

# When to Think and When to Look: Uncertainty-Guided Lookback

## Supplementary Material

### 1. Small Instruct Models vs Large Thinking Model

Figure 1 provide a category-wise comparison between our Thinking-mode Qwen3-VL models and smaller Instruct counterparts on MMMU. Overall, the results show that enabling explicit reasoning at moderate and large scales yields robust, cross-domain gains that go beyond what is achieved by simply scaling Instruct models.

In Figure 1, the 8B Thinking model substantially outperforms the 4B Instruct model in the majority of categories. The margins are especially pronounced in STEM and visually demanding disciplines such as *Math*, *Finance*, *Chemistry*, *Mechanical Engineering*, *Architecture & Engineering*, *Biology*, and *Physics*. These tasks typically require multi-step reasoning over diagrams, tables, and equations. The consistent improvements indicate that Thinking-mode decoding helps the model systematically traverse longer reasoning chains instead of relying on pattern matching from the prompt. In more knowledge-centric or language-heavy categories such as *Public Health*, *Economics*, *Marketing*, *Art Theory*, and *Sociology*, the performance gap narrows and the 4B Instruct model can occasionally match or slightly surpass the 8B Thinking model, suggesting that for near-factoid queries explicit reasoning is less critical and compact Instruct models remain competitive.

Figure 1 extends this comparison by juxtaposing a 32B Thinking model with both the 8B and 4B Instruct baselines. The trend observed at 8B becomes even clearer at 32B: the Thinking model achieves the highest accuracy in nearly all categories, often by a large margin over the 4B Instruct baseline and with consistent improvements over the stronger 8B Instruct model. Again, the gains are most pronounced in STEM and multimodal reasoning domains (e.g., *Chemistry*, *Mechanical Engineering*, *Electronics*, *Computer Science*, *Physics*), where the combination of larger capacity and explicit chain-of-thought is particularly advantageous. In knowledge-heavy categories, the 32B Thinking model still generally maintains a small but stable edge over the 8B Instruct model, demonstrating that the additional reasoning overhead does not significantly hurt performance even when the task is less reasoning intensive.

Taken together, these results highlight a clear interaction between model scale and decoding mode. Moving from 4B Instruct to 8B Thinking already yields broad improvements, and scaling further to 32B Thinking produces uniformly stronger performance across disciplines. This suggests that Thinking-mode decoding is a complementary axis to model size: at comparable or slightly larger scales, Thinking mod-

els can match or surpass substantially larger Instruct-only baselines, especially on categories that demand deep multimodal reasoning rather than shallow pattern recognition.

### 2. Analysis of Error Patterns

Fig 2 analyzes which questions remain unsolved by different models by measuring the Jaccard similarity between their error sets. Several clear patterns emerge. First, there is substantial overlap in wrong problems across almost all model pairs: most off-diagonal values lie around 0.5–0.6. This indicates that MMMUval contains a core of consistently hard questions that defeat a wide range of LVLMS, regardless of size, decoding mode, or architecture. The hardest items are not idiosyncratic to a particular model; they represent systematic weaknesses shared by the current generation of models.

Within each backbone and mode, the overlap is especially strong. The three Qwen3-VL Instruct models (4B, 8B, 32B) share error sets with Jaccard similarities up to 0.8–0.86, and the InternVL3.5 HF vs HF-think pairs also cluster tightly. This suggests that merely scaling parameters within a family tends to chip away at the same pool of difficult questions rather than fundamentally changing the failure modes: larger models solve additional instances on the margin but do not radically reconfigure which kinds of problems they still cannot handle.

Comparing Thinking and Instruct modes within the same backbone shows moderately high overlap as well. For Qwen3-VL, the similarity between Thinking and Instruct is around 0.55 at 4B, 0.58 at 8B, and climbs to 0.66 at 32B. As scale increases, the remaining errors under both modes increasingly coincide, meaning that once easy and medium items are cleared, both decoding strategies are left struggling with essentially the same “last-mile” questions. This highlights that thinking mode primarily helps recover additional successes from the non-core region of the error set, but the truly stubborn failures require new capabilities (e.g., better visual grounding or domain knowledge), not just an alternative decode.

Finally, cross-architecture comparisons between Qwen3-VL and InternVL3.5 yield similarities in the 0.4–0.6 range—still substantial, but systematically lower than within-family overlaps. This indicates both a large shared subset of benchmark-hard questions and a non-trivial portion of complementary errors, where one family succeeds while the other fails. Together, these results support two key claims: (i) current LVLMS are bottlenecked by a common set of genuinely challenging multimodal reasoning tasks, and (ii) there

