LATENTWEAVE: TEST-TIME LATENT OPTIMIZATION FOR MULTIMODAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in test-time optimization have shown that the multimodal reasoning performance of large multimodal language models (MLLMs) can be improved during inference without modifying backbone model parameters. However, existing approaches primarily operate at the prompt or output level, leaving the latent reasoning process largely uncontrolled and insufficient to prevent error accumulation during autoregressive generation. To address this limitation, we propose `LatentWeave`, a test-time framework that treats multimodal reasoning as a controllable latent trajectory. `LatentWeave` performs structured *Latent Weaving*, progressively refining intermediate hidden states segment-wise to correct emerging deviations, and further introduces *Reasoning Anchors* via lightweight WEAVE tokens to stabilize reasoning structure and guide answer-oriented generation. These mechanisms are designed to enable fine-grained, step-aware intervention in the latent reasoning process while keeping the main MLLM backbone parameters fixed. Extensive experiments across diverse multimodal reasoning benchmarks report improvements over the evaluated baselines; the contribution of latent optimization, anchor training, and additional test-time computation is analyzed separately in the ablation and efficiency studies. Our code is available at https://anonymous.4open.science/r/LatentWeave-7777.

1 INTRODUCTION

Multimodal large language models (MLLMs) (Yin et al., 2024) increasingly support visual understanding and generation within unified systems (Ma et al., 2025; Wu et al., 2025), alongside specialized capabilities such as image-quality reasoning (Li et al., 2025b). These advances have broadened their use as general-purpose foundation models (Yang et al., 2023; Comanici et al., 2025). Research is consequently expanding from perception to multi-step reasoning over multimodal information (Li et al., 2025c; Viveiros et al., 2026). Maintaining a structured reasoning process remains challenging: LLaVA-CoT studies staged visual reasoning (Xu et al., 2025), while CoMT evaluates the integration of visual operations into intermediate reasoning (Cheng et al., 2025). As reasoning traces grow longer, the evolving evidence and intermediate representations become important targets for inference-time refinement (Jiang et al., 2026b; Zhang et al., 2026).

To improve the test-time performance of multimodal reasoning (Zhu et al., 2025; Jiang et al., 2026b), existing studies have explored intervention strategies from three core dimensions: ❶ **Output-level refinement:** Focuses on answer post-processing and self-correction (Madaan et al., 2023), iteratively optimizing model-generated outputs via answer rewriting, candidate filtering, or reasoning-trace refinement (Jiang et al., 2026a). ❷ **Prompt-level control:** Leverages concise guidance from prompts or reasoning structures (Xia et al., 2026), guiding models to follow predefined reasoning paths by constructing explicit reasoning chains (Wang et al., 2025), decomposing sub-problems (Acuna et al., 2025), or introducing stepwise textual guidance (Xia et al., 2026). ❸ **Parameter-level adaptation:** Relies on parameter updates and adaptive adjustments (Zhao et al., 2025), incorporating reinforcement updates (Zuo et al., 2025) or temporary adapter optimization (Yuan et al., 2026) during inference to enhance performance on specific reasoning tasks. Although these strategies yield improvements, their primary intervention targets are outputs, prompts, or parameters rather than individual latent states. Recent methods also optimize continuous representations (Li et al., 2025a; Zhang et al., 2026), motivating step-aware intervention in multimodal reasoning.

These observations motivate treating multimodal reasoning as a control problem over latent trajectories. Output- and prompt-oriented methods address errors through output refinement or external guidance (Jiang et al., 2026a; Xia et al., 2026). We hypothesize that some errors accumulate in

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

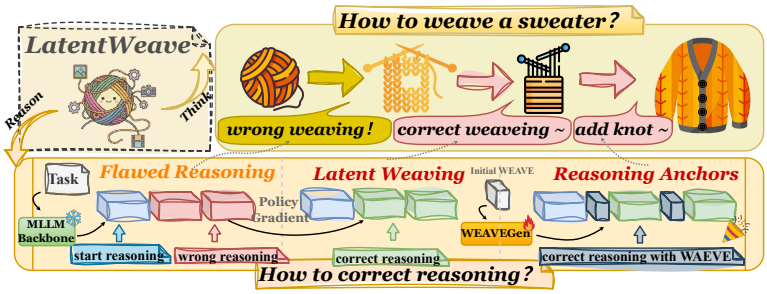

Figure 1: `LatentWeave`'s Latent Weaving functions much like rectifying faulty knitting, preventing accumulation of errors and subsequent failure, whereas Reasoning Anchors mirror adding buttons at appropriate positions to a finished sweater body, enhancing its structural integrity.

intermediate hidden states and become harder to correct during subsequent generation. This motivates refining the evolving trajectory directly (Zhang et al., 2025; 2026). Here, latent reasoning refers to iteratively refining the model's internal hidden states during autoregressive generation. Such intervention can revise intermediate decisions while retaining the pretrained backbone parameters. The central question is how to localize these updates, preserve useful preceding context, and propagate corrections into the remaining continuation.

Given the prior limitations, we propose **`LatentWeave`**, a test-time framework that models multimodal reasoning as an evolving latent trajectory and directly optimizes its hidden states. As in knitting a sweater (Figure 1), an early deviation can be amplified by subsequent steps and distort the final result. Based on this insight, **`LatentWeave`** regulates reasoning trajectories via two paired core mechanisms: **Latent Weaving** refines hidden-state segments during generation. Unlike ACTS (Xia et al., 2026), which steers a frozen reasoner through textual strategies and starter phrases, **`LatentWeave`** directly optimizes each active segment, preserves accepted segments, and re-decodes the remaining suffix. Rectified trajectories supervise the lightweight *WEAVEGen* model to generate textual reasoning anchors. **Reasoning Anchors** differ from PILOT's latent-vector injection (Zheng et al., 2026): WEAVEGen serializes its guidance with `<WEAVE>` delimiters and passes it through the backbone's ordinary textual context, avoiding hidden-dimension transfer between models. WEAVEGen is trained offline and generates one anchor sequence per input. Both mechanisms keep the evaluated backbone fixed at test time; auxiliary generation costs are reported separately.

We summarize our contributions in the following three points:

❶ ***Problem formulation.*** We formulate multimodal reasoning as a latent trajectory control problem, hypothesizing that some reasoning failures arise from progressively accumulated deviations in intermediate latent states.

❷ ***Method.*** We propose **`LatentWeave`**, a frozen-backbone test-time framework that directly intervenes in latent representations during autoregressive generation. **Latent Weaving** refines hidden-state segments in sequence, while textual **Reasoning Anchors** are generated by a lightweight model (*WEAVEGen*) trained on rectified trajectories. Their text-based interface supports backbones with different hidden dimensions.

❸ ***Experimental evaluation.*** Experiments cover seven benchmarks and five backbones (4B–27B). Over vanilla decoding, **`LatentWeave`** gains 6.57 percentage points on MathVision with `Qwen3.5-4B`, and 6.57 and 5.22 points on MathVision and RealWorldQA, respectively, with `Qwen3.6-27B`, demonstrating improvements across model scales and task domains.

2 Related Work

Multimodal reasoning representations. Visual Sketchpad uses executable sketches as intermediate visual reasoning steps (Hu et al., 2024), while Coconut establishes continuous hidden-state reasoning in language models (Hao et al., 2024). Recent multimodal methods learn task-relevant latent representations: LanteRn interleaves text with visually grounded latent blocks (Viveiros et al., 2026); Pearl internalizes tool-use trajectories through predictive embedding objectives (Adhikari & Lapata, 2026); and SLVR enriches region-level visual latents with semantic supervision and multi-query reinforcement learning (Xu et al., 2026). These methods principally learn how to generate useful representations. Complementing this line, Li et al. (2026b) identify weak input-to-latent and latent-to-answer dependencies in the visual reasoners they examine. Their findings motivate checking whether intermediate states influence answers, rather than equating latent tokens with effective reasoning.

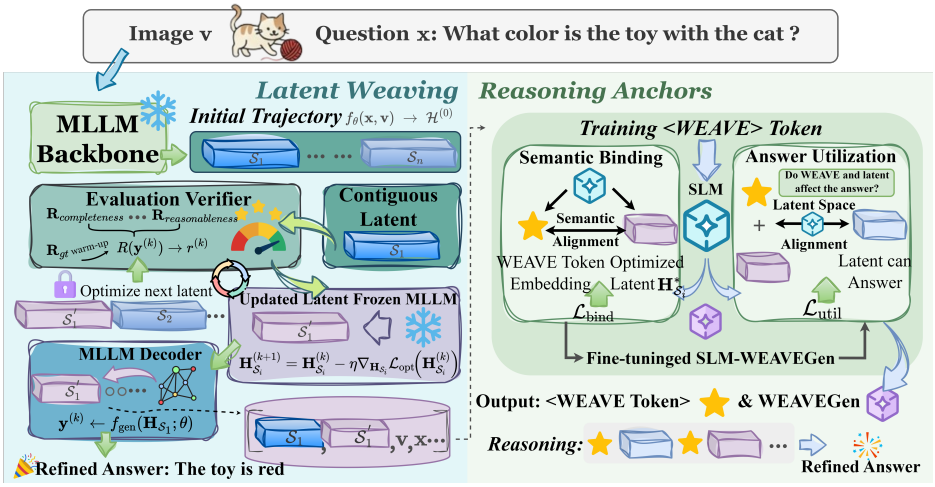

Figure 2: The overall framework of our proposed `LatentWeave`.

Test-time computation and adaptation. Additional inference computation can target visual evidence, reasoning traces, or model parameters. TTSP explores visual-tool trajectories and maintains a correctable evidence ledger (Jiang et al., 2026b). Trajectory-guided structured sampling retrieves reasoning patterns and refines discrete traces through structured MCMC (Jiang et al., 2026a). These approaches improve explicit evidence or generated sequences. Parameter adaptation is a separate axis: TTRL updates model parameters using rewards estimated from unlabeled test data (Zuo et al., 2025), while ReLAT adapts temporary LoRA parameters using a question-reconstruction objective for latent collaboration (Yuan et al., 2026). Thus, test-time adaptation need not preserve model parameters. `LatentWeave` instead updates instance-specific hidden states while keeping the evaluated backbone and the offline-trained anchor generator fixed during evaluation.

