Diffusion Thinking for Fast Long-Form Spatial Reasoning in Vision–Language Models

Zhitao Zeng
National University of Singapore

Weitao Du
SCI-AGI Lab, CATL

Yueming Jin*
National University of Singapore

Abstract

Long-form spatial reasoning in vision–language models (VLMs) is often bottlenecked by autoregressive chain-of-thought (CoT) decoding, where intermediate reasoning traces and final answers are generated token by token. This sequential decoding path makes high-quality spatial reasoning expensive in latency-sensitive embodied and interactive settings. We propose *Diffusion Thinking*, a post-training framework that converts pretrained autoregressive VLMs into diffusion-style reasoners for fast long-form spatial inference without changing the visual encoder or language backbone. Diffusion Thinking partitions each CoT trace into blocks and refines tokens within the current block in parallel through discrete denoising, while preserving causal dependence across blocks and reusing KV cache for streaming generation. This creates a high-speed operating point, but aggressive parallel refinement can weaken fine-grained reasoning. To address this, we introduce *Diffusion Rethinking*, a cached test-time scaling mechanism that reallocates part of the saved latency budget to self-correction rounds, yielding a controllable speed–accuracy frontier. We construct a 561K-instance Long-CoT Spatial Reasoning dataset with verified final answers and generated CoT traces across six spatial task types. Across Qwen2.5-VL and InternVL3 backbones, Diffusion Thinking with block size $D = 32$ achieves 38.18–41.81$\times$ effective wall-clock speedup over autoregressive Long-CoT decoding. With eight rethinking rounds, it retains 4.20–4.60$\times$ speedup and matches or surpasses autoregressive Long-CoT accuracy on the largest backbones. These results show that diffusion-style compute reallocation is a practical path toward fast, accurate long-form spatial reasoning in VLMs, by changing the speed–accuracy operating point rather than strengthening intrinsic spatial reasoning.

1 Introduction

Spatial reasoning is a core capability for embodied and interactive vision–language intelligence. Agents must infer geometric and relational structure from visual observations, reason about object locations, distances, directions, appearance order, and routes, and convert these inferences into decisions under limited time budgets. In robotics and embodied AI, latency is not merely an efficiency metric: delayed reasoning can postpone action prediction, weaken closed-loop control, and reduce the usefulness of otherwise accurate models [Brohan et al., 2023b, Driess et al., 2023, Brohan et al., 2023a]. This makes fast spatial reasoning an important objective for deploying vision–language models (VLMs) in real-time settings.

Recent VLMs often solve complex spatial questions by producing long-form intermediate reasoning before the final answer. Such rationales can be useful for multi-object comparison, multi-view aggregation, route-style planning, and interpretable decision making. However, long-form reasoning is typically generated autoregressively, one token at a time. As a result, even when a model has strong

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

spatial reasoning ability, the token-level sequential decoding path can make inference too slow for latency-sensitive settings. Our goal is different: we seek to reallocate inference-time compute under a fixed latency budget, reducing the sequential decoding path while preserving enough intermediate computation for long-form spatial inference.

We propose *Diffusion Thinking*, a post-training framework that converts pretrained autoregressive VLMs into diffusion-style VLM reasoners. Rather than generating every reasoning token left-to-right, Diffusion Thinking formulates textual reasoning as block-wise discrete denoising. A response is partitioned into fixed-size blocks; within each block, multiple tokens are refined in parallel through a small number of denoising steps, while previously generated blocks remain clean and serve as causal context. This design preserves the left-to-right structure of long reasoning at the block level, supports streaming generation, and enables KV-cache reuse across blocks. Consequently, Diffusion Thinking changes the inference-time operating point from token-by-token sequential decoding to block-wise parallel refinement.

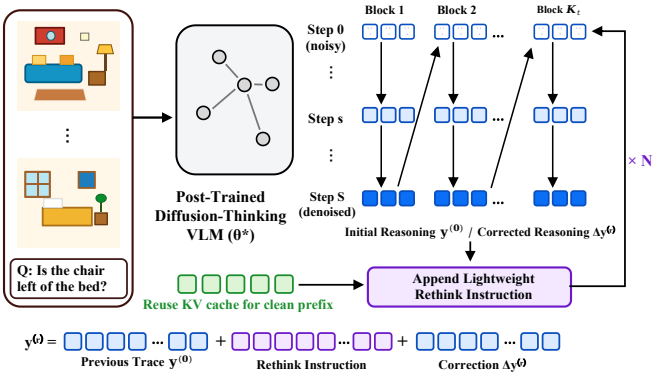

Figure 1: Diffusion Thinking inference pipeline. A post-trained VLM generates long-form reasoning through block-wise denoising. Rethinking appends a lightweight correction instruction and reuses the clean-prefix KV cache for additional cached refinement rounds.

