PROGRAM-VERIFIED SELF-EVOLUTION FOR VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving vision-language models train on questions they generate from unlabeled images. Since these questions have no gold answers, prior methods label them by majority vote over sampled answers or by a model judge. In a human evaluation, we find that 24% of majority-vote labels and 18% of model-judge labels produced during self-evolution are wrong. To address this problem, we present Verifiable QA Generation for Self-Evolving Models (VQS), which changes how the model judges answers. Instead of voting on an answer, the model parses each image into a structured record, such as a scene graph, a chart table, or a diagram graph. Fixed programs then write a question from the record and compute its answer. The model still acts as a visual checker, but it only confirms the individual facts the program reads, one short claim at a time. These claim-level checks select the parser's training targets, so the parser also improves without labels. Human raters find 94% of VQS answers correct, against 76% for majority voting. Across ten benchmarks, VQS improves Qwen3-VL by up to 3.18 points at the 2B, 4B, and 8B scales and outperforms the strongest self-evolving baseline at each. Gains keep growing over three training rounds, reaching 3.84 points at 2B. Our code and models will be released.

1 INTRODUCTION

A vision-language model can improve without human labels by writing the questions it trains on. Recent work does this by splitting one model into a proposer that asks and a solver that answers (He et al., 2026; Thawakar et al., 2025; Sunil et al., 2026; Huang et al., 2026). In each round, the proposer generates questions from unlabeled images, the solver answers them, and the solver is trained on the pairs that match a pseudo-label. Answer verification is the hardest step, since gold answers do not exist in this loop.

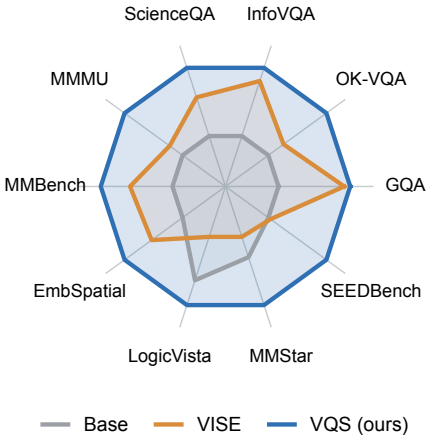


Figure 1: **VQS improves every benchmark.** Scores are for Qwen3-VL-2B. VISE loses on LogicVista and MMStar.

Standard reinforcement learning with verifiable rewards compares each answer with a gold label (Shao et al., 2024; Yu et al., 2026). However, self-generated questions have no gold label, so prior work falls back on proxy labels such as a majority vote over samples (Wei et al., 2026; Zuo et al., 2026; He et al., 2026), agreement among reasoning steps (Sunil et al., 2026), consistency under an image transformation (Venkatraman et al., 2026), or a model judge (Chen et al., 2025; Wu et al., 2026; Yuan et al., 2024).

Prior works use proxies that accept many wrong answers. First, in our human evaluation of 500 generated questions (Table 2), 24% of majority-vote labels and 18% of model-judge labels are wrong. Second, the errors also grow as the loop runs. VisPlay's pseudo-label accuracy falls from 72% to 61% over three rounds (He et al., 2026), and R-Zero's falls from 79% to 63% (Huang et al., 2026). Third, prior work reaches the same conclusion independently, showing that majority-vote self-reward eventually favors confident but incorrect answers and collapses (Shafayat et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

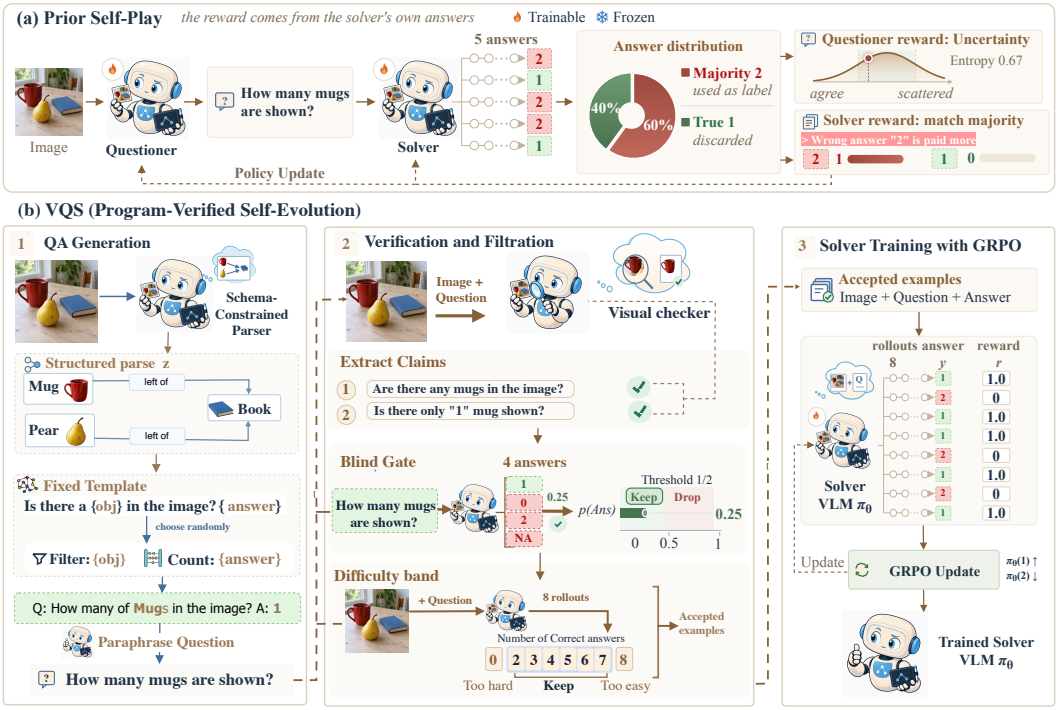


Figure 2: **Majority-vote labels vs. computed answers. (a)** In prior self-play, the solver's majority answer becomes the label. Here three of five answers say 2 mugs, so the wrong answer is rewarded and the correct answer "1" is discarded. **(b)** VQS computes the answer instead. **(1)** A schema-constrained parser turns the image into a structured parse. A fixed template reads the parse (filter mugs, then count) to write a question and compute its answer "1" and the parser then paraphrases the question. **(2)** A visual checker verifies each claim behind the answer, a blind gate drops questions answerable without the image, and a difficulty band keeps those the solver gets right in some but not all of 8 rollouts. **(3)** GRPO rewards each rollout by exact match against the computed answer, so training reinforces the correct answer "1" and suppresses the wrong one "2".

We present VQS (**V**erifiable **Q**A Generation for **S**elf-Evolving Models), a framework that computes answers from verified visual facts rather than estimating them from model outputs. As illustrated in Fig. 2, a parser first converts an input image into a navigable structure, such as a scene graph for a photograph, a structured table for a chart, or a set of nodes and directed edges for a diagram. Fixed templates (i.e., deterministic programs) then traverse these structures to generate questions alongside their computed answers. To catch wrong answers, a checker verifies the atomic visual facts behind an answer one at a time. We then train the solver on these verified QA pairs. These claim-level checks select the parser's training targets by choosing the most precise of K parses, so the parser also improves without labels. A single model is used for the parser, solver, and checker.

Our contributions are: (1) VQS converts photographs, charts, diagrams, and infographics into structured records, then generates questions and computes answers through deterministic programs. (2) A label-free algorithm improves the parser by selecting its training set from candidate parses with claim-level visual verification. (3) VQS improves the average over ten benchmarks by up to **3.18** points (Fig. 1) at the 2B, 4B, and 8B scales and beats the strongest self-evolving method at every scale. (4) VQS keeps improving over three training cycles on the same unlabeled images by generating new QA pairs from templates each cycle, raising the 2B gain from 3.18 to **3.84** points.

2 Related Work

Self-evolving vision-language models. R-Zero (Huang et al., 2026) uses a questioner and a solver trained against each other. VisPlay (He et al., 2026) trains a questioner with a reward that peaks when the solver is maximally uncertain. EvoLMM (Thawakar et al., 2025) rewards the solver by how often its answer is the majority among five samples. iReasoner (Sunil et al., 2026) adds agreement

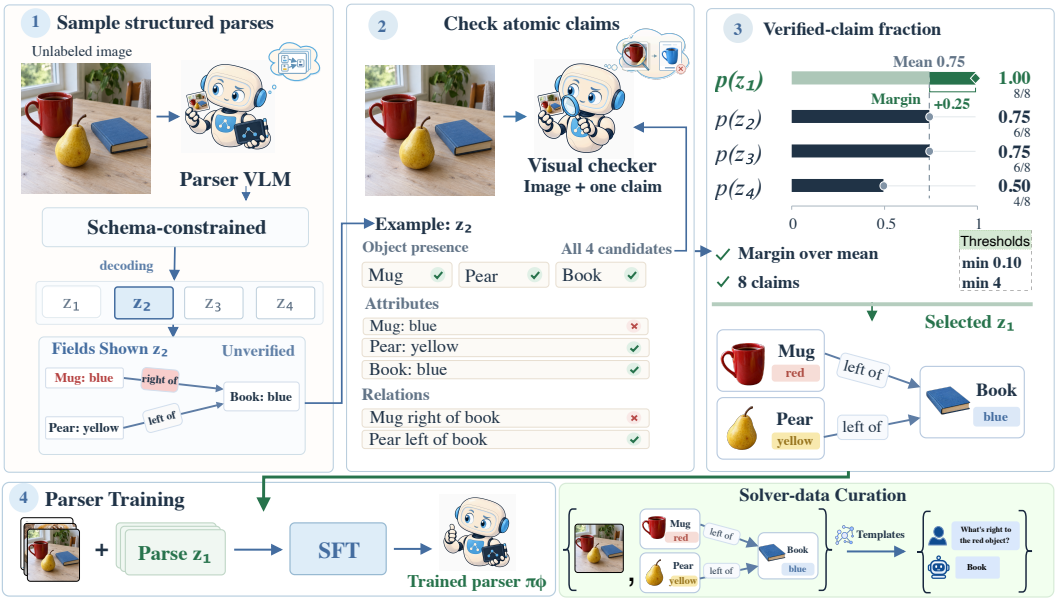


Figure 3: **Label-free parser training. (1)** The parser samples K structured parses of an unlabeled image under a schema-constrained decoder. **(2)** A visual checker verifies each parse one claim at a time. In the example, z_2 wrongly calls the mug blue and places it to the right of the book, so it passes 6 of 8 claims. **(3)** Each parse is scored by its fraction of verified claims. We keep the best parse z_1 (8/8) only because it beats the mean of all four parses (0.75) and contains at least m=4 claims. **(4)** The kept parses train the parser with SFT. The trained parser then feeds the templates that write questions and compute their answers for the solver.

between reasoning steps. Vision-Zero (Wang et al., 2025) turns the loop into a checkable game. VISE (Venkatraman et al., 2026) drops the two-role setup and rewards the model for predicting the same box under a known image transformation. Without gold answers, all of these methods take their reward from the model's outputs, and this reward can become less accurate as training goes on. For example, VisPlay's estimated pseudo-label accuracy falls from 72% to 61% over three rounds as its questioner gets stronger, and R-Zero's falls from 79% to 63%.

