLM-based	Use the specified tense consistently.
no_contractions	Rule-based	Avoid contractions (e.g., use “do not” instead of “don’t”).
		Content
summarize	LLM-based	Require abstraction or a high-level summary rather than literal transcription.
focus	LLM-based	Emphasize the specified aspects and allocate most of the content to them.
content_inclusion	LLM-based	The response must mention specified visible items, which should remain small in scope and checkable.
exact_word_inclusion	Hybrid	The response must include the specified exact word or phrase verbatim.
sequence_marker_inclusion	LLM-based	The response must use specified ordering or connective markers.
tracking	LLM-based	Re-identify recurring entities across scenes while maintaining cross-scene identity consistency.
lexical_diversity	Hybrid	Avoid repeating key terms; this may specify banned terms or a maximum repetition count.
content_exclusion	LLM-based	The response must not mention specified items.
exact_word_exclusion	Hybrid	The response must not include the specified exact word or phrase.
personal_information_exclusion	LLM-based	Do not reveal personal information, only when people or identity-bearing details are visible.
harmful_content_exclusion	LLM-based	Do not include harmful or unsafe content.
		Style
tone	LLM-based	Use the specified tone.
target_audience	LLM-based	Adapt the response for the specified audience.
persona	LLM-based	Write from the specified role or persona.
output_language	LLM-based	Produce the entire response in the specified language or script.
rhetoric	LLM-based	Use the specified rhetorical devices.
		Structure
chronological	LLM-based	Organize the response in real-time chronological order.
causal	LLM-based	Present cause before effect with explicit linking.
spatial_grouping	LLM-based	Organize the response according to spatial layout.
hierarchical	LLM-based	Organize the response by abstraction level.
counterfactual	LLM-based	Organize the response into three parts: observed facts, intervention, and predicted outcome.

Example 1.

Instruction

Describe the transformation of the plant from earlier to later frames based on the video, choosing between two options: **Option A** (the plant is reviving) and **Option B** (the plant is wilting).

If the plant is reviving, describe the observed transformation of the plant from earlier frames to later frames, focusing on the overall improvement trend across frames and using the present continuous tense consistently when describing the frames.

If the plant is wilting, describe how the plant changes from earlier to later frames, focusing on the overall decline trend across frames, ensure that no descriptive word about the flower's condition is repeated more than twice, and emphasize visible signs of deterioration by marking them with double asterisks (**).

Model Output

Option A: The plant is reviving. The leaves, which earlier appeared drooping and limp, are now gradually lifting and standing more upright. The overall posture of the plant is improving, with each frame showing a subtle but consistent upward movement of the foliage. The greenery appears to be gaining vigor, with the edges of the leaves becoming less curled and more taut. The plant's stems are visibly straightening, and the canopy is expanding slightly, indicating renewed energy. The transformation continues with the leaves regaining their natural shape, and the plant's presence becomes more vibrant and lively across the frames.

LLM-BASED CHECKS

`focus` **Pass**

Question 1: Does the output describe an overall improvement trend, specifically noting that the leaves are gradually lifting, consistent with the plant reviving?

Result: Pass

`tense` **Pass**

Constraint: Use present continuous tense consistently when describing the frames.

Result: Pass

`selection_accuracy` **Pass**

Question 1: Does the model select the correct branch consistent with: *The plant is reviving*?

Result: Pass

Example 2.

Instruction

Describe the transformation of the plant from earlier to later frames based on the video, choosing between two options: **Option A** (the plant is reviving) and **Option B** (the plant is wilting).

If the plant is reviving, describe the observed transformation of the plant from earlier frames to later frames, focusing on the overall improvement trend across frames and using the present continuous tense consistently when describing the frames.

If the plant is wilting, describe how the plant changes from earlier to later frames, focusing on the overall decline trend across frames, ensure that no descriptive word about the flower's condition is repeated more than twice, and emphasize visible signs of deterioration by marking them with double asterisks (**).

Model Output

Option B: The plant is wilting. Across the frames, its leaves are visibly drooping and sagging, with some edges curling inward. The overall posture of the plant is increasingly limp, and the leaves appear less vibrant, with a noticeable loss of turgidity. **The stems are bending downward**, and **the foliage is losing its upright stance**. The transformation indicates a steady decline in vitality, with no signs of recovery or improvement.

