# ReaGEN: Adaptive Generation of Structured Chains-of-Thought for Efficient Multimodal Reasoning

Supplementary Material

## 6. Definition of Action Space

Table 4 defines the *stage pool* that serves as the discrete action space for ReaGEN. We factorize multimodal reasoning into four functional groups that mirror the progression of human problem solving from perception to verification, and align with modular vision–language architectures.

**Category 1: Perception & Input Understanding.** `VISUAL.OBSERVATION (VO)` (stage prompt shown in Fig. 8) detects salient objects, text, and regions in the image, creating a structured visual inventory that later stages can reference instead of repeatedly scanning raw pixels. `TEXTUAL.UNDERSTANDING (TU)` (stage prompt shown in Fig. 9) parses the question and options into main concepts, keywords, and entities, revealing the core semantics and reducing noise from surface-level phrasing.

**Category 2: Linking & Fact Extraction.** `TASK.INTERPRETATION (TI)` (stage prompt shown in Fig. 7) classifies the reasoning type (e.g., locate, count, compare, reason) and canonicalizes the question, guiding which downstream stages are most relevant and what output format is expected. `CONTEXTUAL.LINKING (CL)` (stage prompt shown in Fig. 10) grounds textual terms in visual evidence by linking entities to visual objects or regions, turning free-form language into grounded references. `FACT.EXTRACTION (FE)` (stage prompt shown in Fig. 11) collects directly observable facts and measurements into an explicit evidence buffer, so later reasoning can operate on a compact, structured fact set rather than raw descriptions. `VARIABLE.DEFINITION (VD)` (stage prompt shown in Fig. 12) defines named variables for key quantities or categorical attributes, making subsequent arithmetic and logical operations more interpretable and easier to control.

**Category 3: Reasoning & Hypothesis Handling.** `RELATIONAL.REASONING (RR)` (stage prompt shown in Fig. 13) infers structured relations (e.g., larger-than, left-of, part-of, caused-by) between entities, enabling compositional reasoning over object interactions. `QUANTITATIVE.REASONING (QR)` (stage prompt shown in Fig. 14) performs arithmetic and quantitative inference over defined variables and measurements, enabling explicit numerical computation instead of implicit "in-head" math. `LOGICAL.FILTERING (LF)` (stage prompt shown in Fig. 15) eliminates options that contradict the collected facts or relations, shrinking the hypothesis space and preventing later stages from revisiting clearly invalid answers. `HYPOTHESIS.GENERATION (HG)` (stage prompt shown in Fig. 16) proposes one or more candidate answers with brief justifications and confidence scores, making the model's uncertainty and alternative explanations explicit. `CROSSMODAL.ALIGNMENT (CA)` (stage prompt shown in Fig. 17) checks consistency between visual, textual, and numeric evidence, surfacing conflicts when modalities disagree and encouraging evidence-based resolutions. `SELFCONSISTENCY.CHECK (SC)` (stage prompt shown in Fig. 18) recomputes key values and validates earlier steps, catching internal contradictions and mitigating error propagation. `COMPARATIVE.EVALUATION (CE)` (stage prompt shown in Fig. 19) scores and ranks candidate options given all accumulated evidence, turning a set of hypotheses into a calibrated preference ordering.

**Category 4: Final Answer & Explanation.** `EXPLANATION.GENERATION (EG)` (stage prompt shown in Fig. 20) produces a concise, user-facing explanation that highlights the most relevant evidence and reasoning steps, improving transparency and debuggability. `FINAL` (stage prompt shown in Fig. 21) aggregates information from all prior stages to select the final answer and its confidence, acting as the decision head of the pipeline.

Under this view, a chain-of-thought (CoT) is an ordered sequence of stage actions that structures multi-step, visually grounded reasoning. Compared with prior multimodal CoT approaches, our stage pool provides finer-grained, typed actions tailored to modern MLLMs, improving controllability, interpretability, and reuse across tasks.

## 7. Evolution of Query-Specific CoT Structures

ReaGEN constructs a query-specific CoT dataset through a teacher-guided evolutionary search (Figure 3(a)) that discovers, for each image–query pair, an effective sequence of reasoning stages. For a candidate CoT $\tau = (s_1, \ldots, s_{|\tau|})$, we execute the frozen student VLM in a multistage manner: each stage $s_t$ is associated with a dedicated system prompt and operates on the image $I$, question $Q$, and an accumulated memory $M_t$ of all previous stage outputs, producing $y_t = f_\theta(I, Q, M_t, s_t)$ and updating $M_{t+1} = M_t \cup \{y_t\}$. The final stage consumes $M_{|\tau|+1}$ to output the answer and full intermediate traces. From this run, we extract cross-stage attention and compute stage importance scores as described in Sec. 3.2, yielding a structure-aware measure of how much each stage contributes (directly or indirectly) to the final prediction.

| Category | Stage | Goal |
|---|---|---|
| Perception & Input Understanding | VISUAL.OBSERVATION (VO) | Extract key objects and text in the image (key regions). |
| | TEXTUAL.UNDERSTANDING (TU) | Parse the question and options for main concepts and keywords. |
| Linking & Fact Extraction | TASK.INTERPRETATION (TI) | Classify the type of reasoning (locate / count / compare / other). |
| | CONTEXTUAL.LINKING (CL) | Connect textual terms with visual cues. |
| | FACT.EXTRACTION (FE) | Extract measurable factual information. |
| | VARIABLE.DEFINITION (VD) | Define symbols / variables used in reasoning. |
| Reasoning Handling | RELATIONAL.REASONING (RR) | Infer logical or spatial relations between entities. |
| | QUANTITATIVE.REASONING (QR) | Perform numerical calculations or quantitative inference. |
| | LOGICAL.FILTERING (LF) | Eliminate inconsistent or irrelevant options. |
| | HYPOTHESIS.GENERATION (HG) | Generate plausible answer hypotheses. |
| | CROSSMODAL.ALIGNMENT (CA) | Ensure consistency across visual, textual, and numeric evidence. |
| | SELFCONSISTENCY.CHECK (SC) | Verify the internal consistency of the reasoning process. |
| | COMPARATIVE.EVALUATION (CE) | Score and rank candidate options based on evidence. |
| Final Answer & Explanation | EXPLANATION.GENERATION (EG) | Generate a concise explanation summarizing the reasoning. |
| | FINAL | Integrate prior reasoning to produce the final answer. |

Table 4. Stage pool used in ReaGEN. Each stage corresponds to a modular reasoning primitive with a clear goal and I/O semantics.

**Reward Calculation for a CoT.** To compare different CoTs for the same sample, we assign each candidate a scalar reward that balances prediction quality, compactness, and structural diversity. Let $\tau = (s_1, \ldots, s_{|\tau|})$ denote a candidate CoT. Given its teacher prediction score $s(\tau) \in [0, 1]$, length $|\tau|$, and empirical frequency $n(\tau)$ among all previously evaluated CoTs, we define

$$R(\tau) \;=\; \alpha\, s(\tau) \;-\; \beta\, \ell(\tau) \;-\; \gamma\, d(\tau), \tag{10}$$

where $\alpha, \beta, \gamma > 0$ are scalar weights and

$$\ell(\tau) = \frac{|\tau|}{L_{\max}}, \qquad \text{(normalized length penalty)}, \tag{11}$$

$$d(\tau) = \frac{\log\big(1 + n(\tau)\big)}{\log\big(1 + n_{\max}\big)}, \qquad \text{(normalized diversity penalty)}. \tag{12}$$

Here $L_{\max}$ is a fixed upper bound on CoT length, $n(\tau)$ is the number of times $\tau$ has appeared in the search history, and $n_{\max} = \max_{\tau'} n(\tau')$ is the maximum frequency across all chains seen so far. The term $s(\tau)$ encourages CoTs that yield accurate predictions, $\ell(\tau)$ penalizes longer chains and therefore favors more compact reasoning, and $d(\tau)$ grows with how often a CoT has been reused, so the search is biased toward less frequent (more diverse) CoT structures.

