CAN VISUAL CHAIN-OF-THOUGHT TRUST ITS VISUAL EVIDENCE? PRE-CONSUMPTION EVIDENCE VERIFICATION FOR INTERLEAVED VLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Interleaved vision-language models revisit images during reasoning, yet typically consume newly selected evidence without checking whether it matches the current visual request. We show through controlled evidence interventions that such mismatches are consequential: replacing erroneous evidence with ground-truth evidence substantially improves answer accuracy, whereas counterfactual evidence degrades it. Moreover, correcting errors after consumption poses a dilemma: appending correct evidence only partially recovers accuracy, while rollback requires discarding already-generated reasoning. These findings motivate a simple principle: *verify visual evidence before consumption*. We introduce VERIFYCOT, which checks whether selected evidence matches the model's current visual request. This request is derived directly from the model's hidden states, eliminating the need to extract a textual reference. Evidence that passes is kept, and evidence that fails is corrected before reasoning continues. Across interleaved VLM architectures and five visual reasoning benchmarks, VERIFYCOT improves end-to-end accuracy in 14 of 15 evaluated settings, with gains of up to 15.7 percentage points. Compared with unconditional grounding, it achieves higher accuracy with fewer grounding calls, while also offering a better accuracy–compute trade-off than the evaluated post-hoc correction methods. We provide code in the supplementary material and will release it on GitHub.

1 INTRODUCTION

Chain-of-thought (CoT) reasoning enables language models to tackle complex problems through intermediate reasoning steps (Wei et al., 2022). Multimodal CoT extends this paradigm to vision-language models (VLMs), incorporating visual and textual information into the reasoning process (Zhang et al., 2023). However, intermediate reasoning does not guarantee that the model has acquired the visual evidence required at each step. In particular, the initial visual representation may fail to capture fine-grained information needed by later reasoning steps. This motivates *interleaved visual CoT*, in which models revisit the image during reasoning to acquire additional visual evidence.

Many recent VLMs follow this pattern, from region-based reasoning to approaches that interleave visual tokens or tool observations with text (Shao et al., 2024; Li et al., 2025; Wang et al., 2026b; Su et al., 2026; Gao et al., 2025; Chen et al., 2026). Despite differences in architecture and evidence format, these methods share a common workflow: the model revisits the image, selects visual information, and incorporates it into subsequent reasoning.

These advances enable more adaptive visual reasoning by allowing models to acquire additional visual evidence as reasoning progresses. During inference, selected evidence is typically consumed immediately, becoming part of the context for subsequent reasoning steps. However, perceptual ambiguity and learned contextual priors may lead models to select the wrong visual evidence (Wu & Xie, 2024; Leng et al., 2024). This raises two central questions: **Should selected visual evidence be trusted before it is incorporated into subsequent reasoning? If the evidence is incorrect, how and when should it be corrected?**

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.

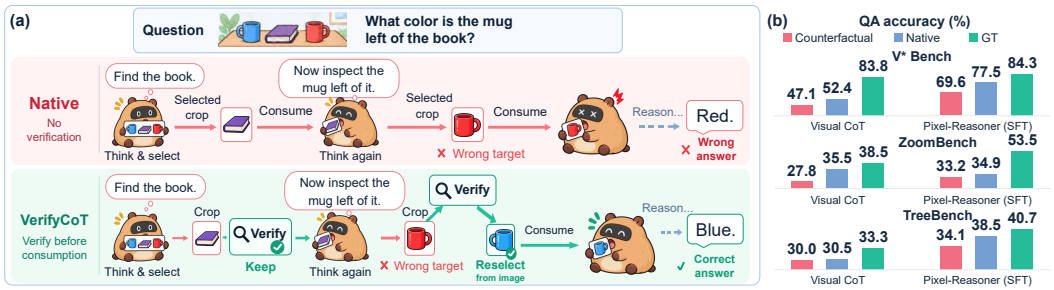


Figure 1: Why verify before consumption? (a) VERIFYCOT preserves correct evidence and corrects errors before use. (b) Correcting visual evidence with ground truth (GT) improves QA accuracy, whereas incorrect evidence lowers it.

To examine these questions, we intervene on intermediate visual evidence while holding the preceding reasoning and current visual request fixed. We compare the model's natural evidence with two controlled alternatives: counterfactual evidence that does not support the current reasoning step, and oracle evidence from the annotated target corresponding to the same visual request. As illustrated in Figure 1, counterfactual evidence reduces final-answer accuracy, whereas oracle evidence improves it relative to the model's natural evidence. This controlled intervention confirms that evidence-selection errors can directly affect downstream reasoning and that correcting them can improve final predictions.

Once erroneous evidence has been consumed, correction faces a dilemma. Appending correct evidence cannot undo the influence of the error on the reasoning already generated, and therefore only partially recovers accuracy—even when the appended evidence is ground truth. Rolling back to the erroneous step removes this influence, but this wastes computation already spent on the discarded continuation. We therefore introduce VERIFYCOT, a framework built on the principle of *verify before consumption*: selected evidence is checked against the model's current visual request before it enters the reasoning context; evidence that passes is retained, and evidence that fails is selectively corrected before reasoning proceeds.

To verify or correct the evidence, we first need to know what visual information the model is seeking at that step. Textual references can describe this target, but their wording varies across models and fine-tuning settings. Extracting them can require model-specific rules, and even cleaned references may omit relevant reasoning or visual context. We therefore derive verification and grounding requests from the model's hidden states before it selects visual evidence. Separate request interfaces are learned for the two tasks, without requiring a textual reference at inference.

When natural visual evidence passes verification, the model keeps it without calling a grounding specialist. Otherwise, the specialist proposes alternative evidence for further verification, while the natural evidence remains a candidate for the final selection. This design preserves reliable evidence and avoids unnecessary grounding calls. If a replacement is selected, the model continues reasoning with the corrected evidence. The framework applies to interleaved VLMs in which visual evidence can be intercepted and verified before consumption, regardless of how that evidence is represented. Our contributions are threefold:

- We identify and characterize evidence-selection errors in interleaved visual reasoning: model-selected evidence can mismatch the current visual request and, once consumed, affect final predictions. Controlled interventions further show that post-consumption correction faces a dilemma between incomplete recovery and discarded reasoning.
- We introduce VERIFYCOT, a pre-consumption evidence verification framework that derives verification and grounding requests from the model's pre-selection hidden states. It uses a lightweight verifier to retain reliable evidence and calls a grounding specialist only when verification fails, while keeping the base VLM frozen.
- Across multiple interleaved VLM architectures, training regimes (SFT and RL), and visual reasoning benchmarks, VERIFYCOT improves end-to-end accuracy in 14 of 15 evaluated settings. Evidence-level audits further show that it corrects invalid evidence while preserv-

ing valid support, and selective correction achieves a better accuracy–compute trade-off than unconditional grounding and the evaluated post-hoc correction baselines.

2 Related Work

Acquiring visual evidence. Interleaved VLMs acquire and use visual evidence in different ways. Visual CoT and VoCoT use localized regions, ICoT and MINT-CoT interleave visual tokens with text, and VGR replays region-linked visual memory. Pixel-Reasoner and DeepEyes (Zheng et al., 2026) return new observations from visual operations. We focus on a complementary question: whether the evidence selected at a reasoning step should be retained or corrected before it is used.

Verifying and correcting visual reasoning. Huang et al. (2024) show that self-correction without external feedback fails to improve reasoning accuracy. CRITIC uses external tools to obtain feedback for verifying (Gou et al., 2024). LookBack checks visual information in generated reasoning (Wu et al., 2025). ReGround revisits the image after an initial reasoning round (Peng et al., 2026), while Trace, Verify, and Correct checks spatial claims and guides reasoning revision (Yang et al., 2026b). Verifier-Guided Decoding instead uses internal attention signals to selectively roll back unsupported object mentions during generation (Yang et al., 2026a). Our intervention concerns the selected visual evidence itself, before it enters subsequent reasoning. VerifyCoT checks whether this evidence matches the current request, retains evidence that passes verification, and requests alternatives only when needed.

Hidden states for grounding and verification. LISA connects language-model hidden representations to a segmentation decoder (Lai et al., 2024). VGent uses MLLM hidden states to select detector proposals (Kang et al., 2026). V-Reflection learns state-conditioned visual probes (Zhou et al., 2026). We use hidden states before evidence selection to guide verification and selective correction without extracting textual references at inference. This reduces reliance on model-specific wording and mitigates the information bottleneck of textual references, which may omit context needed to identify the current target.

3 Method

As illustrated in Figure 2, VerifyCoT introduces verification and selective correction between evidence selection and consumption. It checks whether the selected visual evidence matches the model's current request and completes any correction before reasoning continues with it.

3.1 Decoupling evidence selection from consumption

Consider an interleaved VLM M given an image I and a question Q. At step t, C_t contains the question, preceding reasoning, and visual evidence already available to the model. We denote the model's evidence-selection action by A_t, which may take the form of a visual tool call, special tokens, or direct region prediction. The selected natural visual evidence e_t^{nat} may be represented as bounding-box coordinates, cropped images, visual features, or textual descriptions of visual observations (Zhang et al., 2026).