Continuous test-time optimization. LatentSeek applies policy-gradient updates to hidden states (Li et al., 2025a), and LatentEvolve incorporates retrieval and consolidation of prior latent optimizations (Zhang et al., 2025). More recently, ∇-Reasoner optimizes continuous token logits under reward and likelihood objectives during iterative decoding (Wang et al., 2026); its optimization variables are relaxed output representations. In multimodal models, Zhang et al. (2026) combine visual alignment with reward-guided optimization of reserved latent slots before answer decoding, keeping the backbone frozen. These are direct predecessors for inference-time continuous refinement. `LatentWeave` focuses on sequentially optimizing hidden-state segments of a multimodal generation trajectory, preserving accepted segments and regenerating the suffix after each update, with textual reasoning anchors guiding the generation of the remaining suffix.

Learned guidance and structural supervision. PILOT trains an auxiliary network to produce a query-conditioned latent anchor for a frozen language model (Zheng et al., 2026). ACTS trains a controller that supplies strategies and starter phrases to a frozen reasoner at successive reasoning steps (Xia et al., 2026). PMPS uses shared reasoning prototypes to align latent and explicit chains of thought during training (Li et al., 2026a). These works establish precedents for latent anchoring, textual steering, and structural supervision. We combine segment-wise multimodal hidden-state optimization with an offline-trained textual guidance mechanism: WEAVEGen learns from optimized training trajectories, generates one textual anchor sequence per input before refinement, and conditions the evaluated backbone through ordinary text. This interface differs from PILOT's latent injection and ACTS's online stepwise controller, while online optimization refines the backbone's hidden states and keeps all auxiliary-network parameters fixed throughout evaluation.

3 Methodology

The overall workflow of our method is illustrated in Figure 2. Without updating the evaluated backbone parameters online, `LatentWeave` operates directly in the hidden-state space to incrementally optimize reasoning-critical regions. This latent-level control allows the reasoning process to be adjusted at inference time while preserving the pretrained backbone. Section 3.1 introduces the autoregressive formulation and latent optimization. Section 3.2 presents ***Latent Weaving***, which is used during offline trajectory collection and evaluation-time refinement. Section 3.3 describes the reward signals guiding latent weaving, and Section 3.4 introduces ***Reasoning Anchors***, an offline-trained

auxiliary generator whose textual output conditions the evaluated backbone. The overall procedure is summarized in Algorithm 1 in Appendix C, which separates offline trajectory collection (Phase I), offline anchor training (Phase II), and evaluation-time latent refinement (Phase III). Additional implementation details, notations, and experimental configurations are provided in Appendix A.

3.1 PRELIMINARY

We begin by introducing the multimodal autoregressive formulation and the latent representations optimized in subsequent sections, using vision–language backbones such as Qwen2.5-VL (Bai et al., 2025). This formulation provides the foundation for the latent trajectory manipulation performed during offline collection and evaluation-time refinement in Algorithm 1. The notation is summarized in Table 4 in Appendix A.

Latent Formulation. We study multimodal autoregressive generation given textual input $\mathbf{x}$ and visual context $\mathbf{v}$. Given a pretrained model f_θ, decoding runs token by token, producing intermediate hidden states that encode the multimodal inputs and the generated prefix. In `LatentWeave`, these hidden states are viewed as latent variables and explicitly optimized during inference.

At decoding step t, the model computes a hidden state $\mathbf{h}_t$ from the input modalities and previously generated tokens, and predicts the next-token distribution as:

$$\begin{aligned} \mathbf{h}_t &= f_\theta(\mathbf{x}, \mathbf{v}, \mathbf{y}_{<t}), \\ p_\theta(\cdot \mid \mathbf{x}, \mathbf{v}, \mathbf{y}_{<t}) &= \mathrm{softmax}(\mathbf{W}_o \mathbf{h}_t), \end{aligned} \tag{1}$$

where f_θ denotes the pretrained vision–language model and $\mathbf{W}_o$ is the output projection matrix. Unlike standard decoding, `LatentWeave` exposes $\mathbf{h}_t$ as manipulable latent variables rather than treating them as fixed activations throughout generation.

The probability of an output sequence $\mathbf{y} = (y_1, \ldots, y_T)$ factorizes autoregressively as:

$$p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{v}) = \prod_{t=1}^{T} p_\theta(y_t \mid \mathbf{x}, \mathbf{v}, \mathbf{y}_{<t}). \tag{2}$$

Test-Time Latent Optimization. Rather than updating model parameters at inference time, we directly operate on the latent hidden-state sequence $\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_T)$ defined in Eq. (1). We test whether refining these states can correct reasoning errors with the backbone fixed. The objective is to find a latent trajectory maximizing the task-specific reward of its generated output:

$$\mathbf{H}^\star = \arg\max_{\mathbf{H} \in \mathcal{H}} R(f_{\mathrm{gen}}(\mathbf{H}, \mathbf{x}, \mathbf{v}; \theta)), \tag{3}$$

where $\mathcal{H}$ denotes the space of latent trajectories with dimensions compatible with the pretrained model, and f_{gen} performs deterministic decoding given a fixed latent sequence. The optimization operates solely on $\mathbf{H}$ while keeping θ fixed, enabling reward-guided refinement at test time without retraining. The fixed generator f_{gen} defines how a candidate latent sequence is decoded; the quadratic anchoring term in Eq. (5) penalizes deviations from the initial backbone-induced trajectory. Algorithm 1 applies this objective sequentially to each active latent segment.

3.2 LATENT WEAVING: SEQUENTIAL SEGMENT OPTIMIZATION

Building on the latent optimization objective in Eq. (3), we introduce a structured strategy to optimize long latent trajectories efficiently and stably. The same segment procedure is used in the offline collection stage and in evaluation-time refinement (Phases I and III of Algorithm 1), with ground-truth reward restricted to offline collection and verifier-only reward used at evaluation.

Structured Latent Optimization. Optimizing the entire latent sequence $\mathbf{H}$ couples many decoding positions and can be costly, while early perturbations may propagate through later generation. We therefore propose *Latent Weaving*, which decomposes latent optimization into localized, sequential segments. This decomposition allows each portion of the latent trajectory to be refined while leaving the reasoning states in all previously accepted segments unchanged.

Specifically, we partition the generation timeline into contiguous segments $\mathcal{S}_i$:

$$\mathcal{S}_i = \{s_i, s_i + 1, \ldots, e_i\}, \qquad |\mathcal{S}_i| = \min\{\max(1, \lfloor \rho T \rfloor), T - s_i + 1\}, \tag{4}$$

where T is the current generation length and $\rho \in (0, 1)$ controls the nominal segment length. Contiguous segments cover the trajectory in temporal order, with $s_1 = 1$, $s_{i+1} = e_i + 1$, $N_{\mathrm{seg}} = \lceil T / \max(1, \lfloor \rho T \rfloor) \rceil$, and a possibly shorter final segment. Appendix A.2 specifies the four-segment configuration used in the component comparison.

Segment-wise Optimization Objective. For each segment, only the corresponding hidden states receive gradient updates; the accepted prefix remains fixed, and the suffix is regenerated after the update. The optimization objective uses a score-function estimator (Williams, 1992) with a reward baseline, following the sequence-level training principle of Rennie et al. (2017):

$$\mathcal{L}_{\text{opt}}(\mathbf{H}_{\mathcal{S}_i}) = -\frac{1}{M}\sum_{m=1}^{M} A^{(m)} \sum_{t\in\mathcal{S}_i} \log p_\theta\left(y_t^{(m)} \mid \mathbf{h}_t^{(m)}\right) + \frac{\lambda}{2}\left\|\mathbf{H}_{\mathcal{S}_i} - \mathbf{H}_{\mathcal{S}_i}^{(0)}\right\|_2^2, \tag{5}$$

where $M \geq 2$ stochastic inner-loop rollouts are sampled at temperature τ, and $A^{(m)} = R(\mathbf{y}^{(m)}) - b^{(m)}$ is a centered advantage. The leave-one-out baseline is $b^{(m)} = (M-1)^{-1}\sum_{n\neq m} R(\mathbf{y}^{(n)})$. The superscript m indexes a rollout and its hidden states. Rewards and baselines are treated as scalar feedback when differentiating the log probabilities; gradients update the active latent variables while the backbone and verifier remain fixed. The regularizer anchors the optimized states to the backbone-induced trajectory. Stochastic rollouts supply the optimization signal, whereas the final benchmark answer is decoded deterministically. After each inner update, the suffix beginning at the active segment is re-decoded autoregressively, while the prefix cache and previously accepted segments remain fixed. Appendix A.2 reports M, τ, and λ and the re-decoding policy.

The hidden states are updated by gradient descent on Eq. (5):

$$\mathbf{H}_{\mathcal{S}_i}^{(k+1)} = \mathbf{H}_{\mathcal{S}_i}^{(k)} - \eta\nabla_{\mathbf{H}_{\mathcal{S}_i}}\mathcal{L}_{\text{opt}}\left(\mathbf{H}_{\mathcal{S}_i}^{(k)}\right). \tag{6}$$

Sequential Weaving. Segments are optimized in temporal order. When optimizing the i-th segment, all previously optimized segments are fixed and act as latent context. This freezing mechanism preserves accepted prefix states, although the reward of the completed answer can still change as the suffix is refined. Formally, the active segment is optimized under the following fixed-prefix constraint:

$$\mathbf{H}_{\mathcal{S}_i}^{\star} = \arg\max_{\mathbf{H}_{\mathcal{S}_i}} R\big(f_{\text{gen}}\big(\mathbf{H}_{\mathcal{S}_{<i}}^{\star}, \mathbf{H}_{\mathcal{S}_i}, \mathbf{H}_{\text{ctx},i}; \theta\big)\big), \tag{7}$$

where $\mathbf{H}_{\mathcal{S}_{<i}}^{\star}$ denotes previously optimized segments and $\mathbf{H}_{\text{ctx},i}$ denotes the remaining states, which receive no direct gradient update in the current step. The regenerated continuation can nevertheless change through autoregressive decoding. The input $(\mathbf{x}, \mathbf{v})$ is implicit in this expression. Algorithm 1 summarizes the segment-wise freezing and regeneration schedule.

3.3 Optimization Signals for Latent Weaving

We describe the reward signals for latent weaving. During offline trajectory collection (Phase I), the declared cold-start term may be used on the training split; during evaluation-time refinement (Phase III), only verifier rewards are permitted. In both cases, the scalar feedback drives gradient-based updates of the active segment while the backbone remains fixed.