Programs that write questions. CLEVR (Johnson et al., 2017) attaches a functional program to every question and runs it on the scene graph it rendered from. GQA (Hudson & Manning, 2019) scales this to real photographs using human-annotated graphs. ProVision (Zhang et al., 2024) and SpatialVLM (Chen et al., 2024a) run generators over a *model-produced* parse at very large scale, but both produce fine-tuning data and neither verifies a generated claim.

3 METHOD

Below we describe how VQS builds the solver's training data, trains the solver, and trains the parser. In each training cycle, the parser is trained first. Algorithm 1 gives the full procedure.

Setup. The only training data is a set of unlabeled images $\mathcal{X}$. We initialize two models from the same pretrained model: (1) the **parser** π_ϕ, which maps an image to a structured description that we call a *parse*, and (2) the **solver** π_θ, which answers questions about the image. The parser also serves as the checker and the text-only model described below (details in Section 4). A third component, the **program library** $\mathcal{T}$, is a fixed set of templates written once by hand and never trained (See A.5).

From images to questions. To build the solver's training data, the parser first maps each image x to a parse z. Each parse must follow a schema $\mathcal{S}_{d(x)}$ fixed per domain $d(x) \in \{\text{photo}, \text{chart}, \text{diagram}, \text{infographic}\}$, and we decode it with a constrained decoder so that it is always well formed. A photo parse holds objects with names, attributes, boxes, and relations. A chart parse holds series of (category, printed value, numeric value) points and the axis ticks. A diagram

Algorithm 1 VQS (one training cycle)

Require: unlabeled images $\mathcal{X}$, templates $\mathcal{T}$, pretrained weights π_0
$\phi, \theta \leftarrow \pi_0$ ▷ parser and solver start from the same model
Train the parser (label-free best-of-K)
for $x \in \mathcal{X}$ **do**
  sample $z^{(1..K)} \sim \pi_\phi(\cdot \mid x)$ under the schema
  score each $z^{(k)}$ by its fraction of verified claims $P(z^{(k)})$ ▷ Eq. (5)
  keep $(x, z^\star)$ if it beats the mean of all K parses by δ and has $\geq m$ claims ▷ Eq. (6)
end for
fine-tune ϕ on the kept targets with cross-entropy loss
Build the solver training data
for $x \in \mathcal{X}$ **do**
  $z \leftarrow \pi_\phi(x)$, decoded greedily under the schema
  for each applicable template t: $(q, a, h) \leftarrow t(z)$, then paraphrase $q \rightarrow \tilde{q}$
  drop the question unless it passes the fact-check, the blind gate, and the difficulty band ▷ Eq. (1)
end for
Train the solver
for each GRPO step **do**
  draw n rollouts per question and score each against a ▷ Eq. (3)
  normalize rewards within each group and update θ with GRPO ▷ Eq. (4)
end for
return π_θ ▷ π_θ can re-parse the images and start the next cycle

parse holds nodes and directed arrows. An infographic parse holds entries with their text, number, and unit. Figure 4 shows examples, and Appendix A.4 gives the full schemas for each domain.

Each applicable template $t \in \mathcal{T}$ then reads the parse z and returns a question q, its answer a, and its hop count h, which is the number of operations the template performs: $(q, a, h) = t(z)$. Templates combine these operations: filter a set, read a field or attribute, count a set, take a maximum or minimum, follow a relation, rank, compare, do arithmetic, read a trend, count a node's arrows, and turn a box into a coarse position. Finally, the parser rewrites q in natural wording, $\tilde{q} \sim \pi_\phi(\cdot \mid x, z, q)$, while the answer a stays the same.

Which questions enter the training data. We keep a question only if it passes three checks. The *fact-check* verifies the facts behind the answer. The template asserts one claim per field it reads, giving a set of atomic claims $\mathcal{C}(t, z)$, and a binary checker $v(c, x) \in \{0, 1\}$ answers yes or no to each claim. The *blind gate* removes questions that do not need the image. A text-only model π_{blind} answers the question J times without the image, $u_j \sim \pi_{\text{blind}}(\cdot \mid \tilde{q})$, and we drop it if more than a fraction λ of these answers are correct. The *difficulty band* removes questions that are too easy or too hard. The solver about to be trained answers the question n times, $y_i \sim \pi_\theta(\cdot \mid x, \tilde{q})$, and we drop it if the solver is always right or always wrong, since all rollouts then get the same correctness score and give almost no learning signal (Eq. (3)). Formally, we keep the question only if

$$\underbrace{\textstyle\prod_{c \in \mathcal{C}(t,z)} v(c, x) = 1,}_{\text{fact-check}} \quad \underbrace{\tfrac{1}{J}\textstyle\sum_j \mathrm{eq}(u_j, a) \leq \lambda,}_{\text{blind gate}} \quad \underbrace{0 < \hat{p} < 1, \;\; \hat{p} = \tfrac{1}{n}\textstyle\sum_i \mathrm{eq}(y_i, a),}_{\text{difficulty band}} \tag{1}$$

where $\mathrm{eq}(y, a) \in \{0, 1\}$ is exact match between an answer y and a.

Solver Training. The reward is an exact match with the computed answer, plus a small format term,

$$r(y; a) \;=\; 0.9 \cdot \mathrm{eq}(y, a) \;+\; 0.1 \cdot \mathrm{fmt}(y), \tag{2}$$

where $\mathrm{fmt}(y) \in \{0, 1\}$ checks that y follows the required answer format. We train the solver with GRPO (Shao et al., 2024) rather than SFT. GRPO learns on-policy, while SFT on template-written text may pull the model away from its distribution. For each question, GRPO normalizes the rewards of the n rollouts within their group,

$$A_i \;=\; \frac{r_i - \mathrm{mean}(r_1 \ldots r_n)}{\mathrm{std}(r_1 \ldots r_n) + \varepsilon}, \qquad r_i = r(y_i; a), \tag{3}$$

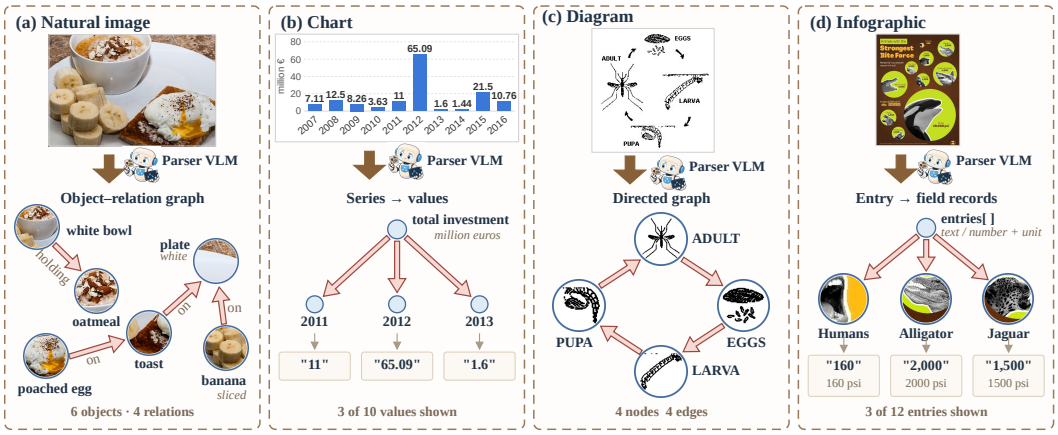


Figure 4: **Examples of structured scene parsing.** The parser converts natural images, charts, diagrams, and infographics into domain-specific JSON records that preserve spatial relations, chart values and ticks, graph structure, and verbatim strings for deterministic QA construction.

and updates the solver with a clipped objective and a KL penalty against a frozen reference π_{ref},

$$\mathcal{L}_{\text{solver}}(\theta) = -\,\mathbb{E}\left[\min\left(\frac{\pi_\theta(y_i)}{\pi_{\theta_{\text{old}}}(y_i)}A_i,\ \text{clip}\left(\frac{\pi_\theta(y_i)}{\pi_{\theta_{\text{old}}}(y_i)}, 1-\epsilon_{\text{lo}}, 1+\epsilon_{\text{hi}}\right)A_i\right)\right] \;+\; \beta\,\text{KL}[\pi_\theta \,\|\, \pi_{\text{ref}}]\,, \tag{4}$$

where ϵ_{lo} and ϵ_{hi} set the clipping range and β weights the KL penalty.

Parser Training. The parser is trained first, before the solver data is built, and without labels (Fig. 3). For each image x, we sample K parses and score each by the fraction of its claims $\mathcal{F}(z)$, one per field, that the checker confirms:

$$z^{(1..K)} \sim \pi_\phi(\cdot \mid x), \qquad P(z) = \frac{1}{|\mathcal{F}(z)|}\sum_{c\in\mathcal{F}(z)} v(c,x), \qquad z^\star = \arg\max_k P(z^{(k)}). \tag{5}$$

We keep the best parse $z^\star$ only when it clearly beats the average of all K parses, so that each kept target carries a real learning signal. If all K parses score the same, the chosen parse is no better than what the parser already produces. We also require the best parse to have at least m claims, since small parses usually get inflated verification scores:

$$\text{keep } (x, z^\star) \text{ iff } P(z^\star) - \tfrac{1}{K}\sum_k P(z^{(k)}) \geq \delta \;\text{ and }\; |\mathcal{F}(z^\star)| \geq m. \tag{6}$$

Finally, we fine-tune the parser on the kept targets $(x, z^\star)$ with the standard cross-entropy loss.