Parallel refinement creates a new speed–accuracy trade-off. Because tokens inside a block are denoised jointly rather than generated under strict token-level causality, aggressive block-wise diffusion may degrade fine-grained reasoning quality, especially for numerical estimation or tightly ordered constraints. To address this, we introduce *Diffusion Rethinking*, a test-time scaling mechanism that spends part of the saved latency budget on cached self-correction, as illustrated in Figure 1. After an initial Diffusion Thinking pass produces a reasoning and answer, the model appends a lightweight rethinking instruction and performs additional cached diffusion-thinking passes. This gives a controllable speed–quality knob: the initial diffusion pass provides a fast draft, and rethinking reallocates part of the saved latency to improve the final answer. To train and evaluate this framework, we construct a Long-CoT Spatial Reasoning dataset containing 561K instances across six spatial task types: absolute distance, object size, relative distance, relative direction, route planning, and appearance order.

Throughout, we view Diffusion Thinking as an acceleration method that changes the speed–accuracy operating point of Long-CoT spatial reasoning rather than as evidence of stronger intrinsic spatial reasoning: its very fast draft can be less accurate, especially for metric estimation, and on sufficiently capable backbones cached Rethinking reallocates the saved latency to recover or numerically exceed the autoregressive Long-CoT operating point.

Our contributions are summarized as follows: (1) We propose *Diffusion Thinking*, a post-training paradigm that adapts pretrained autoregressive VLMs into block-wise diffusion reasoners, enabling parallel refinement of long reasoning traces while preserving causal structure across blocks. (2) We introduce *Diffusion Rethinking*, a cached test-time scaling strategy that reallocates saved latency to self-correction rounds, yielding a controllable speed–accuracy frontier. (3) We construct a 561K-instance Long-CoT Spatial Reasoning dataset with verified answers across six core task types to facilitate training and systematic evaluation of fast reasoning methods.

2 Related Work

Discrete Diffusion Modeling Early discrete formulations include multinomial diffusion [Hoogeboom et al., 2021] and D3PM [Austin et al., 2021]. More recent masked diffusion language models have substantially improved scaling and practicality [Sahoo et al., 2024, Nie et al., 2026], and work on faster inference explores adaptive/parallel decoding and improved unmasking schedules [Israel

et al., 2026, Ben-Hamu et al., 2026]. Diffusion formulations have also been explored for multimodal generation and understanding, yielding diffusion-based VLMs where denoising is conditioned on visual inputs [Yang et al., 2026a, Tian et al., 2026].

Chain-of-Thought Reasoning Chain-of-thought (CoT) prompting and supervision have become a standard mechanism to elicit multi-step reasoning in LLMs and VLMs [Wei et al., 2022]. By encouraging intermediate steps, CoT can improve performance on tasks requiring compositionality, multi-hop inference, and structured constraint satisfaction, and it has inspired methods such as self-consistency sampling [Wang et al., 2022b], process-level supervision and verification [Lightman et al., 2023], and test-time self-refinement [Wang et al., 2022a]. In multimodal reasoning, CoT-style decompositions are frequently used to bridge perception and symbolic inference, e.g., by enumerating objects, relations, and intermediate spatial states before producing an answer [Yang et al., 2025, Du et al., 2024].

Spatial Reasoning Spatial reasoning is central to vision, robotics, and embodied AI, encompassing relative position understanding, referring expression grounding, spatial question answering, navigation, and manipulation. Recent benchmarks highlight that spatial understanding remains a major bottleneck for multimodal models in embodied or multi-view/multi-image settings [Du et al., 2024, Yang et al., 2025, 2026c, Li et al., 2026]. Modern VLMs have improved spatial reasoning via instruction tuning and by leveraging explicit intermediate reasoning to handle compositional relations, multi-object constraints, and viewpoint-dependent queries [Yang et al., 2026b].