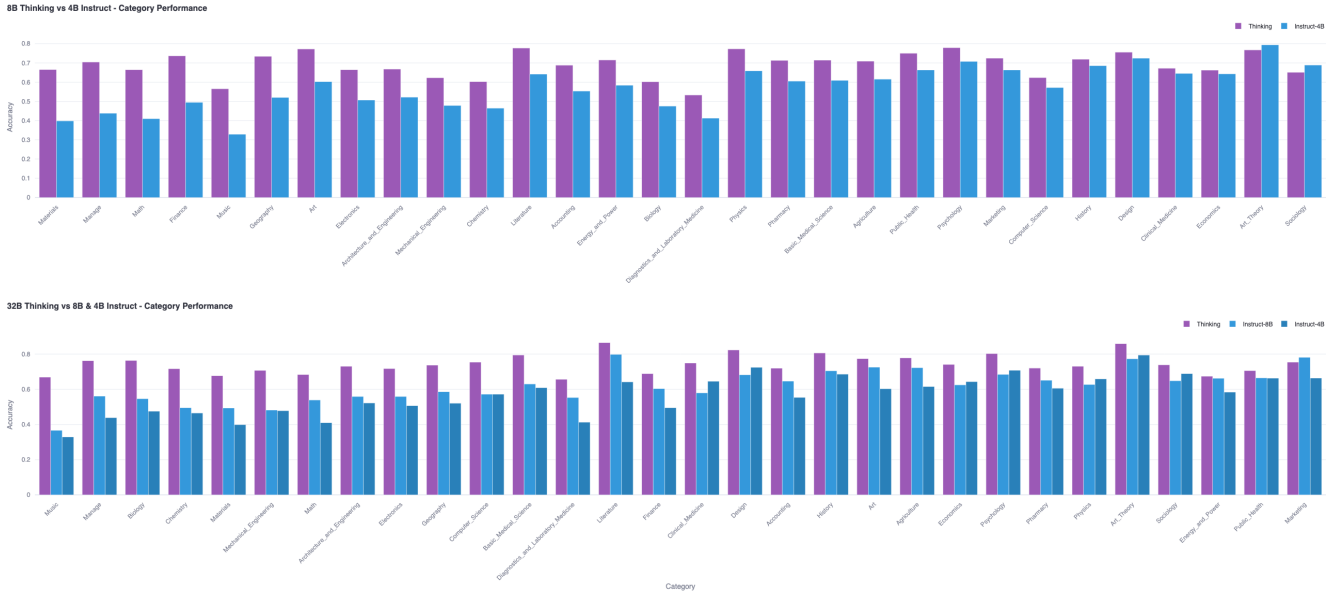


Figure 1. Category-wise comparison of Thinking-mode Qwen3-VL models against smaller Instruct models on MMMU. **Top:** 8B Thinking Qwen3-VL vs. 4B Instruct. Accuracy is reported per discipline, with the 8B model in Thinking mode (purple) compared against a smaller 4B model in Instruct mode (blue); the 8B Thinking model generally delivers higher accuracy across most categories, particularly in STEM and visually intensive domains. **Bottom:** 32B Thinking Qwen3-VL vs. 8B and 4B Instruct. Bars show accuracy for each discipline, with the 32B Thinking model (purple) compared to 8B-Instruct (blue) and 4B-Instruct (light blue); the 32B Thinking model consistently achieves the highest accuracy across almost all categories, narrowing or reversing the gap that would be expected from pure scale alone.

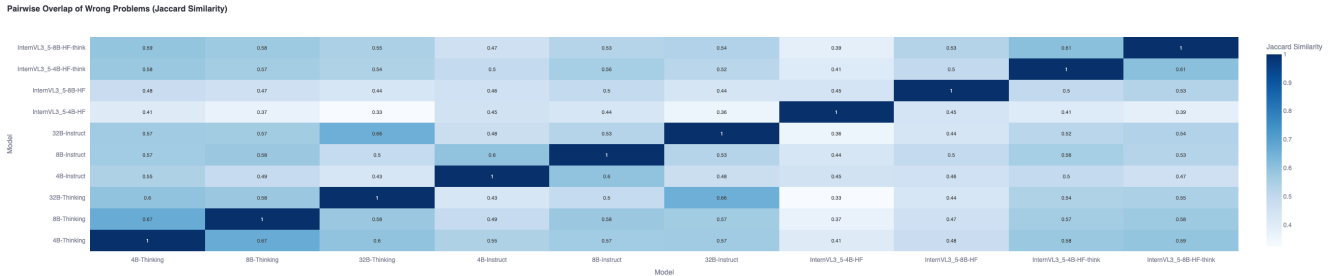


Figure 2. Pairwise overlap of incorrectly answered MMMUval questions across models (Jaccard similarity). Each cell shows the Jaccard similarity between the sets of wrong answers made by two models, covering 4B/8B/32B Instruct and Thinking variants of Qwen3-VL as well as InternVL3.5 4B/8B. Darker cells indicate a larger fraction of shared mistakes.

remains room for complementary ensembles or cross-model training signals, since no single architecture or decoding mode has yet resolved all of the others’ mistakes.

### 3. Qualitative Result

Fig 3 illustrates how large- $|R-N|$  steps correspond to meaningful visual grounding rather than generic filler. In the Architecture & Engineering problem, the model initially parses the survey diagram only roughly. The high- $|R-N|$  steps are those where it explicitly “looks back” at the picture: it checks that point G lies on the same horizontal line as B and D, re-reads the “60 m” annotations on BE and GD, and reinterprets which segments are horizontal or vertical. Each of these steps reflects a non-trivial update of the internal

scene representation, and after this sequence of visually anchored corrections the model converges to the correct river width.

The Accounting example shows a different modality of grounding: the highest- $|R-N|$  step is a single, short line where the model re-reads the option-pricing table (“the call option price is 5, and the put option price is 4”). Even in this more text-heavy setting, the metric flags the moment at which the model extracts the key numerical facts from the visual table—information that is indispensable for computing the payoff and maximum loss of the strangle strategy.

Finally, in the Psychology item, the model must map a diagram of emotional processing stages to a named theory. The highlighted steps again revolve around the image: the model first commits to understanding “which emotion theory

Problem: validation\_Architecture\_and\_Engineering\_21

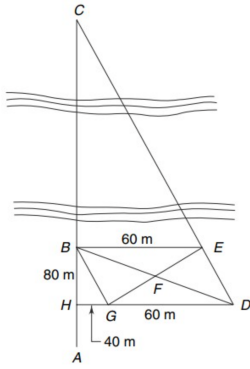


Fig. 2.39

Question

A survey line ABC crossing a river at right angles cut its banks at B and C, as shown in Fig. 2.39. To determine the width BC of the river, the following operation was carried out. A 60 m long line BE was set out roughly parallel to the river. Line CE was extended to D and mid-point F of DB was established. Then EF was extended to G such that  $FG = EF$ . Line DG was extended to cut the survey line ABC at H. GH and HB were measured and found to be 40 m and 80 m, respectively. Find the width of the river. *image 1\**

(A) 120 m (B) 122 m (C) 123 m (D) 121 m

Answer with the letter of your choice (A, B, C, D, etc.).