The model obtains this evidence through its original procedure. It then updates its reasoning context: $C_t^+ = \mathcal{U}_M(C_t, I, e_t^{\mathrm{nat}})$, where $\mathcal{U}_M$ is the model's context-update operation. It adds the selected visual evidence to the context through the model's original inference process, allowing the model to continue reasoning with the updated context C_t^+. We refer to using this evidence in subsequent reasoning as *evidence consumption*.

We decouple evidence selection from consumption by verifying and selectively correcting the natural visual evidence before adding it to the reasoning context. We verify whether the selected evidence satisfies the current visual request, rather than whether the request itself is appropriate.

We record H_t from the final decoder layer's hidden states at language-token positions before evidence selection A_t. When the model expresses A_t as a token sequence, we use the last L hidden states before its first token: $H_t = [h_{j_t-L}, \ldots, h_{j_t-1}]$, where j_t is the position of that token. The

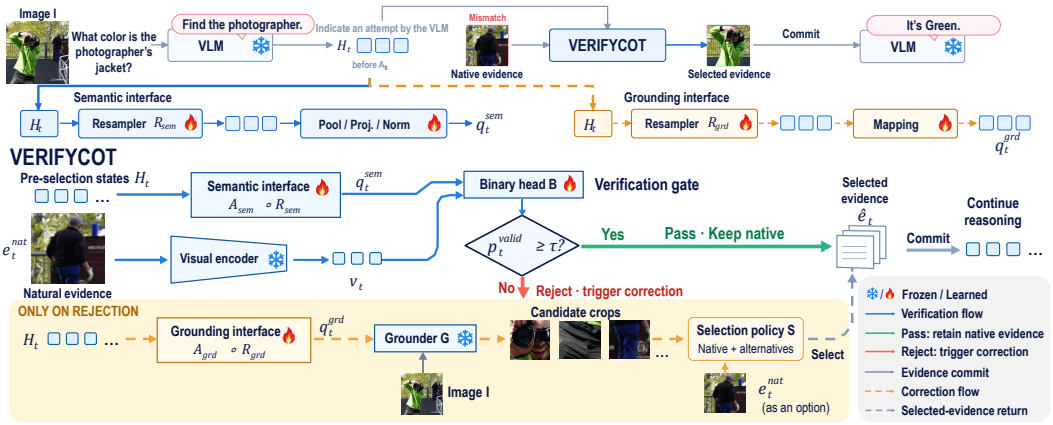


Figure 2: Overview of VerifyCoT. Pre-selection hidden states guide verification and selective grounding. Evidence passing verification is retained; otherwise, the grounder proposes alternatives. The highest-scoring proposal replaces the original evidence only if it passes re-verification. Reasoning then resumes with the selected evidence.

model then completes its selection and obtains e_t^{nat}. The verifier receives this evidence as a separate input, while H_t reflects the context before the current selection.

Let Π denote our verification and selective correction policy. Rather than using natural visual evidence directly, the model first applies Π and then adds the final selection $\hat{e}_t$ to its reasoning context:

$$\hat{e}_t = \Pi(H_t, I, e_t^{\text{nat}}), \qquad C_t^+ = \mathcal{U}_M(C_t, I, \hat{e}_t). \tag{1}$$

The model then continues reasoning with the checked visual evidence $\hat{e}_t$. This reduces the risk of incorrect evidence misleading subsequent reasoning. Evidence that passes verification is retained without unnecessary replacement.

3.2 Constructing Requests from Hidden States

Both evidence verification and correction require a representation of the model's current visual request. Verification uses this request to assess a candidate, while correction uses it to seek alternatives. The request can include target semantics and contextual cues that distinguish the evidence needed for the current step.

We construct the verification and grounding requests from H_t, recorded before evidence selection A_t. Inspired by the learned-query designs of Perceiver and BLIP-2's Q-Former (Jaegle et al., 2021; Li et al., 2023), we use two task-specific interfaces:

$$q_t^r = \mathcal{A}_r(\mathcal{R}_r(H_t)), \qquad r \in \{\text{sem}, \text{grd}\}. \tag{2}$$

Here $\mathcal{R}_r$ uses learned queries to extract information from H_t, and $\mathcal{A}_r$ converts the resulting tokens into the representation required by the corresponding external model. Together, $\mathcal{R}_r$ and $\mathcal{A}_r$ form a *request interface* that translates the VLM's hidden states into a request the external model can use. We use annotated target regions to train these interfaces for evidence matching and localization.

The semantic request interface produces q_t^{sem} for matching candidate visual evidence encoded by a separate, frozen semantic encoder. The grounding request interface produces q_t^{grd} as a token sequence that the frozen grounding model can accept. The two interfaces use the same hidden states H_t and resampling design. For each base VLM, we train its own pair of interfaces using this design. Architectural details are deferred to Appendix C.4.

3.3 Evidence Verification and Selective Correction

Evidence verification. Using the semantic request q_t^{sem}, we check whether the natural visual evidence matches the current visual request:

$$p_t^{\text{valid}} = \mathcal{B}\big(q_t^{\text{sem}}, \Phi(I, e_t^{\text{nat}})\big). \tag{3}$$

Here Φ encodes the candidate visual evidence using a semantic encoder separate from the base VLM, and $\mathcal{B}$ is a learned binary verification head. This separate encoding provides a complementary visual representation, aiming to reduce the risk of reinforcing the base model's own selection errors. The verifier checks the natural visual evidence directly, without first generating alternatives. We train the verifier using SFT data to distinguish visual evidence that matches the current request from evidence that does not.

Let τ be the verification threshold. If $p_t^{\text{valid}} \geq \tau$, we retain the natural visual evidence and continue reasoning without invoking the grounding model. Otherwise, we proceed to the candidate generation and selection steps below.

Selective correction. If the natural visual evidence fails verification, the grounding model uses q_t^{grd} to propose K alternatives, $\{e_{t,k}\}_{k=1}^{K} = \mathcal{G}(I, q_t^{\text{grd}})$. Because grounding proposals can also be incorrect, we retain the natural visual evidence as a fallback: $\mathcal{C}_t = \{e_t^{\text{nat}}\} \cup \{e_{t,k}\}_{k=1}^{K}$.

A selection policy $\mathcal{S}$ determines whether to retain the natural evidence or adopt a grounding proposal. It can use grounding scores, candidate verification, or a learned selector. Our implementation is described in Section 3.4. Together, verification and selection implement the policy Π:

$$\Pi(H_t, I, e_t^{\text{nat}}) = \begin{cases} e_t^{\text{nat}}, & p_t^{\text{valid}} \geq \tau, \\ \mathcal{S}(H_t, I, \mathcal{C}_t), & p_t^{\text{valid}} < \tau, \end{cases} \tag{4}$$

The model then continues reasoning with the selected visual evidence $\hat{e}_t$. Runtime details are deferred to Appendix C.5.

3.4 Training and Selection

Supervision and pretraining. In our implementation, we pair the hidden states H_t with the annotated target region for each visual-evidence step (i.e., a step where the model requests visual evidence for subsequent reasoning). Supervision concerns the evidence requested at that step, not the final answer. For each request interface, we encode the textual reference for each SFT step with the corresponding external model's frozen text encoder. We pretrain the interface to match these embeddings. These references are used only during training; at inference, both interfaces construct requests directly from H_t.

Learning for verification and correction. After pretraining, a contrastive retrieval loss trains the semantic request interface to rank target regions above negative regions, while binary cross-entropy trains the interface and verification head to distinguish valid from invalid pairs. The grounding request interface is trained separately with the grounder's standard bounding-box loss. Only the two request interfaces and the binary head are updated; the base VLM, semantic encoder, and grounding backbone remain frozen.

Selection policy. For $\mathcal{S}$, we rank alternative proposals by the grounder's native scores and recheck the highest-ranked proposal with the same verifier and threshold τ used for the natural visual evidence. We replace the natural visual evidence only if this proposal passes verification. If the proposal fails verification, or no proposal is available, we retain the natural visual evidence. Retaining the natural visual evidence in this case does not mean that it has passed verification.

4 Experiments

We evaluate whether VerifyCoT improves answer accuracy and corrects invalid visual evidence while preserving valid support. We also compare correction policies and post-hoc answer revision, and examine how requests derived from hidden states support verification and grounding.

4.1 Experimental Setup

Models and benchmarks We evaluate VerifyCoT on Pixel-Reasoner (SFT and RL) and Visual CoT, preserving their original inference procedures. V* Bench, HR-Bench (4K and 8K) (Wang et al., 2025), TreeBench (Wang et al., 2026a), and ZoomBench (Wei et al., 2026) span high-resolution perception, fine-grained recognition, and visual reasoning.

Table 1: End-to-end accuracy (%). The full table and published-result sources are in Appendix B.3. VERIFYCOT gains compare adjacent pairs using the same base checkpoint.