In practice, we modulate the length and diversity penalties based on prediction quality. For perfectly correct chains with $s(\tau) = 1$, we turn off the diversity term (i.e., set $\gamma = 0$), so the reward compares only accuracy and length, preferring shorter CoTs among equally correct candidates. For completely incorrect chains with $s(\tau) = 0$, we downweight the length penalty (effectively using a smaller $\beta$), so the search focuses more on exploring diverse CoT structures via $d(\tau)$ instead of aggressively favoring short but uninformative failures.

## 8. Generator Training Architecture.

We instantiate the generator as a compact transformer-based encoder-decoder, which has only 18.27M parameters. First, both image and question embeddings are projected into this space via a linear–LayerNorm–GELU block. The question sequence is further encoded by a 2-layer Transformer encoder, and a cross-modal encoder then conditions question tokens on image tokens. The resulting joint sequence is pooled into a small set of memory tokens via a token-pooling module, producing a compact multimodal memory. In parallel, the attention matrix $A$ is flattened and projected into a "reasoning token" embedding, and stage tokens (obtained from a learnable stage embedding table) are adjusted by an $A$-guided aggregation that propagates attention patterns over the current stage sequence. Depending on configuration, the final memory passed to the decoder is formed by concatenating the multimodal memory, the projected structural token, and/or the $A$-guided stage context.

On top of this memory, a 2-layer Transformer decoder with causal masking takes the (teacher-forced) stage-ID embeddings as input and produces contextualized decoder states. We attach two heads: a stage classifier that predicts a distribution over $\mathcal{V} \cup \{\text{EOS}\}$ at each position, and a length head that predicts a distribution over possible CoT lengths from the pooled memory representation. The detailed summary of layers and dimensionalities in GEN is shown in Table 5.

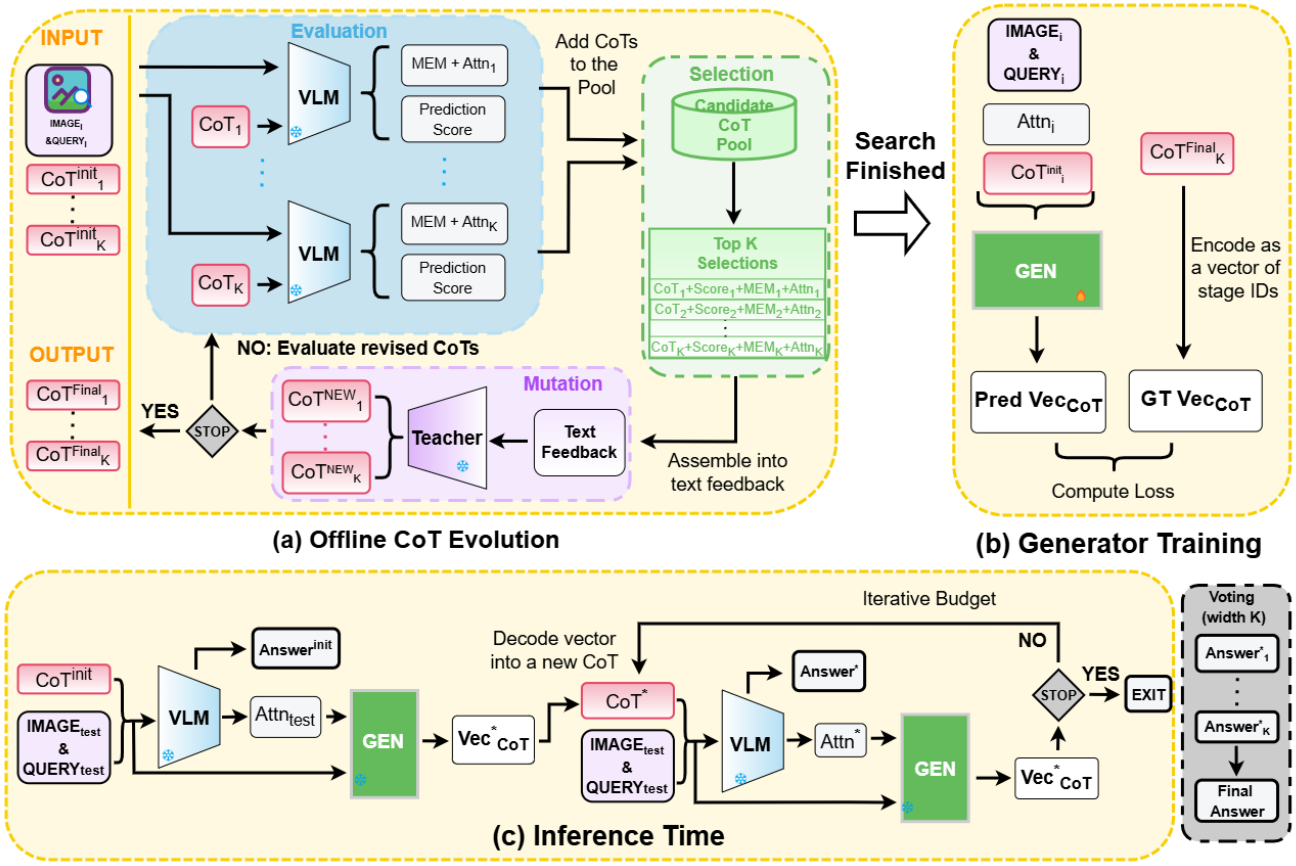



Figure 3. **(a) Offline CoT Evolution.** We generate training data through a teacher-guided search. Randomly initialized CoTs are first evaluated by the VLM to obtain prediction scores, memory outputs (MEM), and attention traces (Attn). The top-$K$ candidates are converted into text feedback, and the teacher rewrites them into improved CoTs. These rewritten CoTs are sent back to the VLM for re-evaluation, and the evolution loop continues until the correct answer is reached or the search budget is exhausted. **(b) Generator Training.** A lightweight generator (GEN) is trained on search data to map $(I, Q, \text{Attn}, \text{CoT}^{\text{init}}) \rightarrow \text{CoT}^{\text{final}}$, by minimizing the loss between the predicted and search-derived stage-ID vectors. **(c) Inference Time.** GEN predicts a sample-adaptive $\text{CoT}^{*}$ (single pass or few-step refinement) by conditioning on the initial attention traces and $\text{CoT}^{\text{init}}$, identifying the most impactful stages from the initial observations. The frozen VLM then follows this $\text{CoT}^{*}$ to produce the answer, and we use majority voting with width-$K$ (i.e., $K$ parallel runs) to report the final result.