4 Experimental Setup

Models and data. We train Qwen3-VL-2B, 4B and 8B Instruct (Bai et al., 2025). We take the training images from Venkatraman et al. (2026) and discard their questions, answers, and labels, so no human annotation is used at any stage.

Benchmarks. We use lmms-eval (Zhang et al., 2025) to evaluate on 10 benchmarks: GQA (Hudson & Manning, 2019), OK-VQA (Marino et al., 2019), InfoVQA (Mathew et al., 2022), ScienceQA (Lu et al., 2022), MMMU (Yue et al., 2024), MMBench (Liu et al., 2024), EmbSpatial (Du et al., 2024), LogicVista (Xiao et al., 2024), MMStar (Chen et al., 2024b), SEEDBench (Li et al., 2024). We use the default hyperparameters for each benchmark.

Implementation Details. We used 8xAMD MI210 cards for training and evaluation. Every reinforcement learning run uses GRPO via EasyR1 (Zheng et al., 2025) and the parser is fine-tuned with LLaMA-Factory (Zheng et al., 2024). For parsing, we sample each image $K{=}4$ parses at temperature 1.0 and top-p 0.95, up to 2048 tokens, under a per-domain JSON schema the decoder enforces, and score each by the fraction of its claims the checker confirms. We keep the best parse only if

Table 1: **Main results across three scales.** Every model trains on unlabeled images only, with no human question, answer or gold label at any stage. Subscripts give the change against that backbone's own base, in green when positive and red when negative. *VQS gives the largest average gain at all three scales*, +3.18 at 2B, +2.32 at 4B and +2.43 at 8B, against +1.01, +1.04 and +0.66 for the strongest published method. InfoVQA stands for infographicsVQA, SQA for ScienceQA, MMB for MMBench, ESB for EmbSpatial, and LogicV for LogicVista.

Method	GQA	OK-VQA	InfoVQA	SQA	MMMU	MMB	ESB	LogicV	MMStar	SEED	Avg.
Qwen3-VL-2B-Instruct											
Base	58.25	40.76	69.02	79.42	38.92	74.48	68.54	35.04	55.62	72.16	59.22
VisPlay	58.65+0.40	41.15+0.39	69.96+0.94	80.74+1.32	39.27+0.35	74.52+0.04	68.56+0.02	34.93-0.11	55.37-0.25	71.62-0.54	59.48+0.26
Vision-Zero	58.98+0.73	41.39+0.63	70.93+1.91	81.96+2.54	39.58+0.66	75.07+0.59	69.72+1.18	35.28+0.24	55.58-0.04	71.53-0.63	60.00+0.78
EvoLMM	59.01+0.76	38.03-2.73	70.69+1.67	83.01+3.59	39.08+0.16	74.62+0.14	69.32+0.78	34.99-0.05	55.50-0.12	71.17-0.99	59.54+0.32
iReasoner	59.13+0.88	38.13-2.63	70.82+1.80	83.12+3.70	39.11+0.19	74.75+0.27	69.67+1.13	35.09+0.05	55.59-0.03	71.25-0.91	59.67+0.45
VISE	59.41+1.16	41.24+0.48	71.43+2.41	83.61+4.19	40.67+1.75	76.72+2.24	70.14+1.60	31.92-3.12	55.02-0.60	72.18+0.02	60.23+1.01
VQS (ours)	**59.52**+1.27	**42.60**+1.84	**72.01**+2.99	**86.81**+7.39	**46.78**+7.86	**78.26**+3.78	**71.54**+3.00	**36.83**+1.79	**57.04**+1.42	**72.61**+0.45	**62.40**+3.18
Qwen3-VL-4B-Instruct											
Base	60.32	47.86	77.73	87.51	45.17	83.42	76.79	49.78	62.67	75.71	66.70
VisPlay	60.06-0.26	48.04+0.18	77.96+0.23	87.30-0.21	45.42+0.25	83.47+0.05	76.11-0.68	49.34-0.44	62.13-0.54	74.83-0.88	66.47-0.23
Vision-Zero	61.30+0.98	48.41+0.55	78.28+0.55	88.46+0.95	46.27+1.10	**84.83**+1.41	77.02+0.23	49.11-0.67	62.45-0.22	75.01-0.70	67.19+0.49
EvoLMM	61.20+0.88	46.72-1.14	78.58+0.85	88.46+0.95	46.02+0.85	84.21+0.79	76.94+0.15	49.57-0.21	62.37-0.30	74.77-0.94	66.88+0.19
iReasoner	61.32+1.00	46.89-0.97	78.76+1.03	88.62+1.11	46.13+0.96	84.34+0.92	76.98+0.19	49.65-0.13	62.46-0.21	74.84-0.87	67.00+0.30
VISE	61.82+1.50	**48.81**+0.95	**81.45**+3.72	90.04+2.53	48.89+3.72	84.51+1.09	77.46+0.67	47.66-2.12	61.73-0.94	75.00-0.71	67.74+1.04
VQS (ours)	**62.86**+2.54	48.19+0.33	80.23+2.50	**93.11**+5.60	**54.78**+9.61	83.93+0.51	**77.80**+1.01	**50.10**+0.32	**63.29**+0.62	**75.82**+0.11	**69.01**+2.32
Qwen3-VL-8B-Instruct											
Base	61.54	49.84	81.23	90.88	50.12	84.71	77.66	52.23	64.96	74.96	68.81
VisPlay	61.66+0.12	49.93+0.09	81.36+0.13	91.02+0.14	52.61+2.49	84.62-0.09	78.24+0.58	51.88-0.35	64.52-0.44	74.34-0.62	69.02+0.21
Vision-Zero	62.12+0.58	50.31+0.47	82.44+1.21	92.35+1.47	52.89+2.77	85.11+0.40	77.96+0.30	52.33+0.10	64.75-0.21	74.33-0.63	69.46+0.65
EvoLMM	61.94+0.40	50.18+0.34	82.02+0.79	91.96+1.08	51.56+1.44	85.02+0.31	77.51-0.15	52.01-0.22	64.59-0.37	74.12-0.84	69.09+0.28
iReasoner	62.02+0.48	50.24+0.40	82.15+0.92	92.11+1.23	51.83+1.71	85.09+0.38	77.82+0.16	52.09-0.14	64.67-0.29	74.18-0.78	69.22+0.41
VISE	**62.43**+0.89	50.61+0.77	82.83+1.60	92.81+1.93	52.69+2.57	85.46+0.75	**78.62**+0.96	50.78-1.45	64.17-0.79	74.30-0.66	69.47+0.66
VQS (ours)	61.97+0.43	**53.00**+3.16	**83.13**+1.90	**94.65**+3.77	**57.56**+7.44	**85.82**+1.11	78.49+0.83	**54.24**+2.01	**68.14**+3.18	**75.39**+0.43	**71.24**+2.43

it beats the mean of all K parses by δ=0.10 and asserts at least m=4 claims, which turned 12,000 images into 4,792 targets. The parser is then fully fine-tuned on those targets with the vision tower frozen, per-device batch 4 with 8 gradient accumulation steps, learning rate 1e$-$5, 2 epochs, 284 steps. At data-curation time it decodes greedily. The fact-checker v is the same as the parser answering yes or no to one claim at a time, greedy, with four new tokens. The blind gate π_{blind} is the same as the parser with no image. It generates J=4 answers and drops a question if more than a fraction λ=0.5 of them are correct. The difficulty band runs the base solver at the training settings, 8 rollouts at temperature 1.0 with a 256-token budget, and keeps a question only when it is answered correctly at least once and not every time. We train the solver with a curriculum that sorts each template family's questions by hop count h, from fewest to most (Table 6).

5 Results

We answer five questions: (1) Does VQS beat prior self-evolving methods, and how do its gains change with model size? (2) How many generated questions pass the filters, and are their answers correct? (3) Does label-free training make the parser more accurate? (4) Which filters and questions help the solver most? (5) Does the model keep improving over more training cycles?

Main Results. Table 1 compares VQS with five self-evolving methods on ten benchmarks at three Qwen3-VL scales. Existing consistency-based baselines using majority voting or sample agreement show relatively little improvement, as they depend significantly on the base model performance. For example, EvoLMM and iReasoner raise ScienceQA, where the 2B base is already strong, but lower OK-VQA, where the base is weak. VisPlay stays close to the base scores on most benchmarks, because its questioner is rewarded for questions the solver is least sure about, and a majority vote is least reliable on such questions.

VQS gives the largest average gain at every scale (+3.18 over base at 2B, against +1.01 for the strongest baseline) and lifts all ten benchmarks at each. Its labels come from programs run on checked parses, not from the solver, so training can move the model toward answers it did not favour and rarely rewards a wrong one (Table 2). This helps most where the base is weak. At 2B, the majority-vote methods gain almost nothing on MMMU, while VQS gains more than twice as much as any baseline (+7.86 vs +1.75). MMMU and ScienceQA have the largest gains as they



Figure 5: **What scale and coverage change.** **(a)** Share of benchmark answers RL changes, measured per sample against each model's base. **(b)** Smoothed training reward. **(c)** 2B gain over base against the number of template families.

ask the model to read a chart, table, or figure and combine several facts, exactly what the multi-step templates train (Table 5). VISE, the strongest baseline, lowers the scores on LogicVista and MMStar at every scale, while VQS raises both. Like MMMU and ScienceQA, both benchmarks contain many questions that require combining several facts. See A.3 for results on non-Qwen model families.

Effect of model scale. Gains from VQS are largest at the smallest backbone and shrink with size, from $+3.18$ at 2B to $+2.43$ at 8B. All three models reach a training reward of about 0.80 (Fig. 5b), so the larger models learn the generated task as well as the 2B model, but less of what they learn transfers to the benchmarks. Since an average gain cannot distinguish a model that barely changed from one that changed many answers in opposite directions, we also compare each run with its base sample by sample and count the benchmark answers that change. This share falls from 7.82% at 2B to 5.58% at 4B and 4.93% at 8B (Fig. 5a), which matches the smaller gains of the larger models.