Reward-Guided Control. The reward combines structural verification with a weak correctness signal during offline collection, and uses verification alone during evaluation:

$$R(\mathbf{y}) = \begin{cases} \alpha\, R_{\text{ver}}(\mathbf{y}, \mathbf{x}, \mathbf{v}) + \beta\, R_{\text{gt}}(\mathbf{y}, \mathbf{y}^{\star}), & \text{cold-start}, \\ \alpha\, R_{\text{ver}}(\mathbf{y}, \mathbf{x}, \mathbf{v}), & \text{test-time}, \end{cases} \tag{8}$$

where α scales verifier feedback and β controls the ground-truth term. Cold-start collection uses disjoint offline training data; development and test evaluation set $\beta = 0$ and use verifier feedback alone. Appendix A.1 specifies the verifier composition and reward weights.

Verification Reward. The verifier reward aggregates four complementary checks: reasoning consistency, final-answer correctness, answer completeness, and question understanding. The correctness check judges the proposed answer from the input and visual evidence; the separate ground-truth term compares against a reference answer during offline collection. With nonnegative weights and a positive total weight, the normalized verifier reward lies in $[-1, 0]$, and a larger value indicates fewer failed checks:

$$R_{\text{ver}} = -\frac{\sum_{k\in\mathcal{F}} w_k \mathbb{1}[k(\mathbf{y}) = 0]}{\sum_{k\in\mathcal{F}} w_k}, \tag{9}$$

where $\mathcal{F}$ is a predefined set of fixed verification functions for the current task, and w_k denotes the relative importance of each individual verifier. These verifiers provide structured, task-specific feedback without updating the evaluated backbone parameters. Appendix A.4 lists the complete prompt templates for each of the four task-specific verifier checks used during optimization.

Table 1: **Main results** under two MLLM backbones across mathematical reasoning, visual QA, and diagram understanding. The best and second-best results per benchmark are shown in **bold** and underline, respectively.

Method	Mathematical Reasoning			General Visual QA			Diagram
	Math Vision	Math Verse	Math Vista	MMMU	RealWorld QA	MME	AI2D
InternVL2.5-4B							
Vanilla	19.58	34.49	60.17	46.82	61.04	2316	79.63
CoT	20.93$_{\uparrow 1.35}$	35.95$_{\uparrow 1.46}$	60.81$_{\uparrow 0.64}$	48.22$_{\uparrow 1.40}$	62.19$_{\uparrow 1.15}$	2331$_{\uparrow 15.0}$	80.10$_{\uparrow 0.47}$
SFT	19.28$_{\downarrow 0.30}$	35.13$_{\uparrow 0.64}$	60.82$_{\uparrow 0.65}$	49.51$_{\uparrow 2.69}$	63.29$_{\uparrow 2.25}$	2309$_{\downarrow 7.0}$	80.87$_{\uparrow 1.24}$
GRPO	20.07$_{\uparrow 0.49}$	35.66$_{\uparrow 1.17}$	61.34$_{\uparrow 1.17}$	48.86$_{\uparrow 2.04}$	61.35$_{\uparrow 0.31}$	2302$_{\downarrow 14.0}$	78.41$_{\downarrow 1.22}$
Self-Rewarding	21.32$_{\uparrow 1.74}$	36.17$_{\uparrow 1.68}$	60.67$_{\uparrow 0.50}$	49.42$_{\uparrow 2.60}$	62.90$_{\uparrow 1.86}$	2315$_{\downarrow 1.0}$	81.04$_{\uparrow 1.41}$
IVT-LR[†]	22.31$_{\uparrow 2.73}$	36.19$_{\uparrow 1.70}$	61.20$_{\uparrow 1.03}$	49.68$_{\uparrow 2.86}$	66.41$_{\uparrow 5.37}$	2333$_{\uparrow 17.0}$	82.12$_{\uparrow 2.49}$
LaCoT (NeurIPS 2025)	20.87$_{\uparrow 1.29}$	41.23$_{\uparrow 6.74}$	59.34$_{\downarrow 0.83}$	50.45$_{\uparrow 3.63}$	64.28$_{\uparrow 3.24}$	2326$_{\uparrow 10.0}$	81.52$_{\uparrow 1.89}$
Self-Refine	23.38$_{\uparrow 3.80}$	38.88$_{\uparrow 4.39}$	62.72$_{\uparrow 2.55}$	50.91$_{\uparrow 4.09}$	62.78$_{\uparrow 1.74}$	2331$_{\uparrow 15.0}$	82.48$_{\uparrow 2.85}$
Self-Consistency	23.06$_{\uparrow 3.48}$	37.89$_{\uparrow 3.40}$	61.62$_{\uparrow 1.45}$	51.02$_{\uparrow 4.20}$	63.08$_{\uparrow 2.04}$	2329$_{\uparrow 13.0}$	81.05$_{\uparrow 1.42}$
TTRV (CVPR 2026)	25.30$_{\uparrow 5.72}$	40.12$_{\uparrow 5.63}$	64.14$_{\uparrow 3.97}$	**52.29**$_{\uparrow 5.47}$	66.02$_{\uparrow 4.98}$	2338$_{\uparrow 22.0}$	82.17$_{\uparrow 2.54}$
LatentWeave	**26.59**$_{\uparrow 7.01}$	**43.22**$_{\uparrow 8.73}$	**66.19**$_{\uparrow 6.02}$	52.14$_{\uparrow 5.32}$	**67.80**$_{\uparrow 6.76}$	**2345**$_{\uparrow 29.0}$	**84.64**$_{\uparrow 5.01}$
Qwen2.5-VL-7B							
Vanilla	23.65	47.98	66.15	56.96	66.02	2328	80.58
CoT	24.38$_{\uparrow 0.73}$	48.66$_{\uparrow 0.68}$	66.91$_{\uparrow 0.76}$	57.92$_{\uparrow 0.96}$	67.35$_{\uparrow 1.33}$	2344$_{\uparrow 16.0}$	82.98$_{\uparrow 2.40}$
SFT	22.81$_{\downarrow 0.84}$	47.03$_{\downarrow 0.95}$	66.13$_{\downarrow 0.02}$	57.18$_{\uparrow 0.22}$	65.95$_{\downarrow 0.07}$	2338$_{\uparrow 10.0}$	81.49$_{\uparrow 0.91}$
GRPO	23.69$_{\uparrow 0.04}$	48.55$_{\uparrow 0.57}$	67.70$_{\uparrow 1.55}$	58.61$_{\uparrow 1.65}$	67.36$_{\uparrow 1.34}$	2340$_{\uparrow 12.0}$	81.65$_{\uparrow 1.07}$
Self-Rewarding	23.92$_{\uparrow 0.27}$	48.47$_{\uparrow 0.49}$	68.14$_{\uparrow 1.99}$	59.44$_{\uparrow 2.48}$	66.99$_{\uparrow 0.97}$	2356$_{\uparrow 28.0}$	82.12$_{\uparrow 1.54}$
IVT-LR[†]	26.36$_{\uparrow 2.71}$	49.48$_{\uparrow 1.50}$	67.26$_{\uparrow 1.11}$	59.43$_{\uparrow 2.47}$	69.30$_{\uparrow 3.28}$	2351$_{\uparrow 23.0}$	84.02$_{\uparrow 3.44}$
LaCoT (NeurIPS 2025)	27.85$_{\uparrow 4.20}$	48.70$_{\uparrow 0.72}$	67.59$_{\uparrow 1.44}$	58.32$_{\uparrow 1.36}$	70.14$_{\uparrow 4.12}$	2368$_{\uparrow 40.0}$	83.31$_{\uparrow 2.73}$
Self-Refine	24.40$_{\uparrow 0.75}$	49.53$_{\uparrow 1.55}$	68.16$_{\uparrow 2.01}$	57.97$_{\uparrow 1.01}$	68.20$_{\uparrow 2.18}$	2355$_{\uparrow 27.0}$	86.30$_{\uparrow 5.72}$
Self-Consistency	25.01$_{\uparrow 1.36}$	50.82$_{\uparrow 2.84}$	69.78$_{\uparrow 3.63}$	58.57$_{\uparrow 1.61}$	67.96$_{\uparrow 1.94}$	2358$_{\uparrow 30.0}$	84.94$_{\uparrow 4.36}$
TTRV (CVPR 2026)	27.08$_{\uparrow 3.43}$	51.76$_{\uparrow 3.78}$	69.26$_{\uparrow 3.11}$	61.40$_{\uparrow 4.44}$	**71.28**$_{\uparrow 5.26}$	2375$_{\uparrow 47.0}$	86.21$_{\uparrow 5.63}$
LatentWeave	**30.11**$_{\uparrow 6.46}$	**53.80**$_{\uparrow 5.82}$	**72.03**$_{\uparrow 5.88}$	**63.15**$_{\uparrow 6.19}$	70.89$_{\uparrow 4.87}$	**2388**$_{\uparrow 60.0}$	**88.93**$_{\uparrow 8.35}$

Ground-Truth Penalty. The auxiliary correctness signal is defined as:

$$R_{\text{gt}}(\mathbf{y}, \mathbf{y}^\star) = \mathbb{1}[\text{Ans}(\mathbf{y}) \neq \mathbf{y}^\star] \cdot (-1), \tag{10}$$

where Ans extracts the benchmark-formatted final answer from the generated reasoning trace. An answer mismatch incurs -1, and a match incurs 0; this signal is used only on the offline training split.

3.4 Reasoning Anchors for Latent Weaving

Finally, we introduce *Reasoning Anchors*, an offline-trained mechanism that complements test-time latent weaving with explicit reasoning structure. This procedure corresponds to Phase II in Algorithm 1 and is applied after collecting optimized trajectories on the training split. The trajectories are converted to textual reasoning traces and bound to explicit <WEAVE> delimiters. At inference, WEAVEGen produces one anchor sequence from $(\mathbf{x}, \mathbf{v})$; that text is inserted into the evaluated backbone's context before latent weaving. The interface is textual, so WEAVEGen and the evaluated backbone need not share hidden size or parameters.