| Module | Type | Key dims / params | Input → Output |
|---|---|---|---|
| Input projection | Linear + LayerNorm + GELU | $d_{\text{emb}} = 3584,\ d_{\text{hid}} = 512$ | $(B, T, 3584) \rightarrow (B, T, 512)$ |
| Question encoder | TransformerEncoder (2 layers) | $d_{\text{model}} = 512,\ n_{\text{head}} = 8,\ L = 2$ | $(B, T, 512) \rightarrow (B, T, 512)$ |
| Cross-modal encoder | CrossModalEncoder | $d_{\text{model}} = 512,\ n_{\text{head}} = 8$ | $(B, \cdot, 512) \rightarrow (B, \cdot, 512)$ |
| Token pooling | TokenPool | $d_{\text{model}} = 512,\ n_{\text{head}} = 8,\ \text{mean\_tok\_len} = 2$ | $(B, \cdot, 512) \rightarrow (B, \text{mean\_tok\_len}, 512)$ |
| Stage embedding | Embedding | $\text{num\_classes} = 15,\ d_{\text{hid}} = 512$ | $(B, \cdot) \rightarrow (B, \cdot, 512)$ |
| Reasoning adjacency proj | Linear + LayerNorm + GELU | $14 \times 15 = 210 \rightarrow 512$ | $(B, 210) \rightarrow (B, 512)$ |
| Decoder pos. emb. | Embedding | $\text{max\_cot\_len} = 14,\ d_{\text{hid}} = 512$ | $(B, L) \rightarrow (B, L, 512)$ |
| Memory pos. emb. | Embedding | $\text{mean\_tok\_len} + 2 = 4,\ d_{\text{hid}} = 512$ | $(B, L') \rightarrow (B, L', 512)$ |
| Decoder | TransformerDecoder (2 layers) | $d_{\text{model}} = 512,\ n_{\text{head}} = 8,\ L = 2$ | $(\text{tgt}, \text{mem}) : (B, L, 512) \rightarrow (B, L, 512)$ |
| Stage head | Linear | $512 \rightarrow \text{num\_classes} = 15$ | $(B, L, 512) \rightarrow (B, L, 15)$ |
| Length head | Linear | $512 \rightarrow \text{max\_cot\_len} = 14$ | $(B, L, 512) \rightarrow (B, L, 14)$ |
| Transition bias | Linear → ReLU → Linear | $15 \rightarrow 128 \rightarrow 15$ | $(B, 15) \rightarrow (B, 15)$ |

Table 5. Summary of layers and dimensionalities in GEN. Here $d_{\text{emb}} = 3584$ is the input embedding size, $d_{\text{hid}} = 512$ is the shared hidden size, $\text{num\_stages} = 14$, and $\text{num\_classes} = \text{num\_stages} + 1 = 15$.

**Generator Training objective.** Let $\hat{\mathbf{c}}$ denote the predicted stage sequence and $\hat{L}$ the predicted length. As illustrated in Figure 3(b), GEN is trained to map image, question embeddings, attention signals, and $\text{CoT}^{\text{init}}$ into a compact discrete $\text{CoT}^{\text{final}}$ representation. We train GEN with a combination of stage-level and length-level cross-entropy losses:

$$\mathcal{L} = \mathcal{L}_{\text{stage}} + \lambda_{\text{len}} \mathcal{L}_{\text{len}}, \tag{13}$$

where $\mathcal{L}_{\text{stage}}$ is the token-wise cross-entropy between the predicted logits and the ground-truth stage IDs (including EOS), and $\mathcal{L}_{\text{len}}$ is the cross-entropy between the predicted length logits and the true CoT length. In this way, the generator learns to map image–question embeddings and structural signals into a compact, discrete CoT representation, enabling fast, search-free CoT prediction at test time.

## 9. Definition of Hyper-parameters

Table 6 summarizes the main hyper-parameters used in the ReaGEN pipeline, grouped into search, GEN training, and inference. For the teacher-guided evolutionary search, we specify the reward weights for accuracy, length, and diversity $(\alpha, \beta, \gamma)$, the maximum CoT length $L_{\max}$, the stage-importance discount $\lambda$, the numerical stabilizer $\varepsilon$, and the search width–depth budget via the number of branches $K_{\text{search}}$ and iterations $\text{Iter}_{\text{search}}$, together with decoding temperatures and token limits for both the frozen student VLM and the teacher model. For GEN training, we list the optimization hyper-parameters, including learning rate, batch size, optimizer (AdamW), weight decay, the total number of training epochs $E$, and the coefficient $\lambda_{\texttt{len}}$ used to balance the auxiliary predicted-length loss. At inference time, we mirror the search budget with a smaller number of branches $K_{\text{Inference}}$ and iterations $\text{Iter}_{\text{Inference}}$, and reuse the same decoding temperature and token cap for the student VLM to generate stage-wise outputs. Architectural dimensions of GEN (e.g., embedding sizes and transformer depth) are reported separately in Table 5.

| **Hyper-parameter** | **Symbol / Value** |
|---|---|
| **Search (Evolutionary CoT discovery)** | |
| Reward accuracy weight | $\alpha = 1.0$ |
| Reward length weight | $\beta = 0.1$ |
| Reward diversity weight | $\gamma = 0.2$ |
| Max CoT length (search) | $L_{\max} = 14$ |
| Stage importance discount | $\lambda = 0.7$ |
| Numerical stability constant | $\varepsilon = 10^{-4}$ |
| Number of branches (width) | $K_{\text{search}} \in [1, 5]$ |
| Search Iterations (depth) | $\text{Iter}_{\text{search}} = 20$ |
| Temperature for student VLM | $T_{\text{stu}} = 0.0$ |
| Max tokens for student VLM | $L_{\text{tok,stu}} = 256$ |
| Temperature for teacher model | $T_{\text{tea}} = 0.6$ |
| Max tokens for teacher model | $L_{\text{tok,tea}} = 1024$ |
| **GEN training** | |
| Learning rate | $\eta = 1 \times 10^{-4}$ |
| Batch size | $B = 64$ |
| Optimizer | `AdamW` |
| Weight decay | $\lambda_{\text{wd}} = 0.001$ |
| Training epochs | $E = 200$ |
| Predicted Length Loss decay | $\lambda_{\texttt{len}} = 0.3$ |
| **Inference** | |
| Number of branches (width) | $K_{\text{Inference}} \in [1, 5]$ |
| Inference Iterations (depth) | $\text{Iter}_{\text{Inference}} = 4$ |
| Temperature for student VLM | $T_{\text{stu}} = 0.0$ |
| Max tokens for student VLM | $L_{\text{tok,stu}} = 256$ |

Table 6. Hyper-parameters used in the ReaGEN pipeline, grouped by search, GEN training, and inference.

| **Dataset** | **# Chains** | **Mean CoT length** |
|---|---|---|
| MATHVISION | 6009 | 3.05 |
| MATHVERSE | 2927 | 2.01 |
| MMMU | 2615 | 1.89 |
| MMSTAR | 3942 | 1.97 |

Table 7. Statistics of searched CoT candidates per benchmark. For each dataset, we report the number of retained chains and their average stage length.