Do the generated answers match the image? Since each answer is computed from the parse, a parsing error can still produce a wrong answer, so human raters check the generated answers against the images. We compare three answer sources on 500 shared questions, with 125 per visual domain and coverage across template families. VQS executes templates on checked parses. Majority voting selects the most frequent of five solver answers and keeps it only if $p(y_{majority}) > 0.3$. A model judge selects an accepted answer from the same five candidates. Table 2 reports both correctness and retention: VQS reaches 94.4% correctness while keeping 76% of questions, compared with 76.4% for majority voting and 82.2% for the model judge.

Answer source	Correct (%)	Kept (%)
Templates + checker	**94.4%**	76.0
Majority vote	76.4%	92.0
Model judge	82.2%	90.8

Table 2: **Human evaluation.** Correct is accuracy among retained questions.

Does paraphrasing change the question? The parser rewrites each template question in natural wording, but neither the checker nor the blind gate tests whether the rewrite still asks what the program computed. We check this on 437 questions from 70 template families. The final question still matches its program in 99.8% of cases at 2B, 100% at 4B, and 99.1% at 8B. All four 8B errors swap the two years in a yes/no comparison, so the stored answer becomes wrong.

Parser training selection. We test how to pick the parser's training target from its K samples without labels. We compare three rules against the untrained parser: a random parse, the parse a checker accepts as a whole with one yes or no, and the parse with the highest fraction of verified claims. We rate parse precision and answer accuracy by hand on 150 samples. We also train a 2B solver on the questions built from each parser and report its average over the ten benchmarks. Table 3 shows that per-claim checking is best on all three measures. Over whole-parse checking, it raises parse precision from 85.0 to 87.6 and answer accuracy from 94.0 to 96.1. Solver accuracy follows the same order. Training the parser with any rule lifts it from 58.0 to at least 61.5, and the three rules differ by less than one point. Questions from the untrained parser even drop the solver below the 2B base (59.22, Table 1), likely because 13.2% of their answers are wrong. One reason per-claim checking works better is that a single yes

Parser target	**Parse Prec.**	**Answer Acc.**	**Solver Acc.**
Untrained	80.8	86.8	58.0
Random	82.1	91.0	61.5
Whole parse	85.0	94.0	62.0
Per claim	**87.6**	**96.1**	**62.4**

Table 3: **Parser target selection.** Parse precision, correct answer accuracy, and solver accuracy (%).

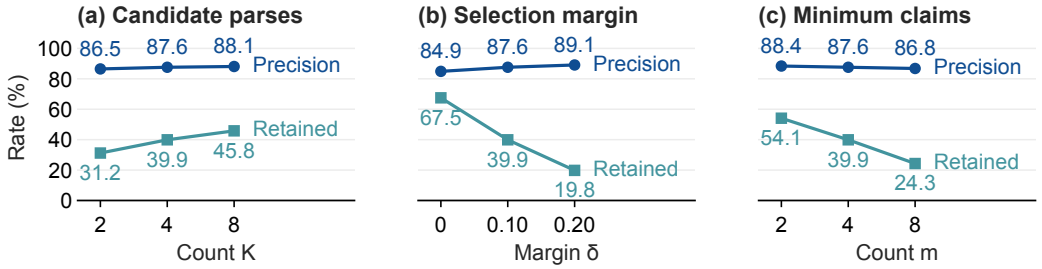


Figure 7: **Parser target selection: quality and coverage.** Each panel varies one setting around $(K, \delta, m) = (4, 0.10, 4)$. Parse precision is the percentage of correct visual claims after training. Images retained is the percentage of source images yielding a selected training target.

or no cannot tell a parse with one error from a parse with five. Hence, when many samples tie, the retained parse is often chosen by chance. In contrast, the fraction of verified claims gives a graded score that separates them. Another reason is that a checker judges one short claim more reliably than a long structure (Min et al., 2023; Jing et al., 2024), in line with findings that checking each fact separately reduces errors (Dhuliawala et al., 2024).

Does label-free training improve the parser? Every VQS answer is computed from the parse, so a parsing error becomes a wrong answer. The parser is also trained on the checker's verdicts, so it could learn to please the checker without becoming more accurate. We therefore evaluate the parser against human annotations. We measure F1 against GQA scene graphs (Hudson & Manning, 2019), ChartQA tables (Masry et al., 2022), and AI2D-RST diagram annotations (Hiippala et al., 2021). Figure 6 compares F1 by domain. Label-free training improves F1 in all three domains, by 5.0 points on GQA, 6.6 on ChartQA, and 7.4 on AI2D-RST. The largest gain comes on diagrams, where the untrained parser is weakest (75.2), although diagrams remain the hardest domain after training (82.6).

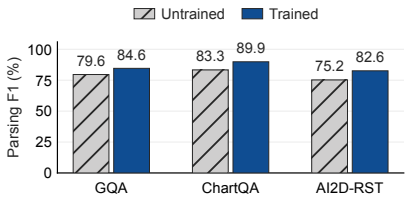


Figure 6: **Parser Structured-F1 Score.**

Parser hyperparameter selection. We also vary the candidate count K, the selection margin δ, and the minimum claim count m (Fig. 7). Raising K or δ improves precision, but only slightly, so we choose a middle setting that keeps computation affordable. Raising m, however, slightly lowers precision. Parses with more claims can produce more questions, but longer parses are more likely to contain hallucinated content (Li et al., 2025).

Does VQS keep improving over several rounds? All results so far (Table 1) come from a single round, or training cycle, in which the parser and the solver are each trained once. Here we test whether more cycles help the solver. At the start of each new cycle, we merge the LoRA weights from the previous solver cycle, generate new QA pairs from our templates on the same images, and combine them with the QA pairs from earlier cycles. We then sample 8 answers to each question, and compute s as the fraction of these answers that are correct. We weight each question by $w(s) = \exp(-((s - 0.5)/0.15)^2)$, give zero weight to questions that are always or never solved, and resample 8K training items with these weights. As a result, questions solved about half the time appear most often. Finally, we train the model from the previous cycle on these items for 96 GRPO steps, with the same settings as all other runs. The average gain over base grows with every cycle (Fig. 8), from 3.18 points after cycle 1 to 3.41 and 3.84 after cycles 2 and 3. The third cycle adds more than the second (0.43 vs 0.23 points), so the curve has not flattened. As with a single cycle, knowledge and reasoning benchmarks gain the most, 5.94 points after three cycles, led by ScienceQA (+8.53) and MMMU (+8.44). Perception and general benchmarks gain 2.62 and 2.24 points. Nine of the ten benchmarks improve in every cycle, and the only exception, MMStar, peaks after cycle 1 but still ends 0.93 points above base.

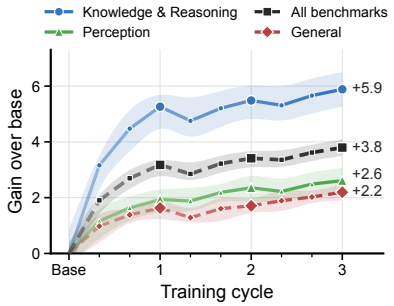


Figure 8: **Gains across cycles.** Average gain over base per benchmark category for Qwen3-VL-2B.

Which filters improve solver accuracy? Table 4 removes one component at a time at 2B. The filters act after parsing, so removing one changes solver accuracy but not parser precision. Removing parser training causes the largest drop, lowering parser precision from 87.6 to 80.8 and solver accuracy from 62.4 to 58.0. Removing the fact-checker causes the second-largest drop, lowering solver accuracy to 60.9 while parser precision stays at 87.6, so unchecked answers alone cost 1.5 points. Each remaining component costs 0.3 to 0.6 points.

Pipeline	Solver Acc.	Parser Prec.
Untrained parser	58.0	80.8
Full VQS	**62.4**	**87.6**
✗ constrained decoding	61.9	85.3
✗ fact-checker	60.9	87.6
✗ blind gate	61.8	87.6
✗ difficulty band	62.1	87.6

Table 4: **Component ablations at 2B.**

Which families gain the most? We measure the solver's gain from each template family, and Table 5 lists the two families with the largest gains and the two with the smallest. The largest gains come from families that combine several reads, such as subtracting two chart values or following diagram arrows. Families that need a single read, such as object existence and attribute lookup, barely change, likely because the base model already answers them reliably. This matches the main results, where reasoning-heavy benchmarks such as MMMU and ScienceQA gain the most. Lastly, Fig. 5c shows that adding template families keeps raising the gain over base. The average gain grows from 1.9 points with 10 families to 3.18 with all 70, and it is still rising. Reasoning benchmarks gain more than perception benchmarks at every family count, reaching 4.7 against 2.4 points.

Rank	Family	Gain
▲ 1	Chart differences	7.4
▲ 2	Diagram connectivity	5.8
▼ 1	Object existence	-0.2
▼ 2	Attribute lookup	0.4

Table 5: **Per-family gains.** The two largest and two smallest.

Which questions are most useful for training? Beyond a correct answer, a question helps the solver only if it needs the image and the current solver answers it correctly sometimes, but not always. We test four design choices at equal training budgets (Table 6). For difficulty, we split template families into easy, middle, and hard thirds by the base solver's accuracy and train on each third alone. For the generator, we replace fixed templates with model-written questions and programs on the same parses, following PROVE (Prabhu et al., 2025). Both generators use the same checkers, and human raters check whether each question asks what its program computes. For the curriculum, we compare shuffled order with ordering by hop count, across all questions or within each family. For the difficulty band, we compare selecting questions once with the base solver against reselecting them with the current solver during training. Middle families work best (62.4), ahead of hard (61.1) and easy (60.8), which matches prior findings that questions of intermediate difficulty give the most learning signal (Bae et al., 2026; Gao et al., 2026). Generated programs lose 1.0 point because only 85.3% of their questions match their program. Ordering by hop count helps, and ordering within each family works best, raising accuracy from 61.3 (shuffled) to 62.4. Reselecting during training brings no gain (62.3), so one selection with the base solver is enough.