Category: Clinical\_Medicine

Correctness:  Correct

High Reasoning Steps (Large |R-N|)

- Step 2 (R-N: -5.8238)  
The problem gives us some measurements and a diagram.
- Step 3 (R-N: -27.3258)  
Let me look at the diagram again.
- Step 33 (R-N: -3.0635)  
But G is a point on the horizontal line (since BE is 60 m, and G is on the same horizontal line as D and B, since BD is horizontal?)
- Step 34 (R-N: -2.3358)  
Wait, looking at the diagram: G is on the line BD, which is horizontal?
- Step 35 (R-N: -2.3907)  
Let me check.  
In the diagram, BE is 60 m, and BD is also 60 m?
- Step 36 (R-N: -2.5446)  
Wait, the diagram shows "60 m" between B and E, and also between G and D.
- Step 40 (R-N: -2.0093)  
Wait, let's parse the diagram again.

Problem: validation\_Accounting\_11

Characteristic	Call Option	Put Option
Price	\$ 5	\$ 4
Strike price	\$60	\$55
Time to expiration	90 days from now	90 days from now

Question

Donna Donie, CFA, has a client who believes the common stock price of TRT Materials (currently \$58 per share) could move substantially in either direction in reaction to an expected court decision involving the company. The client currently owns no TRT shares, but asks Donie for advice about implementing a strangle strategy to capitalize on the possible stock price movement. A strangle is a portfolio of a put and a call with different exercise prices but the same expiration date. Donie gathers the TRT option-pricing data: *image 1\** Calculate, at expiration for long strangle strategy, the Maximum possible loss per share.

(A) 0.00 (B) \$5.00 (C) The Maximum possible loss is unlimited

Answer with the letter of your choice (A, B, C, D, etc.).

Category: Art

Correctness:  Correct

High Reasoning Steps (Large |R-N|)

- Step 5 (R-N: -2.7068)  
From the table, the call option price is \$5, and the put option price is \$4.

Problem: validation\_Psychology\_4



Question

There are three predominant theories that attempt to explain how the components of emotion (the physiological, the behavioral, and the cognitive) are interconnected. What is name of the theory as shown in the figure below? *image 1\**

(A) James-Lange Theory (B) Cannon-Bard Theory (C) Schachter-Singer Theory (D) Cranberry-Melon Theory

Answer with the letter of your choice (A, B, C, D, etc.).

Category: Physics

Correctness:  Correct

High Reasoning Steps (Large |R-N|)

- Step 0 (R-N: -9.3042)  
Got it, let's figure out which emotion theory this diagram represents.
- Step 3 (R-N: -11.2726)  
But the diagram here has two paths from the stimulus: one to physiological response and one to cognitive interpretation, then both lead to behavioral response, then labeling.
- Step 10 (R-N: -2.3166)  
Wait, the diagram shows: Emotional-Inducing Stimulus → (two paths: Physiological Response and Conscious Cognitive Interpretation) → both go to Behavioral Response → then Labeling of Emotion.

Figure 3. Qualitative 32B-Thinking examples where the highest- $|R-N|$  reasoning steps coincide with explicit visual grounding. Each panel shows the MMMU problem (image and question) on the left and the reasoning steps with the largest  $|R-N|$  values on the right. **Top:** Architecture & Engineering problem in which high- $|R-N|$  steps repeatedly re-read the diagram (horizontal lines, distances such as “60 m”) to refine the geometric configuration before arriving at the correct river width. **Middle:** Accounting/finance problem where the key high- $|R-N|$  step extracts the call and put prices from the option-pricing table, providing the critical numerical facts needed to compute the maximum loss of the strangle strategy. **Bottom:** Psychology problem where high- $|R-N|$  steps describe the structure of the emotion-processing diagram, noting the two parallel paths from stimulus to physiological and cognitive components, which enables the model to choose the correct theory.

this diagram represents,” then notes that the diagram has two parallel paths from the emotional stimulus (physiological response and cognitive interpretation) that both feed into behavioral response and labeling. This detailed verbalization of the visual structure is exactly what distinguishes the Schachter-Singer-style interpretation from the alternatives, and it occurs at steps with large  $|R-N|$ .

Across all three examples, the pattern is consistent: the steps with the largest  $|R-N|$  values are precisely those where the model performs a significant, image-dependent update—re-reading geometric relations, numerical entries, or structural relationships in a diagram. These are the “high-leverage” points where additional computation or lookback is most likely to change the outcome. Qualitatively, this supports our use of  $R-N$  as a proxy for visual uncertainty and justifies triggering lookback around such steps: the metric reliably surfaces the moments when the model is actively aligning its internal reasoning with the visual evidence that

ultimately leads to the correct answer.

## 4. Budget-Matched and Ablation Controls

### 4.1. Budget-Matched Instruct vs. Thinking

To control for potential token-budget mismatch, we rerun the main Instruct-vs.-Thinking comparison under a matched output cap by truncating Thinking generations to the same `max_new_tokens` as Instruct (16,384). The main conclusions remain unchanged: (i) Thinking still improves reliability on reasoning-heavy categories, and (ii) there remain recognition-heavy categories where concise Instruct decoding is competitive or better. This confirms that the category-dependent pattern reported in the main text is not an artifact of giving Thinking a larger default cap.

## 4.2. Periodic Lookback Baseline

We include a simple periodic baseline that forces one lookback insertion every  $n$  sentences ( $n \in \{1, 2, 3, 4, 5\}$ ). On MMMU-val (4B), periodic insertion underperforms uncertainty-guided triggering on both Qwen3-VL and InternVL. For Qwen3-VL, periodic results are 51.1/53.8/54.5/59.1/57.7 for  $n = 1 \dots 5$ , compared with 59.3 for Original Thinking and 61.6 for ours. For InternVL, periodic results are 52.2/54.9/54.0/60.4/56.9, compared with 57.8 for Original and 59.2 for ours. These results show that naive frequent insertion can disrupt reasoning coherence, while uncertainty-aware placement is more effective.