## 10. Properties of the Searched CoT Dataset

Figure 4 and Figure 5 analyze the searched CoTs that constitute the training data for GEN. Figure 4 projects each discovered CoT into a bag-of-stages embedding and visualizes the first two principal components via PCA. Points from different benchmarks (MathVision, MathVerse, MMMU, and MMStar) form overlapping yet partially separated clusters, indicating both cross-dataset sharing of common reasoning patterns and substantial dataset-specific structure. This suggests that the search process does not collapse into a single template, but instead uncovers a diverse family of CoTs adapted to different tasks. Figure 5 summarizes how often each stage appears across all collected CoTs. The distribution is clearly non-uniform: perception and grounding stages such as `VO`, `TI`, and `TU` are heavily used, as are higher-level summarization and decision stages like `EG` and `CE`, while more specialized modules (e.g., `CA`, `LF`, `SC`, `HG`) appear less frequently but are still present. Together, these plots show that the searched dataset covers a broad, heterogeneous space of reasoning structures, with a core set of commonly reused stages and a long tail of more specialized reasoning skills.

Table 7 summarizes additional statistics about searched CoTs across four benchmarks. The number of collected chains for each dataset exceeds the number of underlying samples because the teacher-guided search can discover

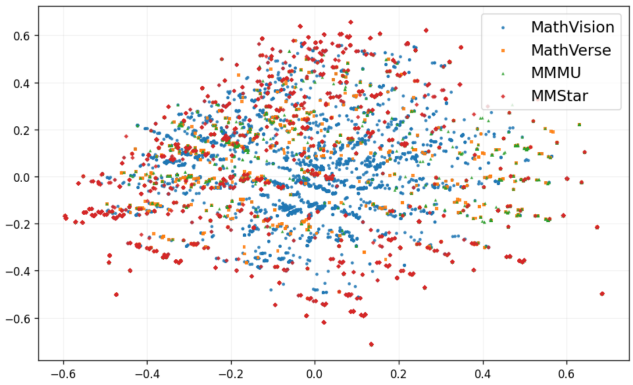



Figure 4. **CoT diversity via PCA (bag-of-stages encoding).** Each point is a searched Chain-of-Thought embedded as a bag-of-stages vector and projected to the first two principal components (PC1, PC2). Colors/markers denote datasets: ● **MathVision**, ■ **MathVerse**, ▲ **MMMU**, ♦ **MMStar**. The spread and partial clustering indicate substantial structural variety across tasks while retaining cross-dataset overlap, evidencing broad CoT diversity.

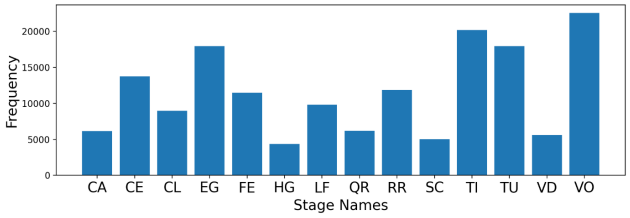



Figure 5. **Stage frequency across searched CoTs.** Bars show how often each stage appears among the collected chains, indicating non-uniform but broad usage. Abbreviations: **CA** = Cross-modal Alignment, **CE** = Comparative Evaluation, **CL** = Contextual Linking, **EG** = Explanation Generation, **FE** = Fact Extraction, **HG** = Hypothesis Generation, **LF** = Logical Filtering, **QR** = Quantitative Reasoning, **RR** = Relational Reasoning, **SC** = Self-Consistency Check, **TI** = Task Interpretation, **TU** = Textual Understanding, **VD** = Variable Definition, **VO** = Visual Observation.

multiple high-reward CoTs for a single image–question pair, and we retain multiple valid reasoning paths whenever available. As a result, the training corpus forms a many-to-one augmentation of the original benchmarks, with several alternative reasoning structures per example. The mean CoT lengths (ranging from roughly 1.9 to 3.1 stages) further indicate that, despite this structural diversity, the discovered CoTs remain relatively short and focused, making the multi-stage calls to the student VLM computationally efficient.

## 11. Ablation Studies

**Ablation on Base Model Size.** Table 8 reports an ablation on the underlying VLM size for ReaGEN on an evenly categorized subset of MATHVISION with GEN trained on the compositie dataset from MMMU and MATHVERSE. We compare direct answering with Qwen2.5-VL-7B, Qwen3-VL-4B, and Qwen3-VL-8B against the same models augmented with our reasoning framework (+ReaGEN). Across all three backbones, ReaGEN consistently yields substantial absolute accuracy gains (e.g., from 16.67% to 21.88% for Qwen2.5-VL-7B, and from 29.90% to 47.42% for Qwen3-VL-8B), demonstrating that structured stage-wise reasoning provides complementary benefits beyond scaling the base model alone. The improvement is especially pronounced for the stronger Qwen3 variants, suggesting that more capable VLMs can better exploit the search-derived CoTs generated by ReaGEN.

| Method | Mathvision |
|---|---|
| **Qwen2.5-VL-7B** | |
| Direct Answer | 16.67% |
| +ReaGEN | 21.88% |
| **Qwen3-VL-4B** | |
| Direct Answer | 24.74% |
| +ReaGEN | 43.30% |
| **Qwen3-VL-8B** | |
| Direct Answer | 29.90% |
| +ReaGEN | 47.42% |

Table 8. **Ablation on base model size.**: Accuracy (%) of Qwen2.5-VL-7B, Qwen3-VL-4B, and Qwen3-VL-8B with direct answering vs. with the proposed ReaGEN reasoning framework (+ReaGEN) on a subset of Mathvision.

| Method | M10 | M4 | VStar | MMStar | MVision | MVerse |
|---|---|---|---|---|---|---|
| ReaGEN | 52.54 | 64.51 | 84.49 | 75.77 | 44.60 | 47.59 |
| w/o Attn | 48.03 | 63.47 | 77.01 | 74.03 | 38.72 | 45.67 |
| **Drop (↓)** | **4.51** | **1.04** | **7.48** | **1.74** | **5.88** | **1.92** |

Table 9. **Ablation: Effect of removing attention guidance.** M10 = MMMU-Pro (10-option), M4 = MMMU-Pro (4-option), MVision = MathVision, MVerse = MathVerse(only vision). Removing attention signals consistently lowers performance across all benchmarks.

**Ablation on $K$ and $A$** Figure 6 analyzes the effect of search hyperparameters on the same balanced MATHVISION subset under the same student VLM and teacher model, focusing on branch width $K$ and the use of attention-guided stage importance. The left panel shows that, with attention enabled, increasing $K$ from 2 to 5 steadily improves final accuracy and accelerates convergence, as wider search explores more CoT candidates in parallel while still being

