STRUCTURED LATENT REASONING: REFACTORING REASONING WITH REUSABLE PRIMITIVES FOR MULTIMODAL LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating step-by-step reasoning has significantly advanced problem solving in multimodal large language models (MLLMs), yet autoregressive generation of lengthy reasoning traces incurs substantial token-level redundancy and computational overhead. Existing approaches compress reasoning into continuous latent states or discrete latent representations, but typically represent heterogeneous reasoning content within a shared latent space, making it difficult to efficiently encode both reusable reasoning patterns and instance-specific information. In this paper, we present **SLR**, a **S**tructured **L**atent **R**easoning framework that separates *reusable reasoning operations* from their *instance-specific executions*. SLR segments reasoning traces into unit spans, decomposes each span into an abstract operation and its concrete execution, and clusters recurring operations to construct a compact discrete *Reasoning Codebook*. During training, operation descriptions are replaced by discrete codebook tokens and interleaved with explicit execution text, allowing recurring procedural patterns to be represented compactly while preserving problem-specific computations. We evaluate SLR across diverse multimodal reasoning benchmarks with three MLLM configurations. Compared with standard SFT, SLR reduces mean output length by 10–35%, while matching or exceeding standard SFT accuracy in all evaluated settings. These results demonstrate that explicitly separating reusable reasoning structure from instance-specific execution provides an effective approach to reducing reasoning redundancy while maintaining multimodal reasoning performance.

1 INTRODUCTION

Generating explicit, step-by-step rationales has substantially advanced the reasoning capabilities of Large Language Models (LLMs) and Multimodal LLMs (MLLMs) (Wei et al., 2022; Kojima et al., 2022; Chung et al., 2024; Muennighoff et al., 2025). By externalizing intermediate reasoning into sequential natural language tokens, models can break down complex visual and symbolic tasks into manageable deductive steps (Mitra et al., 2024; Zhang et al., 2024; Kao et al., 2025). Despite its success, explicit natural language reasoning suffers from substantial redundancy and computational overhead. Auto-regressive generation over lengthy reasoning paths is computationally expensive, as standard reasoning traces repeatedly verbalize repetitive procedural steps and generic problem-solving strategies across instances.

To alleviate this verbosity, recent works have explored replacing explicit reasoning traces with latent representations, broadly following two paradigms: continuous latent reasoning and discrete latent compression. Continuous approaches replace token-level generation with recurrent continuous hidden states (Hao et al., 2024; Shen et al., 2025; Li et al., 2026a; Wang et al., 2026), while discrete approaches compress reasoning into learned discrete latent tokens or representations (Su et al., 2025; Zhang et al., 2025; Zheng et al., 2025). Despite their different formulations, a fundamental challenge remains: *what information should a compact latent reasoning representation preserve?* Reasoning traces inherently combine reusable reasoning patterns with instance-specific information, such as numerical values, entity bindings, and intermediate results. Existing approaches largely compress these heterogeneous forms of information into latent representations without explicitly distinguish-

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

ing their different roles, making it difficult to simultaneously achieve compactness and preserve the information required for instance-level reasoning.

These limitations motivate a different perspective on reasoning efficiency: rather than treating an entire reasoning trace as a homogeneous sequence to be compressed, we ask whether its reusable and instance-specific components can be explicitly disentangled. We observe that a reasoning span typically combines a reusable procedural pattern, capturing *how* to reason, with instance-specific content, determining *what* the reasoning is applied to. In other words, a reasoning pattern can be viewed as a reusable *operation or function*, while its execution instantiates that operation with *instance-specific arguments and intermediate execution content*. This suggests representing the two components differently: reusable patterns can be abstracted to a compact discrete vocabulary, while instance-specific information is retained explicitly rather than encoded into the same finite codebook.

Building on this insight, we introduce SLR, a **S**tructured **L**atent **R**easoning framework that provides a simple and controllable way to extract reusable reasoning patterns while preserving instance-specific information. SLR disentangles recurring reasoning operations from their instance-specific executions and represents the operations with discrete latent tokens. The resulting vocabulary can be explicitly controlled through the minimum frequency required for introducing a token, providing a way for controlling the granularity of abstraction and compression. During reasoning, latent operation tokens are interleaved with explicit instance-specific text, allowing the model to reuse compact procedural patterns together with explicit problem-specific reasoning.

Concretely, SLR first segments the full reasoning trace into a sequence of unit reasoning spans, and decomposes each span into an instance-agnostic operation and its corresponding instance-specific execution. SLR then clusters operations across reasoning traces and assign each sufficiently frequent cluster a discrete operation token. The original operation description is replaced by this token, while its instance-specific execution remains explicit. Thus, a single token can represent the same reasoning operation across different instances without encoding their entities, values, or intermediate results. By replacing recurring procedural text while preserving instance-specific computation, SLR reduces redundant reasoning tokens without discarding the details needed to solve individual problems.

We evaluate SLR across a diverse set of multimodal reasoning tasks and find that it substantially reduces reasoning token lengths while maintaining competitive or improved reasoning accuracy. Beyond token efficiency, our analyses show that the learned latent vocabulary captures recurring reasoning operations that can be reused across instances, while instance-specific information remain preserved during execution. Our contributions can be summarized in three aspects:

- We introduce a structured latent tokenization framework SLR that explicitly disentangles reusable reasoning patterns from instance-specific execution and represents the former with discrete latent operation tokens.
- SLR provides explicit control over the granularity of abstraction and compression through the latent vocabulary size and frequency threshold used to introduce operation tokens.
- SLR enables interleaved latent and explicit reasoning, allowing compact reusable operations to be seamlessly combined with explicit instance-specific computation, thereby reducing reasoning length and improving inference efficiency while preserving reasoning performance.

2 Related Work

Recent work has explored reducing the cost of explicit chain-of-thought reasoning by performing intermediate reasoning in latent representations. Continuous latent reasoning replaces explicit token generation with continuous recurrent hidden states (Hao et al., 2024; Shen et al., 2025; Li et al., 2026a; Wang et al., 2026). However, several studies have identified challenges in learning effective continuous reasoning trajectories. In particular, recursive latent steps can exhibit high representational similarity, limiting the additional information gained from further latent computation (Li et al., 2026b; Cui et al., 2026). Moreover, continuous latent representations may be poorly aligned with the discrete vocabulary space, which can make transitions between latent reasoning and language generation difficult to learn reliably (Cui et al., 2026).