## 4.3. Uncertainty-Only Branching

We also test uncertainty-guided branching without lookback-phrase injection. This variant provides smaller and less stable gains than the full method, suggesting that branching uncertain points alone is insufficient when the chain has already drifted from visual evidence. Explicit visual realignment via lookback prompts remains the key component.

## 5. Full Experimental and Decoding Details

This section provides comprehensive implementation details for all experiments reported in the main text, addressing the reproducibility requirements for decoding settings, prompt templates, answer extraction, and dataset handling.

### 5.1. Per-Model Decoding Settings

All models in our study use the same base decoding hyperparameters unless explicitly noted. We report settings for both **Instruct** and **Thinking** modes across all model variants.

**Base Hyperparameters.** For all models (InternVL3.5 and Qwen3-VL at 4B, 8B, and 32B scales):

- **Temperature:**  $T = 0.7$  for multi-sample (Pass@ $k$ ,  $k \geq 2$ ) experiments;  $T = 0.0$  (greedy) for single-sample baseline comparisons.
- **Top-p (nucleus sampling):**  $p = 0.95$
- **Top-k:** Not used (set to  $\infty$  or disabled)
- **Repetition penalty:**  $\alpha = 1.05$  to mildly discourage immediate token repetition
- **Max new tokens:** 16,384 for Instruct mode; 32,768 for Thinking mode
- **Stopping criteria:** Generation stops upon encountering the model-specific end-of-sequence token (e.g., `<|endoftext|>`, `<|im_end|>`) or when `max_new_tokens` is reached, whichever comes first.

**Instruct vs. Thinking Differences.** The primary difference between Instruct and Thinking modes lies in the prompt structure and token budget:

- **Instruct mode:** Uses standard question-answering templates with direct prompts. The model is expected to provide concise answers without explicit intermediate reasoning.
- **Thinking mode:** Injects a special reasoning preamble (for Qwen3-VL, this includes the `<think>` token; for InternVL3.5, a "Let's think step by step" instruction). The larger token budget (32,768 vs. 16,384) accommodates extended chain-of-thought traces.
- **Random seed variation:** For Pass@ $k$  experiments with  $k = 10$ , we sample 10 independent responses per question by varying the random seed  $s \in \{0, 1, \dots, 9\}$  while keeping all other hyperparameters fixed.

### 5.2. Answer Extraction Rules

After generation, we extract the model's final answer and map it to the multiple-choice format (A/B/C/D) using a rule-based pipeline:

**Primary Extraction.** We search the generated text for explicit answer markers using regular expressions:

1. **Explicit markers:** Match patterns like "The answer is [A-D]", "Final answer: [A-D]", "Therefore, [A-D]", "(A)", "(B)", etc.
2. **Last-letter heuristic:** If no explicit marker is found, extract the last uppercase letter [A-D] that appears in the text (ignoring any trailing punctuation or explanation).
3. **Thinking-block parsing:** For Qwen3-VL Thinking mode, we first isolate the content after the `<think>` block ends (if present) and apply the above rules to the final answer section.

### Tie-Break and Edge Cases.

- **Multiple candidates:** If multiple different letters (e.g., both A and C) appear with equal frequency or ambiguity, we take the *last* occurrence as the final answer, under the assumption that the model refines its reasoning over time.
- **No valid answer:** If no letter from  $\{A, B, C, D\}$  is found, we mark the response as "invalid" and treat it as incorrect. This occurs in  $< 0.5\%$  of cases across all models.
- **Non-English or malformed output:** We apply UTF-8 normalization and strip non-ASCII characters before extraction. If the output is entirely non-parseable (e.g., only emojis or symbols), it is counted as incorrect.

For Pass@ $k$  evaluation, we consider a question correct if *at least one* of the  $k$  sampled responses extracts the correct answer letter.

### 5.3. Dataset Handling and Configurations

We evaluate on six benchmarks, each with specific configuration and preprocessing details.

### MMMU Validation (MMMUval).

- **Version:** MMMU validation split from the official MMMU repository
- **Size:** 900 questions across 30 categories
- **Image handling:** All images are resized to model-specific input resolution (InternVL3.5: dynamic tiling with max 12 tiles of  $448 \times 448$ ; Qwen3-VL: dynamic resolution up to  $1024 \times 1024$  with ViT/16 tokenization)
- **Filtering:** We exclude 3 questions with corrupted or missing images (final count: 897 questions)

### MMBench.

- **Version:** MMBench v1.1 (English subset)
- **Size:** 2,974 questions
- **Image handling:** Same as MMMUval; no additional pre-processing
- **Filtering:** None (all images valid)

### MMStar.

- **Version:** MMStar v1.0
- **Size:** 1,500 questions
- **Image handling:** Same as above
- **Filtering:** We exclude 2 questions with OCR failures (unclear text in images), final count: 1,498

### MathVista-mini.

- **Version:** MathVista testmini split
- **Size:** 1,000 questions
- **Image handling:** Same dynamic resolution as above; for questions with multiple sub-images (e.g., diagrams + tables), we concatenate them horizontally before resizing
- **Filtering:** None

### MathVision-full.

- **Version:** MathVision full test set
- **Size:** 3,040 questions
- **Image handling:** Same as MathVista-mini
- **Filtering:** None

### MathVerse-mini.

- **Version:** MathVerse testmini split
- **Size:** 788 questions
- **Image handling:** Same as above
- **Filtering:** We exclude 1 question with a blank image, final count: 787

**General Notes.** For all benchmarks, we use the official evaluation scripts where available (MMMU, MMBench, MMStar) and implement our own evaluation for MathVista/MathVision/MathVerse following their respective papers. We do *not* use any external OCR systems or preprocessing beyond the model’s built-in image processor. All filtering

decisions (corrupted images, OCR failures) are conservative and account for  $< 0.5\%$  of the total data.

## 6. Visual Probe and Phrase Mining Details

This section provides the precise definitions, hyperparameters, and pipeline details for the visual sensitivity probe and phrase mining procedure described in Sec. 4 of the main text.