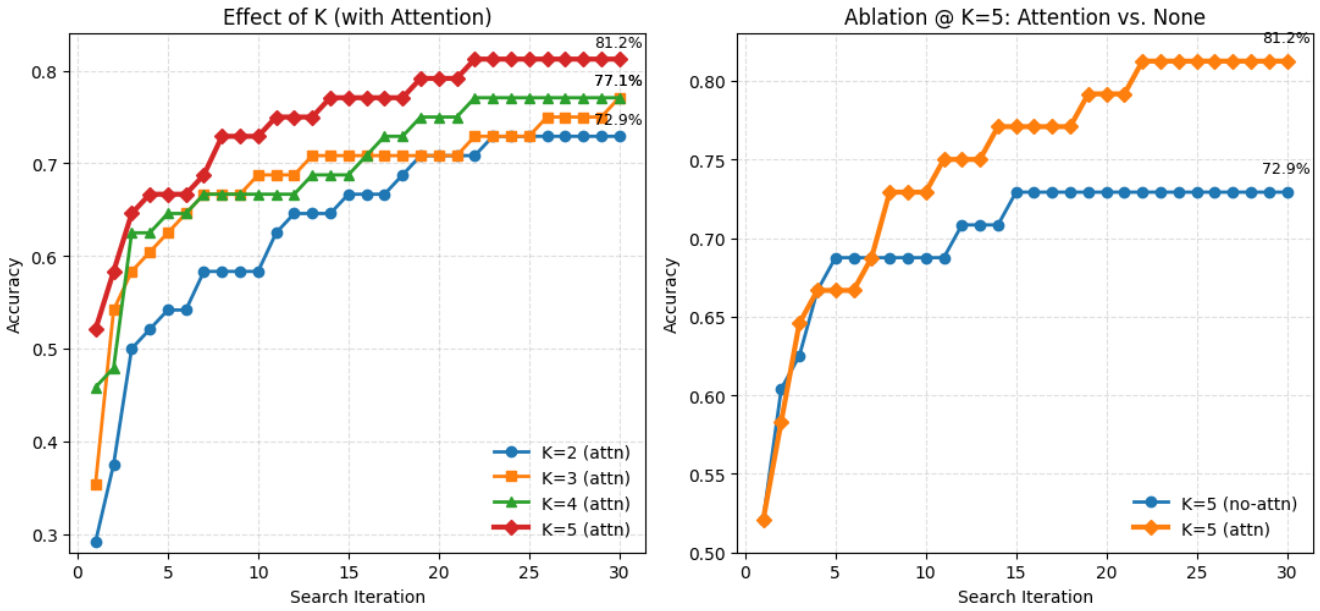



Figure 6. **Effect of branch width $K$ and attention on search performance.** *Left:* With attention guidance, increasing the parallel width from $K{=}2$ to $K{=}5$ steadily improves accuracy and speeds convergence across search iterations (final accuracies $\approx$73%, 77%, 77%, and 81% for $K{=}2, 3, 4, 5$). *Right:* Ablation at $K{=}5$: using attention markedly outperforms the no-attention variant throughout the search and reaches a higher final accuracy ($\sim$81% vs. $\sim$73%), showing that attention signals are crucial for effective CoT search.

| **Ablation setting** | **Accuracy** |
|---|---|
| Optimal CoT (no ablation) | 100.00% |
| Remove least important stage | 72.50% |
| Remove top-2 least important stage | 66.25% |
| Remove most important stage | 47.92% |
| Remove top-2 most important stage | 45.84% |

Table 10. **Stage-importance ablation on MATHVISION.** Accuracy (%) of Qwen3-VL-4B+REAGEN on a subset where the original CoT (length 3–7) answers correctly. Removing stages with higher importance scores causes substantially larger performance drops than removing low-importance stages, validating the proposed stage-importance metric.

effectively guided by the reward. The right panel fixes $K = 5$ and compares variants with and without attention-based guidance: the attention-aware search dominates the no-attention baseline throughout training and reaches a higher final accuracy (about 81% vs. 73%). Together, these results highlight that ReaGEN's gains arise not only from the staged reasoning interface but also from an effective search strategy that leverages cross-stage attention to prioritize promising CoT structures.

To further evaluate the effectiveness of our stage-importance scores, we perform a stage-ablation study on the searched CoTs. We select a subset of MATHVISION examples where Qwen3-VL-4B+REAGEN produces correct answers with CoTs of length 3–7, so the original chains achieve 100% accuracy. For each CoT, we rank its stages by importance $\mathrm{Imp}(i)$ and construct new chains by deleting a single stage or a pair of stages: the most important stage, the top-2 most important stages, the least important stage, or the top-2 least important stages. As shown in Table 10, removing highly important stages leads to a drastic drop in accuracy (from 100% to about 46–48%), whereas removing the least important stages still hurts performance but much less severely (to about 66–73%). This monotonic relationship between predicted importance and performance degradation indicates that our stage-importance metric is meaningful and that high-scoring stages are genuinely critical for successful reasoning.

## 12. Prompt Overview

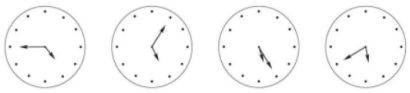
Figure 23 and Figure 24 illustrate the basic variant of our prompt design without importance signals, where the *Teacher System Prompt (w/o Importance Signal)* (Figure 23) specifies the model's role as a Vision–Language Pipeline Mutator that must propose a single, concise and novel CoT based on the question, image(s), past and current CoTs, and their outputs and feedback, while the *Teacher Input User Content (w/o Importance Signal)* (Figure 24) provides a concrete instantiation of these inputs, including a worked example and explored chains. In contrast, Figure 25 and Figure 26 show the extended variant with importance signals: the *Teacher System Prompt (with Importance Sig-*

*nal)* (Figure 25) augments the specification with per-stage Direct/Indirect/Total influence metrics and an editing strategy over `add`/`delete`/`swap` operations, requiring the system to decide whether the current CoT is already optimal or to output exactly one improved chain; the *Teacher Input User Content (with Importance Signal)* (Figure 26) then supplies the corresponding enriched input, including per-stage influence values, and instructs the model to exploit these signals to detect weak, redundant, or misplaced stages and refine the CoT accordingly, under strict JSON, stage-pool, uniqueness, and final-`FINAL` constraints.

## 13. Examples

In this section, we present examples where ReaGEN corrects an incorrect direct VLM answer through structured reasoning stages.

**Example: Clock Reasoning**

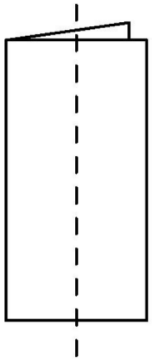


**Question.** In this picture there is what I saw on four different clocks at the same time. Only one of them had the right time. One was 20 minutes fast. Another 20 minutes slow. One had stopped some time ago. What was the right time?

**Options:** A: 4:45, B: 5:05, C: 5:25, D: 5:40, E: 12:00.
**Direct Answer**: **C: 5:25**. ×

**ReaGEN Output:**
**Stage 1 (Task Interpretation):** Reason-type question. The task is to determine the correct time from four clocks, where one clock is accurate, one is 20 minutes fast, one is 20 minutes slow, and one has stopped.
**Stage 2 (Textual Understanding):** The main concept is determining the correct time from clocks. Key terms include *clock*, *time*, *accurate*, *fast*, *slow*, and *stopped*. The question is a reasoning task with candidate options 4:45, 5:05, 5:25, 5:40, and 12:00.
**Stage 2 (Hypothesis Generation):** The main hypotheses are 5:05, 5:25, and 5:40. The strongest hypothesis is 5:05: clock 2 shows 5:05 and is likely the accurate one, while the remaining clocks can be explained by the $+20$ minute, $-20$ minute, and stopped-clock conditions. Alternative hypotheses 5:25 and 5:40 are less consistent with all constraints. The confidence scores are 0.85, 0.75, and 0.65.
**Final Answer: B: 5:05**. ✓

**Example: Folded Paper Perimeter Ratio**



**Question:** A square piece of paper, 4 inches on a side, is folded in half vertically. Both layers are then cut in half parallel to the fold. Three new rectangles are formed, a large one and two small ones. What is the ratio of the perimeter of one of the small rectangles to the perimeter of the large rectangle?