HYBRID CHECKS

`lexical_diversity` **Pass**

Function:
`check_lexical_diversity(model_output, words, max_repeats)`

Extracted parameters:

- `words = [wilting, drooping, sagging, limp, vibrant, decline, deterioration]`
- `max_repeats = 2`

Result: Pass

LLM-BASED CHECKS

`emphasis` **Pass**

Constraint: Emphasize visible signs of deterioration by marking them with double asterisks.
Result: Pass

Figure 8: A representative example of selection.

`selection_accuracy` **Pass**

Question 1: Does the output select the correct branch consistent with: *The plant is wilting*?
Result: **Pass**

Example 3.

Instruction

From the perspective of a person riding a bicycle on a rural gravel path, identify and describe the key events involving the animals encountered along the route. Summarize these events into a concise narrative, focusing on the encounter and interactions. Write the entire narrative in lowercase and end with a closing sentence that follows the pattern: "thus ends the ride with [encounter description]."

Model Output

the journey begins with a solitary skunk appearing on the gravel path, its black and white fur contrasting against the grey surface. as the rider approaches, the skunk moves closer, its bushy tail held high. the encounter intensifies as the skunk's scent glands become visible, indicating a defensive posture. the rider slows down, and the skunk's movements become more erratic, suggesting a sense of alarm. the skunk's tail flares out, and it begins to move away, its body low to the ground. the rider stops, and the skunk pauses, its head turned towards the rider, creating a moment of tension. the skunk then scurries off the path, disappearing into the grassy verge. the rider remains stationary, watching the skunk retreat. the encounter concludes with the skunk's disappearance, leaving the rider alone on the path. thus ends the ride with a skunk's defensive display and subsequent retreat into the grass.

HYBRID CHECKS

`letter_case` **Pass**

Function:
`check_letter_case(model_output, case)`

Extracted parameters:

- `case = lowercase`

Result: **Pass**

LLM-BASED CHECKS

summarize — Fail

Question 1: Does the output mention the rider stop on a rural gravel path bordered by greenery and trees?

Result: Pass

Question 2: Does the output summarize the approach and interaction with the skunks, including their movement toward the rider and bicycle?

Result: Fail

Reason:

- The output describes only a solitary skunk, while the intended event structure involves multiple skunks.
- It does not clearly summarize their approach and interaction with the rider and bicycle, and instead mainly emphasizes a defensive posture.

Question 3: Does the output mention the skunks eventually dispersing toward the edge of the path and into the vegetation?

Result: Pass

pattern_closing — Pass

Constraint: End with a closing sentence that follows the pattern: "thus ends the ride with [encounter description]."

Result: Pass

Example 4.

Instruction

Present your prediction of the key factor that will enable a business to expand sustainably and positively impact society in the future as an unordered bullet list. Emphasize the forward-looking factor that ensures both growth and societal contribution, clearly linking the visible business patterns in the video to the predicted factor. Include at least one notable detail from the success story examples shown. Use an informative and professional tone, and avoid using contractions entirely.

Model Output

* **Deeply integrated, ethically governed artificial intelligence and adaptive learning systems** will be the pivotal factor enabling businesses to expand sustainably and positively impact society. This forward-looking factor signifies a fundamental shift from merely deploying technology for efficiency to embedding intelligence, continuous learning, and a robust ethical framework into the very core of business operations and strategic decision-making. For sustainable expansion, businesses will leverage AI to predict market shifts with unprecedented accuracy, optimize resource allocation to minimize waste across complex global supply chains, and personalize customer experiences while rigorously respecting privacy and data sovereignty. Simultaneously, ethical AI frameworks will ensure that these advancements contribute positively to society by promoting fairness, transparency, and accountability in algorithms, actively addressing complex global challenges such as climate change and healthcare disparities, and fostering inclusive economic growth. The ability to learn, adapt, and self-optimize with an inherent ethical compass will be paramount. Visible business patterns, such as the escalating investment in big data analytics, the proliferation of personalized marketing, the increasing demand for supply chain transparency, and the nascent but growing focus on explainable AI, are all precursors to this future state. Businesses are already collecting vast quantities of data and experimenting with AI, but the future demands a more holistic, ethical, and adaptive integration where AI systems continuously learn from their environment and societal feedback, driving both profitability and profound positive impact. A notable visual detail from successful future enterprises will be the vibrant, real-time digital dashboards prominently displayed in corporate lobbies, illustrating not only financial metrics but also environmental footprint reductions, community engagement scores, and ethical algorithm performance, serving as a transparent testament to their dual commitment to prosperity and planetary well-being.