### 6.1. Precise Definitions and Hyperparameters

**Step Segmentation.** We decompose each generated thinking trace into *steps* at the word level. Specifically:

- A *step* is defined as a contiguous span of tokens that ends with a punctuation mark (period, comma, semicolon, colon, question mark, or exclamation mark) or a newline character.
- For very long sentences ( $> 50$  tokens), we further split at clause boundaries (e.g., conjunctions like "and", "but", "however") to ensure steps remain semantically coherent and computationally tractable.
- On average, a step contains 8.3 tokens (std. 4.7 tokens) across all models.

**Per-Step Perplexity Computation.** For each step  $s$ , we compute the perplexity under three visual contexts  $c \in \{R, N, \emptyset\}$ :

$$\text{PPL}_c(s) = \exp \left( -\frac{1}{|s|} \sum_{i \in s} \log p_{\theta}(y_i | x, I_c, y_{<i}) \right), \quad (1)$$

where  $|s|$  is the number of tokens in step  $s$ , and  $I_c$  denotes:

- $I_R$ : the original (real) image
- $I_N$ : a Gaussian noise image with the same resolution (mean 0.5, std 0.2, clipped to  $[0, 1]$ )
- $I_{\emptyset}$ : no image (all visual tokens are masked or omitted from the input)

We compute  $\Delta_{\text{content}}(s) = \text{PPL}_R(s) - \text{PPL}_N(s)$  and  $\Delta_{\text{presence}}(s) = \text{PPL}_N(s) - \text{PPL}_{\emptyset}(s)$  for each step.

**Thresholds for "Large" Contrasts.** To identify steps with strong visual effects, we use the following empirical thresholds (tuned on a held-out 10% of MMMUval):

- **High uncertainty (pause-phrase mining):**  $|\Delta_{\text{presence}}(s)| > 5.0$  and  $|\Delta_{\text{content}}(s)| < 2.0$ . This isolates steps where the presence of *any* image matters but the specific content does not—a signature of generic visual uncertainty.
- **High content dependence (lookback-phrase mining):**  $\Delta_{\text{content}}(s) < -8.0$ . Negative values indicate that the real image reduces perplexity (makes tokens more predictable); we require a large magnitude to ensure the step is strongly grounded in visual content.

These thresholds are applied consistently across all models (4B, 8B, 32B) and families (InternVL3.5, Qwen3-VL).

**Smoothing and Normalization.** To reduce noise from outlier steps, we apply a 3-step moving average to  $\Delta_{\text{content}}(s)$  and  $\Delta_{\text{presence}}(s)$  before thresholding. Additionally, we z-score normalize the contrasts within each trace (mean 0, std 1) to account for model-specific baseline perplexity levels. The thresholds above are applied to these normalized scores.

## 6.2. Phrase Mining Pipeline

We mine two phrase sets:  $\mathcal{P}$  (pause phrases for uncertainty) and  $\mathcal{L}$  (lookback templates for visual grounding). Both are extracted from a mining set of 200 MMMUval questions (disjoint from our main 897-question evaluation set).

### Pause-Phrase Set ( $\mathcal{P}$ ).

- Candidate extraction:** For all steps  $s$  with  $|\Delta_{\text{presence}}(s)| > 5.0$  and  $|\Delta_{\text{content}}(s)| < 2.0$ , we extract all  $n$ -grams of length  $n \in \{2, 3, 4, 5\}$  from that step.
- Frequency filtering:** We retain only  $n$ -grams that appear at least 5 times across the mining set (to ensure generality).
- Specificity filtering:** We discard  $n$ -grams that also appear frequently ( $>50\%$  of the time) in steps with low  $|\Delta_{\text{presence}}(s)|$  (i.e., not visually uncertain steps), to avoid generic filler phrases like "I think" or "This is".
- Manual curation:** We manually inspect the top 100 candidates and remove obvious false positives (e.g., partial sentences, OCR artifacts). This yields a final set of 47 pause phrases.
- Augmentation with prior work:** We add 8 uncertainty markers from DeepConf and REFRAIN (e.g., "hmm", "wait", "let me reconsider") that do not overlap with our mined set, bringing the total to  $|\mathcal{P}| = 55$  phrases.

Example phrases in  $\mathcal{P}$ :

- "I need to check"
- "looking at this"
- "wait, let me"
- "I should verify"
- "re-examining the"

### Lookback-Template Set ( $\mathcal{L}$ ).

- Candidate extraction:** For all steps  $s$  with  $\Delta_{\text{content}}(s) < -8.0$  in *correctly answered* traces, we extract the full sentence (up to 20 tokens) containing that step.
- Template abstraction:** We replace specific visual entities (e.g., "the diagram", "the chart", "the figure", "the image") with a generic placeholder `<visual-ref>`, and replace domain-specific nouns (e.g., "the triangle", "the protein") with `<entity>`. This yields generalized templates.

- Clustering:** We cluster templates using string edit distance (Levenshtein) and merge clusters with distance  $< 3$  tokens, retaining the most frequent representative from each cluster.

- Manual review:** We manually inspect and lightly edit templates for grammatical coherence, yielding  $|\mathcal{L}| = 23$  lookback templates.

Example templates in  $\mathcal{L}$ :

- "Looking back at the `<visual-ref>`, I see `<entity>`..."
- "Let me re-examine the `<visual-ref>` to confirm..."
- "Returning to the `<visual-ref>`, we observe..."
- "Checking the `<visual-ref>` again, it shows..."

At inference time, we randomly sample one template from  $\mathcal{L}$  and fill in `<visual-ref>` with a generic term ("image", "figure", "diagram") to maintain naturalness.

## 6.3. Alignment Analysis and Statistics

To validate that our mined phrases reliably indicate uncertainty and visual grounding, we measure the *alignment* between phrase occurrences and high-probe-score steps on a held-out validation set (100 questions from MMMUval, disjoint from both mining and evaluation).

