# Perturb, Repair, Verify: Self-Play Vision-Language Verifiers for Compositional Understanding

**Zijun Huang**[1] **Yingjun Du**[2]* **Cees Snoek**[2]
[1]University of Southern California [2]VisLab, University of Amsterdam

## Abstract

Vision-language verifiers judge whether an image faithfully matches a textual description, yet existing training pipelines typically rely on static supervision from stronger external judges or curated hard-negative data. Such supervision requires external generation or annotation, is teacher-bounded, and is difficult to adapt as model failures evolve. We introduce PROVE (**P**erturb-and-**R**epair **O**ptimization for **V**ision-Language **E**valuation), a self-play framework that trains verifiers through a closed perturb–verify–repair–verify loop. Starting from a known-correct image-caption pair, a Mutator introduces a typed perturbation over objects, attributes, counts, or relations. The verifier then judges the perturbed caption and provides textual feedback, which a Repairer uses to recover a faithful caption for re-verification. These two verification rounds produce structured rewards for informative mismatches and visually grounded repairs. Since perturbation, verification, and repair are role-conditioned behaviors of the same evolving multimodal backbone, updating the shared parameters also improves the verifier itself. Experiments show that PROVE improves fine-grained compositional verification across model scales and families and strengthens visual-detail robustness on the reference backbone, with limited, benchmark-dependent changes on broader multimodal tasks.

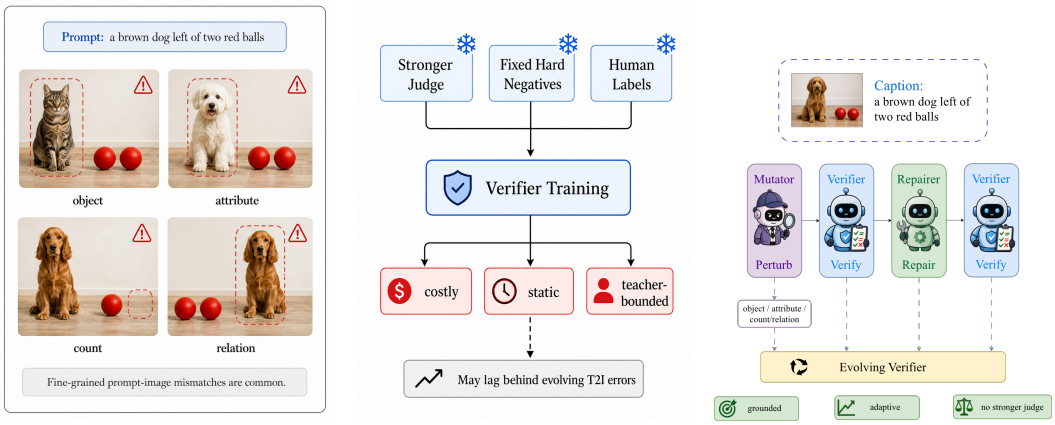
## 1 Introduction

Vision-language models have made rapid progress in image-text understanding, yet they still often miss subtle but consequential mismatches between an image and a textual description. A caption may differ from the image by a substituted object, a misbound attribute, an incorrect count, or a reversed relation, while still retaining high global semantic similarity. As illustrated in Fig. 1a, these errors lie along fine-grained visual-semantic axes that are difficult to capture with coarse image-text scores. This limitation is especially important because VLMs are increasingly used as *verifiers*: given an image and a description, the verifier judges whether the description faithfully matches the image. Such verifiers support VLM-as-judge evaluation [49], reward modeling for RLHF [29] and RLAIF [16], generated-image filtering, and hallucination detection [9, 19]. A central question is therefore how to train verifiers that recognize localized image-text misalignment, rather than merely score global semantic compatibility.

The dominant recipe imports supervision from outside the target model, as summarized in Fig. 1b. A stronger reference judge, typically a larger VLM or a curated process reward model, scores candidate image-text pairs, and the target verifier is trained to imitate or optimize against that score [20, 39, 46]. Alternatively, training relies on human-labeled or synthetically generated hard negatives [14, 17, 30]. Although effective, both routes provide *static* supervision. The student is bounded by the perceptual distinctions exposed by the external judge or negative set; the supervision may lag behind evolving model failures; and extending verification to new axes such as counting, relations, or attribute binding often requires new annotation or generation pipelines. The core challenge is to obtain image-

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).




(a) (b) (c)

Figure 1: **Motivation of PROVE.** (a) Fine-grained image-text verification requires detecting localized object, attribute, count, and relation mismatches. (b) Existing verifier training relies on static external supervision, while (c) PROVE creates adaptive supervision through a perturb–verify–repair–verify self-play loop.

text mismatches that are simultaneously *grounded* in the image, *hard* enough to train fine-grained discrimination, and *gradable* without relying on a stronger external judge.

We introduce PROVE (**P**erturb-and-**R**epair **O**ptimization for **V**ision-Language **E**valuation), a self-play framework that trains a verifier through a closed perturb–verify–repair–verify loop (Fig. 1c). Starting from a known-correct image-caption pair, a Mutator creates a minimally perturbed caption by editing one visual-semantic axis, such as object identity, attribute, count, or relation. The verifier then judges the perturbed caption and provides feedback about the mismatch. Conditioned on this feedback, a Repairer attempts to recover a faithful caption, which is verified again. Crucially, perturbation, verification, and repair are not separate models: they are three role-conditioned behaviors of the same evolving VLM. The two verification rounds produce rewards for informative perturbations and visually grounded repairs, while shared-parameter updates also improve the verifier used at evaluation time.

This formulation is well matched to fine-grained image-text verification. The Mutator does not generate arbitrary text corruptions; it creates controlled, localized mismatches whose semantic axes mirror common compositional failures. The verifier must diagnose the mismatch before repair and recognize the corrected caption after repair, while the Repairer must localize the error and infer the visually faithful description. In this way, PROVE converts ordinary clean image-caption pairs into adaptive hard negatives and repair targets. Compared with external-judge distillation, the training signal is not bounded by a stronger frozen oracle. Compared with fixed hard-negative training, the mismatch distribution evolves with the current verifier.

We evaluate PROVE on a Qwen3.5-2B backbone across fine-grained compositional alignment benchmarks and broader vision-language robustness benchmarks. PROVE consistently improves the same backbone on compositional verification, with the largest gains appearing on axes aligned with the perturbation space, such as attribute and relation understanding. It also improves visual-detail sensitivity and maintains hallucination robustness, while broader capability changes are limited and benchmark-dependent. Training diagnostics further show that perturbation and repair co-adapt through verifier feedback. These results suggest that vision-language verifiers can be strengthened through verifier-guided self-play rather than static judge distillation.

Matched supervision baselines, independent verifier audits, and additional Qwen3.5-4B and InternVL3-2B experiments further support this teacher-free adaptive alternative. The present study establishes caption-based verification and benchmark transfer; task-specific self-play for VQA, grounding, and referring expressions remains future work.

## 2 Related Work

**Vision-language verifiers for image-text understanding.** Evaluating whether an image faithfully matches a textual description is a central problem in vision-language understanding. Coarse image-text similarity metrics are often insufficient, because they can miss fine-grained errors in object identity, attribute binding, counting, and spatial relations. Recent work therefore uses vision-language models as verifiers or judges for multimodal evaluation, reward modeling, generated-image filtering, and hallucination detection [8, 9, 19, 49]. These verifiers are attractive because they can provide semantic judgments beyond fixed embedding similarities, but their reliability depends on whether they can detect subtle image-text mismatches. PROVE targets this capability directly by training the verifier through controlled perturbation, verification, and repair over the same compositional axes that commonly challenge image-text understanding.

**Training verifiers with static external supervision.** A common way to train multimodal verifiers is to import supervision from outside the target model. For example, stronger VLMs or curated reward models can score candidate image-text pairs, after which the target model is optimized by preference learning or reward-based training [18, 20, 37–39, 46]. Another line constructs fixed hard-negative datasets or synthetic preference pairs, then trains the verifier to distinguish faithful captions from controlled corruptions [21]. These approaches provide useful supervision, but the signal is static: it is bounded by the external judge, fixed negative generator, or annotation pipeline. As models and evaluation distributions evolve, their failure modes may shift away from this supervision. In contrast, PROVE does not distill a stronger frozen judge or rely on a fixed hard-negative set. It uses the current verifier itself inside a perturb–verify–repair–verify loop, so the training signal changes with the evolving model.

**Self-play and self-evolving multimodal learning.** Self-play has been widely used to improve agents and reasoning systems, from game-playing methods such as AlphaZero [36] to debate and recent self-evolving LLMs [12, 13, 47]. More recent asymmetric self-play methods train one behavior to propose tasks and another to solve them, creating an adaptive curriculum without a fixed external teacher [23]. In multimodal learning, self-evolving VLMs have also been explored through questioner–reasoner interactions and pseudo-labeling from model samples [10]. However, consensus- or uncertainty-based supervision can still reinforce confident mistakes, especially when the model fails to perceive the relevant visual evidence. PROVE differs in both task and interaction structure: it trains a vision-language verifier rather than a general reasoner, and it grounds self-play in clean image-caption pairs. The Mutator creates typed image-text mismatches, the verifier diagnoses them, and the Repairer attempts to correct them; all three are role-conditioned behaviors of the same evolving MLLM.

**Compositional image-text alignment.** Vision-language models often struggle with compositional understanding despite strong global image-text alignment. Benchmarks such as Winoground [40], ARO [44], SugarCrepe [11], and VL-CheckList [48] test whether models can distinguish captions that differ in objects, attributes, relations, or word order. Prior methods improve this ability with hard-negative contrastive learning or synthetic data [7, 44]. PROVE is complementary: it generates controlled perturbations online and uses verifier-guided repair to create adaptive supervision for fine-grained image-text alignment.

## 3 Methods

### 3.1 Overview

PROVE trains a vision-language verifier through a closed perturb–verify–repair–verify loop. Given a clean image-caption pair, a single multimodal policy $\pi_\theta$ is reused under three role prompts: a Mutator that perturbs the caption, a Verifier that judges image-caption faithfulness, and a Repairer that corrects the caption using verifier feedback. These roles are not separate models; they are three behaviors of the same parameter-shared backbone. Thus, every policy update also changes the verifier used at evaluation time. Figure 2 illustrates the overall loop.

The key idea is to turn ordinary positive image-caption pairs into adaptive verification episodes. The Mutator creates localized mismatches along object, attribute, count, or relation axes. The Verifier diagnoses whether the perturbed caption still matches the image. The Repairer then attempts to recover a faithful caption from the image and verifier feedback. This interaction produces grounded hard negatives and repair targets without a stronger external judge or a fixed hard-negative dataset.