An alternative direction is compressed reasoning using discrete latent representations. Existing approaches have explored vector-quantized codebooks (Su et al., 2025), condensed latent tokens (Zhang et al., 2025), and learned discrete reasoning strategies (Zheng et al., 2025). These approaches offer a more compact and controllable representation and reduce the token-level redundancy of explicit reasoning. However, existing discrete approaches generally use a shared latent vocabulary to represent diverse reasoning content. This design requires the same discrete representation space to capture both recurring reasoning patterns and instance-specific information. Increasing the vocabulary size can provide additional representational capacity, but does not explicitly distinguish between information that can be reused across instances and information that must remain instance-dependent.

3 Preliminaries and Problem Setup

Let $\mathcal{D} = \{(I^{(i)}, X^{(i)}, Y^{(i)})\}_{i=1}^{|\mathcal{D}|}$ denote a dataset of multimodal reasoning tasks, where $I^{(i)}$ is the visual input, $X^{(i)} = (x_1, \ldots, x_{|X^{(i)}|})$ is the textual problem description, and $Y^{(i)} = (y_1, \ldots, y_{|Y^{(i)}|})$ is the ground-truth response consisting of a detailed reasoning sequence enclosed within explicit thinking tags followed by the final answer: $Y^{(i)} = \texttt{<think>} \circ Y^{(i)}_{\text{think}} \circ \texttt{</think>} \circ Y^{(i)}_{\text{ans}}$, of which the tokens are drawn from the base vocabulary $\mathcal{V}_{\text{base}}$.

Standard SFT optimizes the parameter set θ of a language model by minimizing the auto-regressive negative log-likelihood over full CoT sequence:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{|Y^{(i)}|} \log P_\theta \left(y_t^{(i)} \,\middle|\, I^{(i)}, X^{(i)}, y_{<t}^{(i)} \right). \tag{1}$$

Since Y_{think} repeatedly verbalizes recurrent reasoning routines, theorem definitions, and transitional filler phrases, the resulting sequence length $|Y|$ becomes excessively large. SLR transforms Y_{think} into a concise, interleaved sequence $\widetilde{Y}_{\text{think}}$ by discretizing recurrent reasoning operations into special tokens from a learned **reasoning codebook** $\mathcal{V}_{\text{codebook}}$, yielding a compact target sequence $\widetilde{Y} = (\tilde{y}_1, \ldots, \tilde{y}_{|\widetilde{Y}|})$ where $|\widetilde{Y}| < |Y|$, drawn from an expanded vocabulary $\mathcal{V}_{\text{total}} = \mathcal{V}_{\text{base}} \cup \mathcal{V}_{\text{codebook}}$.

4 Structured Latent Reasoning

4.1 Overview

SLR transforms conventional reasoning traces into a structured sequence of reusable operation tokens and explicit instance-specific executions (See Figure 1). As summarized in Algorithm 1, given a reasoning trace, we first segment it into unit reasoning spans and decompose each span into a reusable operation and its corresponding instance-specific execution. We then aggregate operations across the training corpus and perform frequency-prioritized semantic clustering to identify recurring reasoning patterns and construct a discrete reasoning codebook. Finally, we use the codebook to replace the original traces with their corresponding latent tokens while preserving the instance-specific executions, producing interleaved sequences for autoregressive SFT. The resulting model learns to invoke compact operation tokens for recurring reasoning patterns while generating problem-specific computations explicitly.

4.2 Step Segmentation and Decomposition

Atomic Step Segmentation. We segment the raw reasoning trace Y_{think} into an ordered sequence of K atomic reasoning steps:

$$Y_{\text{think}} = s_1 \circ s_2 \circ \cdots \circ s_K, \quad s_k \in \mathcal{S}, \tag{2}$$

where $\circ$ denotes string concatenation and each s_k represents an atomic reasoning operation, such as applying a single geometric theorem or setting up an algebraic equation.

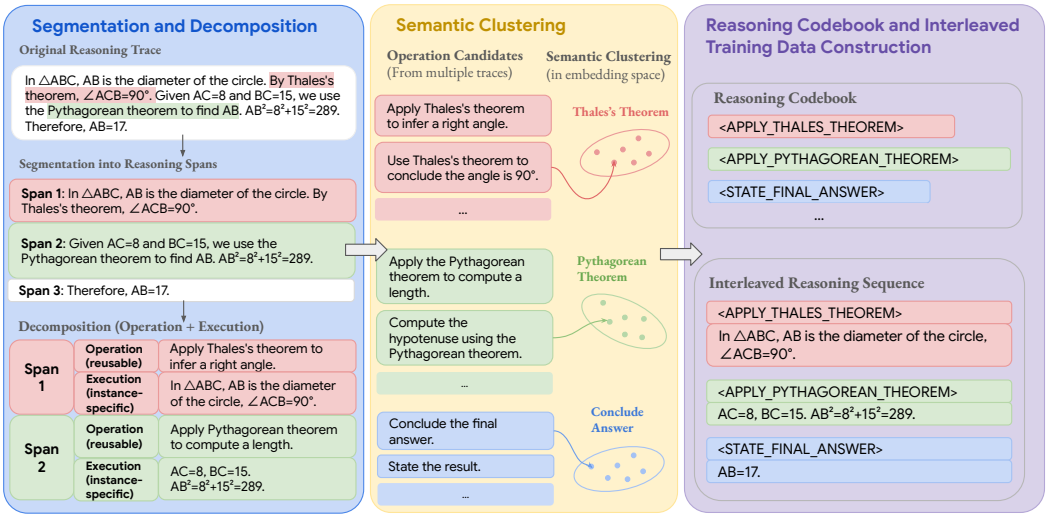


Figure 1: **Overview of Structured Latent Reasoning (SLR).** Reasoning spans are decomposed into reusable operations and instance-specific executions, then clustered to form a discrete reasoning codebook, and finally reconstructed as interleaved sequence for training.

Operation-Execution Decomposition. Each atomic step s_k is decoupled into two complementary components:

$$s_k \mapsto (o_k,\, e_k), \tag{3}$$

where $o_k \in \mathcal{O}_{\text{raw}}$ is the **abstract operation**, a concise procedural description specifying *how* to reason (the invariant rule, theorem, strategy, or deductive operator), decoupled from instance-specific numbers or entities; $e_k \in \mathcal{E}$ is the **instance-specific execution**, concise text containing *what* the operation acts upon (grounded coordinates, numerical assignments, intermediate calculations). When a step is purely strategic, $e_k = \epsilon$ (the empty string).