3 Method

We propose *Diffusion Thinking*, a post-training framework for fast long-form spatial reasoning in VLMs. The central idea is inference-time compute reallocation: we reduce the strictly sequential token-level decoding path by replacing autoregressive generation with block-wise discrete denoising, and then spend part of the saved wall-clock budget on self-correction through *Diffusion Rethinking*.

3.1 Problem Setup

Given an image or multi-image observation I and a text prompt P, a VLM forms a multimodal context $c = \Phi(I, P)$, where Φ includes the vision encoder, projector, and prompt-token embedding pipeline. The model then generates a response $x_{1:L} = (x_1, \ldots, x_L) \in \mathcal{V}^L$, which contains both an intermediate reasoning trace and a final answer. In the long-CoT setting, L can be large, so autoregressive decoding becomes a major latency bottleneck.

For any generation procedure $\mathcal{G}$, let $\tau_{\mathcal{G}}(I, P)$ be its end-to-end wall-clock latency. We define its effective speedup relative to autoregressive long-CoT decoding as

$$\gamma_{\mathcal{G}} = \tau_{\text{AR-CoT}}(I, P)/\tau_{\mathcal{G}}(I, P).$$

When methods have very different output lengths, such as direct answering and long-CoT generation, token throughput alone is not a reliable speedup measure. Therefore, ***Eff. Speedup*** in our experiments always refers to this end-to-end latency ratio, while token/s is reported as a descriptive throughput statistic.

3.2 Block-wise Diffusion Thinking

A standard autoregressive VLM factorizes generation as $p_\theta(x_{1:L} \mid c) = \prod_{\ell=1}^{L} p_\theta(x_\ell \mid c, x_{<\ell})$, requiring L sequential decoding steps. Diffusion Thinking replaces this token-level generation interface with block-wise denoising while keeping the same VLM backbone.

We pad the response to length $L' = D\lceil L/D \rceil$, where D is the block size, and split it into $K = \lceil L/D \rceil$ blocks: $x_0 = [x_0^{(1)}, \ldots, x_0^{(K)}]$, with each $x_0^{(k)} \in \mathcal{V}^D$. For block k, previous blocks $x_0^{(<k)}$ are treated as clean causal context, while the current block is corrupted and reconstructed in parallel. Let t_k be the sampled noise level, $\mathcal{M}_k$ the corrupted positions, and $\tilde{x}_{t_k}^{(k)}$ the corrupted current block. The

block-wise denoising objective is

$$\mathcal{L}_{\mathrm{BDM}} = -\mathbb{E}_{(c,x_0),k,t_k}\left[\frac{\lambda(t_k)}{|\mathcal{M}_k|}\sum_{i\in\mathcal{M}_k}\log p_\theta\big(x_0^{(k)}[i] \mid c, x_0^{(<k)}, \tilde{x}_{t_k}^{(k)}\big)\right]. \tag{1}$$

Here, $\lambda(t_k)$ is a non-negative diffusion loss weight. We use block-uniform corruption: all positions in the same corrupted block share the same noise level t_k. This matches inference, where an entire block is refined for a fixed number of denoising steps, and keeps the training distribution aligned with the generation procedure.

Same backbone, new generation interface. Diffusion Thinking does not introduce a new vision encoder, projector, language backbone, or output head. Starting from a pretrained autoregressive VLM, we finetune all parameters with Eq. (1). The post-training stage changes only the token corruption process and the attention pattern used for denoising.

3.3 Attention Pattern and Inference

The attention mask is designed to be bidirectional inside the current block and causal across blocks. Let C denote multimodal context tokens, $P^{(<k)}$ previous response blocks, and $P^{(k)}$ the current block. When denoising block k, each query token $i \in P^{(k)}$ may attend to

$$\mathcal{A}^{(k)}(i) = C \cup P^{(<k)} \cup P^{(k)}. \tag{2}$$

Thus, tokens inside $P^{(k)}$ can interact bidirectionally, enabling parallel refinement, while no token can condition on future blocks. Since C and $P^{(<k)}$ are clean prefix tokens, their key/value states can be cached and reused for later blocks.

At inference time, Diffusion Thinking causally generates blocks from left to right. For each block, we initialize D positions as `[MASK]` and perform S denoising steps. At each step, the model efficiently predicts all masked positions in the current block in parallel, and an update operator $\mathcal{U}_s$ fills or re-masks positions according to a monotone denoising schedule. After S steps, the block is committed as clean text, its KV cache is stored, and generation proceeds to the next block. When `<eos>` is produced, the generation process finishes and tokens after `<eos>` are ignored. All diffusion variants in our experiments use the same update rule, so the comparisons isolate block size, denoising steps, and rethinking rounds.