WEAVEGen Model. WEAVEGen is a compact auxiliary generator parameterized by ϕ. It is initialized from `Qwen2-VL-2B-Instruct` (Wang et al., 2024b) and trained separately to generate structured textual reasoning. Building on explicit intermediate-step supervision (Wei et al., 2022; Nye et al., 2021), we serialize each step between <WEAVE> delimiters. This 2B auxiliary model has its own parameters and tokenizer; its generated text is consumed through the ordinary language-token interface of the evaluated 4B–27B backbone f_θ. Appendix B.2 accounts for the tokens, latency, and memory of its single auxiliary generation call.

$$\mathcal{L}_{\text{bind}} = -\sum_{j=1}^{J} \log p_\phi\big(\text{step}_j \mid \mathbf{x}, \mathbf{v}, \mathbf{a}_{<j}\big)\,, \tag{11}$$

[†]IVT-LR: ModalityDance Research, 2026-1-29 (project repository).

Table 2: **Performance gains** of `LatentWeave` over vanilla decoding across five MLLM backbones on diverse multimodal reasoning benchmarks. Δ denotes the absolute improvement on each benchmark.

Backbone	Method	MathVision	MathVerse	MathVista	MMMU	RealWorldQA	MME	AI2D
`Qwen2.5-VL-7B`	Vanilla	23.65	47.98	66.15	56.96	66.02	2328	80.58
	+`LatentWeave`	30.11	53.80	72.03	63.15	70.89	2388	88.93
	Δ	+6.46	+5.82	+5.88	+6.19	+4.87	+60.0	+8.35
`Qwen3.5-4B`	Vanilla	66.12	66.37	75.80	68.32	73.33	2372	85.91
	+`LatentWeave`	72.69	71.44	79.30	72.81	76.71	2404	89.25
	Δ	+6.57	+5.07	+3.50	+4.49	+3.38	+32.0	+3.34
`Qwen3.6-27B`	Vanilla	75.33	74.36	82.30	72.56	77.12	2456	90.31
	+`LatentWeave`	81.90	78.04	85.90	76.38	82.34	2482	92.48
	Δ	+6.57	+3.68	+3.60	+3.82	+5.22	+26.0	+2.17
`InternVL2.5-4B`	Vanilla	19.58	34.49	60.17	46.82	61.04	2316	79.63
	+`LatentWeave`	26.59	43.22	66.19	52.14	67.80	2345	84.64
	Δ	+7.01	+8.73	+6.02	+5.32	+6.76	+29.0	+5.01
`STEP3-VL-10B`	Vanilla	68.77	67.20	80.59	75.12	73.45	2432	87.29
	+`LatentWeave`	70.29	72.33	81.98	78.64	77.70	2440	89.93
	Δ	+1.52	+5.13	+1.39	+3.52	+4.25	+8.0	+2.64

where J is the number of anchor steps and $\mathbf{a}_{<j}$ is their serialized text prefix, including `<WEAVE>` delimiters. The step likelihood under WEAVEGen, p_ϕ, is factorized over the tokens of step_j. Here J is distinct from the rollout count M and the update count K.

To encourage WEAVEGen to encode answer-relevant information in its anchors, we introduce an auxiliary answer-prediction objective:

$$\mathcal{L}_{\text{util}} = -\mathbb{E} \log p_\phi(\mathbf{y}^\star \mid \mathbf{x}, \mathbf{v}, \mathbf{a}) \,. \tag{12}$$

Here $\mathbf{a}$ is the serialized anchor sequence and the expectation is over offline training examples and their anchor targets. Together, the objectives train WEAVEGen to emit a structured textual scaffold that conditions the backbone before latent optimization, without transferring hidden states. Appendix A.3 specifies serialization, insertion position, anchor length, and caching.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of `LatentWeave` for multimodal reasoning, organized around explicit research questions to analyze component contributions and reasoning behaviors. Specifically, we aim to answer the following research questions:

- **RQ1.** How well does `LatentWeave` improve multimodal reasoning compared to baselines?
- **RQ2.** How do Latent Weaving and Reasoning Anchors affect `LatentWeave`'s performance?
- **RQ3.** How sensitive is `LatentWeave` to key design choices and hyperparameters?

4.1 Experiment Setup

Tasks & Benchmarks. We evaluate three task categories: ◆ *Mathematical reasoning*, covering MathVision (Wang et al., 2024a), MathVerse (Zhang et al., 2024), and MathVista (Lu et al., 2023); ◆ *General Visual QA*, including MMMU (Yue et al., 2024), RealWorldQA (X.AI, 2024), and MME (Fu et al., 2023); ◆ *Diagram Understanding*, evaluated on AI2D (Kembhavi et al., 2016). We report accuracy (%) for all benchmarks except MME, which uses its aggregate benchmark score. Accordingly, accuracy gains are expressed in percentage points and MME gains in score points. Appendix A.6 lists the evaluation splits and dataset sizes.

Baselines. `LatentWeave` is compared against key methods across different reasoning paradigms:

- **Prompt-based Inference**: Vanilla decoding and Chain-of-Thought (CoT) (Wei et al., 2022).
- **Learning-based Optimization**: supervised fine-tuning (SFT), Group Relative Policy Optimization (GRPO) (Shao et al., 2024), and Self-Rewarding (Yuan et al., 2024).
- **Latent-space Reasoning**: methods operating on implicit reasoning representations, including IVT-LR (Chen et al., 2025) and LaCoT (Sun et al., 2025).
- **Test-time Scaling and Adaptation**: sampling and refinement through Self-Consistency (Wang et al., 2023), Self-Refine (Madaan et al., 2023), and same-verifier Best-of-N; and test-time parameter updates through TTRV (Singh et al., 2026), which uses frequency- and entropy-based rewards.

Backbones. To assess `LatentWeave`'s generalization capability, we conduct experiments on MLLMs drawn from several diverse model families, including Qwen2.5-VL-7B-Instruct (Bai et al.,

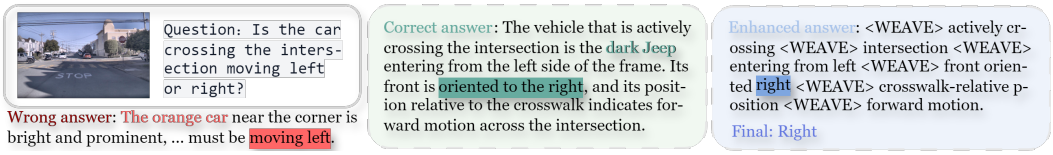

Figure 3: Qualitative case study of reasoning error accumulation and correction. **Left**: Incorrect early reasoning leads to error propagation. **Middle**: Latent Weaving corrects the reasoning trajectory. **Right**: <WEAVE> tokens enhance the clarity and structure of the corrected reasoning trajectory.

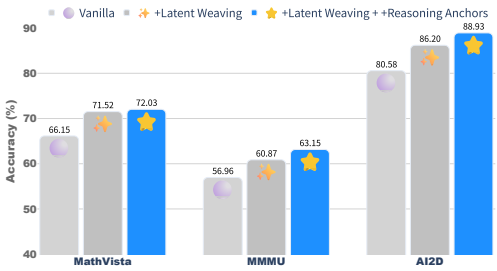

Figure 4: Component-wise performance comparison with Latent Weaving and Reasoning Anchors.

Table 3: Latent Weaving by stage, measured in *reasoning steps* of the generated chain of thought.

Weaving Steps	MathVista	MMMU
Vanilla	66.15	56.96
Early (first 30%)	67.02	57.41
Middle only	64.47	54.92
Late (last 30%)	58.85	51.31
Full trajectory	**72.03**	**63.15**

2025), Qwen3.5-4B (Qwen Team, 2026a), Qwen3.6-27B (Qwen Team, 2026b), InternVL2.5-4B (Chen et al., 2024), and STEP3-VL-10B (Huang et al., 2026).

Implementation details. Final benchmark answers use deterministic decoding with maximum length 1024. Inner-loop optimization instead uses the stochastic estimator in Eq. (5); its rollout count, temperature, baseline, clipping, regularization, segment coverage, update count, and suffix re-decoding policy are reported per configuration in Appendix A.2. Ground-truth reward is restricted to offline trajectory collection on a training split and is disabled for every evaluation example. WEAVEGen is trained offline with $\mathcal{L}_{\text{bind}} + \mathcal{L}_{\text{util}}$ and supplies textual anchors through the interface in Appendix A.3. In the four-segment component comparison, Best-of-N and the latent variants each use 96 candidate-level verifier evaluations; their tokens, latency, and peak memory are reported in Appendix B.2. Experiments use one machine with 8 NVIDIA A100 (80GB) GPUs.

4.2 Performance and Main Results (RQ1)

Consistent improvements across task domains. Table 1 shows that **LatentWeave** improves over vanilla decoding across all evaluated task categories. With `InternVL2.5-4B`, accuracy increases by 7.01, 8.73, and 6.02 percentage points on MathVision, MathVerse, and MathVista, respectively, and by 5.32, 6.76, and 5.01 points on MMMU, RealWorldQA, and AI2D. With `Qwen2.5-VL-7B`, all three mathematical benchmarks improve by more than five percentage points, alongside gains of 6.19 points on MMMU and 8.35 on AI2D. The method improves every evaluated benchmark over its corresponding vanilla baseline, although individual competing methods remain stronger on some tasks, such as TTRV on Qwen2.5-VL-7B RealWorldQA.

Robust generalization across model backbones. As summarized in Table 2, **LatentWeave** improves all seven benchmarks across five MLLM backbones spanning 4B to 27B parameters. On MathVision, gains range from +1.52 percentage points (`STEP3-VL-10B`) to +7.01 (`InternVL2.5-4B`), with +6.57 for `Qwen3.5-4B` and +6.57 for `Qwen3.6-27B`. Gains also remain positive on MMMU (+3.52 to +6.19 points), RealWorldQA (+3.38 to +6.76 points), and AI2D (+2.17 to +8.35 points), while MME increases by 8 to 60 score points. These results extend the observed benefits to newer backbones and a larger 27B model, supporting the applicability of latent trajectory refinement across the evaluated model families and scales.

Effectiveness under strong baseline performance. The improvements achieved by **LatentWeave** persist even when Vanilla performance is already strong. On `Qwen3.6-27B`, MathVision rises from 75.33% to 81.90% (+6.57 percentage points), while RealWorldQA increases from 77.12% to 82.34% (+5.22 points). Even on AI2D, where Vanilla reaches 90.31%, **LatentWeave** achieves 92.48% (+2.17 points). The smaller `Qwen3.5-4B` also improves from 66.12% to 72.69% on MathVision and from 75.80% to 79.30% on MathVista. Together with the gains on `STEP3-VL-10B`, these results indicate that the benefits are not confined to backbones with low initial accuracy, although the size of the gain varies across models and benchmarks.