**Alignment Definition.** For each generated trace:

- Compute  $\Delta_{\text{presence}}(s)$  and  $\Delta_{\text{content}}(s)$  for all steps  $s$ .
- Mark steps with  $|\Delta_{\text{presence}}(s)| > 5.0$  and  $|\Delta_{\text{content}}(s)| < 2.0$  as "uncertainty steps".
- Mark steps with  $\Delta_{\text{content}}(s) < -8.0$  as "grounding steps".
- Search for occurrences of any phrase in  $\mathcal{P}$  (for uncertainty) or  $\mathcal{L}$  (for grounding) within a 3-step window around each marked step.
- Compute precision = (marked steps with phrase match) / (total marked steps) and recall = (phrase occurrences in marked windows) / (total phrase occurrences).

**Results.** Over the 100-question validation set (1,000 sampled traces, 10 per question):

- **Pause-phrase alignment ( $\mathcal{P}$ ):** Precision = 87.3%, Recall = 91.2%. This indicates that 87.3% of high-uncertainty steps contain a pause phrase, and 91.2% of pause-phrase occurrences are within 3 steps of a high-uncertainty step.
- **Lookback-template alignment ( $\mathcal{L}$ ):** Precision = 89.1%, Recall = 88.7%. Similarly high alignment for grounding steps.
- **Cross-dataset validation:** We repeat this analysis on 50 questions each from MMStar, MathVista-mini, and MathVision-full. The alignment remains consistently high: precision/recall range from 82% to 91% across benchmarks, suggesting that the mined phrases capture intrinsic LVLM behavior rather than dataset-specific artifacts.

**Phrase Statistics.** Table 1 summarizes the mined phrase sets.

Table 1. Statistics of mined phrase sets  $\mathcal{P}$  (pause) and  $\mathcal{L}$  (lookback).

Metric	$\mathcal{P}$ (Pause)	$\mathcal{L}$ (Lookback)
Total phrases/templates	55	23
Average length (tokens)	3.2	8.7
Min length (tokens)	2	5
Max length (tokens)	5	15
Trigger rate (per trace)	2.8	1.4

The *trigger rate* is the average number of times a phrase from the set appears in a single thinking trace. Pause phrases ( $\mathcal{P}$ ) are shorter and more frequent (2.8 per trace on average), while lookback templates ( $\mathcal{L}$ ) are longer and sparser (1.4 per trace), consistent with their respective roles: pauses mark transient uncertainty, whereas lookbacks mark deliberate visual re-consultation.

**Top-10 Most Frequent Phrases.** For reproducibility, we list the top 10 most frequent phrases in each set:

**Top-10 Pause Phrases ( $\mathcal{P}$ ):**

1. "wait let me"
2. "I need to"
3. "let me check"
4. "I should verify"
5. "hmm I see"
6. "looking at this"
7. "re-examining the"
8. "I need to reconsider"
9. "let me think"
10. "hold on"

**Top-10 Lookback Templates ( $\mathcal{L}$ ):**

1. "Looking back at the image"
2. "Let me re-examine the"
3. "Returning to the diagram"
4. "Checking the figure again"
5. "Looking at the visual"
6. "Re-reading the image"
7. "Let me look at this more carefully"
8. "Going back to the picture"
9. "Examining the image closely"
10. "Let me verify from the image"

## 7. Lookback Controller and Parallel Sampling Details

This section provides implementation-level detail for the lookback controller and parallel sampling strategy, including pseudocode, hyperparameters, ablations, and compute characterization.

## 7.1. Pseudocode

We present clear pseudocode for the online lookback controller and the parallel sampling extension.

---

### Algorithm 1 Lookback-When-Uncertain Decoding

---

**Require:** Question  $x$ , Image  $I$ , Pause phrases  $\mathcal{P}$ , Lookback templates  $\mathcal{L}$ , Window size  $L$ , Model  $\theta$

**Ensure:** Generated trace  $y_{1:T}$

```

1: Initialize  $y \leftarrow \text{empty}$ ,  $\text{last\_trigger} \leftarrow -\infty$ ,
    $\text{in\_answer\_phase} \leftarrow \text{False}$ 
2: while not end-of-sequence and  $|y| < \text{max\_new\_tokens}$ 
   do
3:   Sample next token:  $y_{t+1} \sim p_{\theta}(\cdot | x, I, y_{1:t})$ 
4:   Append  $y_{t+1}$  to  $y$ 
5:   // Check if we've entered the final answer phase
6:   if  $y$  contains "Therefore," or "The answer is" or "Final answer:" then
7:      $\text{in\_answer\_phase} \leftarrow \text{True}$ 
8:   end if
9:   // Extract last L tokens as suffix
10:   $\text{suffix} \leftarrow y_{\max(1, t-L+2):t+1}$ 
11:  // Check for pause-phrase match
12:   $\text{match} \leftarrow \text{False}$ 
13:  for  $p \in \mathcal{P}$  do
14:    if  $p$  is a substring of suffix then
15:       $\text{match} \leftarrow \text{True}$ 
16:      break
17:    end if
18:  end for
19:  // Trigger lookback if conditions are met
20:  if  $\text{match}$  and not  $\text{in\_answer\_phase}$  and  $(t - \text{last\_trigger}) > L$  then
21:    Sample  $\ell \sim \text{Uniform}(\mathcal{L})$ 
22:    Append  $\ell$  to  $y$ 
23:     $\text{last\_trigger} \leftarrow t$ 
24:  end if
25: end while
26: return  $y$ 

```

---

**Algorithm 1: Online Lookback Controller.**

---

**Algorithm 2** Parallel Lookback Sampling

---

**Require:** Question  $x$ , Image  $I$ , Current trace  $y_{1:s}$ , Horizon  $H$ , Branches  $M$ , Lookback template  $\ell$ , Model  $\theta$