RULE-BASED CHECKS

bullet_list — Pass

Function:
`check_bullet_list(model_output)`

Result: Pass

no_contractions — Pass

Function:
`check_no_contractions(model_output)`

Constraint: Avoid using contractions entirely.

Result: Pass

LLM-BASED CHECKS

focus — Pass

Question 1: Does the output primarily emphasize the predictive factor of a sustainable growth strategy as the enabler of future business expansion and societal contribution?

Result: Pass

content_inclusion — Fail

Question 1: Does the output explicitly mention at least one of the following notable visual details from the success story examples: (1) Bakmi GM's daily high customer volume and many employees, (2) Joger's transition from IDR500K to a top souvenir brand, or (3) Johnny Andrean's hundreds of outlets regionally?

Result: Fail

Reason:

- The output introduces an invented futuristic detail about digital dashboards in corporate lobbies but fails to mention any of the required success-story details shown in the video.

tone — Pass

Constraint: Use an informative and professional tone.

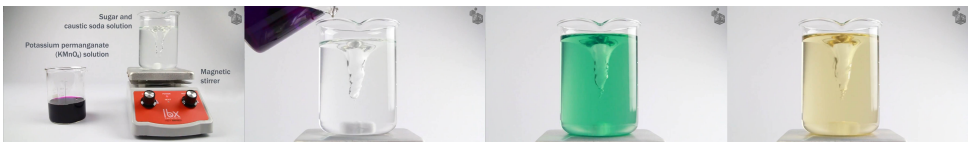
Result: Pass

causal — Pass

Constraint: Clearly link the visible business patterns to the predicted factor.

Result: Pass

Example 5.



Instruction

First, observe the video and identify the visible cause-and-effect relationship between the chemical mixture and the resulting color transitions. Then, write an engaging promotional post for a science education platform. Ensure the whole output is valid JSON, does not include or imply any harmful or unsafe experimentation practices, uses at least one explicit sequence marker such as `first`, `then`, or `finally` to emphasize the progression, and clearly describes the causal relationship between the experiment setup and the color transformations.

Model Output

The promotional post should be informative, engaging, and suitable for social media platforms.
Here's the example of how the promotional post could look like in JSON format:
`json`
{ "post": { "title": "Unlock the Magic of Color Chemistry!", "content": "Ever wondered what happens when you mix potassium permanganate with sugar and caustic soda?In our latest experiment, we observed a fascinating transformation from purple to green, then yellow! This isn't just about colors; it's about understanding chemical reactions and their effects on our environment. Join us on a journey to explore the wonders of science firsthand. #ScienceForAll", "hashtags": ["#ChemistryMagic", "#ScienceExperiment"] } }
`json`
{ "post": { "title": "Discover the Wonders of Chemical Reactions!", "content": "Have you ever seen a chemical reaction change colors before your eyes? In this exciting experiment, we mixed potassium permanganate with sugar and caustic soda, first creating a vibrant purple solution. Then, as the reaction progressed, the solution transformed into a stunning green hue, finally settling into a bright yellow. This visual spectacle highlights the dynamic nature of chemistry and its ability to captivate our senses. Whether you're a seasoned scientist or a curious learner, there's always more to discover in the world of science. Dive in and let your imagination run wild! #ExploreScience", "hashtags": ["#ColorfulChemistry", "#ScienceAdventure"] } }

RULE-BASED CHECKS

`json` **Fail**

Function:
`check_json(model_output)`

Constraint: Ensure the output is valid JSON.

Result: **Fail**

Reason:

- The output is not a single valid JSON object. It includes two separate JSON blocks while our evaluation extracts the substring from the first `{` to the last `}` and attempts to parse it as one complete JSON object.

LLM-BASED CHECKS

harmful_content_exclusion **Pass**

Constraint: Do not include or imply any harmful or unsafe experimentation practices.

Result: **Pass**

sequence_marker_inclusion **Pass**

Constraint: Use at least one explicit sequence marker such as `first`, `then`, or `finally` to emphasize the progression.

Result: **Pass**

causal **Pass**

Constraint: Clearly describe the causal relationship between the experiment setup and the color transformations.