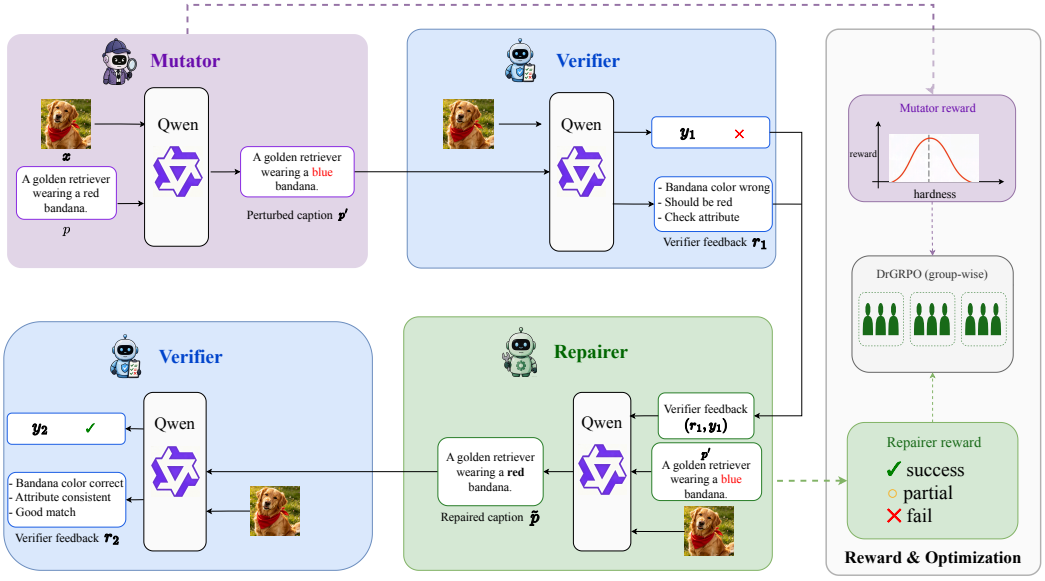



Figure 2: **Overview of PROVE.** A single Qwen backbone is reused under three role prompts: Mutator, Verifier, and Repairer. Given a clean image-caption pair, the Mutator produces a localized caption perturbation, e.g., changing an attribute from "red" to "blue", to create a hard image-text mismatch. The Verifier first judges the perturbed caption and returns a decision-feedback pair $(y_1, r_1)$ that diagnoses the mismatch. Conditioned on this feedback, the Repairer generates a revised caption, which is then checked by the same Verifier to obtain $(y_2, r_2)$. Rewards from the two verification rounds train the shared model with DrGRPO, so perturbation, verification, and repair co-evolve within one multimodal backbone.

### 3.2 Perturb–Verify–Repair–Verify Self-Play

Let $\mathcal{D} = \{(x_i, p_i)\}_{i=1}^{N}$ be a set of image-caption pairs, where $p_i$ faithfully describes image $x_i$. For a clean pair $(x, p)$, PROVE constructs the following self-play episode:

$$(x, p) \xrightarrow{\text{Mutator}} p' \xrightarrow{\text{Verifier}} (y_1, r_1) \xrightarrow{\text{Repairer}} \tilde{p} \xrightarrow{\text{Verifier}} (y_2, r_2). \tag{1}$$

Here $p'$ is a perturbed caption, $(y_1, r_1)$ is the first verifier decision and explanation, $\tilde{p}$ is the repaired caption, and $(y_2, r_2)$ is the second verifier decision and explanation. The binary label $y \in \{0, 1\}$ indicates whether the caption is judged faithful to the image, and $r$ is natural-language feedback.

All stages are sampled from the same policy under different role prompts:

$$\pi_\theta(\cdot \mid x, q, z), \qquad z \in \{\textsc{Mutator}, \textsc{Verifier}, \textsc{Repairer}\}, \tag{2}$$

where $q$ denotes the role-specific text context. This makes the verifier part of the evolving policy, rather than a frozen reward model.

**Caption perturbation.** The Mutator produces a localized edit

$$a_M = (\tau, t, \delta), \tag{3}$$

where $\tau$ is the edit type, $t$ is the target span, and $\delta$ specifies the replacement. We use four edit types:

$$\mathcal{T} = \{\textsf{object}, \textsf{attribute}, \textsf{count}, \textsf{relation}\}. \tag{4}$$

A deterministic editor applies the edit to obtain the perturbed caption:

$$p' = E(p, a_M). \tag{5}$$

Restricting the Mutator to typed local edits prevents arbitrary rewriting and keeps the negative caption close to the original image-caption pair. Invalid edit types, unmatched or ambiguous target spans, unchanged replacements, and non-deterministic edits are filtered; Appendix F describes the role interfaces and editor constraints.

**Verification and repair.** The Verifier first judges the perturbed caption:

$$(y_1, r_1) \sim \pi_\theta(\cdot \mid x, p', \textsc{Verifier}). \tag{6}$$

The feedback $r_1$ identifies the perceived mismatch and is passed to the Repairer, which generates a corrected caption:

$$\tilde{p} \sim \pi_\theta(\cdot \mid x, p', y_1, r_1, \textsc{Repairer}). \tag{7}$$

The same Verifier then evaluates the repaired caption:

$$(y_2, r_2) \sim \pi_\theta(\cdot \mid x, \tilde{p}, \textsc{Verifier}). \tag{8}$$

The first verification tests whether the model detects the perturbation; the second tests whether the repair restores faithfulness. Together, they turn verification into an operational correction task.

### 3.3 Verifier-Guided Rewards

The two verifier calls provide the supervision for self-play. The Mutator should produce perturbations that are informative, grounded, and not overly destructive. The Repairer should recover a faithful caption while making minimal changes.

**Mutator reward.** We define the Mutator reward as

$$r_M = R_{\text{expose}}(y_1) + R_{\text{hard}}(y_2) - \lambda_M C_M(a_M), \tag{9}$$

where $R_{\text{expose}}$ rewards perturbations that expose verifier blind spots, $R_{\text{hard}}$ rewards perturbations that remain difficult after repair, and $C_M$ penalizes overly wide edits. Invalid actions receive a fixed penalty and do not enter $C_M$ (Appendix F). We instantiate it as

$$r_M = \lambda_1 \mathbb{1}[y_1 = 1] + \lambda_2 \mathbb{1}[y_2 = 0] - \lambda_3 C_M(a_M). \tag{10}$$

**Repairer reward.** The Repairer reward encourages a successful and minimal correction:

$$r_R = R_{\text{accept}}(y_2) + R_{\text{flip}}(y_1, y_2) - \lambda_R C_R(p', \tilde{p}), \tag{11}$$

where $R_{\text{accept}}$ rewards repaired captions accepted by the verifier, $R_{\text{flip}}$ rewards changing the verifier judgment, and $C_R$ penalizes unnecessary rewriting. We use

$$r_R = \beta_1 \mathbb{1}[y_2 = 1] + \beta_2 \mathbb{1}[y_1 \neq y_2] - \beta_3 C_R(p', \tilde{p}). \tag{12}$$

Although rewards are assigned to perturbation and repair trajectories, the Verifier is not a frozen judge. Because all roles share $\theta$, updates from these rewards also update the verifier behavior itself. This coupling is what allows PROVE to improve the verifier without distilling a stronger external model.

### 3.4 Role-Conditioned Policy Optimization

We optimize the shared policy with DrGRPO [27]. For each source pair, the model samples multiple self-play trajectories and computes verifier-guided rewards. Let $r_i$ be the reward of trajectory $i$, and let $G(i)$ be the group of trajectories sampled from the same source context. Following DrGRPO, we use mean-centered advantages without standard-deviation normalization:

$$\hat{A}_i = r_i - \frac{1}{|G(i)|} \sum_{j \in G(i)} r_j. \tag{13}$$

This preserves the reward scale and avoids unstable rescaling when within-group variance is small.

The clipped objective over generated tokens is

$$\mathcal{L}(\theta) = \mathbb{E}_{i,t}\left[\min\left(\rho_{i,t}(\theta)\hat{A}_i, \text{clip}\big(\rho_{i,t}(\theta), 1-\epsilon^-, 1+\epsilon^+\big)\,\hat{A}_i\right)\right], \tag{14}$$

where

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t} \mid q_i, a_{i,<t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid q_i, a_{i,<t})}. \tag{15}$$

Here $q_i$ is the role-conditioned multimodal prompt and $a_{i,t}$ is the generated token. Equation 14 is a maximization objective; equivalently, its negative is minimized, with KL regularization specified in the experimental setup. At test time, we use the Verifier prompt of the evolved shared model to judge image-caption faithfulness.

# 4 Experiments

## 4.1 Experimental Setup

**Backbone and training framework.** We instantiate PROVE with Qwen3.5-2B [31] as the shared multimodal backbone. The same parameters are reused across the Mutator, Verifier, and Repairer roles, with role identity specified only by the prompt template. Thus, perturbation, verification, and repair are optimized as role-conditioned behaviors of one evolving MLLM rather than as separate models. At evaluation time, we use the Verifier role of the evolved shared model to judge image-caption faithfulness.

**Training data and self-play construction.** We train on a 48k-example subset of ShareGPT-4o-Image-style image-caption pairs [4], with a held-out validation split of 5k examples. Each self-play episode starts from a known-correct image-caption pair. The Mutator introduces a typed perturbation along one of four visual-semantic axes: object, attribute, count, or relation. The Verifier then judges the perturbed caption and produces a decision-feedback pair. Conditioned on this feedback, the Repairer generates a repaired caption, which is judged again by the same Verifier role. The two verification rounds provide verifier-guided rewards for self-play optimization. No stronger external judge, human-labeled hard negatives, or additional hard-negative annotations are used during training.

**Optimization and compute.** We implement PROVE using the VeRL/HybridFlow training framework [35] and optimize the shared policy with DrGRPO [27], a group-relative policy optimization variant that removes the standard-deviation normalization used in vanilla GRPO [34]. Unless otherwise specified, we use rollout size $n = 3$, learning rate $5 \times 10^{-7}$, global batch size 48, KL coefficient $0.005$, and PPO clip range $[0.20, 0.22]$. All main training runs are conducted on a single node with 8 NVIDIA A100 GPUs using BF16 mixed precision. We report the best validation checkpoint in the main results. Training hyperparameters, role interfaces, and additional experimental results are provided in the appendix. The reference run takes 11 hours on 8 A100 GPUs (88 GPU-hours); Appendix A.4 compares complete pipeline costs. We will release the training and evaluation code to facilitate reproducibility.

**Evaluation protocol.** Our main evaluation focuses on fine-grained image-text understanding, where a verifier must distinguish captions that differ by localized semantic changes. Table 1 compares PROVE with representative compositional understanding methods on four benchmarks shared with [30]: ARO [44], VL-CheckList [48], ColorSwap [3], and FG-OVD [1]. These benchmarks cover relation and attribute reasoning, object-level grounding, color-word order sensitivity, and fine-grained open-vocabulary recognition. For each baseline, we report the values published in the original paper whenever available; when a benchmark was not reported but public code is available, we use our reproduction under the same evaluation protocol.

Beyond the core compositional suites, we evaluate how the learned verifier transfers to broader vision-language understanding. We include Winoground [40] to test role binding under hard image-text contrasts, MMVP [41] to assess sensitivity to fine visual details, and MMBench-EN [25] to measure general multimodal perception and reasoning. For all two-alternative forced-choice tasks, we randomize option order for each example to reduce positional bias. Unless otherwise stated, all benchmarks are evaluated using their official protocols and metrics. Comp. Avg. averages the four core benchmark groups in Table 1; Broader Avg. averages Winoground GS, MMVP pair accuracy, and MMBench in Table 2. Winoground GS and the appendix 2-AFC accuracy are distinct metrics. External evaluation labels are not used for training, rewards, mutation generation, validation, or checkpoint selection.