4.3 Semantic Clustering and Reasoning Codebook Construction

Abstract operations $o \in \mathcal{O}_{\text{raw}}$ represent transferable reasoning primitives that recur across tasks under varying surface phrasings. We consolidate them into a discrete reasoning codebook $\mathcal{V}_{\text{codebook}} = \{c_1, c_2, \ldots, c_M\}$ via frequency-prioritized semantic clustering over normalized operation representations.

Semantic Embedding. To construct the candidate operation space, we gather all extracted abstract operations in $\mathcal{O}_{\text{raw}}$ and filter out rare, non-recurrent ones. Each unique operation description $o \in \mathcal{O}_{\text{raw}}$ is then mapped to a normalized semantic vector using an off-the-shelf text embedding model $f_{\text{enc}} : \mathcal{O}_{\text{raw}} \to \mathbb{R}^d$:

$$\mathbf{z}_i = \frac{f_{\text{enc}}(o_i)}{\|f_{\text{enc}}(o_i)\|_2} \in \mathbb{R}^d, \quad \|\mathbf{z}_i\|_2 = 1. \tag{4}$$

Frequency-Prioritized Abstract Clustering. Unlike standard K-means, which requires prespecifying K and suffers from centroid drift, we employ an adaptive, frequency-prioritized greedy leader clustering algorithm to determine the number of semantic clusters M adaptively from data. We first sort unique operations by corpus frequency in descending order: $f(o_1) \geq f(o_2) \geq \cdots \geq f(o_V)$. This guarantees that high-frequency, prototypical reasoning operations serve as the canonical cluster leaders. Starting with an initial cluster $\mathcal{C}_1 = \{o_1\}$ and anchor vector $\boldsymbol{\mu}_1 = \mathbf{z}_1$, each subsequent operation vector $\mathbf{z}_i$ is sequentially compared against existing cluster anchors via cosine similarity:

$$m^* = \arg \max_{m \in \{1,\ldots,M\}} \mathbf{z}_i^\top \boldsymbol{\mu}_m. \tag{5}$$

If the maximum similarity exceeds a threshold τ_{sim} (i.e., $\mathbf{z}_i^\top \boldsymbol{\mu}_{m^*} \geq \tau_{\text{sim}}$), o_i is assigned to cluster $\mathcal{C}_{m^*}$, $\mathcal{C}_{m^*} \leftarrow \mathcal{C}_{m^*} \cup \{o_i\}$.

Algorithm 1 SLR: Reasoning Codebook Induction

Require: $\{Y_{\text{think}}^{(i)}\}_{i=1}^{N}$: CoT traces, $\mathcal{M}_{\text{seg}}$: step segmentation, $\mathcal{M}_{\text{abs}}$: operation-execution decomposition and abstract generator, $\mathcal{M}_{\text{pattern}}$: pattern name generator, Θ: clustering hyperparams.

$\mathcal{S} \leftarrow \emptyset$ ▷ Initialize atomic reasoning step pool
for each CoT $Y_{\text{think}}^{(i)}$ **do**
  $S_i \leftarrow \textsc{SegmentSteps}(Y_{\text{think}}^{(i)}, \mathcal{M}_{\text{seg}})$ ▷ **Step 1: Step Segmentation**
  for each step $s_k \in S_i$ **do**
    $(o_k, e_k) \leftarrow \textsc{AbstractOperationExecution}(s_k, \mathcal{M}_{\text{abs}})$ ▷ **Step 2: Decomposition**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(o_k, e_k)\}$
  end for
end for
$\mathcal{G} \leftarrow \textsc{ClusterOperations}(\{o_k : (o_k, e_k) \in \mathcal{S}\}, \Theta)$ ▷ **Step 3: Semantic Clustering**
for each cluster $\mathcal{C}_m \in \mathcal{G}$ **do**
  $c_m \leftarrow \textsc{AssignReasoningPattern}(\mathcal{C}_m, \mathcal{M}_{\text{pattern}})$ ▷ Codebook tokens $c_m \in \mathcal{V}_{\text{codebook}}$
end for
for each cluster $\mathcal{C}_m \in \mathcal{G}$ **do** ▷ **Step 4: Pattern Mapping & Replacement**
  $\Pi(o_k) \leftarrow c_m$ for all o_k assigned to $\mathcal{C}_m$ ▷ Mapping from abstract operations to tokens
end for
$\{\widetilde{Y}_{\text{think}}^{(i)}\}_{i=1}^{N} \leftarrow \textsc{ConstructInterleavedDataset}(\{Y_{\text{think}}^{(i)}\}_{i=1}^{N}, \mathcal{S}, \Pi)$
Return $\mathcal{V}_{\text{codebook}} = \{c_m\}_{m=1}^{|\mathcal{G}|}$, $\{\widetilde{Y}_{\text{think}}^{(i)}\}_{i=1}^{N}$

Otherwise, o_i seeds a new cluster $\mathcal{C}_{M+1} = \{o_i\}$ with anchor $\boldsymbol{\mu}_{M+1} = \mathbf{z}_i$, incrementing $M \leftarrow M + 1$. To prevent semantic centroid drift during greedy accumulation, each cluster anchor remains fixed to its founding leader's vector ($\boldsymbol{\mu}_m = \mathbf{z}_{\text{leader}(m)}$) rather than drifting via running averages.

Consolidation and Tail Pruning. To refine the induced codebook, we apply two lightweight post-processing passes:

- **Leader Consolidation:** Clusters whose founding leaders exhibit high semantic similarity ($\boldsymbol{\mu}_j^\top \boldsymbol{\mu}_k \geq \tau_{\text{merge}}$) are merged into the higher-frequency cluster to eliminate redundancy.
- **Support-Based Tail Pruning:** Clusters whose aggregate corpus frequency $F(\mathcal{C}_m) = \sum_{o_i \in \mathcal{C}_m} f(o_i)$ falls below a minimum support threshold $F_{\min}$, are either absorbed into the nearest major cluster if semantically compatible, or discarded to remove noisy, ungeneralizable patterns.

Codebook Tokenization. For each surviving cluster $\mathcal{C}_m$, an LLM judge synthesizes a standardized, human-interpretable pattern name (e.g., `<DETERMINE_OBJECT_BOUNDING_BOX>`), which defines the corresponding discrete codebook token $c_m \in \mathcal{V}_{\text{codebook}}$. The discretization mapping $\phi : \mathcal{O}_{\text{raw}} \to \mathcal{V}_{\text{codebook}} \cup \{\perp\}$ assigns each operation o_k to its corresponding cluster token c_m based on cluster membership. For operations not assigned to any cluster, ϕ performs additional nearest-anchor matching: o_k is mapped to the token associated with the nearest cluster anchor $\max_m \mathbf{z}_k^\top \boldsymbol{\mu}_m \geq \tau_{\text{fallback}}$; otherwise, it's marked as unmapped ($\perp$).