Comparison	Setting	Solver acc.
Family difficulty	Easy	60.8
	Middle	**62.4**
	Hard	61.1
Generator	Fixed templates	**62.4**
	Generated programs	61.4
Curriculum	Shuffled	61.3
	Global hops	61.7
	Within-family hops	**62.4**
Difficulty band	Initial selection	**62.4**
	Refreshed selection	62.3

Table 6: **Question design at 2B.** Each block changes one choice at an equal training budget. Solver accuracy averages the 10 benchmarks in Table 1. Bold marks the VQS default.

6 Conclusion

In this work, we show that self-evolving vision-language models can generate hallucinated questions and answers that still pass consistency-based verifiers. In our human evaluation, 24% of majority-vote labels and 18% of model-judge labels are wrong. To enable principled generation and verification, we propose VQS, a template-based self-evolving framework that computes answers with fixed programs over a structured parse of each image and checks every fact the programs read, lowering this error to 6%. The same checks also train the parser without labels. VQS applies across four image domains, and across ten benchmarks it improves Qwen3-VL by up to 3.18 points, beating the strongest baseline at every scale and reaching 3.84 points at 2B after three cycles.

Reproducibility statement

Section 4 gives the full evaluation protocol and every training hyperparameter, including the framework commit and algorithm variant. All numbers we measure use the single protocol in Section 4. Baseline numbers are taken from (Venkatraman et al., 2026). Appendix A.1 lists the training data, reward, step count, and checkpoint behind every table. The supplementary material contains anonymized code for the generator, filters, and reward functions, along with the statistics needed to rebuild the training data from public images.

Ethics statement

We train on public images from existing vision-language datasets without using their annotations. No personal data are involved. Our questions are generated from the model's description of each image, so they inherit the parser's biases and blind spots. We do not audit these biases here, and downstream users should. By lowering the cost of improving vision-language models, the method can serve both beneficial and harmful uses.

AI use statement

We used generative AI to help edit this paper and writing some of the experimental code. The authors read, ran, and checked all code behind our numbers, and verified that every aggregate reproduces from the per-benchmark outputs.

References

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, 2026.

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

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14455–14465, 2024a.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087, 2024b.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In *Findings of the association for computational linguistics: ACL 2024*, pp. 3563–3578, 2024.

Mengfei Du, Binhao Wu, Zejun Li, Xuan-Jing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 346–355, 2024.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training. In *International Conference on Learning Representations*, volume 2026, pp. 58614–58652, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. Visplay: Self-evolving vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26274–26284, 2026.

Tuomo Hiippala, Malihe Alikhani, Jonas Haverinen, Timo Kalliokoski, Evanfiya Logacheva, Serafina Orekhova, Aino Tuomainen, Matthew Stone, and John A Bateman. Ai2d-rst: a multimodal corpus of 1000 primary school science diagrams: T. hiippala et al. *Language Resources and Evaluation*, 55(3):661–688, 2021.

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*, volume 2026, pp. 130770–130790, 2026.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 2019.

Liqiang Jing, Ruosen Li, Yunmo Chen, and Xinya Du. Faithscore: Fine-grained evaluations of hallucinations in large vision-language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5042–5063, 2024.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13299–13308. IEEE, 2024.

Zhuowei Li, Haizhou Shi, Yunhe Gao, Di Liu, Zhenting Wang, Yuxiao Chen, Ting Liu, Long Zhao, Hao Wang, and Dimitris N Metaxas. The hidden life of tokens: Reducing hallucination of large vision-language models via visual information steering. *arXiv preprint arXiv:2502.03628*, 2025.

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

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521, 2022.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 3190–3199. IEEE, 2019.

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2582–2591. IEEE, 2022.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 12076–12100, 2023.

Viraj Prabhu, Senthil Purushwalkam, An Yan, Caiming Xiong, and Ran Xu. Trust but verify: Programmatic vlm evaluation in the wild. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3258–3267. IEEE, 2025.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Meghana Sunil, Manikandarajan Venmathimaran, and Muthu Subash Kavitha. ireasoner: Trajectory-aware intrinsic reasoning supervision for self-evolving large multimodal models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 29361–29372, 2026.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Omkar Thawakar, Shravan Venkatraman, Ritesh Thawkar, Abdelrahman Shaker, Hisham Cholakkal, Rao Muhammad Anwer, Salman Khan, and Fahad Khan. Evolmm: Self-evolving large multimodal models with continuous rewards. *arXiv preprint arXiv:2511.16672*, 2025.

Shravan Venkatraman, Ritesh Thawkar, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Salman Khan, and Fahad Shahbaz Khan. Paying more attention to visual tokens in self-evolving large multimodal models. In *European Conference on Computer Vision*, pp. 41–59. Springer, 2026.

Qinsi Wang, Bo Liu, Tianyi Zhou, Jing Shi, Yueqian Lin, Yiran Chen, Hai Helen Li, Kun Wan, and Wentian Zhao. Vision-zero: Scalable vlm self-improvement via strategic gamified self-play. *arXiv preprint arXiv:2509.25541*, 2025.

Lai Wei, Yuting Li, Chen Wang, Yue Wang, Linghe Kong, Weiran Huang, and Lichao Sun. First sft, second rl, third upt: Continual improving multi-modal llm reasoning via unsupervised post-training. *Advances in Neural Information Processing Systems*, 38:62293–62318, 2026.

Zhengxian Wu, Kai Shi, Chuanrui Zhang, Zirui Liao, Jun Yang, Ni Yang, Qiuying Peng, Luyuan Zhang, Hangrui Xu, Tianhuang Su, et al. When models judge themselves: Unsupervised self-evolution for multimodal reasoning. *arXiv preprint arXiv:2603.21289*, 2026.

Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts. *arXiv preprint arXiv:2407.04973*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Jieyu Zhang, Le Xue, Linxin Song, Jun Wang, Weikai Huang, Manli Shu, An Yan, Zixian Ma, Juan Carlos Niebles, Silvio Savarese, et al. Provision: Programmatically scaling vision-centric instruction data for multimodal language models. *arXiv preprint arXiv:2412.07012*, 2024.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 881–916, 2025.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pp. 400–410, 2024.

Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, Yuwen Xiong, and Richong Zhang. EasyR1: An efficient, scalable, multi-modality RL training framework. `https://github.com/hiyouga/EasyR1`, 2025.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

A APPENDIX

Contents

Training and evaluation settings . . . 14
Benchmark categories . . . 14
More Results . . . 14
Is the checker independent of the parser? . . . 14
Does VQS transfer to other model families? . . . 15
Prompts . . . 16
Parsing an image into a structure . . . 16
Rewriting the templated question . . . 19
The blind gate . . . 19
Training-time prompts . . . 19
Model-written programs (ablation) . . . 19
Baseline prompts . . . 20
Qualitative examples . . . 21
Traced questions . . . 22
The program behind a template . . . 24
VQS, VISE and the base model on the same benchmark questions . . . 24

A.1 TRAINING AND EVALUATION SETTINGS

We train the solver with GRPO (Shao et al., 2024) on top of a LoRA adapter of rank 64, applied to all linear layers with the vision modules excluded, so the vision tower stays frozen and no visual parameters are trained. We use AdamW at a constant learning rate of $1e{-5}$, a low-variance KL penalty against a frozen reference with $\beta = 0.01$, and clip ratios of 0.2 and 0.3 with dual clipping at 3.0. For each prompt we draw 8 rollouts at temperature 1.0 with a 256-token response budget and cap training images at $1{,}003{,}520$ pixels. We collect 256 prompts per rollout batch, update on a global batch of 128, and train for 96 steps. The three scales share these settings, except that the per-device micro-batch is 16 at 2B and 4 at 8B. Table 7 lists the full settings.

A.2 BENCHMARK CATEGORIES

We group the ten benchmarks into three categories by the main skill each one tests (Table 8). **Perception** benchmarks ask about what is visible in the image, such as objects, relations, printed text, and spatial layout. **Knowledge & Reasoning** benchmarks also need something the image does not supply, such as outside facts, subject knowledge, or logical reasoning. **General** benchmarks mix both kinds of questions, so we keep them as a separate group. A category's score is the unweighted mean of its benchmarks, and the overall score is the mean over all ten, as in Table 1.

A.3 MORE RESULTS

Is the checker independent of the parser? The parser, checker, and solver share the same Qwen3-VL weights, so the checker could share the parser's blind spots and confirm its mistakes. To test this, we grade the same claims with five checkers of different families and sizes (Table 9). From 150 parses, we extract 1,417 atomic claims. Of these, 1,170 come from the parser, and 247 are traps in which we change a number or a label, so the correct verdict is always no. All checkers use the same prompt and greedy decoding. The other models only audit our checker and never touch the training data, so training still uses a single model. Our checker rejects 244 of the 247 traps, so it does not simply approve what the parser writes. Agreement between checkers depends more on size than on family. The two 8B checkers from different families agree with each other more (κ=0.877) than our 2B checker agrees with Qwen3-VL-8B (κ=0.837). SmolVLM2-2.2B agrees least with every other checker (κ from 0.62 to 0.66) and accepts the most traps (5.7%), so it is a weaker checker rather than an independent one. An independent check therefore needs a larger model, not just a different family. The risk also depends on the claim type. Among the claims our checker accepts,

Table 7: **Solver training settings.** Identical across 2B, 4B, and 8B except for the per-device micro-batch.

Setting	Value
Adapter	
LoRA rank, alpha	64, 128
Target modules	all linear layers, `.*visual.*` excluded
Trainable visual parameters	0.00M
Gradient checkpointing	on
Optimiser	
Algorithm	AdamW, betas $(0.9, 0.999)$, weight decay 0.01
Learning rate	$1e{-5}$, constant
GRPO	
Advantage estimator	GRPO
KL penalty	low-variance estimator, $\beta = 0.01$, frozen reference
Clip ratio (low, high)	0.2, 0.3, dual clipping at 3.0
PPO epochs per step	1
Dynamic sampling	off
Rollout	
Rollouts per prompt	8 at temperature 1.0, top-p 1.0
Prompt budget	2048 tokens
Response budget	256 tokens
Training image cap	1,003,520 pixels
Batching and schedule	
Rollout batch	256 prompts
Global update batch	128
Micro-batch per device	16 at 2B, 4 at 8B
GPU memory utilisation	0.60
Steps	96

Table 8: **Benchmark categories.** Each benchmark belongs to one category.