## 4.2 Main Results

**Compositional understanding.** Table 1 compares PROVE with representative compositional understanding methods on ARO, VL-CheckList, ColorSwap, and FG-OVD. Because these methods differ in backbone architecture, training data, objectives, and benchmark coverage, the cross-method comparison should be interpreted as a contextual reference rather than a fully controlled comparison. Our main controlled comparison is between Qwen3.5-2B and PROVE using the same backbone. Under this controlled setting, PROVE improves the overall average from 83.22 to 86.85, with consistent gains across all four benchmark groups. The largest improvement appears on ARO, where the average score increases by $+7.25$, indicating stronger attribute and relation discrimination.

Table 1: **Comparison with compositional understanding methods.** We report all available sub-metrics on ARO, VL-CheckList, ColorSwap, and FG-OVD. "–" indicates that the benchmark was not evaluated. Best and second-best results are shown in **bold** and underline. PROVE achieves the best overall average among methods with complete benchmark coverage and consistently improves over the Qwen3.5-2B backbone across all benchmark groups.

| Model | ARO | | | VL-CheckList | | | | ColorSwap | FG-OVD | Overall |
|---|---|---|---|---|---|---|---|---|---|---|
| | Rel | Attr | Avg | Attr | Obj | Rel | Avg | ITT | Avg | Avg |
| *Contextual comparison: discriminative VLMs* | | | | | | | | | | |
| CLIP [32] | 59.00 | 62.00 | 60.50 | 67.93 | 82.83 | 64.19 | 71.65 | 35.67 | 47.33 | 53.79 |
| NegCLIP [44] | 81.00 | 71.00 | 76.00 | 72.24 | 87.00 | 71.39 | 76.87 | 35.67 | 41.69 | 57.56 |
| GNM [33] | 65.00 | 65.00 | 65.00 | 70.15 | 85.91 | 64.10 | 73.38 | 13.00 | 38.79 | 47.54 |
| PANeg [2] | 65.07 | 67.94 | 66.51 | 76.51 | 88.13 | 69.90 | 78.17 | 17.67 | 44.98 | 51.83 |
| DAC-SAM [7] | 77.16 | 70.50 | 73.83 | 75.80 | 88.50 | 89.80 | 84.70 | 16.33 | 48.36 | 55.81 |
| DAC-LLM [7] | 81.28 | 73.91 | 77.60 | 77.30 | 87.30 | 86.40 | 83.66 | 18.33 | 49.60 | 57.30 |
| XVLM [45] | 73.40 | 86.80 | 80.10 | 75.10 | 85.80 | 70.40 | 76.50 | – | – | – |
| *Contextual comparison: generative VLMs* | | | | | | | | | | |
| InstructBLIP [6] | 69.20 | 50.83 | 60.02 | 56.37 | 80.33 | 53.34 | 63.35 | 40.33 | 26.80 | 47.63 |
| MiniGPT-4 [50] | 46.90 | 55.70 | 51.30 | 71.30 | 84.20 | – | – | – | – | – |
| BLIP2-VGPS [22] | **90.70** | **94.30** | **92.50** | 73.90 | 90.10 | **89.90** | 84.63 | – | – | – |
| CapPa [42] | 86.70 | 85.70 | 86.20 | – | – | – | – | – | – | – |
| LLaVA-1.5-7B [24] | 65.08 | 57.40 | 61.24 | 69.49 | – | 73.50 | – | – | – | – |
| InternVL2-8B [5] | 83.42 | 93.74 | 88.58 | **84.81** | – | 87.10 | – | – | – | – |
| *Matched comparison: shared-backbone verifier models* | | | | | | | | | | |
| Qwen3.5-2B [31] | 69.68 | 79.48 | 74.58 | 76.75 | 91.20 | 82.85 | 83.60 | 93.67 | 81.01 | 83.22 |
| PROVE (Qwen3.5-2B) | 76.92 | 86.74 | 81.83 | 81.46 | **94.14** | 86.95 | **87.52** | **95.83** | **82.23** | **86.85** |

Table 2: **Comparison on broader vision-language benchmarks.** PROVE achieves the best average and improves the Qwen3.5-2B backbone on all three benchmarks, indicating transfer beyond the core compositional suites. Broader task changes are benchmark-dependent (Appendix C.3).

| Model | Wino GS | MMVP Pair | MMBench | Avg |
|---|---|---|---|---|
| *Discriminative VLMs* | | | | |
| CLIP [32] | 8.00 | 15.55 | – | – |
| NegCLIP [44] | 8.00 | 26.67 | – | – |
| PANeg [2] | 6.75 | 21.48 | – | – |
| DAC-SAM [7] | 6.25 | 23.70 | – | – |
| DAC-LLM [7] | 3.25 | 28.14 | – | – |
| *Generative VLMs* | | | | |
| InstructBLIP [6] | 4.75 | 14.81 | – | – |
| LLaVA-1.5-7B [24] | 27.00 | 24.70 | 64.30 | 38.67 |
| InternVL2-8B [5] | 49.75 | 51.33 | **81.70** | 60.93 |
| *Shared-backbone verifier models* | | | | |
| Qwen3.5-2B [31] | 49.00 | 56.00 | 71.31 | 58.77 |
| PROVE (Qwen3.5-2B) | **50.75** | **60.00** | 72.34 | **61.03** |

PROVE also improves VL-CheckList, ColorSwap, and FG-OVD, suggesting that the learned verifier benefits not only the perturbation axes seen during self-play but also broader forms of localized image-text grounding. Compared with prior compositional methods, PROVE achieves the best overall average among complete rows in this table and remains competitive on individual benchmark groups, despite using a compact shared backbone and no external hard-negative dataset. These results support the core design of PROVE. The Mutator creates controlled mismatches over objects, attributes, counts, and relations; the Verifier learns to diagnose these mismatches; and the Repairer learns to recover visually faithful captions. The resulting self-play signal strengthens fine-grained image-text verification without relying on a stronger external judge.

**Broader vision-language understanding benchmarks.** Table 2 evaluates whether the learned verifier transfers beyond the main compositional suites. PROVE improves the Qwen3.5-2B backbone on all three benchmarks, raising the average from 58.77 to 61.03. The most notable gain is on MMVP, where pair accuracy improves by $+4.00$, suggesting improved sensitivity to fine visual

Table 3: **Matched supervision and core loop ablations.** All training variants use Qwen3.5-2B, 48k source pairs, and the same target-model optimization budget and evaluation protocol. Complete ablations are in Appendix B.

| Method / variant | Comp. Avg. | Broader Avg. |
|---|---|---|
| Qwen3.5-2B | 83.22 | 58.77 |
| Fixed typed negatives | 85.02 | 59.68 |
| External-judge distillation | 86.29 | 60.72 |
| Verifier self-training | 85.66 | 60.21 |
| No Repairer or second verification | 84.24 | 59.31 |
| Repairer without verifier feedback | 85.17 | 59.84 |
| Frozen internal Verifier | 85.12 | 59.81 |
| PROVE | **86.85** | **61.03** |

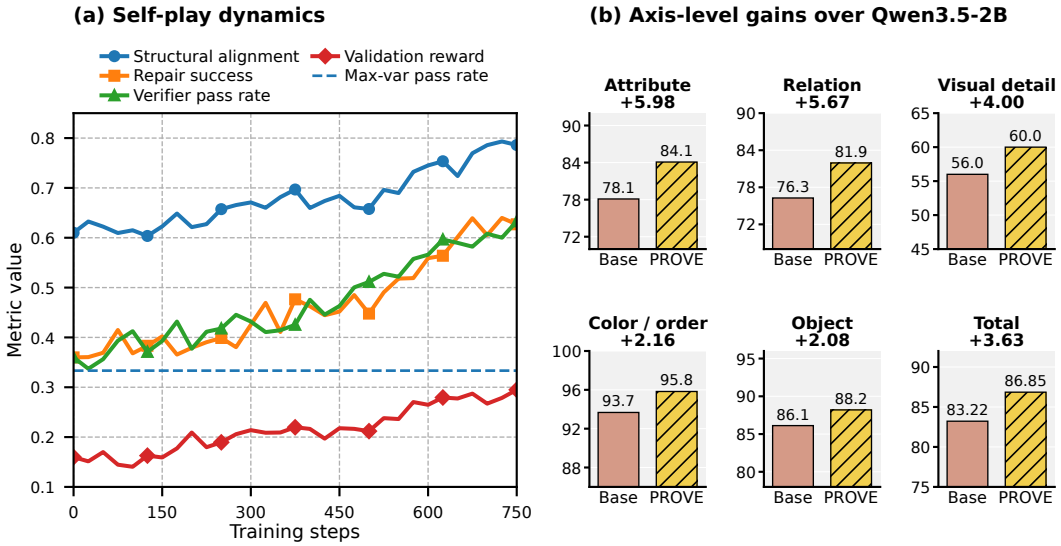



Figure 3: **Self-play dynamics and axis-level gains.** (a) The Repairer steadily improves in repair success and structural alignment, while the verifier pass rate rises as repairs catch up with the current Mutator distribution; the dashed line marks the lower finite-group maximum-variance reference at $1/3$ (Appendix F). (b) Each panel compares Qwen3.5-2B with PROVE on an aggregated visual-semantic capability; Total is the overall average in Table 1. The largest gains appear on attribute and relation verification, matching the typed perturbation axes used during self-play.

details. Winoground also improves, indicating better role binding under hard image-text contrasts, while MMBench-EN shows a modest positive change, suggesting positive transfer to this general multimodal perception and reasoning benchmark.

Together, Tables 1 and 2 show that PROVE mainly strengthens the verifier on localized image-text discrimination, which matches the intended training signal of typed perturbation and verifier-guided repair. At the same time, the gains on MMVP and MMBench indicate positive transfer, while repeated-run results reveal small, task-dependent regressions on other broader benchmarks (Appendix C.3).

**Matched supervision alternatives.** Table 3 compares static negatives generated once, judgments from a frozen larger VLM, and self-generated verifier labels without repair. PROVE improves the compositional/broader averages over these baselines by $1.83/1.35$, $0.56/0.31$, and $1.19/0.82$ points, respectively. These results support adaptive perturb–repair self-play as a teacher-free alternative; its advantage is adaptive supervision rather than uniformly lower compute or universal superiority over distillation.