**Options:** A: $\frac{1}{3}$, B: $\frac{1}{2}$, C: $\frac{3}{4}$, D: $\frac{4}{5}$, E: $\frac{5}{6}$.
**Direct Answer**: **B:** $\frac{1}{2}$. ×

**ReaGEN Output:**
**Stage 1 (Visual Observation):** The image shows a square sheet of paper folded vertically, with a dashed line indicating the fold and cut lines parallel to it. The resulting pieces are one large rectangle and two smaller rectangles.
**Stage 2 (Textual Understanding):** The main concept is the ratio of perimeters. The square has side length 4 inches. Folding vertically halves the width, so the folded shape has width 2 inches and height 4 inches. Cutting both layers in half parallel to the fold creates two small rectangles of width 1 inch and one large rectangle of width 2 inches.
**Stage 3 (Cross-Model Alignment):** The visual and text are consistent: the fold creates a $2 \times 4$ rectangle, and the cuts split it into one $2 \times 4$ large rectangle and two $1 \times 4$ small rectangles. The perimeter of one small rectangle is

$$2(1+4) = 10,$$

and the perimeter of the large rectangle is

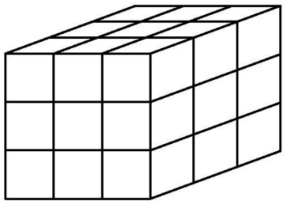
$$2(2+4) = 12.$$

Thus, the required ratio is

$$\frac{10}{12} = \frac{5}{6},$$

which matches option E.
**Final Answer: E:** $\frac{5}{6}$. ✓

**Example: Cube Surface Area**



**Question:** Each corner cube is removed from this 3 cm $\times$ 3 cm $\times$ 3 cm cube. The surface area of the remaining figure is

**Options:** A: 19 sq.cm, B: 24 sq.cm, C: 30 sq.cm, D: 54 sq.cm, E: 72 sq.cm.
**Direct Answer: E:** 72 **sq.cm**. $\times$

**ReaGEN Output:**
**Stage 1 (Task Interpretation):** Reason-type question. The task is to calculate the surface area of a $3 \times 3 \times 3$ cube after removing the corner cubes. The expected output is a multiple-choice selection.
**Stage 2 (Comparative Evaluation):** Start from the original surface area of the cube: $6 \times 3^2 = 54$ sq.cm. Removing a corner unit cube removes 3 unit squares from the outside surface, but also exposes 3 new unit squares on the adjacent inner faces, so the net surface-area change is 0 for each removed corner cube. Therefore, the remaining figure still has surface area 54 sq.cm, which matches option D.
**Final Answer: D:** 54 **sq.cm**. ✓

**Example: Counting Women in an Image**

**Question:** How many women are present in the image?

**Options:** A: 0, B: 1, C: 2, D: 3.
**Direct Answer: B:** 1. $\times$

**ReaGEN Output:**
**Stage 1 (Task Interpretation):** Count-type question. The task is to determine how many women are visible in the image. The expected output is a multiple-choice selection.
**Stage 2 (Contextual Linking):** The concept of *women* is linked to two distinct people in the image: a woman in a white shirt and skirt holding a wedding dress, and a woman in bed in the left background. These links are consistent with the visual scene.
**Final Answer: C:** 2. ✓

**TASK.INTERPRETATION (TI)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- output exactly one json object; no prose/markdown/code fences.
- lowercase keys/enums; trim strings; no extra keys.
- use only visible data and `<mem.*>` context if available.
**Task:** interpret the question to determine its reasoning type and rephrase it concisely.
**Schema:**

```
{
  "task_type": "<classify|count|compare|locate|reason|other>",
  "rephrased_question": "<str>",
  "expected_output_type": "<text|numeric|boolean|choice>"
}
```

Figure 7. System Prompt for TI stage

**VISUAL.OBSERVATION (VO)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- output exactly one json object; lowercase keys; no prose/markdown/fences.
- $\leq$ 8 items per list; concise descriptions only.
**Task:** identify and list key visual elements present in the image.
**Schema:**
{
"objects": ["<str>", ...], "relations": ["<str>", ...],
"text_in_image": ["<str>", ...], "key_regions": ["<str>", ...]
}

Figure 8. System Prompt for VO stage

**TEXTUAL.UNDERSTANDING (TU)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- extract core nouns and verbs.
- $\leq$ 5 keywords; $\leq$ 4 option_entities.
**Task:** analyze the question and options to extract main textual concepts.
**Schema:**
{
"main_concept": "<str>", "keywords": ["<str>", ...],
"question_type": "<classify|compare|count|reason|locate>",
"option_entities": ["<str>", ...]
}

Figure 9. System Prompt for TU stage

**CONTEXTUAL.LINKING (CL)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- link_confidence $\in$ [0,1].
- $\leq$ 6 strong links; omit weak or uncertain matches.
**Task:** link textual entities from the question with visual or textual evidence in the image.
**Schema:**
{
"linked_concepts": [["<term>", "<visual_obj>"], ...],
"unlinked_terms": ["<str>", ...], "link_confidence": <0-1>
}

Figure 10. System Prompt for CL stage

**FACT.EXTRACTION (FE)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- use only direct evidence from the image or question.
- $\leq 6$ concise facts or measurements.
**Task:** extract factual or measurable information directly visible or stated in the question or image.
**Schema:**
{
"facts": ["<str>", ...],
"measurements": [["<quantity>", "<unit>"], ...],
"labels": ["<str>", ...]
}

Figure 11. System Prompt for FE stage

**VARIABLE.DEFINITION (VD)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- $\leq 6$ variables; concise names.
- use numeric value if known, otherwise `null`.
**Task:** define variables representing measurable, categorical, or logical entities relevant to reasoning.
**Schema:**
{
"variables":
[{"name": "<str>", "meaning": "<str>", "value": "<num>", "unit": "<str>"}, ...]
}

Figure 12. System Prompt for VD stage

**RELATIONAL.REASONING (RR)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- example relations: `larger_than`, `left_of`, `part_of`, `caused_by`.
- evidence $\leq 3$ concise phrases.
**Task:** infer logical, comparative, or spatial relations between identified entities.
**Schema:**
{
"relations":
[{"subject": "<str>", "relation": "<str>","object": "<str>"},...],
"supporting_evidence": ["<str>", ...]
}

Figure 13. System Prompt for RR stage

**QUANTITATIVE.REASONING (QR)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- show explicit formulas; $\leq$ 4 equations.
- all quantities must be visible or deducible.
**Task:** perform numerical reasoning using visible data or derived relationships.
**Schema:**
{
"equations": ["<str>", ...],
"derived_values": [["<var>", "<value>", "<unit>"], ...],
"final_numeric": "<num|null>"
}

Figure 14. System Prompt for QR stage

**LOGICAL.FILTERING (LF)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- rationale $\leq$ 25 words.
- keep at least one remaining option.
**Task:** prune answer options inconsistent with observed facts or logic.
**Schema:**
{
"eliminated_options": ["<str>", ...],
"remaining_options": ["<str>", ...],
"rationale": "<str>"
}

Figure 15. System Prompt for LF stage

**HYPOTHESIS.GENERATION (HG)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- 1–3 hypotheses.
- confidence $\in$ [0,1]; justification $\leq$ 20 words each.
**Task:** generate plausible answer hypotheses with short justifications and confidence estimates.
**Schema:**
{
"hypotheses": ["<str>", ...],
"justifications": ["<str>", ...],
"confidence": [<float>, ...]
}