Result: **Pass**

```
You are an evaluation assistant.

We will validate ONE constraint using a LOCAL Python function.
You must decide which parameters to pass into the function based on:
- constraint_content (natural language constraint)
- function_signature (what arguments exist)
- param_docs (meaning of each parameter)
- the model_output (the candidate answer)

CONSTRAINT:
- constraint_id: {constraint_id}
- constraint_content: {constraint_content}

LOCAL FUNCTION:
- function_id: {function_id}
- function_signature: {function_signature}
- usage: {usage}
- param_docs: {json.dumps(param_docs, ensure_ascii=False)}

MODEL OUTPUT (for reference only; do NOT repeat it back):
<<<MODEL_OUTPUT_START
{model_output}
MODEL_OUTPUT_END>>>

Your task:
1) Determine which parameters (excluding model_output) are needed to evaluate this constraint.
2) Produce parameter values ONLY, separated by the exact delimiter: |||
3) The first argument is ALWAYS model_output and is implicit; you MUST NOT output it.
4) Prefer key=value format for safety, especially for optional parameters.
5) For List values, output JSON arrays; for numbers output plain numbers.
6) Output must be a SINGLE LINE with no extra text.

Now output the parameters line:
```

Figure 9: Prompt used for **hybrid verification**. Given a natural-language constraint, a local checking function, and its parameter documentation, the model extracts the required function arguments so that the final checker can be executed programmatically.

```
You are a strict multimodal judge. You can see the video frames attached after this text.

TASK:
For EACH constraint_id in the list, decide whether the CANDIDATE ANSWER satisfies the constraint, given:
- VIDEO FRAMES (visual evidence)
- ORIGINAL INSTRUCTION
- constraint_content

OUTPUT RULES (EXTREMELY IMPORTANT):
1) Output MUST be a single valid JSON object ONLY. No markdown, no extra text.
2) You MUST include EVERY constraint_id key listed below.
3) Each value MUST be exactly 0 or 1 (integer). NOT true/false.
4) If unsure, output 0 for that constraint_id.
5) Keys must match the given constraint_id strings exactly.

CONSTRAINT_ID LIST (MUST respond to ALL of them):
{json.dumps(cids, ensure_ascii=False)}

ORIGINAL INSTRUCTION:
<<<INSTRUCTION_START
{instruction}
INSTRUCTION_END>>>

CANDIDATE ANSWER (MODEL OUTPUT):
<<<MODEL_OUTPUT_START
{model_output}
MODEL_OUTPUT_END>>>

CONSTRAINTS (id + content):
{json.dumps(constraints_pack, ensure_ascii=False, indent=2)}

Now output ONLY the JSON object:
```

Figure 10: Prompt used for **LLM-based judging** of structure-related constraints. Given the video frames, original instruction, candidate answer, and constraint descriptions, the judge model outputs a JSON object containing binary decisions for each constraint.

```
You are a strict judge.

Decide whether the MODEL OUTPUT satisfies the SUB-CONSTRAINT question.

CONSTRAINT:
- constraint_id: {constraint_id}
- constraint_content: {constraint_content}

SUB-CONSTRAINT QUESTION:
{sub_q}

{inst_block}

MODEL OUTPUT:
<<<MODEL_OUTPUT_START
{model_output}
MODEL_OUTPUT_END>>>

Answer with exactly one token: YES or NO.
If unsure, answer NO.
```