Table 4: **Transfer across model scale and family.** The same perturbation and reward formulation improves both evaluation averages.

| | Comp. Avg. | | Broader Avg. | |
|---|---|---|---|---|
| Backbone | Base | PROVE | Base | PROVE |
| Qwen3.5-2B | 83.22 | **86.85** | 58.77 | **61.03** |
| Qwen3.5-4B | 85.15 | **88.71** | 60.92 | **63.26** |
| InternVL3-2B | 82.19 | **85.34** | 58.16 | **60.99** |

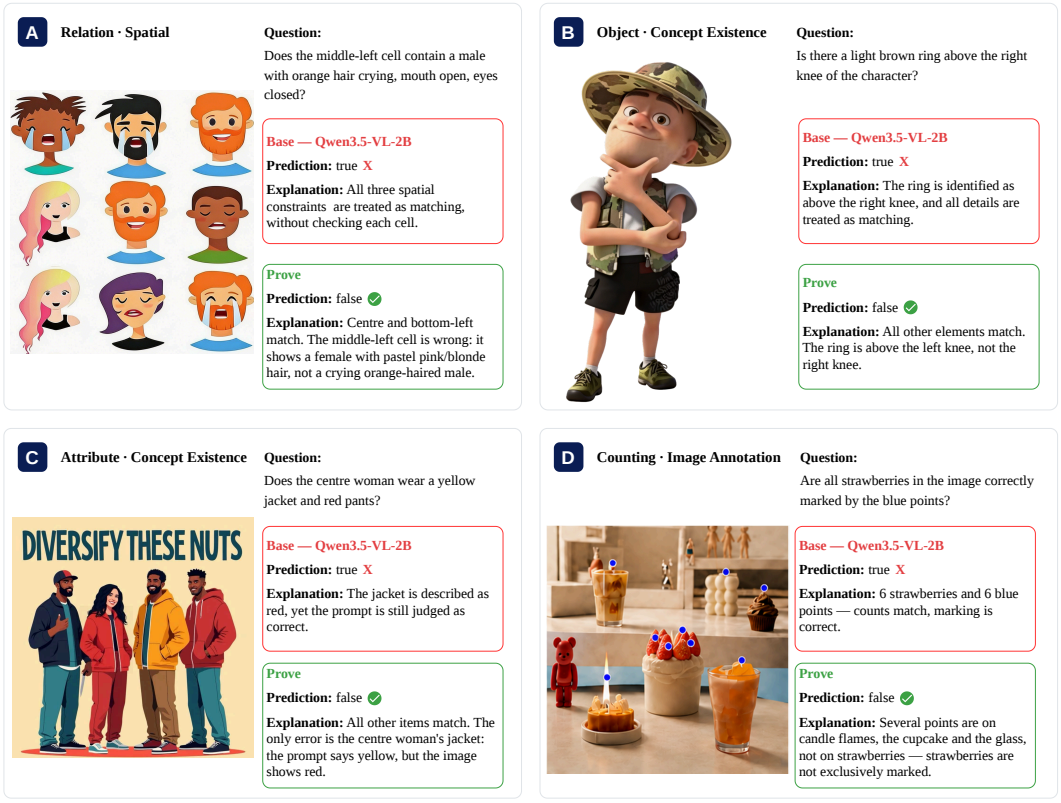



Figure 4: **Qualitative verifier examples.** Each panel compares the base Qwen3.5-2B model with PROVE on a fine-grained verification case. The examples cover relation, object existence, attribute, and counting errors. PROVE better localizes the relevant visual evidence and avoids accepting captions or answers that miss the targeted mismatch.

### 4.3 Analysis

**Self-Play Dynamics Analysis.** Figure 3(a) analyzes how the perturb–verify–repair loop evolves during training. The training dynamics show that the Repairer steadily catches up with the perturbations produced by the Mutator while the overall optimization remains stable. Repair success increases from $36.0\%$ to $62.7\%$, and structural alignment rises from $61.0\%$ to $78.6\%$, indicating that the Repairer becomes increasingly capable of localizing Mutator-induced mismatches and recovering visually faithful captions. At the same time, the verifier pass rate increases from $0.360$ to $0.633$ and remains above the lower finite-group maximum-variance reference point of $1/3$, suggesting that the current four-type perturbation space gradually becomes easier for the evolving Repairer to solve. This trend reveals both the effectiveness and the limitation of the current self-play curriculum: typed perturbations provide a useful learning signal, but a richer action vocabulary may be needed once the Repairer becomes strong enough to saturate the existing perturbation space. Finally, the validation reward improves consistently, suggesting that the verifier-guided rewards provide a stable learning signal.

**Axis-level gains.** Figure 3(b) summarizes where the gains of PROVE come from by aggregating benchmark splits according to the visual-semantic capability they test. The largest improvements appear on attribute and relation verification, which are directly aligned with the typed perturbations used by the Mutator. The full analysis is provided in Appendix G.1.

**Controlled loop ablations.** Removing the Repairer and second verification lowers the two averages by $2.61/1.72$ points; removing verifier feedback costs $1.68/1.19$ (Table 3). Freezing the internal Verifier costs $1.73/1.22$, supporting verifier co-evolution. Reward and role-sharing ablations also favor the full loop, while replacing DrGRPO with GRPO costs only $0.43/0.32$ (Appendix B).

**Independent verification.** Table 12 shows a $7.8$-point verifier accuracy gain and a $23.4$-point increase in fully valid trajectories, requiring a real mismatch, correct diagnosis, faithful minimal repair, and correct final judgment. Fixed external benchmark averages also improve by $3.63/2.26$ points across training; their correlations with self-play pass rate are $0.9254/0.9022$ (Appendix C.2). These independent measurements support improved image-grounded correctness beyond internal role agreement.

**Generality and capability boundary.** Table 4 shows gains of $3.56/2.34$ on Qwen3.5-4B and $3.15/2.83$ on InternVL3-2B. Three Qwen3.5-2B runs yield a compositional average of $86.73 \pm 0.27$, but ViVerBench, MMMU, and MathVista regress while MMBench improves and POPE remains stable (Appendix C.3). Multi-edit curricula, unseen perturbation types, and caption filtering are studied in Appendices D and E. Figure 4 illustrates the learned verifier behavior. Additional failure analysis and task extensions are provided in Appendix G.

**Limitations.** PROVE uses four types of localized caption perturbations, which align with the evaluated compositional axes and may saturate as the Repairer improves. Although independent audits support improved visual correctness, an evolving shared Verifier can still reinforce its own perceptual mistakes. Training depends on faithful starting captions from one corpus and requires four role-conditioned calls per trajectory, increasing compute relative to static supervision. Broader capability changes are benchmark-dependent, including regressions on ViVerBench, MMMU, and MathVista. Task-specific self-play for VQA, grounding, and referring expressions remains untested. Imperfect verifiers can also propagate errors when used for automated evaluation or filtering; these uses require independent validation.

## 5 Conclusion

We introduced PROVE, a self-play framework for training vision-language verifiers through a closed perturb–verify–repair–verify loop. Instead of relying on stronger external judges, fixed hard-negative datasets, or static preference annotations, PROVE converts ordinary clean image-caption pairs into adaptive verification episodes. A single shared MLLM acts as a Mutator, Verifier, and Repairer under different role prompts: the Mutator creates localized image-text mismatches, the Verifier diagnoses them, and the Repairer attempts to recover faithful captions from visual evidence and verifier feedback. This design turns verification into an operational self-improvement problem, where the model learns not only to judge image-text faithfulness but also to expose and correct its own fine-grained failure modes. The resulting training signal is grounded, adaptive, and verifier-guided. Typed perturbations over objects, attributes, counts, and relations provide controllable supervision for compositional understanding, while the two verification rounds produce rewards that update the same backbone used at test time. Experiments on Qwen3.5-2B, Qwen3.5-4B, and InternVL3-2B backbones show that PROVE improves fine-grained compositional understanding across multiple benchmarks. On the reference backbone, it also strengthens visual-detail sensitivity, with limited, benchmark-dependent changes on broader multimodal tasks. These results suggest that self-play over perturbation, verification, and repair offers a practical alternative to static judge distillation for building stronger vision-language verifiers.

## Acknowledgments

This work was supported by the European Union's Horizon Europe research and innovation programme under grant agreement number 101214398 (ELLIOT).

## References


[1] Lorenzo Bianchi, Fabio Carrara, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi. The devil is in the fine-grained details: Evaluating open-vocabulary object detectors for fine-grained understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. arXiv:2311.17518.

[2] Kyle Buettner and Adriana Kovashka. Investigating the role of attribute context in vision-language models for object recognition and detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024. arXiv:2303.10093.

[3] Jirayu Burapacheep, Ishan Gaur, Agam Bhatia, and Tristan Thrush. ColorSwap: A color and word order dataset for multimodal evaluation. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. arXiv:2402.04492.

[4] Junying Chen, Zhenyang Cai, Pengcheng Chen, Shunian Chen, Ke Ji, Xidong Wang, Yunjin Yang, and Benyou Wang. ShareGPT-4o-Image: Aligning multimodal models with GPT-4o-level image generation, 2025. URL `https://arxiv.org/abs/2506.18095`.

[5] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, Ji Ma, Jiaqi Wang, Xiaoyi Dong, Hang Yan, Hewei Guo, Conghui He, Botian Shi, Zhenjiang Jin, Chao Xu, Bin Wang, Xingjian Wei, Wei Li, Wenjian Zhang, Bo Zhang, Pinlong Cai, Licheng Wen, Xiangchao Yan, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. How far are we to GPT-4V? closing the gap to commercial multimodal models with open-source suites, 2024. URL `https://arxiv.org/abs/2404.16821`. Technical report for the InternVL2 series.

[6] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.06500.

[7] Sivan Doveh, Assaf Arbelle, Sivan Harary, Roei Herzig, Donghyun Kim, Paola Cascante-Bonilla, Amit Alfassy, Rameswar Panda, Raja Giryes, Rogerio Feris, Shimon Ullman, and Leonid Karlinsky. Dense and aligned captions (DAC) promote compositional reasoning in VL models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.19595.

[8] Yingjun Du, Wenfang Sun, and Cees G Snoek. Ipo: Interpretable prompt optimization for vision-language models. *Advances in Neural Information Processing Systems*, 37:126725–126766, 2024.

[9] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusion-Bench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385, 2024. arXiv:2310.14566.

[10] Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. VisPlay: Self-evolving vision-language models from images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2511.15661.

[11] Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. SugarCrepe: Fixing hackable benchmarks for vision-language compositionality. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. arXiv:2306.14610.

[12] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2508.05004.

[13] Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate, 2018. URL `https://arxiv.org/abs/1805.00899`.

[14] Zexi Jia, Chuanwei Huang, Hongyan Fei, Yeshuang Zhu, Zhiqiang Yuan, Ying Deng, Jiapei Zhang, Jinchao Zhang, and Jie Zhou. A visual leap in CLIP compositionality reasoning through generation of counterfactual sets. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23498–23507, 2025. arXiv:2507.04699.

[15] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European Conference on Computer Vision (ECCV)*, pages 235–251, 2016.

[16] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2309.00267.

[17] Haoxin Li and Boyang Li. Enhancing vision-language compositional understanding with multimodal synthetic data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24849–24861, 2025. arXiv:2503.01167.

[18] Lei Li, Zhihui Xie, Mukai Li, Shunian Chen, Peiyi Wang, Liang Chen, Yazheng Yang, Benyou Wang, Lingpeng Kong, and Qi Liu. VLFeedback: A large-scale AI feedback dataset for large vision-language models alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. arXiv:2410.09421.

[19] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. arXiv:2305.10355.

[20] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.20050.

[21] Inna Wanyin Lin, Yushi Hu, Shuyue Stella Li, Scott Geng, Pang Wei Koh, Luke Zettlemoyer, Tim Althoff, and Marjan Ghazvininejad. Self-improving VLM judges without human annotations, 2025. URL `https://arxiv.org/abs/2512.05145`.

[22] Zhiqiu Lin, Xinyue Chen, Deepak Pathak, Pengchuan Zhang, and Deva Ramanan. Revisiting the role of language priors in vision-language models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2306.01879; introduces the VisualGPTScore for visio-linguistic reasoning.

[23] Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: Self-play in corpus environments improves reasoning, 2025. URL `https://arxiv.org/abs/2510.24684`.

[24] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26296–26306, 2024. arXiv:2310.03744.

[25] 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. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision (ECCV)*, pages 216–233, 2024. arXiv:2307.06281.

[26] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12), 2024. doi: 10.1007/s11432-024-4235-6.

[27] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025. Proposes Dr.GRPO, which removes the standard-deviation normalization in GRPO; arXiv:2503.20783.

[28] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.02255.