4.3 Component Contribution (RQ2)

Effect of Latent Weaving. Figure 4 compares the component configurations. Latent Weaving alone reaches 71.52% on MathVista, 60.87% on MMMU, and 86.20% on AI2D, improving on Vanilla across all three benchmarks. On MathVista, it exceeds the same-verifier Best-of-N result of 69.44% by 2.08 percentage points. Table 11 reports both methods at a shared budget of 96 candidate-level verifier evaluations.

Complementary role of Reasoning Anchors. Reasoning Anchors are evaluated separately from latent updates. The 2×2 design in Table 9 shows that adding anchors to latent updates raises MathVista from 71.52% to 72.03% (+0.51 points), MMMU from 60.87% to 63.15% (+2.28 points), and AI2D from 86.20% to 88.93% (+2.73 points). The full variant has the highest reported point estimate on all three benchmarks.

Accuracy and inference cost. The same-verifier comparison separates candidate-evaluation budget from execution cost. Best-of-N selects among independently generated candidates, whereas Latent Weaving uses feedback to update the active segment before generating new suffixes. At the shared 96-candidate budget, the full method exceeds Best-of-N by 2.59 points on MathVista. Table 10 reports 52 versus 31 seconds per example and 41 versus 21 GB peak memory on the cost subset. These measurements include the full method's auxiliary anchor-generation call and quantify the resource cost associated with its accuracy gain.

4.4 Sensitivity Analysis (RQ3)

We examine the segment ratio ρ, anchor count J, update count K, and verifier weights. Table 14 transcribes the MathVista and RealWorldQA curves in Figure 5, whose accuracy spreads are at most two percentage points. Each one-factor series compares settings within its reported sweep. The separate delimiter control in Table 14 and the weight study in Table 15 report MathVista and MMMU. Hyperparameters are selected on a development split and frozen before test evaluation.

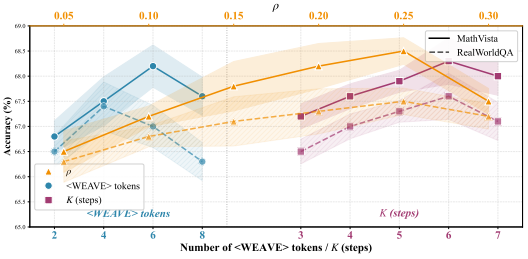

Figure 5: Sensitivity to segment ratio ρ, anchor-step count J (one <WEAVE> opening delimiter per step), and update count K on MathVista and RealWorldQA.

In addition to hyperparameter sensitivity, Table 3 compares intervention at different stages of a generated reasoning trajectory. These stages refer to reasoning steps, not the inner update count K. Early-only weaving improves MathVista and MMMU by 0.87 and 0.45 percentage points over vanilla decoding, whereas middle-only and late-only weaving reduce accuracy. Full-trajectory weaving performs best on both benchmarks, highlighting the role of early intervention in the sequential procedure. Additional ablations and sensitivity analyses are reported in Appendix B.

4.5 Case Study

Figure 3 illustrates how an early reasoning error affects the subsequent trace. The initial inference leads to a wrong final prediction (left), while Latent Weaving revises the active reasoning segment and recovers the correct answer (middle). The anchor-conditioned trace presents the corrected reasoning in explicit <WEAVE>-delimited steps (right). This example connects the latent intervention to the resulting textual continuation and illustrates the distinct roles of trajectory refinement and structured guidance.

5 Conclusion

In this paper, we study **LatentWeave**, a frozen-backbone test-time optimization framework that models multimodal reasoning as a controllable latent trajectory. Structured *Latent Weaving* refines active hidden-state segments while *Reasoning Anchors* provide textual guidance during generation. Across seven benchmarks and five backbones, the results support combining localized latent refinement with textual guidance. In the four-segment component comparison, Best-of-N and the latent variants share the verifier and a 96-candidate evaluation budget, with resource costs reported separately. These results motivate direct control of the hidden-state trajectory that produces the final answer.

AI USE STATEMENT

Writing assistance from language models was limited to polishing grammar and wording in author-written prose and locating a subset of the papers discussed in Related Work. AI coding assistants helped convert author-provided Excel data into LaTeX format for Tables 2 and 3, and produce the bar chart in Figure 4 and the line plot in Figure 5 from author-provided data. AI tools were not used at any stage to generate research ideas, define research objectives, run or interpret experiments, or generate any reported numerical results. The authors retain responsibility for the final manuscript.

ETHICS STATEMENT

This work studies test-time latent optimization for multimodal reasoning and is conducted entirely on publicly available benchmarks (MathVision, MathVerse, MathVista, MMMU, RealWorldQA, MME, and AI2D) under their released evaluation protocols. No new data are collected from human subjects, and no sensitive, private, or proprietary information is used. The proposed framework keeps the evaluated backbone parameters fixed, but this does not guarantee that its output biases remain unchanged. Residual risks include overreliance on automated reasoning outputs and verifier-induced bias. Appendix B.3 reports the verifier reliability and bias audit.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 describe latent optimization, anchor training, and inference. Section 4.1 summarizes the benchmarks, backbones, baselines, and evaluation setup. Appendix A provides reward definitions, optimization and training settings, and prompt templates. Appendix B reports baseline protocols, candidate budgets, computational costs, and ablation results. The anonymous code repository linked in the abstract provides the implementation.

REFERENCES

David Acuna, Ximing Lu, Jaehun Jung, Hyunwoo Kim, Amlan Kar, Sanja Fidler, and Yejin Choi. Socratic-mcts: Test-time visual reasoning by asking the right questions. *arXiv preprint arXiv:2506.08927*, 2025.

Ashutosh Adhikari and Mirella Lapata. Multimodal Latent Reasoning via Predictive Embeddings, 2026. URL https://arxiv.org/abs/2604.08065. arXiv preprint.

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*, 2025.

Chao Chen, Zhixin Ma, Yongqi Li, Yupeng Hu, Yinwei Wei, Wenjie Li, and Liqiang Nie. Reasoning in the dark: Interleaved vision-text reasoning in latent space. *arXiv preprint arXiv:2510.12603*, 2025.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

Zihui Cheng, Qiguang Chen, Jin Zhang, Hao Fei, Xiaocheng Feng, Wanxiang Che, Min Li, and Libo Qin. CoMT: A novel benchmark for chain of multi-modal thought on large vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23678–23686, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. MME: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023. URL https://arxiv.org/abs/2306.13394.

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, 2024. URL https://arxiv.org/abs/2412.06769. arXiv preprint.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah Smith, and Ranjay Krishna. Visual Sketchpad: Sketching as a Visual Chain of Thought for Multimodal Language Models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 139348–139379. Curran Associates, Inc., 2024. doi: 10.52202/079017-4423. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/fb82011040977c7712409fbdb5456647-Paper-Conference.pdf.

Ailin Huang, Chengyuan Yao, Chunrui Han, Fanqi Wan, Hangyu Guo, Haoran Lv, Hongyu Zhou, Jia Wang, Jian Zhou, Jianjian Sun, et al. STEP3-VL-10B technical report. *arXiv preprint arXiv:2601.09668*, 2026.

Tianbao Jiang, Weicong Ni, Gerard de Melo, and Linlin Wang. Aligning Large Vision-Language Models at Test Time: A Trajectory-Guided Structured Sampling Approach, 2026a. URL https://arxiv.org/abs/2608.03204. arXiv preprint.

Zheng Jiang, Yiming Chen, Nan He, Jiahui Chen, Chaoyang Li, Houde Qian, and Lifeng Sun. Test-time Scaling over Perception: Resolving the Grounding Paradox in Thinking with Images, 2026b. URL https://arxiv.org/abs/2604.11025. arXiv preprint.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. Seek in the Dark: Reasoning via Test-Time Instance-Level Policy Gradient in Latent Space, 2025a. URL https://arxiv.org/abs/2505.13308. arXiv preprint.

Weiqi Li, Xuanyu Zhang, Shijie Zhao, Yabin Zhang, Junlin Li, Li Zhang, and Jian Zhang. Q-Insight: Understanding image quality via visual reinforcement learning. *arXiv preprint arXiv:2503.22679*, 2025b.

Yiqi Li, Xu Chen, Chen Ju, Jiangchao Yao, Zhaoyang Li, Jinsong Lan, Xiaoyong Zhu, Bo Zheng, and Yu Wang. Structural Process Supervision for Latent Chain-of-Thought Reasoning, 2026a. URL https://arxiv.org/abs/2609.09928. arXiv preprint.

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, 2026b. URL https://arxiv.org/abs/2602.22766. arXiv preprint.

Yunxin Li, Zhenyu Liu, Zitao Li, Xuanyu Zhang, Zhenran Xu, Xinyu Chen, Haoyuan Shi, Shenyuan Jiang, Xintong Wang, Jifang Wang, et al. Perception, reason, think, and plan: A survey on large multimodal reasoning models. *arXiv preprint arXiv:2505.04921*, 2025c.

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. *arXiv preprint arXiv:2310.02255*, 2023.

Chuofan Ma, Yi Jiang, Junfeng Wu, Jihan Yang, Xin Yu, Zehuan Yuan, Bingyue Peng, and Xiaojuan Qi. UniTok: A unified tokenizer for visual generation and understanding. *arXiv preprint arXiv:2502.20321*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show Your Work: Scratchpads for Intermediate Computation with Language Models, 2021. URL https://arxiv.org/abs/2112.00114. arXiv preprint.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7008–7024, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Akshit Singh, Shyam Marjit, Wei Lin, Paul Gavrikov, Serena Yeung-Levy, Hilde Kuehne, Rogério Feris, Sivan Doveh, James Glass, and M Jehanzeb Mirza. TTRV: Test-time reinforcement learning for vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33153–33163, 2026.

Guohao Sun, Hang Hua, Jian Wang, Jiebo Luo, Sohail Dianat, Majid Rabbani, Raghuveer Rao, and Zhiqiang Tao. Latent chain-of-thought for visual reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 103739–103762. Curran Associates, Inc., 2025. doi: 10.52202/085713-3465. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/95c6ae3f3393786203a4b6dcb9df1036-Paper-Conference.pdf.