Model	Size	V* Bench	HR4K	HR8K	TreeBench	ZoomBench
GPT-4o	–	62.8	59.0	55.5	46.9	–
Gemini-2.5-Pro	–	79.2	87.3	85.4	54.1	–
Qwen2.5-VL	7B	70.4	71.6	67.9	37.0	42.5
Visual Sketchpad (GPT-4o)	–	80.4	–	–	–	–
IVM-Enhance (GPT-4V)	–	81.2	–	–	–	–
SEAL	7B	74.8	38.1	35.6	–	–
Pixel-Reasoner (SFT)	7B	77.5	68.4	58.5	38.5	34.9
+ VERIFYCOT	7B	**83.2**	**70.9**	**63.1**	**40.5**	**48.8**
Pixel-Reasoner (RL)	7B	84.3	72.6	64.5	38.3	47.8
+ VERIFYCOT	7B	**85.9**	**74.8**	**67.4**	**43.7**	**48.0**
Visual CoT	7B	52.4	47.1	36.4	30.4	35.5
+ VERIFYCOT	7B	**68.1**	**52.8**	**47.8**	29.9	**36.8**

Training and implementation. For each VLM, we train the request interfaces and binary verifier on its own SFT data, using image-disjoint training, development, and test sets. Training negatives include regions that miss the target or provide insufficient evidence for the current request. We use SigLIP2 (Tschannen et al., 2025) for semantic encoding and Grounding DINO (Liu et al., 2024) for grounding. Each verification threshold τ is determined on SFT development data before test-set evaluation and kept fixed throughout testing. On Pixel-Reasoner, VERIFYCOT verifies every visual-evidence step. Our main evaluation setup follows Pixel-Reasoner; detailed training and evaluation settings are provided in Appendix B.1.

4.2 MAIN END-TO-END RESULTS

Table 1 shows that VERIFYCOT improves final-answer accuracy across model architectures and training regimes, including Visual CoT and the SFT and RL versions of Pixel-Reasoner.

The improvements span several visual reasoning benchmarks. On V* Bench, VERIFYCOT raises Pixel-Reasoner (SFT) accuracy from 77.5% to 83.2%, a gain of 5.8 percentage points over an already capable base model. It also improves HR4K by 2.5 points and HR8K by 4.6 points. This SFT model gains 2.0 points on TreeBench and 13.9 points on ZoomBench. Overall, VERIFYCOT improves Pixel-Reasoner (SFT) on all five benchmarks, with the largest gain on ZoomBench.

For Pixel-Reasoner (RL), VERIFYCOT also improves on the five evaluated benchmarks, raising TreeBench accuracy from 38.3% to 43.7% and HR8K from 64.5% to 67.4%. The gains are smaller on several RL benchmarks. One possible explanation is that correcting visual evidence does not always resolve errors in the model's subsequent reasoning. Nevertheless, these gains highlight the importance of verifying visual evidence even in stronger, RL-trained models.

The gains also extend beyond Pixel-Reasoner's architecture. With Visual CoT, VERIFYCOT improves four of the five evaluated benchmarks, raising V* Bench accuracy from 52.4% to 68.1% and ZoomBench from 35.5% to 36.8%. TreeBench instead decreases slightly, from 30.4% to 29.9%. These results show that VERIFYCOT improves performance across models with different mechanisms for acquiring and using visual evidence.

4.3 DOES VERIFICATION IMPROVE THE ACCURACY–COMPUTE TRADE-OFF?

To disentangle the contribution of verification from that of the grounding specialist, we compare Native, Always ground, Random gate, NLL gate, and VERIFYCOT on V* Bench with Visual CoT and Pixel-Reasoner. Within each backbone, the correction variants use the same settings. Always ground calls the grounder at every evidence step. NLL gate uses the mean negative log-probability of the generated coordinate tokens as a cheap confidence signal. Random and NLL gates match

Table 2: Grounder-calling policies on V* Bench. Within each backbone, the correction variants use the same settings and selection policy $\mathcal{S}$.

(a) Visual CoT

Variant	QA	Fix	Harm	Calls	TFLOPs	Time
Native	52.4	0.0	0.0	0.00	28.95	1.07
Always	64.9	19.9	7.3	1.00	31.27	1.87
Random	55.5	7.9	4.7	0.46	30.12	1.24
NLL gate	62.3	12.0	2.1	0.46	30.10	1.20
VERIFYCOT	68.1	17.3	1.6	0.46	30.12	1.24

(b) Pixel-Reasoner

Variant	QA	Fix	Harm	Calls	TFLOPs	Time
Native	77.5	0.0	0.0	0.00	212.7	4.93
Always	80.6	11.5	8.4	0.94	230.0	8.55
Random	79.1	7.3	5.8	0.37	214.1	5.30
NLL gate	80.6	7.9	4.7	0.37	214.7	5.40
VERIFYCOT	83.2	10.5	4.7	0.37	215.3	5.40

QA, Fix, and Harm are percentages. QA: answer accuracy. Fix/Harm: wrong-to-right/right-to-wrong changes relative to Native, divided by all 191 questions. Calls: expert requests per question.

Table 3: Evidence interventions on 100 manually audited Visual CoT/V* examples. Groups are defined by native evidence validity: 33 valid and 67 invalid. Invalid and GT evidence are forcibly injected; VERIFYCOT selects the evidence to consume.

(a) Answer quality (%)

Condition	All	Valid	Invalid	Fix	Harm
Native	49.0	90.9	28.4	—	—
Invalid	45.0	69.7	32.8	11.0	15.0
GT	83.0	93.9	77.6	38.0	4.0
VERIFYCOT	65.0	90.9	52.2	17.0	1.0

(b) Changes after VERIFYCOT

Evidence Native → final	n	Fix	Harm	QA (%) Before → after
Valid → Valid	33	0	0	90.9 → 90.9
Invalid → Valid	**31**	**17**	**1**	**16.1 → 67.7**
Invalid → Invalid	36	0	0	38.9 → 38.9

Fix/Harm are percentages of all 100 questions in (a) and counts in (b). Group QA is computed within each column in (a) and each row in (b).

VERIFYCOT's calling budget: 0.46 expert requests per question for Visual CoT and 0.37 for Pixel-Reasoner. Random gate results are averaged over five runs.

Always ground achieves a higher Fix rate than VERIFYCOT (19.9% vs. 17.3%), but also causes more Harm (7.3% vs. 1.6%). By avoiding more harmful corrections, VERIFYCOT achieves higher QA accuracy (68.1% vs. 64.9%) with fewer grounder calls. Compared with budget-matched Random gate, VERIFYCOT achieves a higher Fix rate (17.3% vs. 7.9%) and lower Harm (1.6% vs. 4.7%). At matched calling budgets, VERIFYCOT also achieves higher QA accuracy than NLL gate on both backbones. These comparisons support the verifier's ability to guide useful corrections beyond random selection or the model's coordinate confidence alone.

On both backbones, VERIFYCOT improves QA accuracy by 5.8–15.7 percentage points over Native, with at most 4.0% more estimated compute and 15.9% more runtime.

4.4 NATURAL EVIDENCE ERRORS AND THE BENEFIT OF CORRECTION

Settings. We evaluate Visual CoT on 100 randomly sampled V* questions with assessable evidence, manually labeling each selected region as valid or invalid for the current reasoning step. Its single evidence step allows a direct comparison of natural evidence, deliberately incorrect evidence, annotated GT evidence, and VERIFYCOT. We keep the preceding context and generation settings fixed. GT evidence supports the current step's target, while incorrect crops are manually checked to ensure that they omit the required evidence. Crop construction details are provided in the appendix.

Visual evidence affects answer accuracy. Replacing invalid native evidence with GT increases QA accuracy from 28.4% to 77.6% on the 67 initially invalid examples (Table 3). Conversely, replacing valid native evidence with invalid crops lowers accuracy from 90.9% to 69.7% on the 33 initially valid examples. These interventions show that changing visual evidence can affect final answers, and that correcting evidence errors can improve answer accuracy without changing the model's current target. **Correction improves evidence and answer accuracy.** VERIFYCOT corrects native

Table 4: Correction strategies and timing on Pixel-Reasoner (SFT). (a) Methods on V* and HR8K; TF is estimated TFLOPs per question. (b) TreeBench accuracy (%) under different correction timings; the last three rows append GT after a wrong crop.

(a) Correction methods

	V*			HR8K		
Method	QA	Fix/Harm	TF	QA	Fix/Harm	TF
Native	77.5	—	213	58.5	—	303
Self-corr.	74.9	2.1/4.7	309	60.3	4.1/2.4	361
Zero-shot crop	76.4	6.8/7.9	291	59.8	5.0/3.8	367
LookBack	67.5	5.8/15.7	1639	58.3	12.9/13.1	2558
VERIFYCOT	**83.2**	10.5/4.7	215	**63.1**	6.9/2.3	310

(b) Correction timing

Setting	All	With evidence
Native	38.5	42.6
Wrong crop	34.1	37.3
GT before use	**40.7**	**45.2**
Immediately	37.5	41.4
After first response	38.0	42.0
After full continuation	38.0	42.0

evidence errors while maintaining valid visual support. In this audit, it repairs 31 of the 67 invalid selections, while all 33 initially valid cases still receive valid evidence. These repairs are accompanied by answer-level gains (Table 3(b)): among the 31 repaired cases, 17 wrong answers become correct and one correct answer becomes wrong, yielding a net gain of 16 correct answers. These results support directly correcting intermediate visual evidence as a way to improve subsequent answers while preserving valid visual support.