Algorithm 1 summarizes the full Diffusion Thinking pipeline. During post-training, a pretrained AR-VLM is converted into a block-wise denoising model by corrupting one response block at a time while keeping all previous blocks clean as causal context. During inference, the model generates blocks from left to right through parallel denoising within each block. Diffusion Rethinking reuses the same block-wise diffusion decoder, but conditions each additional pass on the previous reasoning trace and a lightweight self-correction instruction.

What is specific to visual inputs. Appendix G discusses how properties of visual inputs motivate this design.

3.4 Latency Accounting

We now clarify why the theoretical speed expression should not be identified with the empirical speedup in Table 1. Autoregressive long-CoT decoding requires one model invocation per generated token, so its sequential depth is $M_{\mathrm{AR}}(L) = L$. Diffusion Thinking generates $K = \lceil L/D \rceil$ blocks and uses S denoising steps per block, giving $M_{\mathrm{DT}}(L; D, S) = S\lceil L/D \rceil$. The sequential-depth reduction is therefore

$$\rho(L; D, S) = \frac{M_{\mathrm{AR}}(L)}{M_{\mathrm{DT}}(L; D, S)} = \frac{L}{S\lceil L/D \rceil}, \qquad \frac{D}{S(1 + D/L)} \le \rho(L; D, S) \le \frac{D}{S}. \tag{3}$$

Hence, D/S is an upper bound on the *critical-path reduction*, not the measured wall-clock speedup.

The empirical speedup also depends on hardware utilization and kernel efficiency. Let $\bar{\delta}_{\mathrm{AR}}$ be the average latency of one autoregressive decoding step, and $\bar{\delta}_{\mathrm{DT}}$ the average latency of one block-

Algorithm 1 Diffusion Thinking Post-training and Inference

Require: Pretrained VLM parameters θ, training set $\mathcal{D}_{\text{train}}$, block size D, noise sampler $q(t)$, masking policy $\pi(\cdot \mid t)$, denoising steps S, maximum blocks $K_{\max}$, rethinking rounds N

Post-training.
for each minibatch $\{(I, P, x_0)\} \sim \mathcal{D}_{\text{train}}$ **do**
  Encode multimodal context $c \leftarrow \Phi_\theta(I, P)$.
  Pad x_0 to $L' = D\lceil L/D \rceil$ and split it into $K = \lceil L/D \rceil$ blocks: $x_0 = [x_0^{(1)}, \ldots, x_0^{(K)}],\ x_0^{(k)} \in \mathcal{V}^D$.
  Initialize minibatch loss $\mathcal{L} \leftarrow 0$.
  for $k = 1$ **to** K **do**
    Sample noise level $t_k \sim q(t)$ and corrupted positions $\mathcal{M}_k \sim \pi(\cdot \mid t_k)$.
    Corrupt the current block $\tilde{x}_{t_k}^{(k)} \leftarrow \text{Mask}(x_0^{(k)}, \mathcal{M}_k)$.
    Use c and clean prefix blocks $x_0^{(<k)}$ as causal context; apply bidirectional attention only inside block k.
    Predict masked tokens $p_\theta(\cdot \mid c, x_0^{(<k)}, \tilde{x}_{t_k}^{(k)})$.
    Accumulate loss on corrupted positions:

$$\mathcal{L} \leftarrow \mathcal{L} - \frac{\lambda(t_k)}{|\mathcal{M}_k|} \sum_{i \in \mathcal{M}_k} \log p_\theta \left(x_0^{(k)}[i] \mid c, x_0^{(<k)}, \tilde{x}_{t_k}^{(k)} \right).$$

  end for
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$.
end for
Inference with optional Rethinking.
Encode test context $c \leftarrow \Phi_\theta(I, P)$.
Generate an initial reasoning trace and answer: $y^{(0)} \leftarrow \text{BlockDiffuse}(c, D, S, K_{\max})$.
for $r = 1$ **to** N **do**
  Append a lightweight self-correction instruction: $c^{(r)} \leftarrow [\, c;\, y^{(r-1)};\, \mathcal{I}_{\text{rethink}}\,]$.
  Reuse the KV cache for the clean prefix $[c; y^{(r-1)}; \mathcal{I}_{\text{rethink}}]$.
  Generate a correction trace: $\Delta y^{(r)} \leftarrow \text{BlockDiffuse}(c^{(r)}, D, S, K_{\max})$.
  Append the correction: $y^{(r)} \leftarrow [\, y^{(r-1)}; \mathcal{I}_{\text{rethink}}; \Delta y^{(r)}\,]$.
end for