André G. Viveiros, Nuno Gonçalves, Matthias Lindemann, and André Martins. LanteRn: Latent Visual Structured Reasoning, 2026. URL https://arxiv.org/abs/2603.25629. arXiv preprint.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95095–95169, 2024a. doi: 10.52202/079017-3014. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad0edc7d5fa1a783f063646968b7315b-Abstract-Datasets_and_Benchmarks_Track.html.

Peihao Wang, Ruisi Cai, Zhen Wang, Hongyuan Mei, Qiang Liu, Pan Li, and Zhangyang Wang. ∇-Reasoner: LLM Reasoning via Test-Time Gradient Descent in Latent Space. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 45468–45490, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4b8001fc75f0532827472ea5a16af9ca-Paper-Conference.pdf.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution, 2024b. URL https://arxiv.org/abs/2409.12191.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Yaoting Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. Multimodal chain-of-thought reasoning: A comprehensive survey. *arXiv preprint arXiv:2503.12605*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 12966–12977, 2025.

X.AI. Grok-1.5 vision preview. https://x.ai/blog/grok-1.5v, 2024.

Yu Xia, Zhouhang Xie, Xin Xu, Byungkyu Kang, Prarit Lamba, Xiang Gao, and Julian McAuley. Agentic Chain-of-Thought Steering for Efficient and Controllable LLM Reasoning, 2026. URL https://arxiv.org/abs/2606.03965. arXiv preprint.

Guowei Xu, Peng Jin, Ziang Wu, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. LLaVA-CoT: Let vision language models reason step-by-step. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2087–2098, 2025.

Tianrun Xu, Yue Sun, Qixun Wang, Jingyi Lu, Yuan Wang, Tianren Zhang, Longteng Guo, Fengyun Rao, Jing Lyu, Feng Chen, and Jing Liu. Semantic-Enriched Latent Visual Reasoning, 2026. URL https://arxiv.org/abs/2605.19342. arXiv preprint.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of LMMs: Preliminary explorations with GPT-4V(ision). *arXiv preprint arXiv:2309.17421*, 2023.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Xiaopeng Yuan, Haibo Jin, Ye Yu, Peng Kuang, Lijun Yu, Yushun Dong, and Haohan Wang. Closing the Loop on Latent Reasoning via Test-Time Reconstruction, 2026. URL https://arxiv.org/abs/2606.06252. arXiv preprint.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9556–9567, 2024.

Guibin Zhang, Fanci Meng, Guancheng Wan, Zherui Li, Kun Wang, Zhenfei Yin, Lei Bai, and Shuicheng Yan. LatentEvolve: Self-Evolving Test-Time Scaling in Latent Space, 2025. URL https://arxiv.org/abs/2509.24771. arXiv preprint.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. MathVerse: Does your multi-modal LLM truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186. Springer, 2024.

Xin Zhang, Qiqi Tao, Jiawei Du, Moyun Liu, and Joey Tianyi Zhou. Visual Latents Know More Than They Say: Unsilencing Latent Reasoning in MLLMs, 2026. URL https://arxiv.org/abs/2605.02735. arXiv preprint.

Yusheng Zhao, Junyu Luo, Xiao Luo, Jinsheng Huang, Jingyang Yuan, Zhiping Xiao, and Ming Zhang. Attention bootstrapping for multi-modal test-time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22849–22857, 2025.

Haoyu Zheng, Yun Zhu, Yuqian Yuan, Bo Yuan, Wenqiao Zhang, Siliang Tang, and Jun Xiao. PILOT: Planning via Internalized Latent Optimization Trajectories for Large Language Models, 2026. URL https://arxiv.org/abs/2601.19917. arXiv preprint.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-Time Reinforcement Learning, 2025. URL https://arxiv.org/abs/2504.16084. arXiv preprint.

A Full Method and Experimental Configuration Details

Table 4: Notation summary for latent trajectories, reward signals, and reasoning anchors.

Symbol	Description
$\mathbf{x}$	Textual input (question or instruction)
$\mathbf{v}$	Visual input (image or diagram)
$\mathbf{y} = (y_1, \ldots, y_T)$	Generated output token sequence
$\mathbf{h}_t$	Hidden state at decoding step t
$\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_T)$	Latent reasoning trajectory composed of hidden states
f_θ	Pretrained vision–language model with frozen parameters θ
f_{gen}	Deterministic decoding function
$\mathcal{S}_i$	i-th latent segment in Latent Weaving
ρ	Segment length ratio
J, $\mathbf{a}$	Number of anchor steps and their serialized text sequence
M	Stochastic rollouts per inner update
K	Optimization steps per segment
η	Latent step size
$R(\mathbf{y})$	Overall reward
R_{gt}	Ground-truth reward
R_{ver}	Verifier reward
$\mathcal{F}$	Verifier set
w_k	Verifier weight
`<WEAVE>`	Textual reasoning-anchor delimiter
ϕ	WEAVEGen parameters

A.1 Reward Weighting and Verifier Configuration

The verifier reward uses the normalized aggregation in Eq. (9). We distinguish four checks: reasoning consistency (rc), final-answer correctness (ac), answer completeness (cp), and question understanding (uc). Each check contributes $R_k(\mathbf{y}) = -\mathbb{1}[k(\mathbf{y}) = 0]$. The coefficient α scales the normalized verifier reward, while the offline ground-truth term is added separately, exactly as in the main-text reward definition. The ground-truth coefficient β is permitted only during offline trajectory collection on a training split; it is exactly zero for development and test examples, as summarized in Table 5.

$$R_{\text{cfg}}(\mathbf{y}) = \alpha \frac{\sum_{k \in \{\text{rc,ac,cp,uc}\}} w_k R_k(\mathbf{y})}{\sum_k w_k} + \mathbb{1}[\text{offline}]\, \beta R_{\text{gt}}(\mathbf{y}). \tag{13}$$

Table 5: Default reward configuration for offline trajectory collection and evaluation. The verifier weight vector is shared across these settings; the ground-truth coefficient β is zero for every evaluation example. The weight-sensitivity study is reported separately in Table 15.

Phase / split	β	Verifier weights $(w_{\text{rc}}, w_{\text{ac}}, w_{\text{cp}}, w_{\text{uc}})$	Prompt / config ID	Status
Offline collection	0.3	$(0.30, 0.40, 0.15, 0.15)$	`rw-v1/offline`	Verified
Development	0	$(0.30, 0.40, 0.15, 0.15)$	`rw-v1/dev`	Verified
Test evaluation	0	$(0.30, 0.40, 0.15, 0.15)$	`rw-v1/test`	Verified

A.2 Optimization Estimator and Autoregressive Consistency

At every inner update of each active segment, the optimizer draws $M \geq 2$ fresh stochastic suffix rollouts, evaluates the verifier-only reward at test time, centers rewards with the declared baseline, and applies Eq. (5). After each update, tokens and hidden states from the segment start onward are regenerated autoregressively. Tokens before the segment and previously accepted latent segments remain fixed. The final answer is then decoded deterministically. Fresh sampling supplies the stochasticity required by the policy-gradient estimator; suffix regeneration maintains consistency between the updated hidden states and the regenerated continuation.

Table 6: Inner-loop optimization protocol for the four-segment configurations. clip bounds the latent gradient norm, $b^{(m)}$ is the leave-one-out baseline, and λ is the anchoring coefficient in Eq. (5). The realized segment count follows Eq. (4) and can differ for other generation lengths.

Backbone / split	Segmentation (ρ, N_{seg})	Rollouts (M, τ)	Updates (K, η, clip)	Baseline / regularization (b, λ)	Seed / ID
Qwen2.5-VL-7B / test	$(0.25,\ 4)$	$(4,\ 0.8)$	$(6,\ 1{\times}10^{-2},\ 1.0)$	LOO / 0.05	$\{0,1,2\}$ / `opt-v1`
InternVL2.5-4B / test	$(0.25,\ 4)$	$(4,\ 0.8)$	$(6,\ 1{\times}10^{-2},\ 1.0)$	LOO / 0.05	$\{0,1,2\}$ / `opt-v1`

Autoregressive rule: after every update, regenerate the active suffix; retain only the fixed prefix and previously accepted segments.

A.3 Reasoning-Anchor Training–Inference Interface

WEAVEGen uses `Qwen2-VL-2B-Instruct` as its initialization and communicates with the evaluated backbone only through text. Offline, accepted training trajectories are converted to anchor targets delimited by `<WEAVE>`. At inference, WEAVEGen produces one anchor sequence; it is serialized and inserted into the main backbone context before latent optimization. This makes the handoff independent of hidden size and exposes the auxiliary call to budget accounting. Table 7 specifies the serialization format, insertion position, and inference-time caching policy.

WEAVEGen training configuration. The default WEAVEGen training configuration uses full-parameter optimization of the auxiliary model, including its resized token embeddings, with AdamW and BF16 model weights. The settings are a learning rate of 2×10^{-5}, three epochs, batch size 2, maximum sequence length 2048, and seed 42. Each batch triggers one optimizer step, without gradient accumulation. The learning rate increases linearly over the first 10% of optimizer steps and then decays linearly to zero; the gradient-norm bound is 1.0. AdamW retains its library defaults: momentum coefficients $(0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 0.01. Table 6 separately specifies the seeds and update schedule for test-time latent optimization.

Table 7: Reasoning-anchor training–inference interface. The same serialization is used in training and inference.

Stage	Input $\rightarrow$ output	Serialization / insertion	Aux call / cache	Leakage / config
Anchor training	$(\mathbf{x}, \mathbf{v}) \rightarrow$ anchor text	Six spans: `<WEAVE>` step_j `</WEAVE>`, ≤ 80 tokens each	No / N/A	Train split only
Evaluation	$(\mathbf{x}, \mathbf{v}) \rightarrow$ anchor text	Identical string; inserted after visual tokens, before the CoT trigger	Once / no cross-example reuse	No labels / `anchor-v1`

A.4 Prompt Templates Used in Experiments

All verifier prompts are shown verbatim after removing implementation-specific wrappers.