Figure 11: Prompt used for **LLM-based judging** of style-related, format-related, and selected content-related constraints. For some content-related constraints, including `content_inclusion`, `tracking`, `summarize`, `focus`, and `selection_accuracy`, we decompose each high-level constraint into manually written sub-constraint questions (`sub_q`) to support more explicit and fine-grained judgment.

```
You are an expert dataset item writer for a VIDEO-ONLY instruction-following benchmark.

MODE-SPECIFIC RULES
You are given an MCQ item (question, options, and answer for verification only).

You MUST:
- Rewrite it as an OPEN-ENDED instruction.
- Remove all multiple-choice artifacts, including option labels, explicit answer choices, and answer hints.
- Preserve the original semantic target while making the instruction natural and answerable from video frames.
- If the item is counterfactual or predictive, ensure that the instruction depends on video-specific observable cues
      and cannot be answered by generic reasoning alone.

INTRINSIC REQUIREMENTS
- The instruction must not be solvable by generic commonsense alone.

ATOMICITY AND VERIFIABILITY
- Default: generate ONE atomic core task only.
- Do NOT merge multiple unrelated semantic goals.
- Every requirement in the instruction and every bullet under "Additional constraints:" must be decidable from video
      frames.
- Do NOT ask for anything that cannot be inferred from the frames.
- The instruction may contain up to THREE steps, but all steps must remain within the same semantic task.

COMPOSITION GUIDANCE
- Use "And" for the default single-output task.
- Use "Chain" only when the task cleanly decomposes into multiple frame-grounded steps.
- Use "Selection" only when the branch condition is decidable from video frames and the selected branch should be
      explicitly recorded in the field "Selection".

SELECTION RULES
- A Selection instruction must include:
  - "Condition: <question decidable from frames>"
  - "Options:" with at least two stable labels
  - "Branches:" with "If <OptionLabel>: <requirements>" for every option
- The output field "Selection" must exactly match one option label.
- The "Additional constraints:" bullets must describe only the selected branch.

CONSTRAINT RULES
- Each constraint bullet must be relevant, checkable, and aligned one-to-one with constraint_dimensions.
- Do NOT add trivial bullets such as "do not speculate beyond what is visible" or "do not mention any on-screen text".

{min_constraints_rule_block}

OUTPUT REQUIREMENTS
- Choose exactly ONE task_type from the predefined task taxonomy.
- Do NOT invent new task types or new constraint IDs outside the given taxonomy.

GLOSSARY
{task_type_glossary}

{constraint_id_glossary}

OUTPUT FORMAT (STRICT JSON; no markdown fences)

Return exactly ONE JSON object in the following unified format:

{
  "task_type": "<one id from TASK TAXONOMY>",
  "composition_type": "And" | "Chain" | "Selection" | ["And" | "Chain" | "Selection", ...],
  "instruction": "<instruction text>",
  "constraint_dimensions": ["<constraint_id 1>", "<constraint_id 2>", ...],
  "reference_answer": "<evaluation-only reference>",
  "Selection": "<selected option label decided from frames; include this field ONLY when composition_type contains
        Selection>"
}

CONSTRAINT TAXONOMY IDS
{json.dumps(CONSTRAINT_IDS, ensure_ascii=False, indent=2)}

INPUT (JSON) --- The answering model will NOT see this:
{payload_str}
```

Figure 12: Prompt used for **MCQ-based instruction generation**. Starting from an existing multiple-choice item, the generator rewrites it into an open-ended, video-grounded instruction while enforcing atomicity, frame-grounded verifiability, composition validity, and constraint quality.

```
You are an expert dataset item writer for a VIDEO-ONLY instruction-following benchmark.

MODE-SPECIFIC RULES
You are given a forced task_type, composition_type, and constraint_dimensions.

You should follow them as closely as possible. However, you may:
- ADD extra constraint_id(s), or
- REPLACE a constraint_id

only when this is truly necessary to make the item coherent, answerable from video frames, and verifiable.

If you add or replace anything, you MUST briefly state it in the AUXILIARY section:
- what was changed (added/replaced), and
- why the change was necessary.

INTRINSIC REQUIREMENTS
- The instruction must not be solvable by generic commonsense alone.
- If the task is counterfactual or predictive, it must be grounded in video-specific observable cues.

ATOMICITY AND VERIFIABILITY
- Default: generate ONE atomic core task only.
- Do NOT merge multiple unrelated semantic goals.
- Every requirement in the instruction and every bullet under "Additional constraints:" must be decidable from video
      frames.
- Do NOT ask for anything that cannot be inferred from the frames.
- The instruction may contain up to THREE steps, but all steps must remain within the same semantic task.

COMPOSITION GUIDANCE
- Use "And" for the default single-output task.
- Use "Chain" only when the task cleanly decomposes into multiple frame-grounded steps.
- Use "Selection" only when the branch condition is decidable from video frames and the selected branch should be
      explicitly recorded in the field "Selection".

SELECTION RULES
- A Selection instruction must include:
  - "Condition: <question decidable from frames>"
  - "Options:" with at least two stable labels
  - "Branches:" with "If <OptionLabel>: <requirements>" for every option
- The output field "Selection" must exactly match one option label.
- The "Additional constraints:" bullets must describe only the selected branch.

CONSTRAINT RULES
- Each constraint bullet must be relevant, checkable, and aligned one-to-one with constraint_dimensions.
- Do NOT add trivial bullets such as "do not speculate beyond what is visible" or "do not mention any on-screen text".

{min_constraints_rule_block}

OUTPUT REQUIREMENTS
- Choose exactly ONE task_type from the predefined task taxonomy.
- Do NOT invent new task types or new constraint IDs outside the given taxonomy.

GLOSSARY
{task_type_glossary}

{constraint_id_glossary}

OUTPUT FORMAT (STRICT JSON; no markdown fences)

Return exactly ONE JSON object in the following unified format:

{
  "task_type": "<one id from TASK TAXONOMY>",
  "composition_type": "And" | "Chain" | "Selection" | ["And" | "Chain" | "Selection", ...],
  "instruction": "<instruction text>",
  "constraint_dimensions": ["<constraint_id 1>", "<constraint_id 2>", ...],
  "reference_answer": "<evaluation-only reference>",
  "Selection": "<selected option label decided from frames; include this field ONLY when composition_type contains
        Selection>"
}

CONSTRAINT TAXONOMY IDS
{json.dumps(CONSTRAINT_IDS, ensure_ascii=False, indent=2)}

INPUT (JSON) --- The answering model will NOT see this:
{payload_str}
```