4.5 CORRECTION STRATEGIES AND TIMING

Once erroneous evidence has been consumed, correction must either revise the existing reasoning trajectory or roll back to the error and regenerate the subsequent reasoning. We examine both on Pixel-Reasoner (SFT). Table 4(a) compares VERIFYCOT with three baselines using the same frozen model: post-hoc self-correction reviews and revises the answer in one additional call; LookBack reexamines the image to critique the visual facts in the reasoning and revises the answer accordingly; zero-shot crop asks the model to verify its first crop and, if rejected, propose one replacement before continuing. The first two operate on the completed trajectory without regenerating it (settings in Appendix C.2). All three baselines reduce QA accuracy on V* to different degrees. Post-hoc self-correction improves HR8K to 60.3%, still below VERIFYCOT's 63.1%. LookBack reviews the full reasoning trajectory with substantially more computation and fixes more errors than VERIFYCOT on HR8K. However, it also turns more correct answers into wrong ones, outweighing those fixes. In contrast, VERIFYCOT achieves higher final accuracy with fewer harmful revisions and lower computation.

Post-hoc appending only partially recovers accuracy. To assess how much accuracy can be recovered without rewriting the affected reasoning, we inject oracle GT evidence on TreeBench (405 questions, 343 with an evidence step; Table 4(b)). The correction instructions explicitly tell the model that the previous evidence is invalid and ask it to locate the relevant visual evidence again. Supplying GT at the evidence step reaches 40.7%, whereas appending GT with explicit correction instructions after a wrong crop, whether immediately, after the first response, or after the full continuation, reaches only 37.5–38.0%, recovering about half of the gap from the wrong crop alone (34.1%). Once consumed, the wrong crop has already affected the subsequent reasoning state, and appending correct evidence cannot undo this influence.

Rollback pays for discarded reasoning. Rolling back avoids this residual influence of the erroneous evidence, but it requires locating the erroneous step, which the verifier provides, and the discarded continuation has already been generated. In a matched-decision comparison on HR8K (Appendix C.3), pre-consumption correction and post-hoc rollback produce identical answers, yet pre-consumption reduces end-to-end latency by 8.7% over all 800 examples and by 20.9%, with 30.3% fewer output tokens, on the 287 examples where evidence is replaced. Verifying before consumption therefore avoids both the accuracy loss of appending and the cost of rolling back.

Table 5: Text and latent request comparisons on V* Bench (%). (a–b) Verification and grounding on Visual CoT and VoCoT. (c) End-to-end QA on Visual CoT. CA: correct evidence accepted; ED: incorrect evidence rejected. Panel (a) uses thresholds calibrated to 85% correct acceptance on SFT development data and fixed at test time.

(a) Verification

	Visual CoT		VoCoT	
Request	CA	ED	CA	ED
Raw	68.1	32.8	63.7	49.7
Clean	**91.5**	60.7	73.2	52.9
Latent	83.0	**61.5**	**81.3**	**54.8**

(b) Grounding

	Visual CoT		VoCoT	
Request	R@1	R@5	R@1	R@5
Raw	60.6	88.7	46.0	76.6
Clean	66.8	**89.4**	62.8	82.7
Latent	**69.8**	88.2	**69.9**	**87.2**

(c) End-to-end QA

Verifier	Grounder	QA
Raw	Raw	58.1
Latent	Raw	64.9
Raw	Latent	60.2
Latent	Latent	**68.1**
Clean	Clean	63.4

4.6 Comparing Latent and Text Requests

Settings. We compare raw, cleaned, and latent requests on V* Bench using Visual CoT, and additionally evaluate verification and grounding on VoCoT. Raw references use the text preceding the evidence action. For cleaned references, Qwen2.5-VL (Bai et al., 2025) rewrites this text using the question and preceding CoT. Verification is evaluated on native evidence; grounding recall uses the grounder's native ranking on the same steps, independently of verification, with IoU > 0.3 defining a target match. For end-to-end evaluation on Visual CoT, we vary the verification and grounding requests independently while keeping other settings fixed.

Verification and grounding. On Visual CoT, latent requests improve both correct acceptance and error detection over raw references. Compared with cleaned references, latent requests detect more errors with slightly lower correct acceptance, while avoiding the overhead of an additional VLM call for text rewriting (Table 5(a)). For grounding, latent requests achieve the highest R@1 (69.8%), with similar R@5 across request types (Table 5(b)).

Visual CoT mostly uses explicit target descriptions in the examined trajectories. We therefore also test VoCoT, which includes more implicit and context-dependent requests (e.g., "I'll find the object below it"). Latent requests outperform both text baselines in verification and grounding: compared with cleaned references, CA rises from 73.2% to 81.3% and ED from 52.9% to 54.8%; R@1 increases from 62.8% to 69.9% and R@5 from 82.7% to 87.2%. This broader advantage on VoCoT supports the value of latent requests when identifying the current target requires information beyond its explicit wording.

From component accuracy to final answers. Table 5(c) shows that using latent requests only for verification raises QA from 58.1% to 64.9%, while using them only for grounding gives 60.2%. Combining both increases QA to 68.1%. The cleaned-reference baseline reaches only 63.4%. We also observed cases where the rewriting VLM answered the original question instead of extracting the target of the current step. VLM-based rewriting therefore does not guarantee a correct reference for the current request. Both latent interfaces improve final-answer accuracy, with the largest gain from combining them.

5 Conclusion

We introduced VerifyCoT, a framework that separates visual evidence selection from consumption in interleaved VLMs. It uses the model's hidden states to check whether selected evidence matches the current request and selectively corrects it before further reasoning. Experiments across two VLM architectures show improved answer accuracy, while evidence-level analysis shows correction of invalid evidence and preservation of valid support. Selective correction reduces expert calls compared with always grounding. It also achieves higher answer accuracy with lower estimated computation than the evaluated post-hoc correction methods. These findings support verification and selective correction as key steps between acquiring visual evidence and using it for reasoning.

AI USE STATEMENT

In this work, we used generative AI tools to refine the language and readability of the manuscript and to assist with literature discovery, including suggesting potentially relevant studies and search directions. These suggestions were used solely to guide literature searches. The authors independently consulted the original sources of all cited works to verify their bibliographic details and relevance to the associated claims. We did not rely on generative AI to establish the existence, content, or validity of any cited work. We reviewed and revised all AI-assisted text and take full responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we describe the VerifyCoT framework and training procedure in Section 3 and the experimental setup in Section 4.1. Appendix A reports the experimental hardware. Appendix B details data partitioning, negative-region construction, threshold calibration, evaluation protocols, manual evidence audits, and the sources of published baseline results. Appendix C provides the compute-accounting assumptions, correction baseline settings, timing protocols, interface architectures, training configurations, random seeds, and runtime state-handling procedures. The prompt used for clean-reference extraction is provided in Appendix D.3, and statistical uncertainty estimation is described in Appendix D.5. We provide the complete implementation of the VerifyCoT architecture, together with detailed training configurations and evaluation protocols, enabling researchers to reconstruct the system and reproduce our experimental setup.

REFERENCES

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.

Yicheng Bao, Xuhong Wang, Qiaosheng Zhang, Chaochao Lu, Xia Hu, and Xin Tan. Dynamic adversarial reinforcement learning for robust multimodal large language models. *arXiv preprint arXiv:2602.22227*, 2026.

Xinyan Chen, Renrui Zhang, Dongzhi Jiang, Aojun Zhou, Shilin Yan, Weifeng Lin, and Hongsheng Li. Mint-cot: Enabling interleaved visual tokens in mathematical chain-of-thought reasoning. *Advances in neural information processing systems*, 38:69110–69139, 2026.

Jun Gao, Yongqi Li, Ziqiang Cao, and Wenjie Li. Interleaved-modal chain-of-thought. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19520–19529. IEEE, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Weitai Kang, Jason Kuen, Mengwei Ren, Zijun Wei, Yan Yan, and Kangning Liu. Vgent: Visual grounding via modular design for disentangling reasoning and prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41160–41170, 2026.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9579–9589. IEEE, 2024.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. Mitigating object hallucinations in large vision-language models through visual contrastive decoding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13872–13882. IEEE, 2024.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023.

Zejun Li, Ruipu Luo, Jiwen Zhang, Minghui Qiu, Xuan-Jing Huang, and Zhongyu Wei. Vocot: Unleashing visually grounded multi-step reasoning in large multi-modal models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3769–3798, 2025.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024.

Lei Peng, Shuai Lv, and Wei Hu. Reground: Restoring visual grounding in multi-step reasoning through self-diagnosis and visual re-examination. *arXiv preprint arXiv:2608.04385*, 2026.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 37:8612–8642, 2024.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. *Advances in Neural Information Processing Systems*, 38:8222–8251, 2026.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Haochen Wang, Xiangtai Li, Zilong Huang, Anran Wang, Jiacong Wang, Tao Zhang, Sule Bai, Zijian Kang, Jiashi Feng, Zhuochen Wang, et al. Traceable evidence enhanced visual grounded reasoning: Evaluation and method. In *International Conference on Learning Representations*, volume 2026, pp. 148769–148794, 2026a.