Reasoning Consistency Check Prompt.

```
TASK:
Validate the logical coherence and factual accuracy of the reasoning
process presented in the proposed solution.
GUIDELINES:
You should carefully examine the entire reasoning process, with
particular attention to how visual information from the image is
interpreted and integrated.
1.  Extract all reasoning elements, including logical steps,
inferential decisions, and intermediate conclusions.
2.  Verify that each reasoning step follows logically from previous
steps and from the information provided in the question and the image.
```

```
3. Check the factual correctness of all stated concepts,
relationships, and interpretations, especially those related to
visible elements in the image.
4. Assess whether the reasoning process is internally consistent,
rigorous, and logically leads to the stated conclusion, without
relying on unsupported assumptions or misinterpretations of visual
content.
FINAL VERDICT (strict):
- If any logical flaw, factual error, internal inconsistency, or
mismatch between the reasoning and the image content is found, output
exactly: [VERA]False
- If the reasoning is logically coherent, factually correct, and
accurately reflects the image content, output exactly: [VERA]True
IMPORTANT:
Focus on the quality and validity of the reasoning process itself,
rather than on whether the final answer happens to be correct.
```

Final Answer Correctness Check Prompt.

```
TASK:
Determine whether the final answer selected in the proposed solution
is correct.
GUIDELINES:
You should evaluate only the final answer, not the reasoning steps
used to obtain it.
1. Verify whether the selected answer correctly addresses the
question, taking into account both the question text and the visual
information shown in the image.
2. For multiple-choice questions, confirm that the chosen option (A,
B, C, or D) is the correct one.
3. Consider the question type (e.g., mathematical, scientific,
factual, analytical) and assess correctness accordingly.
4. When the question involves visual elements, ensure that the final
answer is consistent with what is actually depicted in the image.
OUTPUT FORMAT (strict):
- If the final answer is correct and consistent with the question and
image: [VERA]True
- Otherwise: [VERA]False
```

Answer Completeness Check Prompt.

```
TASK:
Verify whether the proposed solution provides a complete and
definitive final answer.
GUIDELINES:
1. Confirm that the solution explicitly states a final answer that
directly responds to the question.
2. For multiple-choice questions, the final answer must be clearly
presented in the format: "Final Answer: [A/B/C/D]"
3. Mark the solution as incomplete if any of the following conditions
apply:
- The answer is vague, conditional, or expresses uncertainty.
- No explicit final answer is provided.
- Reasoning is presented without selecting a final option.
- The solution fails to address required visual information when the
question depends on the image.
OUTPUT FORMAT (strict):
- If the solution provides a complete and explicit final answer in the
required format: [VERA]True
- Otherwise: [VERA]False
```

Question Understanding Check Prompt.

```
TASK:
Assess whether the proposed solution correctly understands and
addresses the problem.
GUIDELINES:
1. Problem comprehension:
- Verify that the solution correctly interprets the question intent,
requirements, and constraints.
- Check whether the solution identifies the correct question type
(e.g., mathematical, factual, analytical).
- Examine the image carefully and confirm that all relevant visual
elements are correctly understood and used.
2. Option evaluation:
- Ensure that all provided answer options (A, B, C, D) are
meaningfully considered.
- Check that no option is ignored or dismissed without justification.
- Verify that the selected option aligns with both the question and
the image content.
3. Task alignment:
- Confirm that the solution answers the given question and selects an
answer from the provided options.
- Flag any deviation from the problem context, misinterpretation of
options, or neglect of critical visual information.
OUTPUT FORMAT (strict):
- If the solution demonstrates a correct understanding of the problem,
options, and visual content: [VERA]True
- Otherwise: [VERA]False
IMPORTANT:
Output only the exact string [VERA]True or [VERA]False, with no
additional text.
```

A.5 Evaluation-Time Answer Formatting Prompts

During evaluation, we enforce the answer formatting prompts in Table 8 to ensure unambiguous and consistent model outputs across different task types. Both templates request step-by-step reasoning as well as a final-answer format; they therefore define part of the model's evaluation prompt.

Table 8: Evaluation-time answer formatting prompts used for different question types.

Question Type	Required Output Format
Multiple-Choice Questions	You must answer this multiple-choice question. Think step by step, then at the very end of your response, output: `Final Answer: [X]` where X is the chosen option label (A, B, C, D, E, etc.). Do not include any text after this line.
Open-Ended Questions	You must answer this open-ended question. Think step by step, then at the very end of your response, output: `\boxed{your final answer}` where your final answer is the concise answer to the question. Do not include any text after this line.

The multiple-choice template applies to questions with explicit answer options, including AI2D; the open-ended template applies to free-form questions. Mixed-format benchmarks require routing by question type rather than by dataset name alone. MME uses yes/no questions, so its output convention must also remain compatible with the benchmark answer extractor.

A.6 Dataset Statistics

We report the exact evaluation splits and sample sizes used in our experiments. Each dataset uses its designated test or validation split and the benchmark evaluation protocol.

- **MathVision** (Wang et al., 2024a): We evaluate on the official `test` split, which contains 3,040 questions requiring image-based mathematical reasoning.
- **MathVerse** (Zhang et al., 2024): Evaluation is conducted on the `testmini` split, containing 3,940 samples: five multimodal versions of each of 788 underlying mathematical problems.

- **MathVista** (Lu et al., 2023): We use the `testmini` split with 1,000 samples, covering diagram understanding, visual abstraction, and symbolic reasoning tasks.
- **MMMU** (Yue et al., 2024): We evaluate on the `val` split across 30 subsets. Each subset contains 30 questions, resulting in a total of 900 evaluation samples.
- **RealWorldQA** (X.AI, 2024): Evaluation is performed on the `test` split, which includes 765 questions about visual content in real-world images.
- **MME** (Fu et al., 2023): We use the `test` split comprising 2,374 image–question pairs covering perception, cognition, and reasoning capabilities.
- **AI2D** (Kembhavi et al., 2016): Evaluation is conducted on the `test` split with 3,088 diagram question–answer examples for science diagram understanding, requiring joint interpretation of diagram structure, labels, and their relationships.

B Additional Ablation and Sensitivity Analysis

B.1 Ablation of Key Components

The final component study uses a 2×2 factorial design: latent updates and reasoning anchors are independently enabled or disabled. The backbone, prompt, split, verifier set, decoding cap, and seeds are held fixed across all four configurations in this component study.

Table 9: Component ablation on `Qwen2.5-VL-7B`. Entries with three-seed summaries show mean $\pm$ standard deviation; the latent-only row reports point estimates aligned with Figure 4.

Variant	Latent	Anchors	MathVista	MMMU	AI2D
Vanilla	Off	Off	$66.15_{\pm 0.00}$	$56.96_{\pm 0.00}$	$80.58_{\pm 0.00}$
Anchors only	Off	On	$68.42_{\pm 0.51}$	$59.18_{\pm 0.76}$	$84.10_{\pm 0.44}$
Latent Weaving only	On	Off	71.52	60.87	86.20
`LatentWeave` (Full)	On	On	$\mathbf{72.03}_{\pm 0.55}$	$\mathbf{63.15}_{\pm 0.80}$	$\mathbf{88.93}_{\pm 0.46}$

B.2 Candidate-Evaluation Budget and Same-Verifier Comparison

The same-verifier component comparison allocates 96 candidate-level evaluations to Best-of-N and the four-segment latent variants. For the four-segment configuration, fresh sampling at every update gives $N_{\text{seg}}KM = 4 \times 6 \times 4 = 96$ rollouts and $N_{\text{seg}}K = 24$ updates. Other generation lengths follow Eq. (4), with counts scaling by the realized number of segments. Each verifier evaluation denotes one candidate-level composite reward from the four checks. Best-of-N generates and evaluates 96 candidates. Table 10 reports the corresponding resource profiles, including the auxiliary anchor-generation call; offline training is described separately in Table 13.

Table 10: Inference cost per example on the MathVista-200 cost subset with `Qwen2.5-VL-7B` and one A100-80GB GPU. The latent method uses four segments. Rollouts accumulate over updates; verifier evaluations count external candidate-level rewards. Dashes denote unspecified operation counts. TTRV updates model parameters using its intrinsic frequency- and entropy-based rewards.

Method	Rollouts / updates	Main FWD / BWD	Aux / verifier	Tokens	Time / peak memory
Self-Consistency	8/0	8/0	0/0	2.8k	14 s / 21 GB
Self-Refine	3/0	3/0	0/2	1.3k	7 s / 20 GB
TTRV	–/–	–/–	0/0	3.0k	18 s / 22 GB
Same-verifier Best-of-N	96/0	–/0	0/96	5.6k	31 s / 21 GB
`LatentWeave`	96/24	–/24	1/96	9.2k	52 s / 41 GB

Relative to Vanilla, Best-of-N improves the reported MathVista point estimate by 3.29 percentage points. At the shared 96-candidate verifier-evaluation budget, latent-only is 2.08 points higher than Best-of-N; adding anchors raises the latent variant by another 0.51 points. The full method additionally uses one auxiliary anchor-generation call. Measured latency is 52 versus 31 seconds per example ($1.68\times$), and peak memory is 41 versus 21 GB ($1.95\times$), relative to Best-of-N.

Table 11: Same-verifier comparison on MathVista with `Qwen2.5-VL-7B`. Best-of-N and both four-segment latent variants each use 96 candidate-level verifier evaluations. Entries report point estimates; gains are percentage-point differences from the preceding row. Vanilla provides the unoptimized accuracy reference.

Condition	Latent / anchors	Rollouts / evaluations	Comparison	Accuracy	Gain (pp)
Vanilla	Off / Off	1/0	Reference	66.15	–
Best-of-N	Off / Off	96/96	Same verifier	69.44	+3.29
Latent Weaving only	On / Off	96/96	Same verifier	71.52	+2.08
Full	On / On	96/96 (+1 aux)	Same verifier	72.03	+0.51

B.3 Verifier Reliability and Bias

Verifier scores are audited against a disjoint gold set. Table 12 reports discrimination, calibration, and the two main bias checks; detailed per-example scores remain in the audit artifact.

Table 12: Verifier reliability and bias, audited against a disjoint gold set. Agreement is the fraction of items matching the gold judgment, with a 95% interval; discrimination is AUROC against final answer correctness; calibration is expected calibration error. Bias checks report the Spearman correlation between the verifier score and output length, and the score gain given to same-family generations.