4.4 Interleaved Sequence Construction and Training

Interleaved Sequence Construction. Given the decomposed steps $(s_1, \ldots, s_K)$ of a reasoning chain, each step is transformed into an interleaved representation $\tilde{s}_k$ consisting of its discrete operation token and instance-specific execution context:

$$\tilde{s}_k = \begin{cases} c_{m^*} \circ e_k, & \text{if } \phi(o_k) = c_{m^*} \in \mathcal{V}_{\text{codebook}} \text{ and } e_k \neq \epsilon, \\ c_{m^*}, & \text{if } \phi(o_k) = c_{m^*} \in \mathcal{V}_{\text{codebook}} \text{ and } e_k = \epsilon, \\ \text{span}(s_k), & \text{if } \phi(o_k) = \perp \quad \text{(fallback to original text)}. \end{cases} \tag{6}$$

Concatenating the interleaved steps yields the compressed reasoning sequence $\widetilde{Y}_{\text{think}} = \tilde{s}_1 \circ \cdots \circ \tilde{s}_K$, and results in the final interleaved training target:

$$\widetilde{Y} = \texttt{<think>} \circ \widetilde{Y}_{\text{think}} \circ \texttt{</think>} \circ Y_{\text{ans}}. \tag{7}$$

The complete procedure for sequence transformation, from raw reasoning traces to interleaved training targets, is formalized in Algorithm 1.

Table 1: **Main Results on Qwen3-VL.** Accuracy (%) and mean output tokens per response for standard SFT and SLR trained on the same data. SLR matches or exceeds standard SFT accuracy on most benchmarks while reducing reasoning length by 10–40% across all model sizes and benchmarks.

Method	MathVista		MATH-Vision		MMVP		MMStar	
	Acc (↑)	Tokens (↓)	Acc (↑)	Tokens (↓)	Acc (↑)	Tokens (↓)	Acc (↑)	Tokens (↓)
Base Model Qwen3-VL-2B-Instruct								
Base	57.2	**365**	14.80	**1973**	71.00	**109**	56.07	352
Standard SFT	69.1	534	21.38	2405	72.67	132	59.60	387
SLR	**69.9**	479	**25.00**	1978	**74.67**	110	**60.87**	**347**
Base Model Qwen3-VL-8B-Instruct								
Base	71.7	444	35.53	2514	77.33	112	67.07	360
Standard SFT	77.8	460	43.42	1920	76.67	117	68.27	331
SLR	**78.0**	**386**	**44.41**	**1245**	**79.00**	**94**	**69.40**	**274**

Unified Auto-Regressive Training. Crucially, SLR requires no architectural modifications, auxiliary heads, or custom loss formulations. The model parameter set θ is optimized using standard auto-regressive next-token prediction over the expanded vocabulary $\mathcal{V}_{\text{total}} = \mathcal{V}_{\text{base}} \cup \mathcal{V}_{\text{codebook}}$:

$$\mathcal{L}_{\text{REORCA}}(\theta) = -\sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{|\widetilde{Y}^{(i)}|} \log P_\theta \left(\tilde{y}_t^{(i)} \,\middle|\, I^{(i)}, X^{(i)}, \tilde{y}_{<t}^{(i)} \right). \tag{8}$$

By replacing verbose procedural spans with atomic codebook tokens $c \in \mathcal{V}_{\text{codebook}}$, while preserving concrete instance-specific computations $e \in \mathcal{V}_{\text{base}}$, SLR shortens the reasoning horizon and enforces structured logical trajectory simultaneously.

5 Experiments

In this section, we empirically study whether reasoning can be more efficient by replacing recurring reasoning operations with reusable discrete tokens while preserving instance-specific computation in explicit text. We compare SLR with standard SFT and existing multimodal reasoning methods across multiple multimodal reasoning benchmarks and model scales.

5.1 Experimental Setup

Base Model and Training Setup. We evaluate our framework across three open-source multimodal language models spanning different model scales: Qwen3-VL-2B-Instruct, Qwen3-VL-8B-Instruct (Bai et al., 2025a), and Qwen2.5-VL-7B-Instruct (Bai et al., 2025b). This setup allows us to examine the effectiveness of reusable reasoning tokens across different model capacities and model families. Unless otherwise specified, all models are fine-tuned on OneThinker dataset (Feng et al., 2026) for 3,300 optimization steps under identical training schedules and evaluation protocols. For each operation token, we initialize its embedding by averaging the embeddings of the original vocabulary tokens corresponding to the operation pattern name. These newly introduced token embeddings are then jointly optimized with the model during SFT.

Evaluation Benchmarks. We evaluate on four multimodal reasoning benchmarks: **MathVista** (Lu et al., 2024), **MATH-Vision** (Wang et al., 2024), **MMVP** (Tong et al., 2024), and **MMStar** (Chen et al., 2024), covering mathematical, visual, and general multimodal reasoning. We use the same prompt template without benchmark-specific prompt engineering for evaluation. Models generate responses using greedy decoding.

5.2 Main Results

We evaluate SLR from two complementary perspectives. First, we compare SLR with standard SFT under the same training setup on Qwen3-VL, examining the trade-off between reasoning accuracy and reasoning length. Second, we compare SLR with existing multimodal reasoning approaches on Qwen2.5-VL-7B-Instruct.

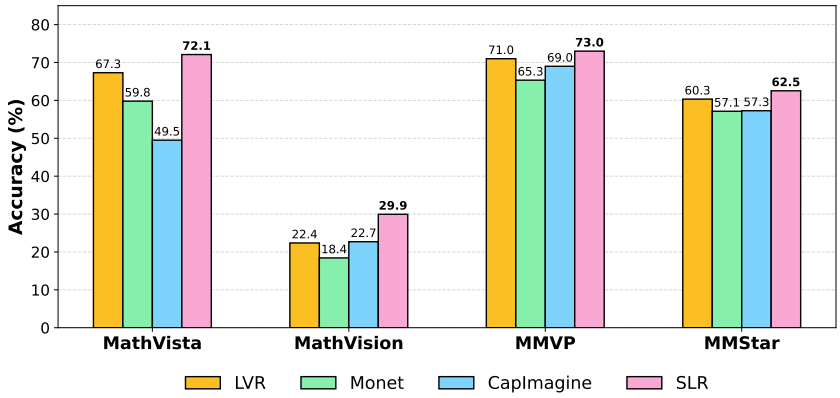


Figure 2: **Comparison with multimodal reasoning baselines.** Accuracy (%) of LVR, Monet, CapImagine, and SLR on MathVista, MathVision, MMVP, and MMStar. All methods are built on the same Qwen2.5-VL-7B-Instruct backbone. SLR achieves the highest accuracy on all four benchmarks.