[29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2203.02155.

[30] Fiorenzo Parascandolo, Nicholas Moratelli, Enver Sangineto, Lorenzo Baraldi, and Rita Cucchiara. Causal graphical models for vision-language compositional understanding. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.09353.

[31] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, February 2026.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. arXiv:2103.00020.

[33] Ugur Sahin, Hang Li, Qadeer Khan, Daniel Cremers, and Volker Tresp. Enhancing multimodal compositional reasoning of visual language models with generative negative mining. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5551–5561, 2024. arXiv:2311.03964.

[34] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`. Introduces Group Relative Policy Optimization (GRPO).

[35] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, pages 1279–1297, 2025. doi: 10.1145/3689031.3696075.

[36] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[37] Wenfang Sun, Hao Chen, Yingjun Du, Yefeng Zheng, and Cees G Snoek. Regionreasoner: Region-grounded multi-round visual reasoning. In *International Conference on Learning Representations*, volume 2026, pages 80583–80603, 2026.

[38] Wenfang Sun, Yingjun Du, and Cees GM Snoek. Less from more: Reinforcing sparse video reasoning from dense references. *arXiv preprint arXiv:2610.10893*, 2026.

[39] Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, Kurt Keutzer, and Trevor Darrell. Aligning large multimodal models with factually augmented RLHF. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. arXiv:2309.14525.

[40] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5238–5248, 2022.

[41] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal LLMs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9568–9578, 2024. arXiv:2401.06209.

[42] Michael Tschannen, Manoj Kumar, Andreas Steiner, Xiaohua Zhai, Neil Houlsby, and Lucas Beyer. Image captioners are scalable vision learners too. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.07915.

[43] 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, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567, 2024. arXiv:2311.16502.

[44] Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.01936.

[45] Yan Zeng, Xinsong Zhang, and Hang Li. Multi-grained vision language pre-training: Aligning texts with visual concepts. In *International Conference on Machine Learning (ICML)*, pages 25994–26009, 2022. arXiv:2111.08276.

[46] Xinchen Zhang, Xiaoying Zhang, Youbin Wu, Yanbin Cao, Renrui Zhang, Ruihang Chu, Ling Yang, Yujiu Yang, and Guang Shi. Generative universal verifier as multimodal meta-reasoner. In *International Conference on Learning Representations (ICLR)*, 2026. Introduces OmniVerifier-7B and the ViVerBench benchmark; arXiv:2510.13804.

[47] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.03335.

[48] Tiancheng Zhao, Tianqi Zhang, Mingwei Zhu, Haozhan Shen, Kyusong Lee, Xiaopeng Lu, and Jianwei Yin. VL-CheckList: Evaluating pre-trained vision-language models with objects, attributes and relations, 2022. URL `https://arxiv.org/abs/2207.00221`.

[49] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. arXiv:2306.05685.

[50] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2304.10592.

# A Additional Experimental Details

## A.1 Training Hyperparameters

Table 5 summarizes the main hyperparameters used for the Qwen3.5-2B reference run. The same shared backbone is used for the Mutator, Verifier, and Repairer roles, with role identity specified by prompt templates.

Table 5: Training hyperparameters for the Qwen3.5-2B reference run.

| Hyperparameter | Value |
|---|---|
| Backbone | Qwen3.5-2B |
| Training data | 48k image-caption pairs |
| Validation data | 5k image-caption pairs |
| max prompt length / response length | 4096 / 1024 |
| rollout batch size | 48 |
| rollout size $n$ | 3 |
| sampling temperature / top-p / top-k | 0.7 / 0.8 / 20 |
| presence penalty | 1.5 |
| actor learning rate | $5 \times 10^{-7}$ |
| global batch size | 48 |
| PPO clip range | $[0.20, 0.22]$ |
| KL coefficient $\beta$ | $0.005$ |
| invalid-action penalty | $-0.3$ |
| precision | BF16 |
| hardware | 8 NVIDIA A100 GPUs |
| reported checkpoint | best validation checkpoint |

## A.2 Additional Benchmark Results

Table 6 reports checkpoint-level results on additional benchmarks [15, 26, 28, 43, 46]. The checkpoints are denoted as PROVE @450, PROVE @600, and PROVE @750 according to the training step. The main paper reports the best validation checkpoint, while this table provides the full trajectory for completeness.

Table 6: Extended checkpoint results for Qwen3.5-2B. PROVE @450, PROVE @600, and PROVE @750 denote checkpoints at steps 450, 600, and 750, respectively.

| Benchmark | Base | PROVE @450 | PROVE @600 | PROVE @750 |
|---|---|---|---|---|
| Winoground (2-AFC) | 69.50 | 74.50 | 74.75 | **75.62** |
| ARO macro | 73.57 | 78.01 | 78.27 | **78.66** |
| SugarCrepe micro | 93.76 | 94.24 | **94.28** | 94.17 |
| VL-CheckList 2-axis avg | 79.80 | 83.58 | 83.87 | **84.20** |
| POPE F1 | 89.57 | 89.52 | 89.58 | **89.66** |
| MMVP Pair Acc. | 56.00 | 59.33 | 57.33 | **60.00** |
| ViVerBench (rule) | **55.16** | 52.81 | 52.79 | 52.98 |
| MMBench-EN dev | 71.31 | 72.16 | 71.65 | **72.34** |
| OCRBench | 84.50 | – | **84.90** | **84.90** |
| MMMU val | **40.78** | – | 40.11 | 40.44 |
| MathVista testmini | 73.80 | **74.50** | 73.60 | 72.80 |
| AI2D | 75.26 | – | 75.71 | **75.91** |

Table 7 reports the VL-CheckList axis-level results. The gains are consistent across attribute, object, and relation axes, showing that PROVE improves localized image-text verification rather than only a single benchmark score.

Table 7: VL-CheckList axis-level results on Qwen3.5-2B.

| Metric | Base | PROVE @450 | PROVE @600 | PROVE @750 |
|---|---|---|---|---|
| Attribute axis | 76.75 | 80.72 | 81.13 | **81.46** |
| Object axis | 91.20 | – | – | **94.14** |
| Relation axis | 82.85 | 86.45 | 86.60 | **86.95** |
| 3-axis average | 83.60 | – | – | **87.52** |

### A.3 Reward Design Discussion

The verifier-guided rewards in PROVE are designed to couple two complementary goals: exposing verifier weaknesses and recovering faithful captions. For the Mutator, the reward contains an exposure term, a hardness term, and an edit-cost penalty. The exposure term encourages perturbations that the current verifier fails to reject, while the hardness term encourages perturbations that remain challenging after repair. The cost term discourages overly wide edits. Actions rejected by the deterministic editor receive a fixed penalty instead of $C_M$ (Appendix F).

For the Repairer, the reward contains an acceptance term, a judgment-flip term, and a repair-cost penalty. The acceptance term encourages repaired captions that the verifier judges faithful, while the judgment-flip term rewards a change in the verifier decision, $\mathbb{1}[y_1 \neq y_2]$; the separate acceptance term favors a faithful accepted repair. The repair-cost penalty discourages unnecessary rewriting and encourages localized correction.

Table 8 summarizes the role of each reward component.

Table 8: Verifier-guided reward components in PROVE.

| Role | Component | Purpose |
|---|---|---|
| Mutator | $R_{\text{expose}}(y_1)$ | Expose verifier blind spots |
| Mutator | $R_{\text{hard}}(y_2)$ | Prefer perturbations that remain challenging after repair |
| Mutator | $C_M(a_M)$ | Penalize overly wide edits |
| Mutator | invalid-action penalty | Fixed penalty when the editor rejects the action |
| Repairer | $R_{\text{accept}}(y_2)$ | Reward repaired captions accepted by the verifier |
| Repairer | $R_{\text{flip}}(y_1, y_2)$ | Reward a change in the verifier judgment |
| Repairer | $C_R(p', \tilde{p})$ | Penalize unnecessary rewriting |

### A.4 Compute Cost

The Qwen3.5-2B reference run is trained on a single node with 8 NVIDIA A100 GPUs using BF16 mixed precision. Each self-play episode executes the perturb–verify–repair–verify trajectory with one shared backbone under different role prompts. The main run uses rollout size $n = 3$ and global batch size 48; the reported trajectory includes checkpoints through step 750. The reference run takes 11 hours on 8 A100 GPUs, totaling 88 GPU-hours. We will release the training and evaluation code, including launch scripts and prompt templates, to facilitate reproducibility.

We profile all complete pipelines on the same hardware and 48k source pairs (Table 9). For PROVE, supervision is generated online and is included in target-training time. Its four role-conditioned calls per trajectory produce more generated tokens than either static negatives or offline external-judge distillation. Thus, the benefit is adaptive supervision without a separate teacher or fixed negative set, rather than universally lower raw compute.

Table 9: Complete pipeline compute costs. Generation and training costs are GPU-hours. Online generation for PROVE is included in its 88.0 GPU-hours.

| Method | Supervision generation | Target training | Total GPU-hours | Generated tokens/source |
|---|---|---|---|---|
| Fixed typed negatives | 4.8 | 43.2 | 48.0 | 46 |
| External-judge distillation | 27.6 | 45.3 | 72.9 | 214 |
| PROVE | Online | 88.0 | 88.0 | 1,126 |

### A.5 Illustrative Examples of the PROVE Pipeline

To complement the method description in Section 3, we provide end-to-end examples for the four supported Mutator perturbation types: **count**, **attribute**, **relation**, and **object**. Each example follows the perturb–verify–repair–verify trajectory:

$$(x, p) \xrightarrow{\text{Mutator}} p' \xrightarrow{\text{Verifier}} (y_1, r_1) \xrightarrow{\text{Repairer}} \tilde{p} \xrightarrow{\text{Verifier}} (y_2, r_2).$$

The Mutator emits a localized edit of the form $(\texttt{edit_type}, \texttt{target_text}, \texttt{replacement})$, which is applied to the clean caption $p$ to obtain the perturbed caption $p'$. The Verifier first judges whether $p'$ matches the image and returns a binary decision $y_1$ with textual feedback $r_1$. Conditioned on the image, the perturbed caption, and this feedback, the Repairer directly generates a repaired caption $\tilde{p}$. The Verifier is then invoked again to obtain $(y_2, r_2)$. These examples illustrate how PROVE converts clean image-caption pairs into grounded hard negatives and repair targets without requiring a stronger external judge.

#### A.5.1 Count

Illustrative PROVE Pipeline – Count Action



**Original prompt $p$ (label $= 1$):**
*"A front view shot, utilizing strong vertical lines for composition, captures four divers before a central altar laden with mysterious offerings, their varied forms: the first, a technical specialist with a backplate and wing, twin steel tanks, a sealed drysuit, separate thick neoprene hood, and powerful blade fins; the second in a full wetsuit employing a sidemount BCD with a single tank, their face obscured by a full-face mask with integrated regulator, and distinguished by camouflage patterned fins; a third clearly recreational, with a jacket-style BCD, single aluminum tank, low-volume dual-lens mask, and standard paddle fins; and the fourth minimally equipped in a shorty wetsuit and full-foot fins—dramatically silhouetted by a strong, warm backlight yet softly defined by diffused ambient light, all rendered in an often high-contrast black and white artistic style."*

---

**Mutator action:** `{"edit_type": "count", "target_text": "four divers", "replacement": "five divers"}`

**Perturbed prompt $p'$ (label $= 0$):**
*"A front view shot ... captures five divers before a central altar laden with mysterious offerings, their varied forms: the first ... and the fourth minimally equipped in a shorty wetsuit and full-foot fins—..."*

**Verifier (1st pass) on $(x, p')$:** `<think> The user wants me to verify whether the description matches the image. Analyze the image: four divers stand in front of a central stone altar; the altar holds a skull, several candles, and small jugs; a strong vertical backlight produces silhouettes against a high-contrast black-and-white scene... Analyze the description: "... captures five divers ..." -- Mismatch: the image clearly shows four divers, not five. The altar with offerings, the warm backlight, and the B&W styling all align with the rest of the prompt; only the count claim is violated... Conclusion: the description does not accurately match the image. </think>`

```
{ "answer": false, "explanation": "The image contains four divers around the altar, not five."
}
```

$\Rightarrow y_1 = \texttt{false}$, $r_1 =$ "The image contains four divers around the altar, not five."