denoising step. Then $\tau_{\text{AR}} = L\bar{\delta}_{\text{AR}}$ and $\tau_{\text{DT}} = S\lceil L/D \rceil \bar{\delta}_{\text{DT}}$, so

$$\gamma_0 = \frac{\tau_{\text{AR}}}{\tau_{\text{DT}}} = \underbrace{\frac{L}{S\lceil L/D \rceil}}_{\rho(L;D,S)} \cdot \underbrace{\frac{\bar{\delta}_{\text{AR}}}{\bar{\delta}_{\text{DT}}}}_{\kappa}. \tag{4}$$

The factor κ captures implementation and hardware effects, including GPU utilization, memory access, KV-cache reuse, and kernel efficiency. This explains why empirical wall-clock speedup can differ substantially from D/S.

3.5 Diffusion Rethinking

Block-wise denoising improves latency but relaxes token-level causality inside each block, which can hurt fine-grained reasoning. Diffusion Rethinking converts part of the saved latency into additional self-correction.

After an initial Diffusion Thinking pass produces a reasoning trace and answer, we append a lightweight instruction $\mathcal{I}_{\text{rethink}}$ asking the model to verify and revise its previous reasoning. The previous context and reasoning trace are treated as clean prefix tokens, so their KV states can be reused. Each rethinking round then generates an additional correction trace using the same block-wise diffusion procedure.

Let τ_0 be the latency of the initial Diffusion Thinking pass and τ_r the latency of the r-th rethinking round. Define the relative cost $\eta_r = \tau_r/\tau_0$. After N rethinking rounds, the total latency and effective speedup are

$$\tau_{\text{Rethink}}(N) = \tau_0 \left(1 + \sum_{r=1}^{N} \eta_r \right), \qquad \gamma_{\text{Rethink}}(N) = \frac{\gamma_0}{1 + \sum_{r=1}^{N} \eta_r} \approx \frac{\gamma_0}{N+1}. \tag{5}$$

The approximation holds when each rethinking round has roughly the same latency as the initial diffusion pass. To remain no slower than autoregressive long-CoT decoding, we require $1+\sum_{r=1}^{N}\eta_r \le \gamma_0$, which gives $N_{\max} \approx \lfloor \gamma_0 - 1 \rfloor$ when $\eta_r \approx 1$. Importantly, this budget depends on the measured wall-clock speedup γ_0, not on the theoretical critical-path reduction ρ.

This formulation matches the behavior observed in our experiments. A single Diffusion Thinking pass gives the largest raw speedup but may lose accuracy. Rethinking trades a controlled fraction of that speedup for self-correction, yielding a practical speed–accuracy frontier for long-form spatial reasoning.

4 Experiments

We evaluate Diffusion Thinking in the long-CoT spatial reasoning regime. Our experiments are designed to answer four questions: (i) whether block-wise diffusion moves long-CoT VLMs from autoregressive low-speed decoding to a high-speed operating region; (ii) whether Rethinking can convert part of the saved latency into accuracy recovery; (iii) how Diffusion Thinking compares with trained answer-only control, correction-style diffusion CoT baselines, and AR acceleration baselines under the same deterministic protocol; and (iv) which design choices control speed–accuracy frontier.

4.1 Dataset and Evaluation

Dataset. We construct a *Long-CoT Spatial Reasoning* dataset for training and evaluating fast long-form spatial reasoning in VLMs. The dataset follows the task taxonomy and evaluation format of VSI-Bench [Yang et al., 2025], while the verified visual-spatial annotations are derived from Visual Spatial Tuning (VST) [Yang et al., 2026b]. It contains **561K** spatial VQA instances across six task types: Absolute Distance, Object Size, Relative Distance, Relative Direction, Route Plan, and Appearance Order. Since the original annotations provide verified answers but no CoT, we generate answer-constrained CoT traces with Qwen3-VL-Plus [Bai et al., 2025] and refine them with Qwen3-Max. Details of CoT generation, filtering, and task statistics are provided in Appendix E.