Verifier	Audit N	Agreement (CI)	Discrimination	Calibration	Bias checks
Consistency (rc)	400	0.77 [0.73, 0.81]	AUROC 0.73	ECE 0.12	len. +0.21; self +2.4
Correctness (ac)	400	0.91 [0.88, 0.94]	AUROC 0.89	ECE 0.06	len. +0.07; self +1.1
Completeness (cp)	400	0.75 [0.71, 0.79]	AUROC 0.70	ECE 0.13	len. +0.34; self +2.9
Understanding (uc)	400	0.80 [0.76, 0.84]	AUROC 0.75	ECE 0.11	len. +0.16; self +3.6

B.4 Baseline Reproduction Protocol

All baselines use the same split and answer extractor. Sample counts and refinement rounds are reported by method in Tables 13 and 10. The four-segment same-verifier comparison uses 96 candidate-level evaluations for Best-of-N and the latent variants; offline training is described separately from inference costs.

Table 13: Baseline reproduction protocol. All rows use the same split, formatting prompt, and answer extractor.

Method	Backbone / checkpoint	Decoding / samples	Train data / budget	Verifier	Seeds / config
CoT	Qwen2.5-VL-7B-Instruct	Greedy, N=1, 1024 tok.	–	–	1 / `base-cot`
Self-Consistency	Qwen2.5-VL-7B-Instruct	τ=0.7, top-p 0.95, N=8	–	Majority vote	3 / `base-sc`
Self-Refine	Qwen2.5-VL-7B-Instruct	Greedy, 2 refine rounds	–	Self-critique	3 / `base-sr`
TTRV	Qwen2.5-VL-7B-Instruct	τ=0.7, N=8	Unlabeled test inputs; online GRPO	Frequency + entropy	3 / `base-ttrv`
SFT / GRPO	Qwen2.5-VL-7B-Instruct	Greedy, N=1	18k CoT traces; 2 ep. / 1.0k steps	– / same reward	3 / `base-sft`, `base-grpo`
IVT-LR / LaCoT	Released checkpoints	Per original paper	Same 18k pool	Per original paper	3 / `base-ivtlr`, `base-lacot`

B.5 Segment and Optimization Sensitivity

Table 14 aligns the three parameter sweeps with Figure 5. The upper panel reports MathVista and RealWorldQA to one decimal place, matching the plotted values. MathVista peaks at $J = 6$, whereas RealWorldQA peaks at $J = 4$; both reach their highest plotted accuracy at $\rho = 0.25$ and $K = 6$ in the respective sweeps. The lower panel preserves the separate anchor-delimiter comparison on MathVista and MMMU, which is not plotted in Figure 5.

Table 14: Parameter sensitivity and anchor-delimiter control. The upper panel matches Figure 5; the lower panel reports the separate delimiter experiment. Accuracies follow the order of the values, and spread is $\max - \min$ for the two listed benchmarks. Bold marks each benchmark's highest accuracy within a sweep.

Parameter sweeps from Figure 5

Factor	Values	MathVista	RealWorldQA	Spread
ρ	0.05/0.10/0.15/ 0.20/0.25/0.30	66.5/67.2/67.8/ 68.2/**68.5**/67.5	66.3/66.8/67.1/ 67.3/**67.5**/67.2	2.0/1.2
J	2/4/6/8	66.8/67.5/**68.2**/67.6	66.5/**67.4**/67.0/66.3	1.4/1.1
K	3/4/5/6/7	67.2/67.6/67.9/**68.3**/68.0	66.5/67.0/67.3/**67.6**/67.1	1.1/1.1

Separate anchor-delimiter control

Factor	Values	MathVista	MMMU	Spread
Anchor delimiters	none/<WEAVE>/ numbered/XML	70.2/**72.0**/71.9/71.6	61.0/**63.2**/63.1/62.7	1.8/2.2

B.6 Sensitivity to Reward Weighting

Table 15 varies the reward weights with the online budget held fixed. The default vector gives the highest reported accuracy among the four configurations, with spreads of 1.49 points on MathVista and 1.73 on MMMU. Both the correctness-heavy and structure-heavy configurations fall below uniform weighting. This comparison favors the default balance within the tested configurations; it does not establish that arbitrary concentrations of reward weight are harmful.

Table 15: Reward-weight sensitivity on `Qwen2.5-VL-7B` with the online budget held fixed. The vector lists weights for reasoning consistency, answer correctness, completeness, and question understanding, in that order.

Configuration	Weight vector	MathVista	MMMU
Default	$(0.30, 0.40, 0.15, 0.15)$	**72.03**	**63.15**
Uniform	$(0.25, 0.25, 0.25, 0.25)$	71.38	62.47
Correctness-heavy	$(0.10, 0.70, 0.10, 0.10)$	71.26	62.31
Structure-heavy	$(0.40, 0.15, 0.30, 0.15)$	70.54	61.42

B.7 Reward Component Ablation

Table 16 removes one verifier at a time on MathVision and RealWorldQA. Each removal lowers the reported accuracy. Removing correctness causes the largest decrease on both benchmarks, followed by consistency, while completeness and understanding have smaller effects. These comparisons support retaining multiple signals within the evaluated reward configuration; the relative ordering of completeness and understanding differs across the two benchmarks.

Table 16: Reward-component ablation on `Qwen2.5-VL-7B`, mean ± standard deviation over three seeds. The last column is the change relative to the full composite reward on MathVision / RealWorldQA.

Reward setting	MathVision	RealWorldQA	Δ vs. full
Full composite reward	$\mathbf{30.11}_{\pm 0.38}$	$\mathbf{70.89}_{\pm 0.52}$	–
Without consistency	$29.04_{\pm 0.42}$	$69.77_{\pm 0.55}$	-1.07 / -1.12
Without correctness	$28.30_{\pm 0.47}$	$69.08_{\pm 0.61}$	-1.81 / -1.81
Without completeness	$29.63_{\pm 0.35}$	$70.46_{\pm 0.49}$	-0.48 / -0.43
Without understanding	$29.72_{\pm 0.40}$	$70.31_{\pm 0.53}$	-0.39 / -0.58

B.8 Result Reconciliation Ledger

The ledger records representative links between table entries and named evaluation artifacts; each entry identifies its task, backbone, split, and configuration.

Table 17: Result-reconciliation ledger with representative entries from selected result tables. Artifact names are the stems of the corresponding `.jsonl`/`.json` files; full command lines and hashes remain in the run manifest.

Location	Task / backbone / split	Prompt / config	Raw artifact	Metric / value	Status
Table 1	MathVista / Qwen2.5-VL-7B / `testmini`	`cot-v1` / `lw-v1`	`qwen7b_mathvista`	Acc. 72.03	Verified
Table 2	MathVision / STEP3-VL-10B / `test`	`cot-v1` / `lw-v1`	`step3_mathvision`	Acc. 70.29	Verified
Table 9	AI2D / Qwen2.5-VL-7B / `test`	`cot-v1` / `lw-anchor-v1`	`qwen7b_ai2d_s{0,1,2}`	Acc. $84.10_{\pm 0.44}$	Verified
Table 10	Cost / Qwen2.5-VL-7B / MathVista-200	`lw-v1`	`cost_qwen7b`	52 s / item	Verified

B.9 Additional Analysis and Discussion

We hypothesize that early latent corrections influence subsequent reasoning decisions. Late-only intervention may leave earlier errors unresolved or alter the remaining continuation. Under the fixed-prefix procedure, it cannot overwrite previously accepted states; the observed accuracy changes instead arise from changes to the remaining suffix and final answer.

The ablations and sensitivity sweeps characterize intervention timing and parameter choices within the evaluated settings. Table 17 lists representative artifact links. The main and same-verifier comparisons report point estimates, while the component and reward ablations also report seed summaries for the configurations shown. The evaluation spans seven benchmarks emphasizing stepwise multimodal reasoning.

C ALGORITHM STATEMENT

Algorithm 1: Algorithm workflow of `LatentWeave`

Input: training split $\mathcal{D}_{\text{tr}}$, evaluation input $(\mathbf{x}, \mathbf{v})$, frozen backbone f_θ, auxiliary initialization ϕ_0, rollout count M, and inner updates K
Output: refined answer $\hat{\mathbf{y}}$, optimized latent trajectory $\mathbf{H}^\star$, trained WEAVEGen model ϕ

```
/* Phase I: Offline trajectory collection                    */
```
Initialize trajectory buffer $\mathcal{B} \leftarrow \varnothing$;
foreach $(\mathbf{x}, \mathbf{v}, \mathbf{y}^\star) \in \mathcal{D}_{\text{tr}}$ **do**
  $\mathbf{y}^{(0)}, \mathbf{H}^{(0)} \leftarrow f_\theta(\mathbf{x}, \mathbf{v})$; ▷ Eq. (1)
  Split $\mathbf{H}^{(0)}$ into contiguous segments $\{\mathcal{S}_i\}_{i=1}^{N_{\text{seg}}}$; ▷ Eq. (4)
  for $i = 1$ **to** N_{seg} **do**
    Freeze the prefix and previously accepted segments;
    for $k = 1$ **to** K **do**
      Draw M fresh stochastic suffix rollouts at temperature τ;
      Evaluate R_{cfg} (ground truth may be used only on this offline split);
      Compute the leave-one-out baseline and update $\mathbf{H}_{\mathcal{S}_i}$ with Eq. (6);
      Re-decode the suffix and retain the prefix cache;
    Freeze $\mathbf{H}^\star_{\mathcal{S}_i}$;
  Store the accepted trajectory and its textual reasoning trace in buffer $\mathcal{B}$;

```
/* Phase II: Offline Reasoning-Anchor training               */
```
Initialize $\phi \leftarrow \phi_0$ and augment the auxiliary tokenizer with `<WEAVE>` delimiters;
foreach *trajectory in* $\mathcal{B}$ **do**
  Update ϕ to minimize $\mathcal{L}_{\text{bind}} + \mathcal{L}_{\text{util}}$; ▷ Eqs. (11) and (12)

```
/* Phase III: Evaluation-time inference                      */
```
Generate one textual anchor sequence $\mathbf{a} \leftarrow f_\phi(\mathbf{x}, \mathbf{v})$ and insert it into the context of f_θ;
Initialize $\mathbf{y}^{(0)}, \mathbf{H}^{(0)} \leftarrow f_\theta(\mathbf{x}, \mathbf{v}, \mathbf{a})$;
Repeat the segment-wise procedure above, but set $\beta = 0$, use verifier-only rewards, and do not access labels;
Decode the final answer deterministically and return $(\hat{\mathbf{y}}, \mathbf{H}^\star)$;