Accuracy is improved over standard SFT with substantially reduced reasoning length. Table 1 compares SLR with standard SFT across four multimodal reasoning benchmarks and two Qwen3-VL model sizes. SLR consistently improves accuracy over standard SFT across different types of multimodal reasoning tasks on both model sizes. In the meantime, SLR produces substantially shorter reasoning traces with an average reduction of approximately 18% tokens. These results show that replacing recurring procedural reasoning with compact reusable tokens provides a more efficient representation while retaining the instance-specific information required in explicit text for effective reasoning.

Comparison with multimodal reasoning baselines. Figure 2 compares SLR with LVR (Li et al., 2026a), Monet (Wang et al., 2026), and CapImagine (Li et al., 2026b). All methods use the same Qwen2.5-VL-7B-Instruct backbone. SLR achieves the highest accuracy among the compared methods on all four benchmarks, with particularly clear gains on MathVista and MATH-Vision. This comparison demonstrates that the reusable-operation representation is effective not only relative to standard SFT, but also against existing approaches for multimodal reasoning.

5.3 In-Depth Analysis and Ablations

5.3.1 Robustness to Codebook Construction

The construction of the reusable operation codebook involves two main hyperparameters: the minimum appearance frequency $F_{\min}$, which filters out clusters that occur fewer than $F_{\min}$ times in the training corpus, and the similarity threshold τ_{sim}, which determines the semantic similarity required for two abstract operations to be grouped together. Figure 3 examines the robustness of SLR to these parameters. Across a broad range of settings, SLR maintains competitive accuracy while producing substantially shorter reasoning traces, demonstrating robustness to the codebook construction hyperparameter choice.

Minimum frequency ensures learnable and reusable operations. The minimum frequency $F_{\min}$ serves two related purposes. First, the newly introduced operation tokens should occur sufficient times during SFT for the model to associate each token with a consistent reasoning behavior. Operations which occur only a few times provide limited training signal for learning their embeddings and appropriate usage. Second, frequency acts as a natural criterion for reusability: operations that recur across many training instances are more likely to represent general reasoning patterns, whereas rare operations may correspond to narrow, instance-specific cases that are less suitable as reusable primitives. Thus, filtering by $F_{\min}$ prevents the codebook from being populated by insufficiently trained or overly specialized operations. Empirically, $F_{\min} \in \{200, 300, 400\}$ contains several of the strongest configurations, with $F_{\min} = 400$ achieving the highest MMVP accuracy and $F_{\min} = 300$ achieving the highest MMStar accuracy.

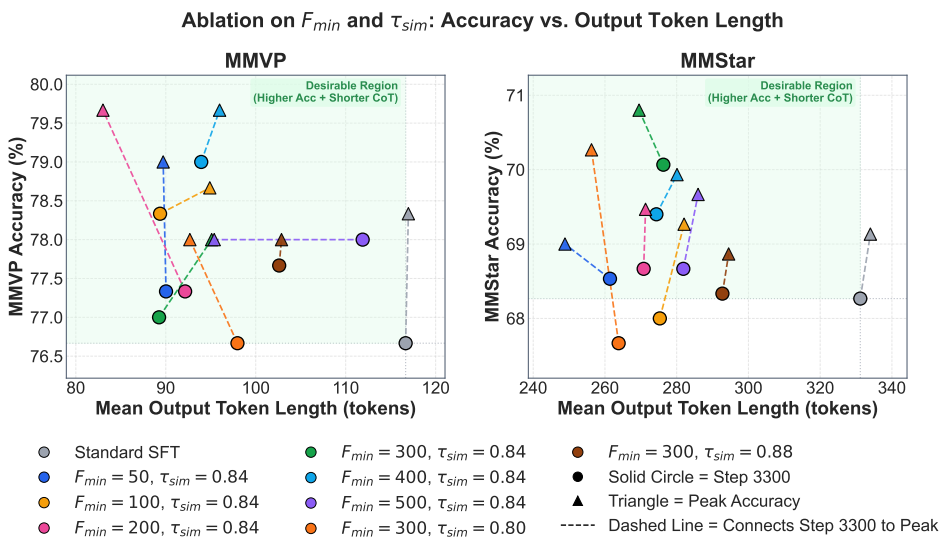


Figure 3: **Robustness to operation vocabulary construction.** Accuracy versus mean output length under different minimum appearance frequencies $F_{\min}$ of operation tokens and clustering similarity thresholds τ_{sim}. Circles denote the 3,300-step checkpoint; triangles denote the peak accuracy during the training; dashed lines connect the two.

Similarity controls the granularity of reusable operations. The similarity threshold τ_{sim} determines how fine-grained the operation vocabulary is. If the threshold is too low, semantically distinct operations may be merged into the same cluster and consequently mapped to a single token, forcing that token to represent different reasoning behaviors. Conversely, an excessively high threshold can fragment semantically related operations into small clusters, which are then more likely to fall below $F_{\min}$ and be discarded. A moderate similarity threshold therefore provides a balance between semantic consistency within each operation token and sufficient cross-instance recurrence. As shown in Figure 3, $\tau_{\text{sim}} = 0.84$ provides the most consistent performance among the tested values.

5.3.2 Ablations and Qualitative Analysis

SLR uses a discrete vocabulary of operation tokens to represent recurring reasoning patterns. We conduct ablations to examine whether (1) semantic correspondence between tokens and their underlying operations matters and (2) the model learns to use these tokens according to their semantic roles. All variants use the same SLR training procedure and schedules, differing only in token construction or initialization: **(i) Random (uniform)** independently replaces each operation token with a uniformly sampled vocabulary token, destroying operation–token correspondence; **(ii) Random (frequency-matched)** globally permutes token assignments, preserving token frequencies while destroying operation–token correspondence; **(iii) Unified** replaces all operation tokens with a single generic `<REASONING_STEP>` marker, preserving segmentation while removing operation-specific identities; and **(iv) Random init** retains the original operation vocabulary but randomly initializes token embeddings. Table 2 reports the results on Qwen3-VL-8B-Instruct.