**Repairer output $\tilde{p}$:**
*"A front view shot, utilizing strong vertical lines for composition, captures four divers before a central altar laden with mysterious offerings . . . "* (recovers the count slot in $p$)
**Verifier (2nd pass) on $(x, \tilde{p})$:** `<think> Re-analyze the image: four figures stand in a row before the central altar with a skull, candles, and small jugs; the vertical god-ray and high-contrast B&W styling are unchanged from the previous pass... Re-analyze the description: "... captures four divers ..." -- the count is now correct, and the altar, the warm backlight, and the B&W rendering all line up with the description... Conclusion: modulo a few gear-level paraphrases the description is consistent with the image. </think>`
`{ "answer": true, "explanation": "Four divers are visible around the altar, matching the description." }`
$\Rightarrow y_2 =$ `true`. Alignment score $s \approx 0.995$ ($\tilde{p}$ near-identical to $p$, count slot fully restored).

### A.5.2 Attribute

Illustrative PROVE Pipeline – Attribute Action



**Original prompt $p$ (label $= 1$):**
*"A bald chibi character stands with hands on his hips, wearing an orange martial arts uniform with a yellow circular logo on the chest, a black belt around the waist, and blue wristbands on both arms."*

**Mutator action:** `{"edit_type": "attribute", "target_text": "an orange martial arts uniform", "replacement": "a green martial arts uniform"}`
**Perturbed prompt $p'$ (label $= 0$):**
*"A bald chibi character stands with hands on his hips, wearing a green martial arts uniform with a yellow circular logo on the chest, a black belt around the waist, and blue wristbands on both arms."*
**Verifier (1st pass) on $(x, p')$:** `<think> The user wants me to verify whether the description matches the image. Analyze the image: a bald chibi character stands with hands on his hips, wearing an unmistakably orange martial-arts gi; the chest carries a yellow circular Dragon-Ball-style logo, a black belt is tied at the waist, and blue wristbands are visible on both wrists... Analyze the description: "... wearing a green martial arts uniform ... yellow circular logo ... black belt ... blue wristbands ..." -- Mismatch on the uniform colour (green vs. orange); the logo, belt, and wristbands all match... Conclusion: the description has a single colour error on the uniform. </think>`
`{ "answer": false, "explanation": "The uniform is orange, not green." }`
$\Rightarrow y_1 =$ `false`, $r_1 =$ "The uniform is orange, not green."

**Repairer output $\tilde{p}$:**
*"A bald chibi character stands with hands on his hips, wearing an orange martial arts uniform with a yellow circular logo on the chest, a black belt around the waist, and blue wristbands on both arms."*
(identical to $p$, attribute slot recovered)
**Verifier (2nd pass) on $(x, \tilde{p})$:** `<think> Re-analyze the image: bald chibi character, hands on hips, orange gi, yellow circular logo on the chest, black belt at the waist, blue wristbands on both wrists... Re-analyze the description: "... orange martial arts uniform ... yellow circular logo ... black belt ... blue wristbands ..." -- every attribute slot now agrees with the image, and no further mismatch remains... Conclusion: the description is an accurate match for the image. </think>`

```
{ "answer": true, "explanation": "All visual attributes (orange gi, yellow logo, black belt, blue wristbands) match the description." }
```

$\Rightarrow y_2 =$ `true`. Alignment score $s \approx 0.99$ (single attribute slot restored).

### A.5.3 Relation

Illustrative PROVE Pipeline – Relation Action




**Original prompt $p$ (label $= 1$):**
*"A person wearing a white GAP hoodie is standing indoors with arms raised, holding a coiled white cable in both hands, near an open window with curtains."*

---

**Mutator action:** `{"edit_type": "relation", "target_text": "standing indoors", "replacement": "standing outside"}`
**Perturbed prompt $p'$ (label $= 0$):**
*"A person wearing a white GAP hoodie is standing outside with arms raised, holding a coiled white cable in both hands, near an open window with curtains."*
**Verifier (1st pass) on $(x, p')$:** `<think> The user wants me to verify whether the description matches the image. Analyze the image: a young man with dark hair and a beard wears a white GAP hoodie; his arms are raised above his head and his hands hold a coiled white cable; behind him, heavy beige curtains hang on the left and a white-framed window is visible on the right, with soft diffuse interior lighting... Analyze the description: "... standing outside ..." -- Mismatch: the curtains, the window frame, and the indoor lighting all indicate an interior space, not an outdoor one. The hoodie, raised arms, and coiled cable do match... Conclusion: the spatial relation in the prompt is inverted with respect to the image. </think>`

```
{ "answer": false, "explanation": "The person is standing indoors, not outside." }
```

$\Rightarrow y_1 =$ `false`, $r_1 =$ "The person is standing indoors, not outside."

---

**Repairer output $\tilde{p}$:**
*"A person wearing a white GAP hoodie is indoors with arms raised, holding a coiled white cable in both hands, near an open window with curtains."*
**Verifier (2nd pass) on $(x, \tilde{p})$:** `<think> Re-analyze the image: indoor scene with beige curtains on the left and a white-framed window on the right; a man in a white GAP hoodie raises his arms above his head and holds a coiled white cable in both hands... Re-analyze the description: "...`

*indoors* with arms raised, holding a coiled white cable ... near an open window with curtains'' -- every clause aligns with the image: the indoor relation, the raised arms, the coiled cable, and the window with curtains... Conclusion: the description accurately matches the image. </think>
{ "answer": true, "explanation": "The person is indoors wearing a white GAP hoodie with arms raised holding a cable, near an open window with curtains." }

$\Rightarrow y_2 =$ `true`. Alignment score $s = 0.958$ (relation slot recovered; "standing" dropped, otherwise close to $p$).

### A.5.4 Object

Illustrative PROVE Pipeline – Object Action



**Original prompt $p$ (label $= 1$):**
*"A blue-painted woman holds a single gold banana up to her face against a warm tan backdrop, in a stark studio portrait style."*

---

**Mutator action:** {"edit_type": "object", "target_text": "gold banana", "replacement": "gold pear"}
**Perturbed prompt $p'$ (label $= 0$):**
*"A blue-painted woman holds a single gold pear up to her face against a warm tan backdrop, in a stark studio portrait style."*
**Verifier (1st pass) on $(x, p')$:** <think> The user wants me to verify whether the description matches the image. Analyze the image: a woman with blue-painted skin holds a gold-coloured object up to her face against a warm tan studio backdrop; the object's silhouette is long, curved, and tapered at one end--unambiguously a banana, not the round, broader-based shape of a pear... Analyze the description: "... holds a single *gold pear* up to her face ..." -- Mismatch on the fruit identity. The blue body paint, the warm tan backdrop, and the stark studio lighting all align with the prompt... Conclusion: the object slot is incorrect (*pear* where the image shows a *banana*). </think>
{ "answer": false, "explanation": "The object held is a gold banana, not a gold pear." }

$\Rightarrow y_1 =$ `false`, $r_1 =$ "The object held is a gold banana, not a gold pear."

---

**Repairer output $\tilde{p}$:**
*"A blue-painted woman holds a single gold banana up to her face ..."* (identical to $p$, object slot recovered)
**Verifier (2nd pass) on $(x, \tilde{p})$:** <think> Re-analyze the image: a blue-painted woman holds a single gold banana up near her face; the background is a textured warm tan/brown, and the lighting and framing are clearly studio-portrait style... Re-analyze the description: "... *gold banana* ... warm tan backdrop ... stark studio portrait style'' -- the fruit identity, the colour, the backdrop, and the studio styling all match... Conclusion: the description accurately reflects the visual elements of the image. </think>
{ "answer": true, "explanation": "The blue-painted woman is holding a gold banana near her face against a warm tan backdrop, matching the prompt." }

$\Rightarrow y_2 =$ `true`. Alignment score $s \approx 0.99$ (object slot fully restored).

**Takeaways.** Across the four edit types, the same minimal-edit principle holds: the Mutator flips exactly one semantic slot (`count`, `attribute`, `relation`, or `object`) via a structured edit, the first

Verifier pass localises the mismatch via a short rationale, and the Repairer uses that rationale to directly generate a faithful caption $\tilde{p}$ that, modulo at most a few-token paraphrase (e.g. "standing indoors" $\rightarrow$ "indoors"), matches the original $p$. The second Verifier pass closes the loop and produces the success signal $y_2$ that drives the Repairer reward $r_R$ (Eq. 12). Because perturbation is a discrete typed action while repair is free-form caption generation, the two roles are not symmetric: the Mutator is constrained to expose verifier blind spots along a known axis, whereas the Repairer must recover the full caption from visual evidence and verifier feedback, which is what makes repair an operational form of verification.

# B Controlled Supervision and Loop Ablations

**Matched supervision settings.** All comparisons use the same Qwen3.5-2B initialization, 48k clean image-caption pairs, target-model optimization budget, and evaluation protocol. Fixed typed negatives are generated once and reused throughout training. External-judge distillation uses judgments and textual feedback from a frozen larger VLM. Verifier self-training uses self-generated labels without the Repairer or second verification. This baseline isolates self-generated verifier supervision under a matched image-text protocol; it is not a reproduction of reasoning-oriented self-play methods with different tasks and supervision. Table 3 reports the matched supervision results. The same optimization budget does not imply equal GPU-hours, since the methods execute different generation pipelines (Table 9).

## B.1 Component and Reward Ablations

Table 10 isolates the role of static versus adaptive perturbations, the repair loop, verifier feedback, the optimizer, and reward components. Removing the Repairer and second verification produces the largest drop, $2.61/1.72$ points on the compositional/broader averages. Removing feedback costs $1.68/1.19$ points, indicating that repair benefits from the verifier's diagnosis. Both role rewards contribute: removing the Mutator or Repairer reward costs $1.82/1.27$ or $1.31/0.94$ points, respectively. Removing the edit and repair costs reduces the averages by $0.77/0.60$ points. Replacing DrGRPO with GRPO produces a smaller $0.43/0.32$ drop; the main gain therefore depends on the closed-loop supervision rather than the optimizer alone.