Figure 16. System Prompt for HG stage

**CROSSMODAL.ALIGNMENT (CA)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- `alignment_score` $\in$ [0,1].
- $\leq 3$ conflicts; describe how inconsistencies were resolved.
**Task:** ensure consistency between textual, visual, and quantitative evidence.
**Schema:**
{
"alignment_score": `<float>`,
"conflicts": ["`<str>`", ...],
"resolved_interpretation": "`<str>`"
}

Figure 17. System Prompt for CA stage

**SELFCONSISTENCY.CHECK (SC)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- `agreement_score` $\in$ [0,1].
- `consistent` if $\geq 0.8$, `inconsistent` if $\leq 0.4$.
**Task:** verify internal consistency across previous reasoning outputs.
**Schema:**
{
"recomputed_values": [["`<var>`", "`<value>`" ], ...],
"agreement_score": `<float>`,
"validation_status": "`<consistent|inconsistent|uncertain>`"
}

Figure 18. System Prompt for SC stage

**COMPARATIVE.EVALUATION (CE)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- all scores $\in$ [0,1].
- ranking must match descending order of scores.
**Task:** evaluate and rank all answer options based on reasoning confidence and supporting evidence.
**Schema:**
{
"option_scores": {"`<option>`": `<float>`},
"ranking": ["`<option>`", ...],
"selection_reasoning": "`<str>`"
}

Figure 19. System Prompt for CE stage

**EXPLANATION.GENERATION (EG)**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- rationale $\leq$ 30 words.
- $\leq$ 5 evidence items; factual, not speculative.
**Task:** generate a concise, factual explanation summarizing how the final answer was derived.
**Schema:**
{
"rationale": "<str>",
"evidence": ["<str>", ...],
"reasoning_summary": "<str>"
}

Figure 20. System Prompt for EG stage

**FINAL**
You are one reasoning stage in a modular vision-language reasoning pipeline.
**Rules:**
- confidence $\in$ [0,1].
- $\leq$ 3 supporting_stages.
- if multiple choice, final_answer must be the option letter <A|B|C|D|...>.
- otherwise, final_answer is the free-form answer string.
**Task:** integrate reasoning results from prior stages to produce the most probable final answer.
**Schema:**
{
"final_answer": "<str>",
"confidence": <float>,
"supporting_stages": ["<str>", ...]
}

Figure 21. System Prompt for FINAL stage

**DIRECT ANSWER**
You are a single-stage module in a multimodal reasoning pipeline.
**Rules:**
- output exactly one json object; no prose/markdown/code fences.
- lowercase keys/enums; trim whitespace; no extra keys.
- rely solely on provided images, question, and options if given.
- if multiple choice, output only the option letter <A|B|C|D|...>.
- otherwise, output the free-form final answer string.
**Task:** directly answer the question concisely.
**Schema:**
{
"answer": "<str>"
}

Figure 22. System Prompt for Direct Answering

**Teacher System Prompt (w/o Importance Signal)**

**Role.** You are a Vision–Language Pipeline Evaluator.

**Output rule.** Emit *only* the final JSON per schema—no commentary.

**Inputs.**

- Question and image(s).
- Past CoT iterations (ordered stage lists) with their scores.
- Current CoT (ordered stages) with per-stage outputs and feedback.
- Stage pool (valid names only; len=14):
  `['TASK.INTERPRETATION','VISUAL.OBSERVATION', ...]`
- Predicted answer and ground-truth answer.

**Goal.** Propose *exactly one* new CoT that is different from all past CoTs.

**Required procedure.**

1. Review prior iterations: note stage sequences and outcomes.
2. Review the current chain and outputs.
3. Identify essential intermediates; mark redundant stages (unused/uncited/no evidence link).
4. Consult the stage pool to add missing capability or improve ordering.
5. Ensure the proposed chain is not identical to any past chain.

**Decision policy.**

**A) If `predicted == ground_truth`:**
If a shorter or equally effective pipeline exists after pruning redundancies and it is novel, propose it.

**B) If `predicted != ground_truth`:**
Think silently; propose *one* improved, meaningfully different CoT likely to fix the error.

**Hard constraints.**

- Use only valid stage names from the pool above.
- The last stage token must be "`FINAL`".
- No duplicate stages inside "`cot`".
- The proposed "`cot`" must not exactly match any past chain.
- Keep pipelines concise and high-impact.

**Output (strict JSON only; no text before/after).**

```
{
  "reason":"≤2 sentences",
  "cot":["STAGE_1","STAGE_2",...,"FINAL"]
}
```

Figure 23. System Prompt for Teacher (w/o Importance Signals)

**Teacher Input User Content (w/o Importance Signal)**

**You are given:** the question, associated image(s), previous CoTs with scores, the current CoT (its stages and outputs), stage-level feedback, the predicted answer, and the ground-truth answer. Return *only* a JSON object that follows the schema (no reasoning text, no commentary, no <think>...</think>).

**Procedure (strict).**

1. Inspect the outputs from earlier iterations (if they exist).
2. Inspect the current iteration output.
3. Determine which intermediate outputs are essential and mark any stages that are unnecessary.
4. Examine the stage pool for missing capabilities or a more effective ordering.
5. Ensure the proposed stage sequence is new (not exactly the same as any previous chain).

**Policy.**

- If `predicted == ground_truth`: prioritize efficiency; when a shorter or equally strong chain is possible, propose that alternative.
- If `predicted != ground_truth`: do not output internal thoughts; provide *one* revised chain that is meaningfully different and aimed at correcting the mistake.

**Constraints.**

- Use only valid stages (total length $= 14$):
  `['TASK.INTERPRETATION','VISUAL.OBSERVATION', ...]`
- The stage `"FINAL"` must appear last; no stage name may occur more than once in `"cot"`.
- The new CoT sequence must differ from all past sequences (no exact matches).
- Keep the pipeline as short as possible while remaining effective.

**Output format (JSON only; no extra text).**

```
{
  "reason":"at most 2 sentences",
  "cot":["STAGE_NAME_1","STAGE_NAME_2",...,"FINAL"]
}
```

**Evaluation Context.**

**Question:** The volume of the cylinder shown is $45\pi$ cubic cm. What is the height in centimeters of the cylinder?
**Ground-truth answer:** 5
**Prediction Correctness:** 1.0
**Current ITERATIONS:**
score=0.0 | stages=TASK.INTERPRETATION, VISUAL.OBSERVATION, FACT.EXTRACTION, VARIABLE.DEFINITION, QUANTITATIVE.REASONING, FINAL
**Stage 1** -- TASK.INTERPRETATION:
Output {"task_type":"reason","rephrased_question":"given the volume and radius, find the height of the cylinder","expected_output_type":"numeric"}
**Stage 2** -- VISUAL.OBSERVATION:
Output of VISUAL.OBSERVATION: {"objects":["cylinder","radius label","height label"],"relations":["radius labeled on top circular face","height labeled along side"],"text_in_image":["r = 3","h"],"key_regions":["top circular face with radius label","side with height label","cylinder body"]}
**Stage 3** -- FACT.EXTRACTION:
Output of FACT.EXTRACTION: {"facts":["volume = $45\pi$","r = 3"],"measurements":[["$45\pi$","cubic cm"],["3","cm"]],"labels":["h"]}
**Stage 4** -- VARIABLE.DEFINITION:
Output of VARIABLE.DEFINITION: {"variables":["name":"volume","meaning":"volume of the cylinder","value":45,"unit":"$\pi$ cubic cm","name":"radius","meaning":"radius of the cylinder","value":3,"unit":"cm","name":"height","meaning":"height of the cylinder","value":null,"unit":"cm"]}
**Stage 5** -- QUANTITATIVE.REASONING:
Output of QUANTITATIVE.REASONING: {"equations":["V = $\pi r^2 h$","$45\pi = \pi(3)^2 h$","$45\pi = 9\pi h$","h = $45\pi$ / $9\pi$","h = 5"],"derived_values":[["h","5","cm"]],"final_numeric":5}
**Final answer:** 5
**Explored chains:** ... (do not repeat any of these sequences exactly).