Jiacong Wang, Zijian Kang, Haochen Wang, Xiao Liang, Ya Wang, Jiawen Li, Bohong Wu, Jiao Ran, Haiyong Jiang, Chao Feng, et al. Vgr: Visual grounded reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 89506–89528, 2026b.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7907–7915, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Lai Wei, Liangbo He, Jun Lan, Lingzhong Dong, Yutong Cai, Siyuan Li, Huijia Zhu, Weiqiang Wang, Linghe Kong, Yue Wang, et al. Zooming without zooming: Region-to-image distillation for fine-grained multimodal perception. *arXiv preprint arXiv:2602.11858*, 2026.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13084–13094. IEEE, 2024.

Xueqing Wu, Yuheng Ding, Bingxuan Li, Pan Lu, Da Yin, Kai-Wei Chang, and Nanyun Peng. Visco: Benchmarking fine-grained critique and correction towards self-improvement in visual reasoning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9527–9537. IEEE, 2025.

Lei Yang, Xinze Liu, Dayan Wu, Ding Wang, Hengjie Zhu, Zihao Zhang, Tianzhu Hu, Hanqi Wu, Peng Fu, and Zheng Lin. Hallucinations leave a grounding signature: Verifier-guided decoding for selective object correction. *arXiv preprint arXiv:2607.27823*, 2026a.

Yang Yang, Jiawei Chen, Tairan Chen, and Zhaoxia Yin. Trace, verify, and correct: A training-free framework for spatial reasoning in multimodal llms. *arXiv preprint arXiv:2608.04759*, 2026b.

Yongchang Zhang, Oliver Ma, Tianyi Liu, Guangquan Zhou, and Yang Chen. See it, say it, sorted: An iterative training-free framework for visually-grounded multimodal reasoning in lvlms. *arXiv preprint arXiv:2602.21497*, 2026.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, and Chao Shen. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 126775–126798, 2026.

Jiazhou Zhou, Yucheng Chen, Hongyang Li, Qing Jiang, Hu Zhou, Ying-Cong Chen, and Lei Zhang. V-reflection: Transforming mllms from passive observers to active interrogators. *arXiv preprint arXiv:2604.03307*, 2026.

SUPPLEMENTARY MATERIAL

A EXPERIMENTAL HARDWARE

Our experimental server uses NVIDIA GeForce RTX 4090 GPUs with 24 GB of memory each, two Intel Xeon Platinum 8336C processors (64 physical cores and 128 logical threads in total), and 512 GiB of host memory, running Ubuntu 22.04 LTS.

B ADDITIONAL DATA AND EVALUATION PROTOCOL

B.1 TRAINING AND EVALUATION SETTINGS

SFT training examples and calibration. For each base VLM, its own SFT dataset supplies the reasoning steps and annotated target regions used to train the verifier and grounding interfaces. We split these SFT data by image into 80% training, 10% development, and 10% test partitions, keeping all steps from the same image in the same partition.

Negative regions. Each SFT verifier step has one positive crop and four same-image negatives. Across the training data, these negatives cover three types: displaced regions that omit the required target; regions containing only a subset of the targets required by a multi-target request; and regions that omit visual information needed to complete the current step. Negative regions may overlap the annotated target region. Partial coverage alone does not make a crop invalid: it must omit information required by the current request. Both contrastive retrieval training and binary cross-entropy training use these three types of negatives.

Unless otherwise specified, one verification threshold τ per VLM is chosen on constructed SFT development pairs to accept approximately 85% of annotated positives. It is determined before test-set evaluation and kept fixed for both natural and replacement evidence throughout testing. This target measures acceptance of correct development pairs, not overall classification accuracy or retention on naturally generated evidence.

External models and candidates. We use SigLIP2 for semantic encoding and Grounding DINO for grounding. Their respective frozen text encoders provide teacher embeddings for the corresponding request interfaces. The grounder returns up to five candidates per call. The interface architectures and training stages are described in Appendix C.4.

Evaluation protocol. Our main evaluation setup follows Pixel-Reasoner, while retaining each base model's native inference procedure. For each checkpoint, the original model and its VERIFYCOT variant use the same inputs, prompts, generation settings, and scoring procedure. We use official test splits and report answer accuracy for the selected benchmarks. Main results include every test question, including those where no correction is triggered.

Efficiency reporting. Estimated inference FLOPs per question include the base VLM, request interfaces, verification, and grounding. Model-specific accounting assumptions are given in Appendix C.1. Latency is measured separately rather than inferred from FLOPs. Complete-task latency includes verification, expert calls, and subsequent reasoning; continuation-only measurements are labeled separately.

B.2 MANUAL AUDIT OF VISUAL EVIDENCE

Evidence validity. For the single-step Visual CoT/V* study, we manually inspected each crop alongside the original image, question, and GT target annotations to judge whether it provided the visual information required by the current step. For single-target questions, the crop had to include the required target. For multi-target questions in this audit, it had to include both required targets; showing only one was insufficient. We applied the same criteria to native and final selected evidence. Labels describe the visual support provided by the crop, not whether the model's final answer was correct. Two annotators independently assessed each crop and resolved disagreements through discussion.

Sampling and uncertain cases. We randomly sampled V* questions and excluded cases whose evidence validity could not be reliably judged. The retained audit set contains 100 questions: 33 with valid native evidence and 67 with invalid native evidence. These exclusions apply only to the audit; the main end-to-end evaluation uses the full test set. The audit is separate from SFT training and threshold calibration.

Crop construction and intervention. GT crops match the current evidence step's target. Incorrect crops have the same dimensions as GT and an IoU below 0.1 with it. We manually confirm that each incorrect crop omits required evidence, such as showing only one of two objects needed for a spatial comparison. Incorrect and GT crops are directly substituted at the same pre-consumption state, with generation settings fixed; VERIFYCOT instead applies its verification and selection policy.

B.3 SOURCES OF PUBLISHED BASELINE RESULTS

The published rows of Tables 1 and 6 are literature reference values, not our reproductions under a common evaluation setting. Different columns can use different model snapshots or inference settings. The V* Bench values follow the Pixel-Reasoner paper.

HR-Bench. GPT-4o's HR4K/HR8K scores, $59.0/55.5$, and SEAL-7B's $38.1/35.6$ come from the original HR-Bench paper's Table 10. That source does not detail SEAL's search configuration; equivalence with its V* configuration reported by Pixel-Reasoner is not established. Gemini-2.5-Pro's $87.3/85.4$ come from the thinking configuration in Table 2 of the Qwen3-VL technical report (Bai et al., 2025), not its separate budget-128 configuration. Qwen2.5-VL-7B's $71.62/67.88$ follow Table 2 of the ZoomBench paper. Gemma3's HR4K score, 57.88, is the unmodified Gemma-3-IT-27B baseline in Table 4 of Bao et al. (2026), not their adversarially trained variant.

TreeBench and ZoomBench. TreeBench scores follow the Overall QA column of its original Table 2: GPT-4o-1120 (46.9), Gemini-2.5-Pro-0605 (54.1), and Qwen2.5-VL-7B (37.0). These are not grounding mIoU or GT-region-assisted results. Qwen2.5-VL-7B's ZoomBench score, 42.49,

Table 6: Full end-to-end accuracy (%). Published references retain their source-specific settings described above; local VERIFYCOT comparisons use matched base checkpoints. Size denotes the base model size. "–" denotes unavailable or unevaluated results.

Model	Size	V* Bench	HR4K	HR8K	TreeBench	ZoomBench
GPT-4o	–	62.8	59.0	55.5	46.9	–
Gemini-2.5-Pro	–	79.2	87.3	85.4	54.1	–
Gemini-2.0-Flash	–	73.2	–	–	–	–
Qwen2.5-VL	7B	70.4	71.6	67.9	37.0	42.5
Video-R1	7B	51.2	–	–	–	–
LongLLaVA	13B	68.5	–	–	–	–
Gemma3	27B	62.3	57.9	–	–	–
Visual Sketchpad (GPT-4o)	–	80.4	–	–	–	–
IVM-Enhance (GPT-4V)	–	81.2	–	–	–	–
SEAL	7B	74.8	38.1	35.6	–	–
PaLI-3-VPD	5B	70.9	–	–	–	–
PaLI-X-VPD	55B	76.6	–	–	–	–
Pixel-Reasoner (SFT)	7B	77.5	68.4	58.5	38.5	34.9
+ VERIFYCOT	7B	**83.2**	**70.9**	**63.1**	**40.5**	**48.8**
Pixel-Reasoner (RL)	7B	84.3	72.6	64.5	38.3	47.8
+ VERIFYCOT	7B	**85.9**	**74.8**	**67.4**	**43.7**	**48.0**
Visual CoT	7B	52.4	47.1	36.4	30.4	35.5
+ VERIFYCOT	7B	**68.1**	**52.8**	**47.8**	29.9	**36.8**

uses the full-image setting in Table 2 of the ZoomBench paper, not the separate cropped-region evaluation.

Table 6 gives the full comparison, including all twelve published reference models and the three local base-model/ VERIFYCOT pairs. It includes the six published baselines omitted from the compact main table and preserves the main table's results.

C ADDITIONAL IMPLEMENTATION DETAILS

C.1 INFERENCE COMPUTE ACCOUNTING

For Pixel-Reasoner, we estimate the major matrix-multiplication and convolution FLOPs from recorded trajectories, counting a multiply–add as two FLOPs. Costs include the base VLM's visual encoder, context processing and generation, the request interfaces, verification, and grounding. We use the actual input sizes, token lengths, and call counts for each trajectory, and average over all evaluated questions, including those without intervention. Frozen modules are included.