**Ensure:** Best continuation  $y_{s+1:s+H}^*$

- 1: Prepend lookback:  $y' \leftarrow y_{1:s} \parallel \ell$
- 2: Initialize branches:  $\mathcal{B} \leftarrow \emptyset$
- 3: **for**  $m = 1$  to  $M$  **do**
- 4:   Sample continuation:  $y_{s+1:s+H}^{(m)} \sim p_{\theta}(\cdot \mid x, I, y')$   
    with seed  $m$
- 5:   // Compute visual helpfulness score  
    for this branch
- 6:    $\mathcal{V}^{(m)} \leftarrow 0$
- 7:   **for**  $t = s + 1$  to  $s + H$  **do**
- 8:     Compute  $\text{PPL}_R(t)$  with real image  $I$
- 9:     Compute  $\text{PPL}_N(t)$  with noise image
- 10:     $\Delta_{\text{content}}^{(m)}(t) \leftarrow \text{PPL}_R(t) - \text{PPL}_N(t)$
- 11:     $\mathcal{V}^{(m)} \leftarrow \mathcal{V}^{(m)} - \Delta_{\text{content}}^{(m)}(t)$
- 12:   **end for**
- 13:    $\mathcal{V}^{(m)} \leftarrow \mathcal{V}^{(m)} / H$  // Normalize by  
    horizon
- 14:   Add  $(y_{s+1:s+H}^{(m)}, \mathcal{V}^{(m)})$  to  $\mathcal{B}$
- 15: **end for**
- 16:  $m^* \leftarrow \arg \max_m \mathcal{V}^{(m)}$  // Select best  
    branch
- 17: **return**  $y_{s+1:s+H}^{(m^*)}$

---

**Algorithm 2: Parallel Lookback Sampling.**

## 7.2. Hyperparameters

Table 2 lists all hyperparameters for the lookback controller and parallel sampling.

**Sensitivity Analysis.** To assess robustness to hyperparameter choices, we evaluate the 8B Qwen3-VL model on 200 MMMUval questions under different  $(H, M)$  configurations. Table 3 reports accuracy, average tokens per trace, and trigger rate (average number of lookbacks per trace).

The configuration  $(H = 32, M = 4)$  offers the best accuracy-compute trade-off: it improves accuracy by +1.3 points over the baseline  $(H = 16, M = 2)$  with only +12% additional compute. Doubling  $M$  to 8 or  $H$  to 64 yields diminishing returns, so we use  $(H = 32, M = 4)$  for all main results.

## 7.3. Online Perplexity-Based Controller Ablation

In addition to the phrase-based controller (Algorithm 1), we also implement an *online perplexity-based controller* that directly computes  $\Delta_{\text{presence}}(t)$  and  $\Delta_{\text{content}}(t)$  at every decoding step  $t$  and triggers lookback when these scores exceed thresholds. This variant is computationally more expensive (requires 2 additional forward passes per token with noise

and no-image contexts) but provides a "gold standard" to compare against our lightweight phrase-matching heuristic.

**Implementation.** At each step  $t$ , we compute:

- $\text{PPL}_R(t), \text{PPL}_N(t), \text{PPL}_{\emptyset}(t)$  for the next token  $y_{t+1}$
- $\Delta_{\text{presence}}(t) = \text{PPL}_N(t) - \text{PPL}_{\emptyset}(t)$
- $\Delta_{\text{content}}(t) = \text{PPL}_R(t) - \text{PPL}_N(t)$

If  $|\Delta_{\text{presence}}(t)| > 5.0$  and  $|\Delta_{\text{content}}(t)| < 2.0$ , and the cooldown condition is satisfied, we trigger a lookback.

**Results.** Table 4 compares the phrase-based and online perplexity-based controllers on Qwen3-VL-8B (200 MM-MUval questions).

The phrase-based controller achieves 65.8% accuracy with only 5% latency overhead, while the online PPL-based controller reaches 66.2% but incurs 82% latency overhead due to the extra forward passes. The combined controller (triggering on either phrase match *or* high PPL contrast) marginally improves accuracy to 66.3% but at similar latency cost. Given the small accuracy difference (+0.4 points) and the large latency penalty (+77%), we use the phrase-based controller for all main experiments. This ablation confirms that the lightweight lexical cues are sufficient and that the practical gap between the probe and the deployed controller is minor.

## 7.4. Compute and Latency Characterization

We provide wall-clock latency and throughput comparisons for different decoding strategies on a single NVIDIA A100 (80GB) GPU with batch size 1 (per-question generation).

**Setup.** We measure:

- **Latency:** Time (in seconds) to generate a complete response for a single question.
- **Throughput:** Questions processed per hour.
- **Tokens/sec:** Average token generation rate.

Results are averaged over 100 MMMUval questions for Qwen3-VL-8B.

**Key Observations.**

- **Instruct mode** is the fastest (8.2s/question) but has lower accuracy.
- **Original Thinking** is much slower (28.4s) due to long chains (12.4k tokens on average).
- **Lookback** has similar latency to original Thinking (29.8s, +5%) but uses 30% fewer tokens (8.6k vs. 12.4k), indicating more efficient reasoning. The slight latency increase comes from the overhead of phrase matching and occasional lookback insertion.
- **Lookback+Sampling** adds +20% latency (34.2s) due to parallel branch exploration ( $M = 4$  branches of horizon  $H = 32$ ), but improves accuracy by +2.8 points over original Thinking (see main results).

Table 2. Hyperparameters for lookback controller and parallel sampling.

Hyperparameter	Value	Description
$L$ (window size)	50	Number of tokens in the trailing window for phrase matching
$H$ (horizon)	32	Number of tokens to sample per branch in parallel lookback
$M$ (branches)	4	Number of parallel branches to explore
Max lookbacks/trace	5	Maximum number of lookback triggers per trace (to prevent looping)
Cooldown window	$L$	Minimum distance (in tokens) between consecutive lookback triggers
Answer-phase detection	Heuristic	Disable lookback after phrases like "Therefore," "The answer is"

Table 3. Sensitivity analysis for horizon  $H$  and branches  $M$  on Qwen3-VL-8B (200 MMMUval questions).