Category	Benchmark	What it tests
Perception	GQA	objects, attributes, and relations
	InfoVQA	text and layout in infographics
	EmbSpatial (ESB)	spatial relations in embodied scenes
Knowledge & Reasoning	OK-VQA	outside knowledge
	ScienceQA (SQA)	school science
	MMMU	college-level subjects
	LogicVista (LogicV)	logical reasoning over images
General	MMBench (MMB)	many perception and reasoning skills
	MMStar	questions that require the image
	SEED-Bench (SEED)	broad visual understanding

at least two other checkers reject 26.7% of diagram arrows, 8.3% of chart values, and none of the object-existence claims. Diagram arrows and chart values are therefore where the checker most needs improvement. Since another checker's rejection is not ground truth, we treat these rates as warning signs rather than error rates.

Does VQS transfer to other model families? We also apply VQS to Gemma3-12B-It (Team et al., 2025), InternVL3-8B-Instruct (Zhu et al., 2025), and Llama-3.2-11B-Vision-Instruct (Grattafiori et al., 2024) (Table 10). VQS improves all ten benchmarks for every family, raising the average by 1.77 points on Gemma, 1.51 on InternVL3, and 1.84 on Llama. Gains are large on MMMU, InfoVQA, and EmbSpatial, about 3 to 4 points per family, and smallest on SEED-Bench and OK-VQA, under 0.7 points. The weakest base, Llama (55.77), gains the most, and the strongest, InternVL3 (64.46), gains the least. All three gains are smaller than on Qwen3-VL (+2.32 to +3.18, Table 1).

Table 9: **Checker independence.** All five checkers grade the same 1,417 claims. Accepts is the share of the 1,170 parser claims a checker calls true. Trap YES is the share of the 247 corrupted claims it wrongly accepts. κ is Cohen's kappa with our checker over all claims.

Checker	Relation to the parser	Accepts (%)	Trap YES (%)	κ vs. ours
Qwen3-VL-2B (ours)	same weights	94.4	1.2	–
Qwen3-VL-8B	same family, 4× larger	93.3	1.6	0.837
InternVL3.5-8B	different family, 8B	93.1	4.0	0.810
Qwen2.5-VL-7B	previous Qwen generation	89.5	0.4	0.807
SmolVLM2-2.2B	different family, same size	86.1	5.7	0.660

Table 10: **Results on other model families.** Every model trains on unlabeled images only. Subscripts give the change against that backbone's base. Abbreviations follow Table 1. *VQS improves all ten benchmarks for every family*, raising the average by +1.77 on Gemma3-12B-It, +1.51 on InternVL3-8B-Instruct and +1.84 on Llama-3.2-11B-Vision-Instruct.

Method	GQA	OK-VQA	InfoVQA	SQA	MMMU	MMB	ESB	LogicV	MMStar	SEED	Avg.
Gemma3-12B-It											
Base	61.20	62.40	52.10	94.30	46.50	76.40	24.50	41.80	58.90	74.20	59.23
VQS (ours)	**62.02**+0.82	**62.71**+0.31	**55.29**+3.19	**95.23**+0.93	**50.84**+4.34	**79.06**+2.66	**27.70**+3.20	**42.87**+1.07	**59.80**+0.90	**74.47**+0.27	**61.00**+1.77
InternVL3-8B-Instruct											
Base	65.40	64.10	68.80	96.50	51.20	81.30	28.10	48.60	63.70	76.90	64.46
VQS (ours)	**66.01**+0.61	**64.36**+0.26	**71.81**+3.01	**96.71**+0.21	**54.87**+3.67	**83.17**+1.87	**31.99**+3.89	**49.42**+0.82	**64.32**+0.62	**77.08**+0.18	**65.97**+1.51
Llama-3.2-11B-Vision-Instruct											
Base	56.00	55.40	46.90	91.80	50.70	72.80	21.20	36.50	55.10	71.30	55.77
VQS (ours)	**57.13**+1.13	**56.05**+0.65	**49.87**+2.97	**93.47**+1.67	**53.64**+2.94	**76.04**+3.24	**24.18**+2.98	**37.79**+1.29	**56.22**+1.12	**71.68**+0.38	**57.61**+1.84

A.4 Prompts

Text in braces, such as `{q}`, is a Python format field filled at call time. The prompts in Section A.4.6 belong to the two baselines we compare against, which are the only places a model is asked to judge or vote on an answer.

A.4.1 Parsing an image into a structure

One prompt per domain, each paired with a JSON schema enforced by constrained decoding, so valid JSON holds by construction rather than by instruction.

Charts

```
Read this chart. List every series and every data point you can see.
Copy `printed_value` exactly as printed on the chart (keep %, $, B,
commas). Put the plain number in `numeric_value`. Use the real
category and axis names.
```

Diagrams

```
Read this diagram. List every labelled part as a node, copying
labels verbatim. Set has_arrows to true ONLY if the diagram really
draws arrows from one labelled part to another, like a cycle or a
food web. Many diagrams only label parts and have no such arrows --
for those set has_arrows to false and leave edges empty. Never
invent an arrow.
```

Natural images

```
Describe this photo as structure. List the main objects with their
visible attributes and boxes in 0-1000 coordinates as [x0,y0,x1,y1].
Then list how the objects relate to each other, using simple
predicates like "on", "holding".
```

Infographics

```
Read this infographic. List every labelled figure or statistic as an
entry. Copy `printed_value` exactly as printed (keep %, $, commas,
K/M/B). Put the plain number in `numeric_value` and the unit in
`unit`.
```

The schemas below are the ones the decoder enforces through vLLM's guided decoding, as JSON Schema with whitespace compacted. Every field is required and no object admits extra fields, so a parse can neither omit a field nor add one. A `numeric_value` may be `null`: a value that is drawn but not legible is recorded as null rather than guessed, and the templates skip it.

Charts: output schema

```
{
  "type": "object",
  "additionalProperties": false,
  "required": ["kind", "title", "x_axis_title", "y_axis_title", "series"],
  "properties": {
    "kind": {"const": "chart"},
    "title": {"type": "string"},
    "x_axis_title": {"type": "string"},
    "y_axis_title": {"type": "string"},
    "series": {
      "type": "array",
      "maxItems": 6,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": ["series_name", "points"],
        "properties": {
          "series_name": {"type": "string"},
          "points": {
            "type": "array",
            "maxItems": 24,
            "items": {
              "type": "object",
              "additionalProperties": false,
              "required": ["category", "printed_value", "numeric_value"],
              "properties": {
                "category": {"type": "string"},
                "printed_value": {"type": "string"},
                "numeric_value": {"type": ["number", "null"]}}}}}}}}}
```

Diagrams: output schema

```
{
  "type": "object",
  "additionalProperties": false,
  "required": ["kind", "title", "has_arrows", "nodes", "edges"],
  "properties": {
    "kind": {"const": "diagram"},
    "title": {"type": "string"},
    "has_arrows": {"type": "boolean"},
    "nodes": {
      "type": "array",
      "maxItems": 20,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": ["node_id", "node_label"],
        "properties": {
```

```
          "node_id": {"type": "integer"},
          "node_label": {"type": "string"}}}},
    "edges": {
      "type": "array",
      "maxItems": 30,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": ["from_node_id", "to_node_id", "edge_label"],
        "properties": {
          "from_node_id": {"type": "integer"},
          "to_node_id": {"type": "integer"},
          "edge_label": {"type": "string"}}}}}}
```

Natural images: output schema

```
{
  "type": "object",
  "additionalProperties": false,
  "required": ["kind", "objects", "relations"],
  "properties": {
    "kind": {"const": "photo"},
    "objects": {
      "type": "array",
      "maxItems": 10,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": ["object_id", "object_name", "attributes", "box_0_1000"],
        "properties": {
          "object_id": {"type": "integer"},
          "object_name": {"type": "string"},
          "attributes": {
            "type": "array",
            "maxItems": 3,
            "items": {"type": "string"}},
          "box_0_1000": {
            "type": "array",
            "minItems": 4,
            "maxItems": 4,
            "items": {"type": "integer"}}}}},
    "relations": {
      "type": "array",
      "maxItems": 12,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": ["subject_id", "predicate", "object_id"],
        "properties": {
          "subject_id": {"type": "integer"},
          "predicate": {"type": "string"},
          "object_id": {"type": "integer"}}}}}}
```

Infographics: output schema

```
{
  "type": "object",
  "additionalProperties": false,
  "required": ["kind", "title", "entries"],
  "properties": {
    "kind": {"const": "infographic"},
    "title": {"type": "string"},
    "entries": {
      "type": "array",
      "maxItems": 20,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": ["entry_label", "printed_value", "numeric_value", "unit"],
        "properties": {
          "entry_label": {"type": "string"},
          "printed_value": {"type": "string"},
          "numeric_value": {"type": ["number", "null"]},
          "unit": {"type": "string"}}}}}}
```

A.4.2 REWRITING THE TEMPLATED QUESTION

The template program emits the question; the model then rewrites it. The instruction forbids adding or removing conditions and restricts the rewrite to words already present, because an unconstrained rewrite silently changes the computation. The gold answer is never shown and never touched.

Paraphrase

```
Rewrite this question so it sounds natural, keeping its meaning
exactly the same.
Do not answer it. Do not add or remove any condition. Use only the
words and entities already in the question. Reply with only the
rewritten question.

Question: {q}
```

A.4.3 THE BLIND GATE

The shipped question is answered $k{=}4$ times by the same model *with the image withheld*.

Blind gate (no image)

```
Answer this question with your best guess. You cannot see the image,
so guess from the wording alone. Reply with the answer only, no
explanation.

Question: {q}
```

A.4.4 TRAINING-TIME PROMPTS

Each training item is the image, the question, and a suffix fixing the answer format. The format string is chosen by the program's answer type.

Answer-format suffix

answer type = word	`Answer the question using a single word or phrase.`
answer type = number	`Answer the question using a single number.`
multiple-choice pool	`Answer with the option's letter from the given choices directly.`

A.4.5 MODEL-WRITTEN PROGRAMS (ABLATION)

In the ablation where the model writes the question, the answer *and* the program, the parse is supplied as JSON and only the executor decides correctness.