To compare Pixel-Reasoner variants under the same caching conditions, these estimates assume that unchanged KV-cache prefixes and previously computed features of identical visual inputs can be reused across rounds. New or changed inputs incur their encoding cost, and invalidated prefixes are recomputed. Discarded reasoning is counted, as is any newly generated continuation. This is an idealized accounting assumption, not a claim that the evaluated runtime implements this reuse. Each policy is estimated from its own trajectory rather than by adding a fixed overhead to the native baseline. Normalization, activations, preprocessing, and data movement are not included in the major-operation count. Latency is measured separately; the matched-decision timing protocol is described in Appendix C.3.

For Visual CoT, compute accounting holds the generated token count fixed. The baseline, including localization and final-answer generation, uses 28.95 TFLOPs per question. The verifier contributes 0.228 TFLOPs, including two SigLIP2 crop encodings; each global DINO call costs 2.0959 TFLOPs. All correction variants use one global view per request, without tiles. Total compute is 31.27 TFLOPs per question for Always ground and 30.12 for Random gate and VERIFYCOT, corresponding to 1 and 0.46 calls per question. These totals hold the verifier budget fixed.

C.2 CORRECTION BASELINE SETTINGS

Shared setup. For both post-hoc baselines, we use the frozen Pixel-Reasoner SFT checkpoint and complete native reasoning with its original visual tools before applying correction. Both methods use only the original image during post-hoc correction, without reattaching crop images, invoking new tools, or accessing ground-truth answers or external critics.

LookBack. Our implementation follows the VISCO pipeline: it generates an independent model reference answer, extracts visual verification questions from the original reasoning steps, answers these questions from the image, synthesizes a step-wise critique, and revises the final answer accordingly. We retain the original reasoning without truncating it to five steps.

Self-correction. The model receives the original image, question, and native text/tool transcript, and reviews and revises its answer in a single additional call, without separate reference generation or critique stages.

Generation and failure handling. Post-hoc calls use temperature 0.7, top-p 1.0, and a maximum of 2,048 generated tokens per call. Malformed or failed calls are retried once with the same prompt; if both attempts fail, the native answer is retained.

Zero-shot self-correction. This baseline uses the same Pixel-Reasoner SFT model for crop verification and correction, without additional training. Before returning the first successful crop to subsequent reasoning, we give PR the crop, original question, and the reasoning intent associated with the crop request. It checks whether the target is clear and the crop contains sufficient context, returning only Yes or No. A Yes decision retains the original crop. For No, the rejected crop is withheld from subsequent reasoning; PR receives the rejection feedback and the original image and is asked to generate new crop coordinates once.

C.3 MATCHED-DECISION TIMING COMPARISON

Unlike the LookBack comparison, this experiment holds correction decisions fixed to isolate the cost of delaying correction. The first-crop restriction below applies only to this timing comparison; in the main evaluations, VERIFYCOT verifies every visual-evidence step.

Correction boundary and rollback. In the Pixel-Reasoner/HR8K timing study, external correction is allowed only after the first successful crop request has been generated and before the crop is returned to the model. Pre-consumption correction returns the selected crop and continues generation. Post-hoc rollback first completes the natural trajectory. If replacement is selected, it discards all continuation text and subsequent crop results after the saved boundary, restores that boundary, and generates a new continuation with the selected crop. Later crop requests are handled by the base model without further external correction. The two procedures use the same correction policy, with matching decisions and final outputs checked across paired executions. Post-hoc decisions use the saved boundary state, not information from the later trajectory.

Timing and cache settings. End-to-end latency is measured by an outer timer around each complete execution, including prefix generation, expert calls, tool execution, rollback where applicable, and continuation generation. Model and service loading are excluded. The generator uses batch size one, tensor parallel size one, BF16, and greedy decoding; Grounding DINO uses FP32. Cross-call prefix caching is disabled, while ordinary KV caching within each generation call remains enabled.

Timing stability. Pre-consumption correction reduces end-to-end latency by 8.74% (95% CI: 6.98–10.65%; Table 7). Intervals use 2,000 paired bootstrap resamples grouped by the 200 original questions, keeping their four answer-option rotations together. Savings are positive under both execution orders and in each of the four shards.

Compute under actual and ideal caching. Table 8 compares estimated major-arithmetic FLOPs under two cache settings for the same 800 paired trajectories. The actual runtime estimate uses the recorded decoder and visual-encoder computations, with cross-call prefix caching disabled and

Table 7: Matched-decision timing on Pixel-Reasoner (SFT)/HR8K. Time is mean seconds per example; output-token counts are totals within each group. Savings are relative to post-hoc rollback. The continuation row covers generator calls after the first crop; the other rows cover complete tasks. Both procedures produce identical answers (505/800 correct).

	Time (s)		Output tokens		Savings (%)	
Scope	Post-hoc	Before use	Post-hoc	Before use	Time	Tokens
Full tasks						
All 800 examples	12.1	11.1	150,398	131,752	8.7	12.4
Full tasks						
287 replaced examples	14.5	11.5	—	—	20.9	30.3
Continuation						
287 replaced examples	—	—	34,754	16,108	53.4	53.7

Overall time saved: 1.1 seconds per example (95% CI: 0.8–1.3). Dashes indicate unreported absolute values.

Table 8: Estimated compute for matched-decision correction on Pixel-Reasoner (SFT)/HR8K. TFLOPs are averaged over all 800 examples; savings are relative to post-hoc rollback.

Cache setting	Pre-consumption (TFLOPs)	Post-hoc rollback (TFLOPs)	Savings (%)
Actual runtime	305.9	377.4	18.9
Ideal cache reuse	151.6	159.7	5.1

ordinary within-call KV caching enabled. The ideal-cache estimate additionally allows reuse of unchanged prefixes and previously computed features of identical visual inputs. Each procedure starts with an empty cache for each example, with no reuse across procedures or examples. Both estimates include all expert costs.

Pre-consumption correction reduces estimated compute by 18.9% under the actual runtime settings and by 5.1% under ideal cache reuse. Even with ideal reuse, post-hoc rollback still incurs the cost of generating the discarded continuation.

C.4 Adaptor architecture and training details

Role-specific latent interfaces. Following learned-query resampling, each branch compresses the hidden-state window H_t into a fixed-length sequence $Z_t^r = \mathcal{R}_r(H_t)$, for $r \in \{\mathrm{sem}, \mathrm{grd}\}$. The branches use the same input-state definition and resampling design but have independently learned parameters. The semantic adaptor pools its request tokens, projects them into the frozen vision–language encoder's shared embedding space, and normalizes the result: $z_t^{\mathrm{sem}} = \mathrm{norm}(\mathcal{P}_{\mathrm{sem}}(\mathrm{pool}(Z_t^{\mathrm{sem}})))$. Here $\mathcal{P}_{\mathrm{sem}}$ denotes the projection component of the semantic request adaptor $\mathcal{A}_{\mathrm{sem}}$. For candidate evidence e, the frozen visual encoder supplies $v_{t,e} = \mathrm{norm}(\Phi(I, e))$. The grounding adaptor instead preserves its request-token sequence and maps it into the specialist's conditioning space.

Interface dimensions and parameter counts. Both implementations use up to 16 of the most recent language-token hidden states from the final decoder layer before evidence selection, four learned queries, two resampling layers, four attention heads, and a hidden width of 256. Each layer contains cross-attention to the input states, a feed-forward block with width 1,024, and query self-attention. Padding and visual-token positions are masked. The semantic interface mean-pools the four output tokens and projects the result to 768 dimensions. The grounding interface applies a $256 \to 512 \to 256$ projection to each token, producing four conditioning tokens for Grounding DINO. The two interfaces have separate parameters.

Verifier structure. The global/local verifier combines request similarity to a pooled crop embedding with request similarity to its local patch embeddings. It uses a $768 \to 256 \to 768$ residual

Table 9: Interface dimensions and parameter counts. The full verifier includes its semantic request interface and visual adaptors. Frozen base models and external encoders are excluded.

Component	Pixel-Reasoner	Visual CoT
Input hidden-state dimension	3,584	4,096
Semantic request interface	3,236,352	3,368,448
Full global/local verifier	4,224,259	4,356,355
Latent grounding interface	3,301,888	3,433,984

adaptor on the pooled embedding and a 768-dimensional projection on the patches. A learned mixing weight combines the two scores; a learned scale and bias produce the binary logit. A sigmoid converts this logit to the score used in the method notation. By default, the threshold is calibrated on constructed SFT development pairs to accept approximately 85% of annotated positives and is kept fixed during testing. The same head and threshold are reused to check the highest-ranked specialist proposal on routed events.

Optimization. We use a learning rate of 10^{-5} and early stopping with patience 5.

Random seeds. Verifier training uses seeds 2026 and 2027; the primary checkpoints use seed 2026. Latent grounding training uses seed 2026. Visual CoT data selection and image-grouped splitting also use seed 2026. Pixel-Reasoner's negative sampler derives a deterministic per-step seed from the step identifier and the fixed seed `2026`.