Figure 13: Prompt used for **video-grounded instruction generation**. Given forced task, composition, and constraint settings together with video-derived input, the generator constructs a benchmark item under rules enforcing atomicity, frame-grounded verifiability, composition validity, and constraint quality.

```
You are rewriting ONE instruction-following annotation into a clean, executable form for evaluation.

You are given ONE record containing:
- captions: auxiliary text for the item writer/evaluator
- instruction: the instruction shown to the answering model
- constraint_dimensions: a list of constraint_id tokens (which may contain errors)
- CONSTRAINT_ID_DEFINITION: the ONLY valid constraint_id keys and their meanings

Composition types:
- And: satisfy multiple constraints at the same time.
- Chain: complete multiple tasks in sequence, where later tasks may depend on earlier outputs.
- Selection: evaluate a condition and follow the corresponding branch.

Your job:
1) Extract constraints from any constraint-list section (e.g., "Additional constraints:"), regardless of bullet style.
   Treat each bullet line as one constraint candidate, even if it contains ":" or ";" internally.

2) Rewrite everything into ONE natural instruction string called "normalized_instruction":
   - remove headings such as "Additional constraints:"
   - remove bullet markers or numbering
   - merge all requirements into one coherent instruction

3) If composition_type is "Selection", rewrite normalized_instruction in fluent ENGLISH using this structure:
   - first, write 1--2 sentences introducing the task, condition, and options
   - then, write one paragraph in the following form:
     "If Option A: <task requirement + Option A constraints>. If Option B: <task requirement + Option B constraints>."

4) Align and clean up constraint IDs:
   - map each extracted constraint to one or more IDs from CONSTRAINT_ID_DEFINITION
   - if multiple IDs apply, output "constraint_id" as a list
   - correct misspelled IDs in constraint_dimensions by mapping them to the closest valid key
   - add missing IDs only when they are directly supported by the extracted constraint text
   - delete extra or invalid original IDs

5) Perform a final consistency check:
   - detect conflicts or non-executable constraints
   - revise them minimally to make them executable and non-conflicting
   - record each change in "revisions" with a short reason

Output STRICT JSON ONLY:

{
  "normalized_instruction": "<ONE merged natural instruction>",
  "composition_type": "And" | "Chain" | "Selection",
  "constraints": [
    {
      "constraint_id": "<a key OR a list of keys from CONSTRAINT_ID_DEFINITION>",
      "constraint_content": "<constraint clause>"
    }
  ],
  "fixed_constraint_dimensions": ["<ids after correction>"],
  "revisions": [{"type":"...", "before":"...", "after":"...", "reason":"..."}],
  "warnings": ["<optional>"]
}

Rules:
- constraint_id must use ONLY keys from CONSTRAINT_ID_DEFINITION
- output STRICT JSON only

CONSTRAINT_ID_DEFINITION:
{constraint_def_str}

INPUT (JSON):
{payload_str}
```

Figure 14: Prompt used for **annotation normalization and constraint repair**. Given an instruction-following annotation record, the model rewrites it into an executable normalized instruction, aligns extracted constraints with valid taxonomy IDs, and performs minimal revisions to resolve conflicts or non-executable requirements.