Question, answer and program written by the model

```
You are given an image and a JSON description of it that was read
off the image.

{graph}

Write ONE question about the image, its short answer, and a Python
program that COMPUTES that answer from the JSON.

Rules for the program:
- exactly one function: def solve(g): ... where g is the JSON above,
as a dict
- return the answer; it will be compared to your "answer" field as a
string
```

```
- read the values out of g. Do NOT write the answer as a literal in
the program
- plain Python only: indexing, loops, if, len, sum, min, max,
sorted, round, abs, str, int, float
- no imports; use g["key"] and for x in g["list"]; string methods
like .lower() and .strip() are fine

Rules for the question:
- answerable from the image alone by someone who cannot see the JSON
- the answer must be a single word, number or short phrase
- do not name the answer in the question

Return JSON with keys question, answer, program.
```

A.4.6 BASELINE PROMPTS

These are the prompts of the two baselines we compare our generator against. Both start from the same free-form question prompt.

Question generation (both baselines)

```
Write one question about this image that can be answered with a
single word, phrase or number by looking at the image. Reply with
only the question.
```

Majority vote. The question is answered 5 times at T=1.0 with the standard format suffix, and the plurality answer is kept as the label. There is no prompt beyond the suffix.

Judge

```
You are grading one answer to a question about an image.
Question: {q}
Proposed answer: {a}

Look at the image and decide whether the proposed answer is correct.
Reply with exactly one word: YES if it is correct, NO if it is not.
```

A.5 QUALITATIVE EXAMPLES

This section shows what the generator produces. Everything in it comes from Qwen3-VL-2B parsing, checking and gating its own images. Table 11 lists twenty template families with a question each one wrote. Section A.5.1 follows single questions from the parse to the answer key, including two that the filters removed, and Section A.5.2 shows the code that computes one family's answer. Levels follow the generator's operand depth: L1 lookup, L2 single hop, L3 two operands, L4 derived operand, L5 comparison of aggregates.

Table 11: **Template families, with one question each.** A slot in ⟨*angle brackets*⟩ is filled from the parse, and a part in [square brackets] appears only when the chart has several series. Each example is a question the family wrote for a real image, shown after paraphrasing where the paraphrase passed the guard, with the key its program computed. We checked every key against its image.

Family	Lv	Template → example question	Answer key
Photos			
`existence`	L2	Are there any ⟨*objects*⟩ in the image? → Are there any windows in the image?	no
`relation_existence`	L2	Are there any ⟨*objects*⟩ ⟨*relation*⟩ the ⟨*anchor*⟩? → Are there any chairs to the right of the painting?	yes
`attribute_filtered_existence`	L3	Is there a ⟨*attribute*⟩ ⟨*object*⟩ in the picture? → Is there white grass in the picture?	no
`state_filtered_existence`	L3	Do you see a ⟨*state*⟩ ⟨*object*⟩ in the picture? → Do you see an open oven door in the picture?	yes
`gated_object_recognition`	L3	Which kind of ⟨*category*⟩ is ⟨*attribute*⟩? → Which kind of furniture is green?	bench
`relation_chain`	L4	Are there any ⟨*objects*⟩ ⟨*relation*⟩ the ⟨*anchor*⟩ that is ⟨*relation*⟩ the ⟨*target*⟩? → Are there any watches to the left of the man that is to the right of the baseball bat?	yes
Charts			
`retrieve`	L1	What is the value of ⟨*category*⟩[for ⟨*series*⟩]? → What is the value of Indiana?	0.53
`extremum`	L2	Which category has the ⟨*highest* | *lowest*⟩ value[for ⟨*series*⟩]? → Which category has the highest value?	2025*
`compare`	L2	Which has a higher value, ⟨*a*⟩ or ⟨*b*⟩[for ⟨*series*⟩]? → Which has a higher value, 2018 or 2020?	2018
`diff`	L3	How much higher is ⟨*a*⟩ than ⟨*b*⟩[for ⟨*series*⟩]? → How much higher is 2019 than 2018 for Stayovers?	31984
`ratio`	L3	What is the ratio of ⟨*a*⟩ to ⟨*b*⟩[for ⟨*series*⟩]? → What is the ratio of Somewhat prefer theater to Strongly prefer streaming for June 2020?	0.44
`series_compare`	L3	In ⟨*category*⟩, which is higher, ⟨*series*⟩ or ⟨*series*⟩? → In 2016, which is higher, China or United States?	China
`ranked_arithmetic`	L5	Is the sum of the two highest values greater than the sum of the three lowest?	No
Infographics			
`ig_chart_metadata`	L1	What is the title of this chart?	THE WORLD'S 10 LARGEST STADIUMS
`top_k`	L2	Which are the ⟨*k*⟩ highest[in ⟨*title*⟩]? → Which are the 3 highest in HOW AMERICANS RATE BUSINESS FIGURES?	Bill Gates, Warren Buffett and Mark Cuban
`multi_item_filter`	L2	Which have a value greater than ⟨*threshold*⟩[in ⟨*title*⟩]? → Which have a value greater than 37 in HOW AMERICANS RATE BUSINESS FIGURES?	Bill Gates, Warren Buffett, Mark Cuban and Elon Musk
`ig_diff`	L3	How much higher is ⟨*a*⟩ than ⟨*b*⟩? → How much higher is China than the U.S.?	994
`compound_lookup`	L3	Which is highest[in ⟨*title*⟩] and what is its value? → Which is highest in HOW AMERICANS RATE BUSINESS FIGURES and what is its value?	Bill Gates, 45
Diagrams			
`diagram_identification`	L1	What is shown in this diagram? → What is depicted in this diagram?	Types of Volcanoes
`existence`	L2	Does every labelled item in this diagram have an arrow attached?	yes

A.5.1 TRACED QUESTIONS

Each card follows one generated question through the pipeline. **Parse** lists the fields the template read, **Program** lists the operations that compute the answer, and **Question** gives the template question with its paraphrase. **Fact check** gives the checker's yes or no verdict on each claim behind the answer, and **Blind gate** lists four answers from the same model without the image (Eq. (1)). The last two cards show questions that the filters removed.

Photo · gated_object_recognition · L3 — KEPT



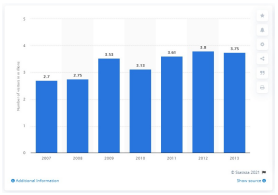
Parse	furniture objects whose color is green: {bench}
Program	`gated_object_recognition(category=furniture, color=green)`, which must find exactly one kind
Question	Which kind of furniture is green?
Fact check	read back the bench's color: "green" ✓ *(attribute read-back)*
Blind gate	sofa · couch · "The green furniture is a chair." · "leafy green furniture" 0/4 match ✓
Answer key	**bench**

Photo · attribute_filtered_existence · L3 — KEPT



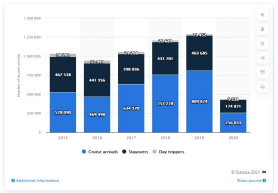
Parse	grass objects carrying the attribute white: none
Program	`attribute_filtered_existence(object=grass, attribute=white)`
Question	Is there a white grass in the picture? → Is there white grass in the picture? *(paraphrase accepted)*
Fact check	asked whether there is white grass: "no" ✓
Blind gate	Yes · Yes · Yes · Yes 0/4 match ✓
Answer key	**no**

Chart · ranked_arithmetic · L5 — KEPT



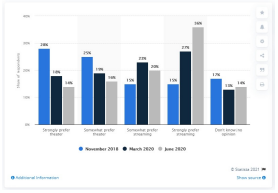
Parse	*Number of visitors in millions*: 2007 2.7 · 2008 2.75 · 2009 3.53 · 2010 3.13 · 2011 3.61 · 2012 3.8 · 2013 3.75
Program	two highest $3.8 + 3.75 = 7.55$; three lowest $2.7 + 2.75 + 3.13 = 8.58$; $7.55 > 8.58$ is false
Question	Is the sum of the two highest values greater than the sum of the three lowest?
Fact check	7/7 values read back exactly, e.g. 2009 → 3.53 ✓
Blind gate	Yes · Yes · Yes · Yes 0/4 match ✓
Answer key	**No**

Chart · diff · L3 — KEPT



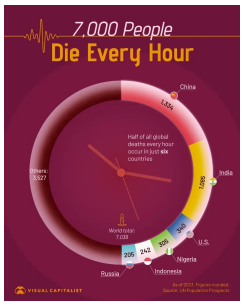
Program	$\lvert \text{Stayovers}(2019) - \text{Stayovers}(2018) \rvert$ on a stacked chart with three series
Question	How much higher is 2019 than 2018 for Stayovers?
Fact check	read back 463,685 and 431,701; the difference recomputed from them, 31,984, matches the key ✓
Blind gate	10 · 1 · 10 · 1 0/4 match ✓
Answer key	**31984**

Chart · ratio · L3 — KEPT



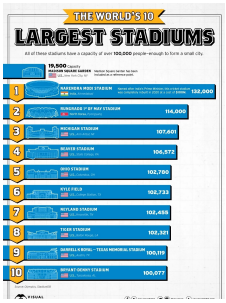
Program	Somewhat prefer theater ÷ Strongly prefer streaming, in the June 2020 series
Question	What is the ratio of Somewhat prefer theater to Strongly prefer streaming for June 2020?
Fact check	read back 16 and 36; the ratio recomputed from them, $16/36 = 0.444$, matches the key ✓
Blind gate	1.2 · 0.22 · 0.67 · 0.50 0/4 match ✓
Answer key	**0.44**

Infographic · ig_diff · L3 — KEPT



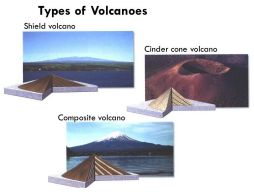
Program	| China − U.S. |
Question	How much higher is China than U.S.? → How much higher is China than the U.S.? *(paraphrase accepted)*
Fact check	read back 1,334 and 340; the difference recomputed from them, 994, matches the key ✓
Blind gate	"10000000" · "China is taller than the U.S. by about 3,000 miles." · two more free-text guesses 0/4 match ✓
Answer key	**994**