Semantic grounding is critical for operation tokens. On MATH-Vision, SLR achieves 44.41% accuracy, whereas Random (uniform) and Random (frequency-matched) drop to 37.17% and 36.51%, respectively. The Unified variant performs better than the randomized variants, reaching 41.45%, suggesting that explicit segmentation itself provides some benefit; however, removing the distinctions between different operations still results in a performance gap relative to the full operation vocabulary. Together, these results indicate that the model benefits from tokens that encode distinct and semantically meaningful reasoning operations.

The model learns to use operation tokens according to their semantic roles. The operation-token usage statistics provide further evidence that the model does not treat the vocabulary as generic segmentation markers. On MMStar, SLR emits an average of 3.68 operation to-

Table 2: **Operation-token ablations** (Qwen3-VL-8B-Instruct). All variants share the SLR recipe and differ only in the interleaved tokens. **Tokens** is the mean output length; **#Ops** is the mean number of operation tokens emitted per response; **Use** (%) is the share of responses emitting at least one operation token. Randomizing the operation token assignments substantially degrades MATH-Vision accuracy despite producing shorter outputs, while also sharply reducing operation token usage.

Method	MATH-Vision				MMStar			
	Acc (↑)	**Tokens** (↓)	**#Ops**	**Use**	**Acc** (↑)	**Tokens** (↓)	**#Ops**	**Use**
Standard SFT	43.42	1920	–	–	68.27	331	–	–
SLR	44.41	1245	2.21	69.1	69.40	274	3.68	96.1
Random (uniform)	37.17	708	0.04	3.3	69.40	265	0.85	38.1
Random (frequency-matched)	36.51	730	0.22	12.5	69.33	272	2.02	71.5
Unified	41.45	1331	1.95	68.1	68.60	263	3.52	96.5
Random init	41.78	1274	2.51	61.2	68.80	280	3.66	95.7

kens per response, with 96.1% of responses containing at least one operation token. When the operation–token correspondence is randomized, operation-token usage sharply decreases. Random (uniform) emits only 0.04 operation tokens per response on MATH-Vision, with tokens appearing in just 3.3% of responses. Random (frequency-matched) shows the same trend. In the latter case, the model frequently falls back to the most common token in the vocabulary, <INITIATE_STEP_BY_STEP_SOLVING> (Figure 6), even though the token no longer corresponds to its original operation. These results suggest that the model learns to invoke operation tokens when their semantic identities provide a useful and predictable signal, rather than simply learning to emit discrete markers.

Semantic initialization provides a useful inductive bias. We further investigate whether initializing operation token embeddings from their corresponding abstract descriptions contributes to the observed gains. Randomly initialized tokens remain usable after training, maintaining relatively high operation-token usage with slightly lower accuracies than default SLR. Therefore, semantic initialization is not necessary for the model to learn and use the operation vocabulary, but provides a useful initialization that leads to stronger final performance.

Qualitative analysis. We examine how the operation vocabulary affects reasoning traces. Figure 6 presents a visual reasoning example where SLR correctly identifies relevant visual evidence through a structured sequence of operations, while comparison variants produce superficially similar but incorrect reasoning. Standard SFT repeatedly revisits its visual interpretation through free-form "wait" statements; Random (uniform) invokes tokens unrelated to the underlying reasoning steps; and Random (frequency-matched) repeatedly selects the most frequent token after shuffling operation–token correspondence, providing no semantic guidance. Unified retains a structured form but replaces every step with the same generic <REASONING_STEP> marker, removing the operation-specific distinctions. Figure 7 shows a geometry example where both SLR and standard SFT reach the correct answer. SLR organizes the solution into a concise sequence of reusable operations, with instance-specific calculations expressed in text, whereas standard SFT expresses the same reasoning through a longer and more free-form natural language trace. Together, these examples provide qualitative evidence that the learned operation vocabulary makes recurring reasoning structure explicit while reducing redundant verbalization.

6 Conclusion

In this paper, we introduced SLR, a structured latent reasoning framework that refactors multimodal reasoning traces into interleaved sequences of discrete reasoning operations and instance-specific executions. Through reasoning span segmentation, operation–execution decomposition, and frequency-aware semantic clustering, SLR constructs a compact reasoning codebook of reusable reasoning operations, without requiring architectural modifications or additional training objectives. Across diverse multimodal reasoning benchmarks and three MLLM configurations, SLR reduces reasoning length by 10–35% while maintaining or improving task accuracy. These results suggest that separating reusable reasoning structure from instance-specific computation provides a simple and effective approach to more concise and efficient multimodal reasoning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and editing the paper, including improving clarity, grammar, and readability. We also used coding agents to assist with the development and implementation of experiment code. AI-assisted content and code were reviewed, edited, and verified by the authors. We did not use generative AI tools to develop the research methodology, formulate hypotheses, interpret experimental results, or make other substantive research decisions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of machine learning research*, 25(70):1–53, 2024.

Jin Cui, Xinyue Long, Xunyong Zhang, Yadong Zhang, Chuanchang Su, Jingye Gan, Boran Zhao, and Pengju Ren. Retrieve, integrate, and synthesize: Spatial-semantic grounded latent visual reasoning. *arXiv preprint arXiv:2605.07106*, 2026.

Kaituo Feng, Manyuan Zhang, Hongyu Li, Kaixuan Fan, Shuang Chen, Yilei Jiang, Dian Zheng, Peiwen Sun, Yiyuan Zhang, Haoze Sun, et al. Onethinker: All-in-one reasoning model for image and video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5432–5443, 2026.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Kuei-Chun Kao, Hsu Tzu-Yin, Yunqi Hong, Ruochen Wang, and Cho-Jui Hsieh. Qg-coc: Question-guided chain-of-captions for large multimodal models. *arXiv preprint arXiv:2511.03206*, 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 148076–148090, 2026a.

You Li, Chi Chen, Yanghao Li, Fanhu Zeng, Kaiyu Huang, Jinan Xu, and Maosong Sun. Imagination helps visual reasoning, but not yet in latent space. *arXiv preprint arXiv:2602.22766*, 2026b.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024.

Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain-of-thought prompting for large multimodal models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14420–14431. IEEE, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

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

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond image and language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12030–12040, 2026.

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

Daoan Zhang, Junming Yang, Hanjia Lyu, Zijian Jin, Yuan Yao, Mingkai Chen, and Jiebo Luo. Cocot: Contrastive chain-of-thought prompting for large multimodal models with multiple image inputs. *arXiv preprint arXiv:2401.02582*, 2024.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13318–13339, 2025.