Stage-wise supervision. Each request interface is first pretrained using raw step references encoded by its corresponding external model's frozen text encoder: SigLIP2 for the semantic request interface and Grounding DINO for the grounding request interface. The two interfaces are independently pretrained to match these teacher embeddings. The contrastive retrieval objective $\mathcal{L}_{\text{align}}$ then refines the semantic request representation, matching the annotated target region against the negative regions described in Appendix B.1. This contrastive task learns request–evidence alignment; online routing uses the binary head trained with binary cross-entropy $\mathcal{L}_{\text{bin}}$ using annotated positives and the same three types of negatives. The grounding interface is trained separately with Grounding DINO's standard bounding-box loss and the annotated target regions. Only the lightweight request interfaces and binary head are learned; the base interleaved VLM, semantic encoder, and grounding backbone remain frozen. These objectives train the verifier and proposal generator; no separate selector is trained.

Top-ranked proposal verification. On a routed event, Grounding DINO provides up to five alternative proposals, ranked by its native scores. We evaluate the highest-ranked proposal with the same semantic request, encoder, binary head, and threshold τ used for the native gate. If it passes, $\mathcal{S}$ adopts it; otherwise, the policy retains the native evidence. An empty proposal set also leaves the native evidence unchanged. Fallback declines replacement rather than certifying the native evidence as valid. The native evidence is not assigned a DINO score. Only the highest-ranked alternative is rechecked, requiring one additional candidate encoding and binary-head evaluation when a proposal is available.

C.5 Evidence state, cache restoration, and commit

The runtime implements $\mathcal{U}_M(C_t, I, \hat{e}_t)$ using a native realization function $\rho_M(I, \hat{e}_t)$ that serializes or materializes the selected evidence. The state-consistency condition is that, before reasoning resumes, the committed action tokens, cached states, and visual context entries representing the current evidence all correspond to the same selection $\hat{e}_t$.

Let S_t be a snapshot of the committed runtime state immediately before the earliest token or state update that would publish the tentative evidence. It retains all previously accepted reasoning. The latent-state window H_t is captured before the first token of the current evidence action, independently of how long provisional proposal generation subsequently takes. For an autoregressive implementation, the snapshot comprises the committed token prefix and key–value cache boundary,

Table 10: Verification on the image-held-out SFT test set for Pixel-Reasoner. All scores are percentages. Verification uses method-specific, development-calibrated thresholds fixed at test time.

Request	Correct accepted	Errors found ↑
Raw ref.	**85.8**	32.6
Clean ref.	82.7	65.1
Latent	82.5	**70.6**

attention masks and positions, and multimodal pointers or metadata needed to restore that same prefix.

The integration follows this sequence:

1. Save S_t and freeze H_t at the pre-action boundary.
2. Decode the natural proposal on a provisional working state. Decode enough of the action to materialize the candidate, while deferring its publication to subsequent reasoning.
3. Verify the natural evidence and, when correction is invoked, apply the candidate-selection policy to obtain the item $\hat{e}_t$ to commit. The proposal and all candidate scores remain provisional at this point.
4. Restore S_t, discarding state introduced by provisional decoding. Apply the native realization $\rho_M(I, \hat{e}_t)$ and evidence update $\mathcal{U}_M$ exactly once. If the interface encodes evidence in action tokens, serialize and replay the selected action so that tokens and cached states agree with the selected evidence. If it introduces no input tokens, apply the corresponding native state update instead.
5. Resume reasoning from the resulting committed state. The same procedure applies to any later evidence event on this updated trajectory.

For Visual CoT, the evidence entry is the selected image crop; for Pixel-Reasoner, it is the crop observation returned by the visual tool. An unchanged accepted proposal may reuse its provisional state when this preserves the same committed state as the restore-and-commit procedure. For matched replay with stochastic decoding, the snapshot additionally includes the random-number state and sampling policy.

D Additional Ablations and Qualitative Analysis

D.1 Additional SFT verification results

Table 10 reports the supplementary Pixel-Reasoner request comparisons on the image-held-out SFT verification test set. All three verifiers use trained adaptors. Each method's threshold is calibrated to approximately 85% acceptance of positive SFT development pairs and then fixed for testing; the table reports actual test-set acceptance, not matched test-set retention. Correct acceptance and errors found denote the fractions of correct regions accepted and incorrect regions rejected, respectively, before replacement or fallback. For SFT verification, cleaned references are generated by Qwen2.5-VL-7B-Instruct from the question and preceding CoT (Appendix D.3).

On the SFT test set, latent requests find more errors than cleaned references (70.6% versus 65.1%) at similar observed correct acceptance (82.5% versus 82.7%).

D.2 Verification threshold sensitivity

We evaluate verification thresholds τ corresponding to different retention targets on Visual CoT/V* Bench (Table 11). For each target, τ is calibrated using only SFT development data before test-set evaluation and kept fixed throughout testing. The native model answers 100 of 191 questions correctly (52.36%). The 80% target yields 68.59% QA, compared with 68.06% at the default 85% target. Lower targets make more replacements, while higher targets reduce harm but also miss

Table 11: Verification-threshold ablation on Visual CoT/V* Bench. Retention targets refer to SFT development data, not test-set retention; each threshold is fixed before test-set evaluation.

Target (%)	QA (%)	Fix / Harm	Gain (pp)	Replacements
70	65.97	34 / 8	+13.61	108
80	**68.59**	34 / 3	**+16.23**	97
85	68.06	33 / 3	+15.71	87
90	67.02	30 / 2	+14.66	77
95	63.35	22 / 1	+10.99	58

Fix/Harm are question counts; gains are relative to Native.

opportunities to correct errors. These test results characterize threshold sensitivity rather than select the threshold used for the main evaluation.

D.3 CLEAN-REFERENCE EXTRACTION PROMPT

The clean-reference baseline uses Qwen2.5-VL-7B-Instruct in text-only mode to turn the question and pre-tool reasoning into a concise description of the current visual target. Earlier reasoning is included when available; for the first-crop examples in this comparison, that field is empty and the current step contains the CoT prefix before the tool call. No image, tool arguments, target annotations, or later answers are supplied. The model uses greedy decoding with at most 128 new tokens. Its output is reused as the cleaned reference for verification.

System message. The wording below is retained from the shared target-extraction prompt, including its reference to the grounder.

```
Rewrite the current step's crop target as a concise English
referring phrase for Grounding DINO. Use earlier reasoning
to resolve pronouns and retries.  Keep the intended objects,
attributes and spatial constraints.  If the current step
examines a relationship, keep both objects.  Do not answer
the visual question or invent missing details.  Return only
the referring phrase.
```

User-message template. Earlier reasoning entries are joined with a blank line; an empty history is written as `(none)`.

```
Question: {question}

Earlier reasoning:
{earlier_reasoning_or_(none)}

Current step:
{current_pre_tool_reasoning}
```

Few-shot examples. The following five synthetic examples are inserted as alternating user and assistant messages after the system message. Each user message uses the template above. The actual evaluation input follows these examples.

1. **Question:** What color is the umbrella?
   Earlier reasoning: `(none)`
   Current step: I cannot tell its color yet. I will zoom in on the umbrella beside the entrance.
   Assistant response: the umbrella beside the entrance
2. **Question:** Where is the kettle relative to the toaster?
   Earlier reasoning: `(none)`
   Current step: I will inspect the kettle and its position relative to the toaster.
   Assistant response: the kettle and the toaster

3. **Question:** What does the label on the package say?
   Earlier reasoning: I will inspect the label on the package held by the person wearing a striped scarf.
   Current step: The crop missed it. I will zoom in again.
   Assistant response: the label on the package held by the person wearing a striped scarf
4. **Question:** Where is the bicycle relative to the mailbox?
   Earlier reasoning: I have inspected the bicycle.
   Current step: Next I will zoom in on the mailbox beside the gate.
   Assistant response: the mailbox beside the gate
5. **Question:** When does registration close?
   Earlier reasoning: `(none)`
   Current step: I will zoom in on the section about the registration deadline.
   Assistant response: the text section describing the registration deadline

D.4 Cross-dataset evidence replacements

Table 12 reports final-answer accuracy with native, GT, and counterfactual evidence for Visual CoT and Pixel-Reasoner (SFT). GT replacement improves accuracy across all three datasets for both models, while counterfactual replacement reduces it. The size of the effect varies across models and datasets. For Visual CoT on TreeBench, end-to-end evaluation uses all 405 questions, whereas GT and counterfactual interventions and their Native baseline use the same 403-question subset.

Table 12: QA accuracy (%) under different evidence conditions. CF − Native reports the change under counterfactual evidence in percentage points (pp). Pixel-Reasoner's Native scores are from Table 1.

Evidence	V* Bench	ZoomBench	TreeBench
(a) Visual CoT			
Questions	191	845	403
Native	52.4	35.5	30.5
GT crop	**83.8**	**38.5**	**33.3**
Counterfactual crop	47.1	27.8	30.0
CF − Native (pp)	−5.2	−7.7	−0.5
(b) Pixel-Reasoner (SFT)			
Native	77.5	34.9	38.5
GT crop	**84.3**	**53.5**	**40.7**
Counterfactual crop	69.6	33.2	34.1
CF − Native (pp)	−7.9	−1.7	−4.4

D.5 Statistical uncertainty of end-to-end results

For each HR-Bench dataset, the 800 evaluation records contain four answer-option rotations of 200 original questions. Accuracy, Fix, and Harm are still computed over all 800 records. Grouping is used only for uncertainty estimation and hypothesis testing; a question is not required to be answered correctly under all four rotations to receive credit.