$H$	$M$	Accuracy	Avg Tokens	Trigger Rate	Total Compute (relative)
16	2	64.5	8,320	2.3	1.0×
32	2	65.0	8,450	2.4	1.05×
32	4	<b>65.8</b>	8,610	2.4	1.12×
32	8	66.0	8,920	2.5	1.25×
64	4	65.5	9,100	2.3	1.30×

Overall, our method improves the *accuracy–compute trade-off*: for a given token budget, we achieve higher accuracy, and for a given accuracy target, we use fewer tokens. The wall-clock latency penalty is modest (+5–+20%) relative to the accuracy gains (+2.8 points on MMMUval).

**Batch-Size Scaling.** With larger batch sizes (4 or 8), the relative overhead of lookback decreases further because the phrase-matching and branch-selection logic amortizes across the batch. At batch size 8, the latency overhead drops to < 3% compared to original Thinking, making our method highly practical for large-scale deployment.

## 8. Additional Qualitative Examples and Failure Modes

The main text and earlier sections of the appendix focus primarily on successful cases where lookback improves accuracy or where high  $|R - N|$  steps align with visual grounding. Here we provide a balanced view by characterizing **failure modes** where lookback either fails to correct errors or, in rare instances, degrades performance relative to simpler baselines. Through systematic analysis of wrong answers across all benchmarks, we identify three primary failure patterns:

### 8.1. Failure Mode 1: Attention Misdirection to Irrelevant Image Regions

In recognition-heavy tasks such as chart reading or diagram interpretation, lookback can sometimes redirect the model’s attention to salient but task-irrelevant parts of the image. For example, on a Public Health question asking “Which region has the highest prevalence of tuberculosis according to the chart?” with a bar chart showing disease prevalence by region, the original Thinking mode correctly identifies Sub-Saharan Africa by comparing bar heights. However, when lookback is triggered at a pause phrase (“wait, let me check”), the model re-examines the chart title (“Global TB Prevalence”) instead of the data bars and incorrectly concludes that “this suggests Europe is a key region,” leading to the wrong answer.

This failure mode occurs in approximately 3-5% of cases where lookback changes the answer, particularly on questions involving charts, tables, or multi-component diagrams. The lookback controller correctly detects visual uncertainty, but our method prompts the model to re-consult the image without specifying *where* to look. When the answer is immediately visible and straightforward, this additional deliberation can introduce noise by surfacing spurious visual features or secondary image elements (titles, captions, legends) that are visually prominent but semantically irrelevant to the question.

Table 4. Comparison of phrase-based vs. online perplexity-based lookback controller.

Controller Type	Accuracy	Avg Tokens	Trigger Rate	Latency (relative)
Baseline (no lookback)	63.0	12,400	0.0	1.0×
Phrase-based	<b>65.8</b>	8,610	2.4	1.05×
Online PPL-based	66.2	8,550	2.6	1.82×
Combined (phrase + PPL)	66.3	8,720	2.7	1.85×

Table 5. Latency and throughput comparison for different decoding modes on Qwen3-VL-8B (A100 GPU, batch size 1).

Decoding Mode	Latency (s)	Throughput (q/hr)	Tokens/sec	Avg Output Tokens
Instruct	8.2	439	52.3	430
Thinking (original)	28.4	127	43.7	12400
Lookback	29.8	121	43.2	8610
Lookback+Sampling	34.2	105	41.5	9120

## 8.2. Failure Mode 2: Overthinking and Self-Doubt on Easy Recognition Tasks

On straightforward recognition or recall questions where the model’s initial instinct is correct, lookback can induce unnecessary second-guessing that leads to wrong answers. For instance, on an Art History question showing a famous painting (e.g., the Mona Lisa) and asking for the artist, the original Thinking mode immediately recognizes the painting and correctly answers Leonardo da Vinci in a short, confident trace (~50 tokens). However, when a pause phrase (“wait, let me verify”) triggers lookback, the model re-examines the image and notes “the dramatic use of shadow and realism, which are hallmarks of Caravaggio’s style,” then revises its answer to Caravaggio, which is incorrect.

This overthinking failure is particularly common in Art, History, Literature, and other knowledge-centric or recognition-based categories, where concise Instruct mode often outperforms Thinking. Our analysis shows that on questions answered correctly by both Instruct and original Thinking modes, adding lookback hurts accuracy in 2-4% of cases, with nearly all such regressions occurring in humanities and social science categories. The model’s initial recognition is typically based on holistic visual features or memorized knowledge; forcing additional visual scrutiny can surface low-level details (textures, local contrasts, specific brush strokes) that mislead the reasoning process.

## 8.3. Failure Mode 3: Inability to Correct Deep Reasoning or Perceptual Errors

Lookback is fundamentally a *decoding-level* intervention that encourages re-consultation of the image but does not enhance the model’s underlying visual perception or domain knowledge. When failures stem from fundamental misunderstandings—such as spatial reasoning errors, counting mistakes, or incorrect application of domain princi-

ples—repeated lookbacks often fail to correct the error. For example, on a Physics problem showing a pulley system and asking for the mechanical advantage, the model may incorrectly count the number of supporting ropes (concluding 3 instead of 4) due to occlusion or poor spatial reasoning. Even after two lookback triggers (“let me re-examine the pulley” and “looking at the diagram again”), the model recounts but makes the same error each time, persisting with the wrong answer.

This limitation affects approximately 15-20% of the questions that remain wrong under lookback (i.e., cases where both original Thinking and Lookback fail). These are typically the hardest items in each benchmark—questions requiring precise spatial reasoning, fine-grained object counting, complex geometric understanding, or deep domain expertise. Our error-overlap analysis (Sec. 2) shows that these “last-mile” failures are shared across models and decoding modes, suggesting that they represent fundamental capability gaps rather than routing or attention issues. Addressing such errors requires model-level improvements (e.g., better vision encoders, geometry-aware architectures, explicit spatial reasoning modules) or external tools (e.g., object detection, OCR, symbolic geometry solvers) rather than purely decoding-based interventions.