Haoyu Zheng, Zhuonan Wang, Yuqian Yuan, Tianwei Lin, Wenqiao Zhang, Zheqi Lv, Juncheng Li, Siliang Tang, Yueting Zhuang, and Hongyang He. Fast thinking for large language models. *arXiv preprint arXiv:2509.23633*, 2025.

A Appendix

A.1 Prompts for Reasoning Codebook Induction

All LLM calls use `Gemini-3.5-Flash` with temperature 0.2; embeddings use `gemini-embedding-001`. In our implementation, step segmentation $\mathcal{M}_{\text{seg}}$ and operation–execution decomposition $\mathcal{M}_{\text{abs}}$ are performed jointly in a single call (Prompt 4): the model partitions the CoT into chronological spans, each an exact sub-string of the original trace, and for each span produces an instance-agnostic abstract operation o_k (`abstract_operation`) and its instance-specific execution e_k (`execution_content`), realizing $\mathcal{M}_{\text{abs}}$.

The pattern name generator $\mathcal{M}_{\text{pattern}}$ (Prompt 5) receives each cluster's leader (most frequent) operation and up to five additional members, and returns a codebook token name with a one-sentence description.

A.2 Evaluation Examples

Prompt 1: Step Segmentation and Operation–Execution Decomposition ($\mathcal{M}_{\text{seg}}$, $\mathcal{M}_{\text{abs}}$)

You are an expert annotator and cognitive scientist specializing in reasoning decomposition for multimodal AI models.
Analyze a Question, an Image (if present), an Answer, and its Chain-of-Thought (CoT), and perform a faithful, sequential **span-by-span decomposition** of the entire CoT.

Core Principles.
1. Faithful chronological mapping. Do not synthesize a new or simplified "shortest-path" solution. Decompose the original CoT sequentially from start to finish, preserving the complete reasoning trajectory, including: initial setup and problem understanding; exploration and trial-and-error; theorem recall and heuristic rules; self-correction and backtracking; concrete variable assignments and calculations; visual landmark perception and boundary reasoning; and sanity checking and final answer selection.
Every `original_span` must be an exact substring of the original CoT.

2. Operation–execution decomposition. For every sequential span, identify:

- **`abstract_operation`:** The underlying reusable operation, transferable method, theorem, visual query, calculation operator, backtracking action, or discourse step. Formulate it generally without instance-specific constants or values.
- **`execution_content`:** The instance-specific calculation, variable substitution, geometric deduction, or visual observation for this problem. For purely reusable spans with no problem-specific entities or values, use an empty string `""`.
- **`original_span`:** The concise corresponding phrase from the original CoT, preserved verbatim for traceability.

Examples of abstract operations.

- Initiate step-by-step problem solving.
- Apply the Tangent Segments Theorem for circle tangents.
- Assign variables to tangent segments from vertices.
- Detect a potential inconsistency and initiate backtracking.
- Determine an object's bounding box from visual boundaries.
- Select positive segmentation points on the target foreground.

3. Visual grounding. For visual reasoning, preserve landmark names and spatial extents in `execution_content`; do not reduce visual observations to bare coordinates. For example, preserve descriptions such as "yellow front cab", "windshield", and "wheels on track", together with relevant spatial extents.

Output. Return **only valid JSON** matching the following schema:

```
{
"steps": [
{
"step_idx": 1,
"abstract_operation": "<reusable operation>",
"execution_content": "<instance-specific content or empty string>",
"original_span": "<exact substring from original CoT>"
}
]
}
```

Figure 4: Prompt used to decompose each CoT into reusable reasoning operations and instance-specific execution content while preserving traceability to the original reasoning trace.

Prompt 2: Pattern Name Generator ($\mathcal{M}_{\text{pattern}}$)

Task. You are an expert in reasoning decomposition for Chain-of-Thought (CoT) solutions in visual and text AI models.
Below is a cluster of similar abstract operation descriptions extracted from CoT reasoning steps.

Cluster.
Leader (most frequent operation): {`leader`}
Other member operations:

- {`member_1`}
- ...
- {`member_5`}

Task. Provide:

- a concise, standardized pattern name in `UPPER_CASE_WITH_UNDERSCORES`, with at most 4–5 words (e.g., `IDENTIFY_TARGET_ATTRIBUTE`, `CALCULATE_QUANTITY`, or `VERIFY_CONSTRAINT`);
- a clear one-sentence description of the abstract reasoning step represented by this cluster.

Output. Return valid JSON with the keys `pattern_name` and `description`:

```
{
"pattern_name":  "<standardized pattern name>",
"description":  "<one-sentence description>"
}
```

Figure 5: Prompt used to consolidate similar abstract reasoning operations into standardized, reusable reasoning patterns.

MMVP: Is there a lemon inside the drink in the cup or are all the lemons outside the drink? A: There is one inside B: All are outside Ground truth: **A**

SLR **A** ✓



<INITIATE_VISUAL_INSPECTION>
<IDENTIFY_VISUAL_ELEMENTS>
A glass containing a drink, with a lemon slice inside the glass (in the drink).
<IDENTIFY_VISUAL_ELEMENTS>
Another lemon slice is on the rim of the glass, outside the drink.
<IDENTIFY_QUESTION_OBJECTIVE>
Determine if there is a lemon inside the drink.
<SELECT_OPTION_VIA_VISUAL_EVIDENCE>
There is at least one lemon inside the drink, which corresponds to Option A.

Standard SFT **B** ×

So, let's look at the image. The cup has a drink, and there's a lemon slice inside the cup (in the drink). Wait, no, wait. Wait, the lemon slice–wait, the image shows a glass with a drink, and there's a lemon slice inside the glass (so inside the drink). [...] Wait, but how many lemons are there? The image shows one lemon slice, which is on the rim (outside the drink). So all lemons (only one) are outside. So the answer would be B.

Random (uniform) **B** ×

So, let's look at the image. The drink is in a glass. There's a lime slice inside the glass, and a lemon slice–wait, the lemon slice is on the rim of the glass, so is that inside the drink? [...] Wait, but let's confirm. The image shows a glass with a drink, a lime slice inside the drink, and a lemon slice on the rim (so outside the drink). So there are no lemons inside the drink.
<FINALIZE_BOUNDING_BOX_COORDINATES>
Option B (''All are outside'') is correct.

Random (frequency-matched) **B** ×

<INITIATE_STEP_BY_STEP_SOLVING> So, let's look at the image.
<INITIATE_STEP_BY_STEP_SOLVING>
A glass containing a drink with a lime slice inside. Wait, the question is about lemons. The lemon slice is on the rim of the glass, not inside the drink. So all the lemons (the lemon slice here) are outside the drink.
<INITIATE_STEP_BY_STEP_SOLVING>
Option B corresponds to ''All are outside''.