Figure 24. Input User Content for Teacher (w/o Importance Signals)

**Teacher System Prompt (with Importance Signal)**

**Role.** You are a Vision–Language Pipeline Architect and Evaluator.
**Output rule.** Emit *only* the final JSON per schema—no extra text.
**Inputs.**

- A question and an image (image content is given in text form).
- Past CoT iterations with their scores.
- The current CoT (ordered stages) with per-stage outputs.
- Per-stage influence metrics: Direct, Indirect, Total (normalized to $[0, 1]$).
- Stage pool (valid modules only; total $= 14$):
  `['TASK.INTERPRETATION','VISUAL.OBSERVATION',...]`
- Predicted answer, score, reward, and ground-truth answer.

**Goal.** Decide whether the current CoT is already optimal. If not, propose *exactly one* improved CoT (an ordered list of stage names).
**Optimality criteria.** Treat the current CoT as optimal *only if all* of the following hold:

- Overall score is near-perfect (e.g., $\geq 0.99$).
- Every stage has a valid, non-empty output (no `None` or empty string).
- No redundant or misleading stages; the chain is concise (not needlessly long relative to the pool).

**When to propose a new CoT.** If the current chain is not optimal, you *must* propose a new one whenever any of the following is true:

- Any stage has invalid output or very low Total influence (e.g., $< 0.1$).
- Helpful evidence exists but a stage has low Direct influence, suggesting it is misplaced.
- Ordering conflicts with causal roles (e.g., HIGH-Direct stages too early; HIGH-Indirect stages too late).
- The chain is overly long, repetitive, or contains misleading stages.
- The answer is correct but the reasoning is inefficient; a shorter, equally strong chain is possible.

**Influence interpretation.**

- Direct: how strongly a stage affects the final output.
- Indirect: how strongly a stage supports or conditions later stages.
- Total: defined as Direct + Indirect, and used as the importance score for that stage.

**Editing strategy (conceptual).** Apply exactly one edit operation `op` $\in$ {`add`, `delete`, `swap`}, chosen using correctness and importance:

- *Operation choice.* If the final answer is correct (score above a correctness threshold), prefer `delete` or `swap` (prune or reorder). If it is likely incorrect, prefer `add` or `swap` (enrich or reorder). If uncertain, treat all three operations roughly equally.
- *Importance.* Use each stage's Total influence as its importance score; low-importance stages are primary candidates for removal or movement.
- *Add.* Choose a stage from the pool that is not yet in the chain and insert it where it best "bridges" existing stages. If this duplicates a previous chain, adjust the insertion position.
- *Delete.* Remove the least-important stage, ensuring the resulting chain is non-empty and novel. If that still matches a past chain, try deleting another low-importance stage.
- *Swap.* Find the weakest adjacent link (adjacent pair with lowest effective importance) and swap those two stages to improve ordering. If the resulting chain already exists, try other non-identical pairs.

**Hard constraints.**

- Use only stage names from the given stage pool.
- The last stage token must be `"FINAL"`.
- No repeated stages inside the `"cot"` list.
- The proposed `"cot"` must differ from all previous CoT sequences.

**Output (strict JSON only; no text before/after).**

```
{
  "reason":"short rationale (≤2 sentences)",
  "cot":["STAGE_1","STAGE_2",...,"FINAL"]
}
```

Figure 25. System Prompt for Teacher (with Importance Signals)

**Teacher Input User Content (with Importance Signal)**

**You are given:** a question and image (text description), past CoTs with scores, the current CoT (its stages and outputs), per-stage influence metrics (Direct, Indirect, Total, normalized to $[0, 1]$ and possibly labeled LOW/MED/HIGH), the stage pool, and the ground-truth and predicted answers. Return *only* a JSON object that follows the schema (no reasoning text, no commentary, no `<think>...</think>`).

**Task.**

- Decide whether the current CoT is already optimal or needs evolution.
- If there are redundant, unused, misplaced, or low-influence stages, propose a better ordered CoT.

**Constraints.**

- Use only valid stage names from `stage_pool` (total length $= 14$):
  `['TASK.INTERPRETATION','VISUAL.OBSERVATION', ...]`
- The stage `"FINAL"` must appear last.
- No repeated stages inside `"cot"`.
- The new CoT must clearly differ from all past CoTs (no exact repeats or simple permutations of explored sequences).

**Output format (STRICT JSON only; no extra text).**

```
{
  "reason":"short reason (≤2 sentences)",
  "cot":["STAGE_1","STAGE_2",...,"FINAL"]  // only if "evo_finish" = "False"
}
```

**Evaluation Context.**

**Question:** The volume of the cylinder shown is $45\pi$ cubic cm. What is the height in centimeters of the cylinder? <image1>

**Ground-truth answer:** 5

**Predicted answer:** 5 | **Prediction Correctness:** 1.0

**Current Iteration:**

score=1.0 | stages=TASK.INTERPRETATION, VISUAL.OBSERVATION, FACT.EXTRACTION, VARIABLE.DEFINITION, QUANTITATIVE.REASONING, FINAL

**Stage 1** -- TASK.INTERPRETATION:

Output: {"task_type":"reason","rephrased_question":"given the volume and radius, find the height of the cylinder","expected_output_type":"numeric"}

Direct: 0.0980, Indirect: 0.1395, Total: 0.2374

**Stage 2** -- VISUAL.OBSERVATION:

Output: {"objects":["cylinder","radius label","height label"],"relations":["radius labeled on top circular face","height labeled along side"],"text_in_image":["r = 3","h"],"key_regions":["top circular face with radius label","side with height label","cylinder body"]}

Direct: 0.1286, Indirect: 0.1352, Total: 0.2638

**Stage 3** -- FACT.EXTRACTION:

Output: {"facts":["volume = 45π","r = 3"],"measurements":[["45π","cubic cm"],["3","cm"]],"labels":["h"]}

Direct: 0.0970, Indirect: 0.1328, Total: 0.2298

**Stage 4** -- VARIABLE.DEFINITION:

Output: {"variables":["name":"volume","meaning":"volume of the cylinder","value":45,"unit":"π cubic cm","name":"radius","meaning":"radius of the cylinder","value":3,"unit":"cm","name":"height","meaning":"height of the cylinder","value":null,"unit":"cm"]}

Direct: 0.1533, Indirect: 0.1057, Total: 0.2590

**Stage 5** -- QUANTITATIVE.REASONING:

Output: {"equations":["V = $\pi r^2 h$","45π = $\pi(3)^2 h$","45π = $9\pi h$","h = 45π / 9π","h = 5"],"derived_values":[["h","5","cm"]],"final_numeric":5}

Direct: 0.1761, Indirect: 0.0000, Total: 0.1761

**Final answer:** 5

**Explored CoTs:** ... (do not repeat these sequences or trivial permutations).

Figure 26. Input User Content for Teacher (with Importance Signals)