Infographic · ig_chart_metadata · L1 — KEPT



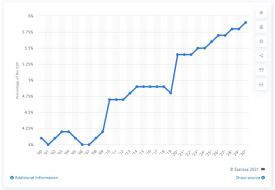
Parse	title: THE WORLD'S 10 LARGEST STADIUMS
Program	read the title field of the parse
Question	What is the title of this chart?
Fact check	read back "The World's 10 Largest Stadiums" ✓ (case is ignored)
Blind gate	"Line Chart" · "Average household income by age group" · two free-text guesses 0/4 match ✓
Answer key	**THE WORLD'S 10 LARGEST STADIUMS**

Diagram · diagram_identification · L1 — KEPT



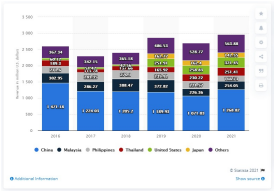
Parse	title: Types of Volcanoes
Program	read the title field of the parse
Question	What is shown in this diagram? → What is depicted in this diagram? *(paraphrase accepted)*
Fact check	read back the title: "Types of Volcanoes" ✓
Blind gate	"The diagram depicts a chloroplast." · "A bird in the sky." · two more free-text guesses 0/4 match ✓
Answer key	**Types of Volcanoes**

Chart · ranked_aggregate · L4 — DROPPED BY FACT CHECK



Parse	*Percentage of the GDP*: '00 4.1 · '01 3.9 · … · '20 4.7 · '21 4.8 · '22 4.8 · '23 4.8 *(24 values; the parse stops at '23)*
Program	three highest $4.8 + 4.8 + 4.8 = 14.4$
Question	What is the sum of the three highest values? *(paraphrase "three largest" rejected: new content word)*
Fact check	21 of 24 read-backs pass; '21 read back as 5.45 ✗, '22 as 5.4 ✗, '23 as 5.45 ✗. The chart shows 5.4 to 5.5 for these years and reaches 5.9 by '30*, so the key 14.4 was wrong.
Answer key	**14.4**, never used

Chart · series_compare · L3 — DROPPED BY BLIND GATE



Program	compare the China and United States series at 2016, on a stacked chart with seven series
Question	In 2016, which is higher, China or United States?
Fact check	read back 1471.18 for China ✓ and 189.3 for United States, where the chart shows 60.12 (189.3 is Thailand's value). The comparison recomputed from the two read-backs, China, still matches the key ✓, so the probe passes despite the misread.
Blind gate	China · China · China · China 4/4 match ✗. The key is correct, but a model without the image answers it from world knowledge, so the question teaches nothing about the image.
Answer key	**China**, never used

A.5.2 The program behind a template

A family is a question template paired with an executor that computes the answer from the parse. Below is the pair for `ranked_arithmetic`, the family of the third card. The executor returns nothing rather than a guess when the parse cannot support a unique answer: too few values, or a tie. The cells it read (`allprov`) are what the fact check later asks the model to verify.

`ranked_arithmetic`: template and executor

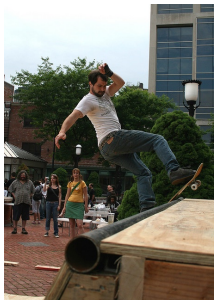
```
# question side: the template, filled from the parse
q = (f"Is the sum of the two highest {subj or 'values'}{sr} greater than the "
     f"sum of the three lowest?")

# answer side: the executor, run on the parsed values of one series
if fam == "ranked_arithmetic":
    k, m = int(prog["k"]), int(prog["m"])        # here k = 2, m = 3
    if len(vals) < k + m:
        return None                              # too few values: no question
    vs = sorted(vals, reverse=True)
    top, bot = sum(vs[:k]), sum(vs[-m:])
    if top == bot:
        return None                              # a tie has no answer: no question
    return ("Yes" if top > bot else "No"), "free", allprov
```

A.5.3 VQS, VISE and the base model on the same benchmark questions

Each card shows one benchmark item answered by three 2B models: the base model, VISE, and VQS after one training cycle. All responses are taken from our lmms-eval runs, where the three models received the same prompt. Red text explains why the other two answers are wrong.

GQA — sample 420



Question	Is the woman near the man standing on the bricks?
Gold	**yes**
VQS (ours)	yes ✓
VISE	no ✗
Base	no ✗
Why wrong	The woman beside the bearded man in the background is standing on the brick pavement.

OK-VQA — sample 3993



Question How old are the boys?
Gold **5**
VQS (ours) 5 ✓
VISE 7 ✗
Base 7 ✗
Why wrong None of the ten human annotators answered 7, so it scores 0 under VQA accuracy; the players are small children.

InfoVQA — sample 150



detail of the "by state" panel

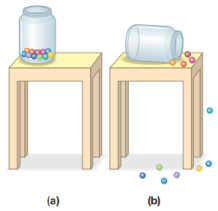
Question How many states are having a poor job Outlook?
Gold **5**
VQS (ours) 5 ✓
VISE 4 ✗
Base 4 ✗
Why wrong Five states sit under the thumbs-down in "Jobs outlook by state": Rhode Island, New Jersey, Washington, Oregon and Alaska. Four is the size of the thumbs-up group (Georgia, Hawaii, Florida, Utah).

ScienceQA — sample 934

Planet	Volume (billions of km^3)	Primary composition
Mercury	60	rock
Venus	930	rock
Earth	1,090	rock
Mars	160	rock
Jupiter	1,431,280	gas
Saturn	827,130	gas
Uranus	68,330	ice
Neptune	62,530	ice

Question Context: Use the data to answer the question below. Is the following statement about our solar system true or false? The volume of Mars is more than three times as large as Mercury's. A. false B. true
Gold **A**
VQS (ours) A. false ✓
VISE B. true ✗
Base B. true ✗
Why wrong The table gives Mars 160 and Mercury 60 billion km^3, so Mars is $160/60 \approx 2.7$ times Mercury's volume, less than three.

MMMU — sample 758



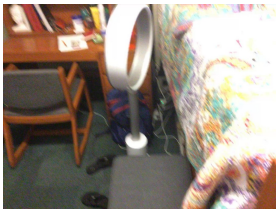
Question Look at the two pictures below. Which picture appears to have more order? A. a B. b C. same of the two
Gold **A**
VQS (ours) A. a ✓
VISE B. b ✗
Base B. b ✗
Why wrong In (b) the jar lies on its side and the marbles are scattered over the table and the floor; in (a) they are all inside the upright jar, the more ordered state.

MMBench sample 98



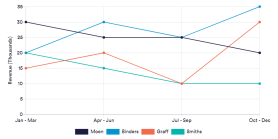
Question	Select the best answer. Which property do these three objects have in common? A. blue B. smooth C. flexible
Gold	**B**
VQS (ours)	B ✓
VISE	C ✗
Base	C ✗
Why wrong	Only the silk tie is flexible; the glass bottle and the wooden baseball bat are rigid. All three are smooth.

EmbSpatial sample 3056



Question	Which object, in relation to your current position, holds the farthest placement in the image? A) chair B) Unknown C) box D) shelves
Gold	**D**
VQS (ours)	D ✓
VISE	C ✗
Base	C ✗
Why wrong	The box is the grey block in the foreground, the object nearest the camera. The shelves on the desk at the back of the room are the farthest.

LogicVista sample 255



Question	Assuming the report has overestimated revenues by 15%, what was the actual revenue generated by Binders in the quarter October to December? Select from A, B, C, and D. (A) 30,435 (B) 25,500 (C) 29,750 (D) 35,000
Gold	**A**
VQS (ours)	A ✓
VISE	C ✗
Base	C ✗
Why wrong	A 15% overestimate means the reported 35 is 1.15 times the actual value, so the actual revenue is $35/1.15 \approx 30.4$ thousand. Both baselines take 85% of the reported value instead, and both say the chart covers three quarters when it shows four. VISE also ends with a malformed `<CAnswer>` tag.

LogicVista, sample 255: full responses

VQS (ours)

...The chart shows the revenue in thousands for four different companies over four quarters. The blue line represents Binders. In the "Oct - Dec" quarter, the chart indicates a revenue of 35 thousand. Since the report has overestimated revenues by 15%, we need to calculate the actual revenue by **dividing the reported value by (1 + 0.15)**. This gives us **35 / 1.15 ≈ 30.435**. Therefore, the actual revenue generated by Binders in the quarter October to December is approximately 30,435. <answer>A</answer>

VISE

...The chart shows the revenue for four companies (Moen, Binders, Graff, and Smiths) **over three quarters**. The y-axis represents revenue in thousands of dollars, and the x-axis represents the time period.

First, we need to determine the reported revenue for Binders in the "Oct - Dec" quarter from the chart. The blue line represents Binders. At the "Oct - Dec" point on the x-axis, the reported revenue is 35,000 (thousands).

The report has overestimated revenues by 15%. This means that **the actual revenue is 100% - 15% = 85% of the reported value**.

So, the actual revenue is:
(85/100) * 35,000 = 0.85 * 35,000 = 29,750

Therefore, the actual revenue generated by Binders in the quarter October to December was 29,750.

<CAnswer> C </CAnswer>

Base

...The chart shows the revenue for four companies (Moen, Binders, Graff, and Smiths) **over three quarters**. The y-axis represents revenue in thousands of dollars, and the x-axis represents the time period.

First, we need to determine the reported revenue for Binders in the "Oct - Dec" quarter from the chart. The blue line represents Binders. At the "Oct - Dec" mark on the x-axis, the reported revenue is 35 thousand dollars.

The report has overestimated revenues by 15%. This means that **the actual revenue is 100% - 15% = 85% of the reported value**.

So, the actual revenue is:
(85/100) * 35,000 = 0.85 * 35,000 = 29,750

Therefore, the actual revenue generated by Binders in the quarter October to December was 29,750.

<answer>C</answer>

MMStar — sample 152



Question Which image is the brightest one? Options: A: upper left, B: upper right, C: down left, D: down right

Gold **C**

VQS (ours) C ✓

VISE B ✗

Base B ✗

Why wrong The upper-right panel is fireworks against a night sky and mostly black; the lower-left panel, a clear daytime sky, is bright across the whole frame.

SEED-Bench — sample 2433



Question Which of the following objects are detected in the image? A. Person B. Yellow and blue sculpture C. Ball D. All of the above

Gold **B**

VQS (ours) B ✓

VISE D ✗

Base D ✗

Why wrong There is no person and no ball in the image, so “All of the above” cannot hold; the human figures are illuminated Statue of Liberty sculptures.