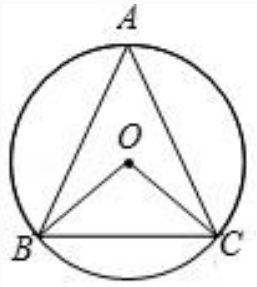
Unified **B** ×

<REASONING_STEP>
So, let's look at the image.
<REASONING_STEP>
A glass containing a drink with a lime slice inside. [...]
<REASONING_STEP>
The drink contains lime, and the lemon is only on the rim (outside).
<REASONING_STEP>
Option B (''All are outside'') is correct because there is no lemon inside the drink.

Figure 6: **Reusable operation tokens enable more reliable visual reasoning.** SLR composes reusable operations for visual inspection, element identification, objective identification, and evidence-based option selection. In contrast, the other variants misidentify the object relationships, reaching the wrong conclusion. This example illustrates how reusable operations can structure visual reasoning toward the relevant evidence rather than merely reproducing surface-level reasoning patterns.

MathVista: As shown in the figure, points A and B are three points on $\odot O$ and $AB = AC$. Connect BO and CO, if $\angle ABC = 65.0$, then the degree of $\angle BOC$ is A: 50° B: 65° C: 100° D: 130°
Ground truth: **C**

SLR **C ✓**



<INITIATE_STEP_BY_STEP_SOLVING>
<CALCULATE_ISOSCELES_TRIANGLE_ANGLES>
∠ABC is isosceles with AB = AC, so base angles ∠ABC = ∠ACB = 65°. ∠BAC = 180° - 65° - 65° = 50°.
<APPLY_INSCRIBED_ANGLE_THEOREM>
Central angle ∠BOC and inscribed angle ∠BAC subtend the same arc BC, so ∠BOC = 2 * ∠BAC.
<MATCH_CALCULATED_RESULT_WITH_OPTIONS>
∠BOC = 2 * 50° = 100°, which corresponds to Option C.

Standard SFT **C ✓**

Got it, let's try to solve this problem step by step. First, let's recall some circle theorems and triangle properties. We know that AB = AC, so triangle ABC is isosceles with AB = AC. That means the base angles at B and C are equal. Wait, the problem says ∠ABC = 65°, so ∠ACB is also 65° because AB = AC. Then we can find ∠BAC, the vertex angle of triangle ABC. The sum of angles in a triangle is 180°, so ∠BAC = 180° - 65° - 65° = 50°.
Now, in a circle, the central angle is twice the inscribed angle that subtends the same arc. The arc here is arc BC. The inscribed angle subtended by arc BC is ∠BAC, and the central angle is ∠BOC. So ∠BOC = 2 * ∠BAC. Let's calculate that: 2 * 50° = 100°. So the measure of ∠BOC is 100°, which corresponds to option C.

Figure 7: **Reusable operation tokens yield structured and concise reasoning with less verbalization.** SLR organizes the solution into a sequence of reusable operations, while expressing only the instance-specific calculations in text. In contrast, standard SFT produces a longer, free-form reasoning trace that repeatedly explains intermediate steps. The example illustrates how SLR makes the reasoning succinct and structured.

Table 3: Top special operation tokens per benchmark with their number of appearances.

#	Token	App. #
	MathVista	
1	INITIATE_STEP_BY_STEP_SOLVING	653
2	SELECT_MULTIPLE_CHOICE_OPTION	221
3	IDENTIFY_VISUAL_ELEMENTS	145
4	EXTRACT_VISUAL_CATEGORY_DATA	137
5	DETECT_ERROR_AND_BACKTRACK	122
6	IDENTIFY_GIVENS_AND_TARGET	100
7	IDENTIFY_AND_COUNT_OBJECTS	94
8	STATE_FINAL_ANSWER	77
9	MATCH_CALCULATED_RESULT_WITH_OPTIONS	74
10	RECALL_GEOMETRIC_PROPERTIES	72
11	IDENTIFY_GRAPH_AXES_AND_VARIABLES	64
12	APPLY_INSCRIBED_ANGLE_THEOREM	62
	MATH-Vision	
1	INITIATE_STEP_BY_STEP_SOLVING	181
2	EVALUATE_OPTIONS_AGAINST_VISUAL_EVIDENCE	129
3	EVALUATE_CANDIDATE_AGAINST_CRITERIA_DEFINED	93
4	INITIATE_CANDIDATE_EVALUATION	45
5	EVALUATE_AND_SELECT_OPTION	38
6	INITIATE_PROBLEM_SOLVING_PROCESS	33
7	DETECT_ERROR_AND_BACKTRACK	32
8	IDENTIFY_GIVENS_AND_TARGET	21
9	EVALUATE_OPTIONS_AGAINST_CHART_DATA	19
10	CALCULATE_ISOSCELES_TRIANGLE_ANGLES	19
	MMVP	
1	SELECT_MULTIPLE_CHOICE_OPTION	246
2	INITIATE_VISUAL_INSPECTION	175
3	DESCRIBE_TARGET_VISUAL_FEATURES	166
4	IDENTIFY_VISUAL_ELEMENTS	59
5	ANALYZE_SPATIAL_LAYOUT	43
6	INITIATE_STEP_BY_STEP_SOLVING	37
7	DETECT_ERROR_AND_BACKTRACK	36
8	IDENTIFY_QUESTION_AND_VISUAL_TARGETS	31
9	LOCATE_TARGET_OBJECT	29
10	EVALUATE_OPTIONS_AGAINST_VISUAL_EVIDENCE	27
	MMStar	
1	SELECT_MULTIPLE_CHOICE_OPTION	858
2	INITIATE_STEP_BY_STEP_SOLVING	836
3	EVALUATE_OPTIONS_AGAINST_VISUAL_EVIDENCE	688
4	IDENTIFY_VISUAL_ELEMENTS	425
5	INITIATE_VISUAL_INSPECTION	326
6	INITIATE_CANDIDATE_EVALUATION	255
7	DETECT_ERROR_AND_BACKTRACK	207
8	DESCRIBE_TARGET_VISUAL_FEATURES	139
9	SELECT_FINAL_OPTION	118
10	ANALYZE_SPATIAL_LAYOUT	114
11	INITIATE_OBJECT_COUNTING	104
12	EVALUATE_CANDIDATE_AGAINST_CRITERIA_DEFINED	97
13	SELECT_OPTION_VIA_VISUAL_EVIDENCE	74