Table 10: Complete component, optimizer, and reward ablations. All variants use the same source data, backbone initialization, target-model optimization budget, and evaluation protocol; the GRPO row changes only the optimization variant.

| Training variant | Comp. Avg. | Broader Avg. |
|---|---|---|
| Qwen3.5-2B | 83.22 | 58.77 |
| Fixed typed negatives | 85.02 | 59.68 |
| No Repairer or second verification | 84.24 | 59.31 |
| Repairer without verifier feedback | 85.17 | 59.84 |
| GRPO instead of DrGRPO | 86.42 | 60.71 |
| Without Mutator reward | 85.03 | 59.76 |
| Without Repairer reward | 85.54 | 60.09 |
| Without edit/repair costs | 86.08 | 60.43 |
| PROVE | **86.85** | **61.03** |

## B.2 Role Configuration and Verifier Updating

Table 11 compares shared role prompts with role-specific adapters and frozen Verifier variants. Freezing the internal Verifier lowers the compositional/broader averages by $1.73/1.22$ points, supporting verifier co-evolution. Role-specific adapters remain competitive but are $0.44/0.25$ points below full parameter sharing. A frozen external Verifier is stronger than the frozen internal Verifier, but remains $0.78/0.49$ points below the shared loop. This frozen external Verifier operates online inside the self-play loop, whereas external-judge distillation uses offline teacher judgments as target supervision. These are distinct comparisons.

Table 11: Role configuration and verifier-update ablations.

| Role configuration | Comp. Avg. | Broader Avg. |
|---|---|---|
| Shared backbone, role prompts | **86.85** | **61.03** |
| Shared backbone, role-specific adapters | 86.41 | 60.78 |
| Frozen internal Verifier | 85.12 | 59.81 |
| Frozen external Verifier | 86.07 | 60.54 |

# C Independent Verification and Repeated-Run Analysis

## C.1 External Trajectory Audit

We audit a held-out trajectory set using labels produced outside the self-play loop. These labels are never used for training, reward computation, mutation generation, validation, or checkpoint selection. The initial Qwen3.5-2B model and PROVE use the same role prompts and deterministic editor. Table 12 reports verifier accuracy on a fixed set and the quality of the generated trajectories.

A valid localized perturbation introduces a real image-caption mismatch confined to the intended edit. A correct verifier judgment identifies whether the caption is faithful to the image. A faithful minimal repair corrects the mismatch without introducing unsupported changes. A fully valid trajectory requires all four stages to be correct: a real localized mismatch, a correct diagnosis, a faithful minimal repair, and a correct final judgment. The fully valid rate increases by $23.4$ points, while fixed-set verifier accuracy increases by $7.8$ points. These external measurements support improved visual correctness, while the shared-model setting still leaves possible self-verification bias (Section 4.3).

Table 12: Complete external audit. All entries are percentages. Fixed-set verifier accuracy and trajectory-quality rates are separate audit measurements.

| External audit | Base | PROVE |
|---|---|---|
| Fixed-set verifier accuracy | 75.8 | **83.6** |
| Valid localized perturbations | 82.4 | **92.1** |
| Correct verifier judgments | 71.5 | **84.7** |
| Faithful minimal repairs | 55.2 | **73.4** |
| Fully valid trajectories | 40.8 | **64.2** |

## C.2 External Benchmark Trajectory

We evaluate checkpoints on fixed, externally labeled benchmarks that are excluded from training, self-play rewards, mutation generation, validation, and checkpoint selection. Table 13 separates the internal self-play pass rate from externally measured benchmark performance. The compositional average increases from $83.22$ to $86.85$, while the broader average increases from $58.77$ to $61.03$. The broader trajectory is not monotonic: its step-600 value is below step 450, illustrating why internal agreement is not a substitute for independent evaluation. Across the six reported checkpoints, Pearson correlations between pass rate and the compositional/broader averages are $0.9254/0.9022$. These descriptive correlations summarize co-movement during training; they do not establish a causal relationship or a significance test. Figure 5 plots the six reported measurements without adding intermediate observations.

Table 13: Internal self-play pass rate and external benchmark averages across training. The pass rate is in percent; benchmark averages use the main-paper definitions.

| Checkpoint | Self-play pass rate | Comp. Avg. | Broader Avg. |
|---|---|---|---|
| Base | 36.0 | 83.22 | 58.77 |
| Step 150 | 39.3 | 84.08 | 59.18 |
| Step 300 | 43.2 | 85.14 | 59.73 |
| Step 450 | 46.3 | 85.97 | 60.41 |
| Step 600 | 56.6 | 86.48 | 60.16 |
| Step 750 | 63.3 | 86.85 | 61.03 |

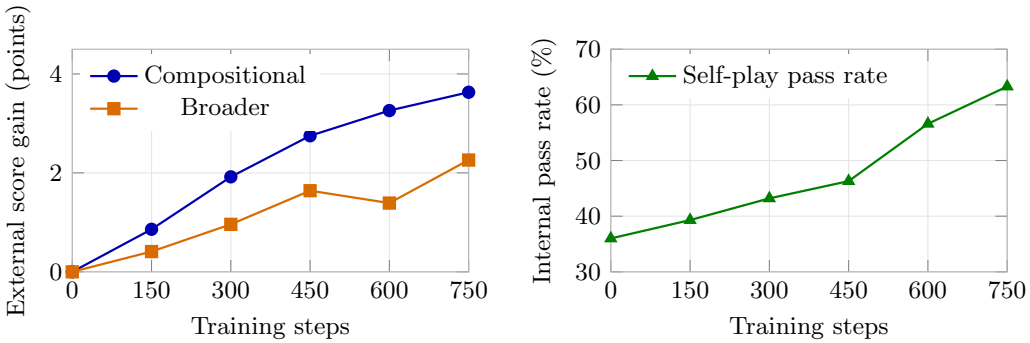



Figure 5: **External benchmark performance and internal self-play pass rate.** Each marker is one reported checkpoint; connecting lines guide the eye. Gains in the left panel are measured against the base checkpoint. The right panel reports internal pass rate separately, preserving the distinction between independent accuracy and role agreement.

### C.3 Repeated-Run Variability and Broader Capability

We repeat Qwen3.5-2B training with three independent seeds. Table 14 reports the mean and standard deviation across the three trained models. These statistics measure training-run variability, not example-level uncertainty, confidence intervals, or statistical significance. The reference-run results in Tables 1 and 2 and the checkpoint table in Appendix A.2 remain single-run measurements.

The core compositional average improves by 3.51 points across seeds. Broader changes are limited and task-dependent: ViVerBench regresses by 2.04 points, MMMU and MathVista decrease by 0.39 and 0.86, MMBench improves by 0.95, and POPE remains nearly unchanged. Thus, PROVE substantially improves compositional verification without implying uniform preservation or improvement of all broader capabilities.

Table 14: Results across three independent Qwen3.5-2B training runs. Change is the mean PROVE result minus the base score.

| Benchmark | Base | PROVE (mean $\pm$ std) | Change |
|---|---|---|---|
| ViVerBench | 55.16 | $53.12 \pm 0.58$ | $-2.04$ |
| MMMU | 40.78 | $40.39 \pm 0.47$ | $-0.39$ |
| MathVista | 73.80 | $72.94 \pm 0.52$ | $-0.86$ |
| MMBench | 71.31 | $72.26 \pm 0.31$ | $+0.95$ |
| POPE | 89.57 | $89.64 \pm 0.09$ | $+0.07$ |
| Core compositional average | 83.22 | $86.73 \pm 0.27$ | $+3.51$ |

## D Generality and Perturbation Curricula

### D.1 Backbone Scale and Family

We apply the same perturbation and reward formulation to Qwen3.5-4B and the scale-matched InternVL3-2B family. Table 4 in the main paper reports the full base and PROVE averages for all three backbones. The Qwen3.5-4B gains are $3.56/2.34$ compositional/broader points, while InternVL3-2B gains $3.15/2.83$. These results show that the gains extend beyond the 2B Qwen reference setting, although all runs still rely on one source training corpus.

### D.2 Leave-One-Type-Out Transfer

For each leave-one-type-out experiment, we remove the held-out type from the Mutator action space and resample the remaining types to preserve the same episode count and optimization budget. Table 15 reports performance on the held-out axis. The retained gain is $(s_{\mathrm{LOO}} - s_{\mathrm{base}})/(s_{\mathrm{full}} - s_{\mathrm{base}})$.

These axis-level scores follow the capability aggregation used in Figure 3(b), rather than the overall averages in the main tables. Leave-one-out training retains $48.2\%$–$52.4\%$ of the full-method gain across the three reported axes. This supports cross-type transfer while showing that direct exposure supplies additional axis-specific benefit.

Table 15: Leave-one-type-out transfer on the three reported axes. The held-out axis is absent from the training action space.

| Held-out type | Base | Leave-one-out PROVE | Full PROVE | Retained gain |
|---|---|---|---|---|
| Object | 86.1 | 87.2 | 88.2 | 52.4% |
| Attribute | 78.1 | 81.1 | 84.1 | 50.0% |
| Relation | 76.3 | 79.0 | 81.9 | 48.2% |

### D.3 Single- and Multi-Edit Curricula

Table 16 compares one-edit, mixed one/two-edit, and two-edit curricula. The mixed curriculum raises the broader average by $0.15$ points but lowers compositional performance and repair success. Using two edits throughout further reduces repair success to $48.6\%$. These results support single localized edits as the default because they simplify diagnosis, minimal repair, and credit assignment. A multi-edit extension may benefit from explicit partial-credit rewards, which are not evaluated in these comparisons.

Table 16: Perturbation curriculum comparison. Repair success is a percentage.

| Curriculum | Comp. Avg. | Broader Avg. | Repair success |
|---|---|---|---|
| One edit | **86.85** | 61.03 | **62.7** |
| Mixture of one/two edits | 86.06 | **61.18** | 57.4 |
| Two edits | 85.72 | 60.31 | 48.6 |

## E Positive-Caption Quality

We independently audit a random subset of the training pool, distinguishing fully faithful captions, minor unsupported details or ambiguity, and material image-caption mismatches. Table 17 reports the three fractions, which sum to $100\%$. Most captions are faithful, while $1.7\%$ contain material mismatches. We then remove low-confidence positives and repeat training (Table 18). The two averages increase by $0.23/0.12$ points, suggesting that the observed verification gain does not depend on retaining noisy positives. This does not remove the method's dependence on sufficiently faithful reference captions.

Table 17: Independent audit of a random subset of the positive-caption pool.

| Caption quality | Fraction |
|---|---|
| Fully faithful | 90.1% |
| Minor unsupported detail or ambiguity | 8.2% |
| Material image-caption mismatch | 1.7% |

## F Role Interfaces and Deterministic Editing

**Role-conditioned interfaces.** The Mutator receives the image and a faithful caption and returns a single structured action with fields `edit_type`, `target_text`, and `replacement`. The type is one of `object`, `attribute`, `count`, or `relation`. The deterministic editor applies the local replacement to construct the caption supplied to the first Verifier. The Verifier receives the image and candidate caption and produces a binary faithfulness decision with textual feedback; the illustrative outputs in Appendix A.5 use the fields `answer` and `explanation`. The Repairer receives the image, perturbed caption, first decision, and explanation, and directly generates a faithful caption with a minimal

Table 18: Training after filtering low-confidence positive captions.

| Training pool | Comp. Avg. | Broader Avg. |
|---|---|---|
| Original 48k | 86.85 | 61.03 |
| Filtered positives | **87.08** | **61.15** |

correction. The same Verifier interface is used for the second verification. The complete templates used in training are listed below.

**Prompt templates.** `<image>` marks the image input. Braced fields are filled at run time: `{p}` is the clean caption $p$, `{caption}` is the candidate caption ($p'$ in the first verification and $\tilde{p}$ in the second), `{p'}` is the perturbed caption, and `{y1}` and `{r1}` are the first Verifier decision and explanation.