HR-Bench confidence intervals use 100,000 paired cluster-bootstrap resamples of the 200 original questions. Each draw retains all four rotations and both models' predictions for each sampled question. We use a two-sided exact paired permutation test at the original-question level, swapping the two models' predictions jointly across all four rotations within a question. The test statistic is the mean paired accuracy difference, equivalent to the difference over all 800 records. Non-HR p-values in Table 13 use McNemar's test. Among the completed HR-Bench comparisons, Pixel-Reasoner (SFT) on HR8K and Visual CoT on both HR4K and HR8K show significant gains at $p < 0.05$ (†).

Table 13: Paired end-to-end accuracy comparisons with Native. For HR-Bench, N counts evaluation records; confidence intervals and tests use original questions as the sampling unit. † marks significant HR-Bench gains ($p < 0.05$); "–" denotes unavailable statistics.

Dataset	N	Native QA (%)	+VERIFYCOT QA (%)	Fix	Harm	Gain (pp)	95% bootstrap CI (pp)	p
Pixel-Reasoner (SFT)								
V* Bench	191	77.5	83.2	20	9	+5.8	[0.5, 11.0]	0.0614
HR4K	800	68.4	70.9	78	58	+2.5	[−1.5, +6.5]	0.2440
HR8K	800	58.5	63.1	55	18	+4.6†	[+2.6, +6.8]	2.72×10^{-5}
TreeBench	405	38.5	40.5	13	5	+2.0	[0.0, +4.0]	0.0963
ZoomBench	845	34.9	48.8	175	58	+13.9	[10.4, 17.3]	7.88×10^{-15}
Pixel-Reasoner (RL)								
V* Bench	191	84.3	85.9	3	0	+1.6	[0.0, 3.7]	0.2500
HR4K	800	72.6	74.8	48	31	+2.1	[−0.6, +5.0]	0.1666
HR8K	800	64.5	67.4	67	44	+2.9	[−0.8, +6.5]	0.1379
TreeBench	405	38.3	43.7	47	25	+5.4	[1.5, 9.6]	0.0128
ZoomBench	845	47.8	48.0	15	13	+0.2	[−0.9, +1.4]	0.8506
Visual CoT								
V* Bench	191	52.4	68.1	33	3	+15.7	[10.0, 21.5]	2.27×10^{-7}
HR4K	800	47.1	52.8	62	17	+5.6†	[+2.6, +8.9]	3.786×10^{-4}
HR8K	800	36.4	47.8	106	15	+11.4†	[+7.3, +15.6]	8.76×10^{-8}
TreeBench	405	30.4	29.9	11	13	−0.5	[−3.0, 2.0]	0.8388
ZoomBench	845	35.5	36.8	40	29	+1.3	[−0.6, 3.2]	0.2284

D.6 QUALITATIVE EVIDENCE-REPLACEMENT EXAMPLES

We show a Visual CoT example from V* and a Pixel-Reasoner example from HR8K. For each example, the question and model checkpoint are unchanged between the natural and replacement runs.

Visual CoT: finding the Apple logo. Question: "What is the color of the Apple logo?" The choices are (A) polychromatic, (B) red, (C) white, and (D) silver; the correct answer is **A**. Visual CoT produces coordinates followed by an answer, without a textual reasoning trace. Its before/after trajectories are:

Before: $[0.005, 0.581, 0.101, 0.621]$ → natural crop → **C (white)**.

After: $[0.075, 0.262, 0.087, 0.276]$ → replacement crop → **A (polychromatic)**.



Figure 3: Visual CoT: the original image with the natural crop (orange, dashed) and replacement crop (blue, solid). Insets enlarge the actual crops. The replacement includes the Apple logo.

The natural crop misses the annotated logo, whereas the replacement crop contains it (Figure 3). Changing the visual evidence changes the answer from incorrect to correct.

Pixel-Reasoner: missing the neon sign without noticing. **Question:** "What color is the neon 'OPEN' sign hanging above the items for sale?" The choices are (A) red and blue, (B) blue and green, (C) white and blue, and (D) yellow and green; the correct answer is **A**. Both runs start with the same request:

> "The question asks about the color of the neon 'OPEN' sign hanging above the items for sale. To inspect this, I need to zoom in on the area where the sign is located." The model then calls `crop_image_normalized` with `bbox_2d = [1600, 400, 1800, 600]` on image 1.

After receiving the crop, each run directly produces its final answer, without another tool call:

> **Before:** "The cropped image focuses on the area above the items for sale, where the neon 'OPEN' sign is located. The sign is illuminated in **blue and green** colors." $\boxed{B}$
>
> **After:** "The cropped image focuses on the area above the items for sale, where the neon 'OPEN' sign is located. The sign is illuminated with red and blue colors." $\boxed{A}$

The natural crop misses the requested neon sign, but Pixel-Reasoner does not identify the mismatch and directly gives the wrong answer. The verifier rejects this crop, and the replacement shows the sign's red lettering and blue outline (Figure 4). With this evidence, the model answers correctly without another crop.

E Discussion and Limitations

E.1 From evidence correction to correct answers

VerifyCoT improves visual evidence quality by correcting invalid selections while preserving valid support. However, correct visual evidence does not guarantee a correct final answer. In our Visual CoT audit, replacing natural evidence with annotated target evidence raises accuracy from 49% to 83%, but also changes four previously correct answers to incorrect ones (Table 3). Similarly, one case repaired by VerifyCoT receives valid evidence but produces an incorrect answer.

These findings show that evidence correctness is important but is not the only factor in answer accuracy. The model must also use the evidence correctly in subsequent reasoning. Verification and selective correction therefore provide an opportunity to improve VLM reasoning, but do not resolve

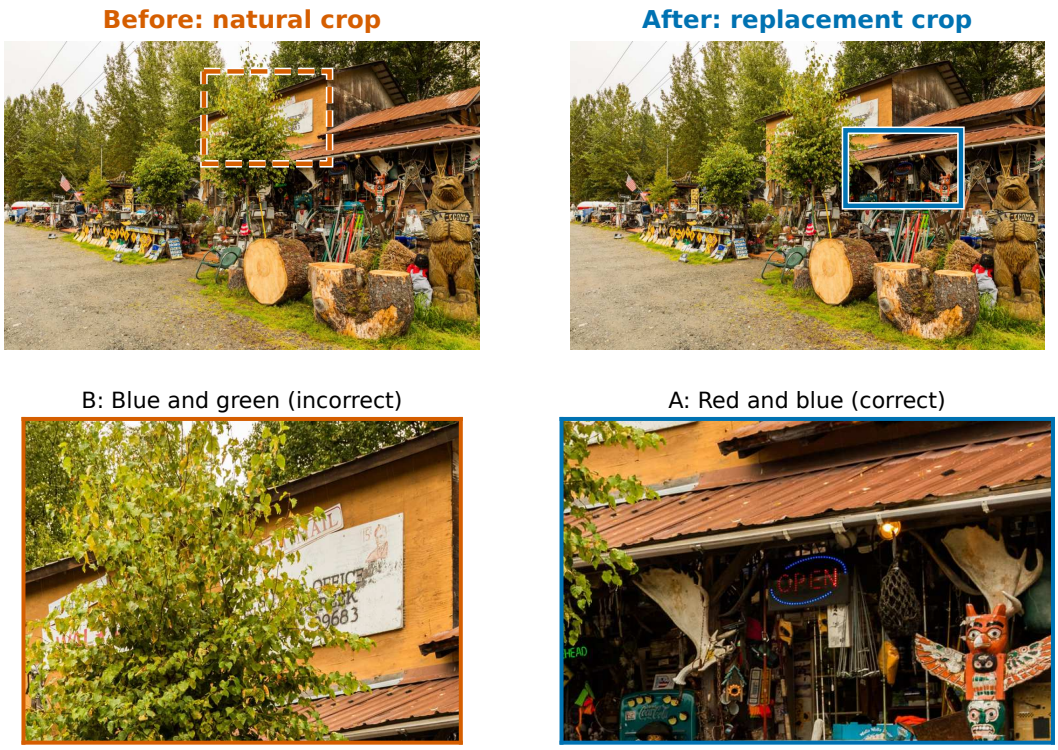


Figure 4: Pixel-Reasoner on HR8K: the natural crop (orange, dashed) shows branches and a wall sign rather than the requested neon 'OPEN' sign. The replacement crop (blue, solid) includes the neon sign. Lower panels show the actual crops.

every source of error. An important direction for future work is to understand when evidence repairs lead to answer improvements and how to help models make better use of corrected evidence.

E.2 Training data for new models

Our latent verifier and grounding interfaces are trained on annotated SFT trajectories from the base model. When such data are unavailable, adaptation could use collected model trajectories with VLM-assisted annotations of the requested targets and corresponding visual evidence. This introduces additional data collection, annotation, and training costs. Alternatively, teacher-generated or rule-extracted textual references can provide requests for verification and grounding. These choices change how the current target is represented, but not the strategy of verifying and selectively correcting evidence before consumption. Developing more accurate and less costly ways to represent what the model is looking for remains an important direction for future work.