Mutator prompt template

```
<image>{p}

You are a Mutator agent.  Look at the image and the caption above, then propose a single minimal
edit that turns the caption into a slightly incorrect description of the image.

Output exactly one JSON object on a single line, with no extra text, no markdown fences, no
comments, no thinking trace:
{"edit_type":  "<object|attribute|count|relation>", "target_text":  "<exact substring from the
caption>", "replacement":  "<modified text, must differ from target_text>"}

Rules:
- edit_type is one of object, attribute, count, relation.
- target_text MUST be an exact substring that occurs once in the caption.
- replacement MUST differ from target_text.
- Keep the edit minimal and locally semantic.
- Output ONLY the JSON object, nothing else.
```

Verifier prompt template (both verifications)

```
<image>{caption}

You are a Verifier agent.  Determine whether the caption faithfully describes the image.

First write the reasoning inside <think> </think>.  Then output exactly one JSON object:
{"answer":  <true|false>, "explanation":  "<one short sentence>"}

Rules:
- answer is true only when every object, attribute, count, and relation in the caption matches the
image.
- answer is false when any object, attribute, count, or relation is wrong.
- explanation names the mismatch, or states that the caption matches.  Example:  "The image
contains four divers around the altar, not five."
- Judge only what is visible.  Do not rewrite the caption.
```

Repairer prompt template

```
<image>{p'}

You are a Repairer agent.  The caption above may not faithfully describe the image.  The verifier
decision is y1={y1}, and its explanation is:  {r1}

Recover a caption that faithfully describes the image.  Make the smallest correction needed and
preserve the rest of the caption.

Output only the repaired caption, with no JSON, no markdown fences, no comments, and no thinking
trace.
```

**Output parsing.** The Mutator output is parsed as $a_M = (\tau, t, \delta)$ from `edit_type`, `target_text`, and `replacement`. The Verifier output follows the format of Appendix A.5: a reasoning trace inside `<think>` tags followed by the JSON object. `answer=true` gives $y = 1$, `answer=false` gives $y = 0$, and `explanation` is the textual feedback $r$; the two verifications yield $(y_1, r_1)$ and $(y_2, r_2)$. The Repairer output is the repaired caption $\tilde{p}$ itself rather than an edit action.

**Deterministic editing constraints.** A valid action has a supported edit type and nonempty target and replacement strings. The target must identify one unambiguous span in the caption, and the replacement must change it. Unsupported types, missing or ambiguous spans, unchanged replace-

ments, and edits that cannot be applied deterministically are invalid. For a unique target span, the editor preserves the caption prefix and suffix and replaces only that span. Algorithm 1 specifies this local-edit interface. An invalid action receives the invalid-action penalty $-0.3$ in Table 5 as its Mutator reward, and $C_M$ is not computed. Invalid edits do not establish a valid negative example.

**Algorithm 1** Deterministic single-span editor specification

**Require:** Caption $p$, structured action $a_M = (\tau, t, \delta)$
1: **if** $a_M$ cannot be parsed, $\tau \notin \mathcal{T}$, or $t, \delta$ are missing or empty **then**
2: **return** invalid action
3: **end if**
4: **if** $t = \delta$ or $t$ does not identify exactly one span in $p$ **then**
5: **return** invalid action
6: **end if**
7: Write $p = u\,t\,v$ using the unique target occurrence
8: **return** $p' = u\,\delta\,v$

**Reward weights.** Equations 10 and 12 instantiate Equations 9 and 11 with $R_{\text{expose}}(y_1) = \lambda_1 \mathbb{1}[y_1 = 1]$, $R_{\text{hard}}(y_2) = \lambda_2 \mathbb{1}[y_2 = 0]$, and $\lambda_M = \lambda_3$ for the Mutator, and $R_{\text{accept}}(y_2) = \beta_1 \mathbb{1}[y_2 = 1]$, $R_{\text{flip}}(y_1, y_2) = \beta_2 \mathbb{1}[y_1 \neq y_2]$, and $\lambda_R = \beta_3$ for the Repairer. Table 19 lists the weights used in all reported runs.

Table 19: Reward weights in Equations 10 and 12.

| Weight | Value | Term |
|---|---|---|
| $\lambda_1$ | 1.0 | Perturbation not rejected by the first verification |
| $\lambda_2$ | 0.5 | Repaired caption still judged unfaithful |
| $\lambda_3$ | 0.3 | Mutator edit cost $C_M$ |
| $\beta_1$ | 0.8 | Repair accepted by the second verification |
| $\beta_2$ | 0.7 | Judgment flip between the two verifications |
| $\beta_3$ | 0.3 | Repairer rewriting cost $C_R$ |

**Cost terms.** $C_M$ penalizes overly wide edit spans, normalized by the character length of the clean caption:

$$C_M(a_M) = 2 \cdot \frac{\text{len}(t)}{\text{len}(p)}, \tag{16}$$

with $C_M = 0$ when $t$ or $p$ is empty. $C_R$ penalizes unnecessary rewriting. Let $T(\cdot)$ split a caption on whitespace and lowercase each token, and let $\text{lev}$ be the token-level Levenshtein distance:

$$C_R(p', \tilde{p}) = \frac{\text{lev}\big(T(p'), T(\tilde{p})\big)}{\max\big(|T(p')|, |T(\tilde{p})|\big)}. \tag{17}$$

$C_R$ is 0 for identical captions, approaches 1 for a complete rewrite, and is set to 1 when either caption has no tokens.

**Reward and diagnostic interpretation.** Equations 10 and 12 specify the exposure, hardness, acceptance, judgment-flip, and cost terms. The flip indicator $\mathbb{1}[y_1 \neq y_2]$ rewards a decision change; it is not independently a guarantee of a visually correct repair. In particular, an evolving Verifier can accept an incorrect caption, motivating the external audit in Appendix C.1. Repair success, structural alignment, and internal pass rate are training diagnostics, whereas the external benchmark and audit labels measure correctness independently. For a group of $n = 3$ binary outcomes, the empirical success fraction can take values $0, 1/3, 2/3, 1$. Its Bernoulli variance $\bar{y}(1 - \bar{y})$ is maximized at both $1/3$ and $2/3$; Figure 3 plots the lower finite-group reference. This differs from the population Bernoulli variance maximum at $1/2$ and is a diagnostic reference, not an additional measurement or an external accuracy target.

# G Axis-level Gains, Qualitative Behavior, and Task Extensions

## G.1 Axis-level Gains

**Axis-level gains.** Figure 3(b) summarizes where the gains of PROVE come from by aggregating benchmark splits according to the visual-semantic capability they test. The largest improvements appear on attribute and relation verification, which are directly aligned with the typed perturbations used by the Mutator. This pattern supports the intended training mechanism: localized perturbations create visually checkable mismatches, forcing the Verifier to diagnose fine-grained inconsistencies and the Repairer to recover faithful captions. The positive gain on visual-detail sensitivity further suggests that verifier-guided repair improves localized evidence grounding beyond the core compositional benchmarks. In contrast, object and color/order gains are smaller but still positive, partly because the base model already performs strongly on these axes.

## G.2 Qualitative Behavior and Failure Cases

**Qualitative verifier behavior.** Figure 4 provides representative examples of how the learned verifier changes after self-play training. The base model often accepts captions or answers that overlook a localized mismatch, such as an incorrect relation, missing object evidence, wrong attribute, or counting error. After PROVE training, the verifier produces judgments that better track the relevant visual evidence and reject mismatches that the base model misses. These examples indicate that PROVE does not merely improve aggregate scores; it changes the verifier's behavior in the intended direction, making it more sensitive to the same visual-semantic factors targeted by the perturb–verify–repair loop.

**Failure cases.** Our error analysis identifies four recurring settings: small or occluded objects that are misperceived; ambiguous relations in crowded scenes or unusual viewpoints; overlapping instances or unclear counting conventions; and descriptions that require external knowledge and invite language-prior judgments rather than image-grounded verification. These settings can affect mutation validity, verifier diagnosis, or repair quality. A localized edit may fail to create an unambiguous mismatch, a Verifier may misidentify the relevant evidence, or a Repairer may alter additional unsupported details. The independent audit evaluates these stages separately rather than treating final acceptance as sufficient evidence of a faithful trajectory.

**Task extensions.** The loop can be adapted when a task provides a known-correct multimodal instance and a structured operation that creates a verifiable error. For VQA, the Mutator could alter an answer, rationale, or question condition and the Repairer could recover the visually supported answer. For visual grounding, it could perturb a region, bounding box, or referring phrase; for referring expressions, it could alter target attributes or relations. The present experiments establish caption-based verification and benchmark transfer, not task-specific self-play training for these extensions.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution of PROVE as a verifier-guided self-play framework for improving vision-language verifiers. The claims are supported by the method description, the main compositional understanding results, the broader benchmark results, and the limitations discussed in Section 4.3.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4.3 discusses the limitations of the current action space, difficulty saturation, general-VL regression at small scale, and the reliance on clean reference captions.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present theoretical results or formal theorems. The contribution is an empirical self-play training framework for vision-language verifiers.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: Sections 3 and 4 and Appendix A describe the method, data size, benchmarks, optimization settings, and reference compute. Appendix F gives the complete role prompts, output formats, reward weights, and cost definitions. Some supplemental audit/baseline settings are not supplied in the current materials; full reproduction requires these details and the planned code release.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The training uses existing image-caption data and public benchmarks. The complete prompts are listed in Appendix F; training and evaluation code and launch scripts are planned for release but are not included in the present supplement.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [No]

   Justification: The backbone, data sizes, principal optimization settings, evaluation metrics, and matched comparisons are specified, and Appendix F gives the full prompt templates, reward weights, and cost definitions. Supplemental audit/baseline settings remain needed for complete reproduction.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix C.3 reports means and standard deviations across three independent Qwen3.5-2B training runs for the core compositional average and five broader benchmarks. These measure training-run variability, not confidence intervals or significance tests.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 and Appendix A report that the main experiments use a single node with 8 NVIDIA A100 GPUs, BF16 mixed precision, rollout size, batch size, and optimizer settings. Appendix A.4 reports wall-clock time, GPU-hours, and generated-token counts for the reference run and supervision baselines.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies verifier training for image-text understanding using existing datasets, public benchmarks, and standard VLM training procedures. We are not aware of any aspect of the research that violates the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses verifier training for detecting image-text mismatches and hallucinations, which can improve reliability of multimodal systems. Potential risks include over-reliance on imperfect verifiers or use of stronger verification models in automated filtering systems; these concerns are partly addressed through the limitations discussion.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not introduce a new high-risk pretrained model, image generator, or scraped dataset. It trains a verifier using existing image-caption data and evaluates on public benchmarks; any released code will be for reproducing the training and evaluation pipeline.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the Qwen3.5 backbone, VeRL/HybridFlow framework, ShareGPT-4o-Image-style data, and all evaluation benchmarks used in the experiments. We will follow the licenses and terms of use of these existing assets when releasing code and reproduction instructions.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

Justification: The current supplement documents the training framework and experiments; the proposed training and evaluation code artifact and its complete reproduction documentation are not yet included.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research, crowdsourcing, or user studies, so IRB approval is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The core method uses a multimodal LLM backbone as the Mutator, Verifier, and Repairer under different role prompts. This usage is central to the method and is described in Sections 3 and 4.