Evaluation protocol. We evaluate all methods with a deterministic VSI-style protocol rather than LLM-as-a-judge scoring. Following VSI-Bench [Yang et al., 2025], Relative Distance, Relative Direction, Route Plan, and Appearance Order are evaluated by exact-match Accuracy (ACC), while Absolute Distance and Object Size are evaluated by Mean Relative Accuracy (MRA). For every model output, we extract the final option letter or numerical value and compare it with the verified source annotation. The test set contains **6,000** examples, with 1,000 examples per task type, and is disjoint from the training split at the question-instance level.

Training and speed measurement. All methods are trained or post-trained on the same Long-CoT Spatial Reasoning training split. AR Answer-only removes generated CoT from the target and directly predicts the verified final answer, serving as a trained low-latency control. AR Long-CoT, DoT, DiffCoT, and DiffThink use long-CoT supervision, while DiffThink replaces next-token prediction with the block-wise denoising objective. Full training details are provided in Appendix F.

4.2 Comparison

Diffusion Thinking creates a high-speed operating point. Table 1 shows that Diffusion Thinking substantially changes the latency profile of long-CoT spatial reasoning. With block size $D = 32$, DiffThink achieves 38.18–41.81$\times$ effective wall-clock speedup over AR Long-CoT across five VLM backbones. This speedup is not a small decoding optimization: it comes from replacing token-by-token autoregressive generation with block-wise parallel denoising. The high-speed point incurs an accuracy drop on some fine-grained numerical tasks, especially Absolute Distance and Object Size, but remains competitive on relational and global-structure tasks such as Relative Direction and Route Plan.

Rethinking reallocates saved latency to accuracy recovery. Rethinking moves the diffusion operating point back toward the high-accuracy region while retaining multi-fold speedup. With R$\times$8, DiffThink retains 4.20–4.60$\times$ effective speedup. On smaller backbones, Rethinking mainly

Table 1: Comparison on the 6,000-sample VSI-style test set. AR Answer-only is trained to directly predict the final answer, while the other methods are trained with Long-CoT. Eff. Speedup denotes effective end-to-end wall-clock speedup over AR Long-CoT. R$\times$8 denotes eight rethinking rounds.

Model	Method	Speed (tok/s)	Eff. Speedup	Avg.	Abs. Dist. (MRA)	Obj. Size (MRA)	Rel. Dist. (ACC)	Rel. Dir. (ACC)	Route Plan (ACC)	Appr. Order (ACC)
Qwen2.5-VL-3B	AR Answer-only	121	2.53$\times$	27.38	25.60	23.40	31.60	36.40	24.80	22.50
	AR Long-CoT	116	1.00$\times$	30.88	28.90	27.40	35.10	38.70	29.80	25.40
	DoT	101	0.87$\times$	31.12	29.10	27.70	35.30	38.90	30.10	25.60
	DiffCoT	95	0.82$\times$	**31.28**	**29.30**	**27.90**	**35.40**	39.00	30.30	25.80
	DiffThink ($D = 32$)	**4230**	**38.18**$\times$	29.47	23.60	18.10	34.80	38.60	**33.80**	**27.90**
	DiffThink ($D = 32$) + R$\times$8	3722	4.20$\times$	30.80	27.60	24.80	34.90	**39.00**	31.20	27.30
Qwen2.5-VL-7B	AR Answer-only	60.5	2.47$\times$	33.33	32.40	30.80	38.20	43.10	29.50	26.00
	AR Long-CoT	58	1.00$\times$	36.63	35.60	34.10	41.50	45.70	34.30	28.60
	DoT	50.5	0.87$\times$	36.95	35.90	34.30	41.70	45.90	34.70	28.70
	DiffCoT	47.3	0.82$\times$	**37.18**	**36.10**	**34.50**	**41.90**	**46.10**	35.00	28.90
	DiffThink ($D = 32$)	**2402**	**41.41**$\times$	33.98	26.10	23.40	40.30	44.00	**40.80**	**29.30**
	DiffThink ($D = 32$) + R$\times$8	2114	4.55$\times$	36.18	33.50	32.00	41.30	44.90	36.10	**29.30**
Qwen2.5-VL-32B	AR Answer-only	15.2	2.41$\times$	37.82	37.10	35.70	42.30	47.00	33.60	31.20
	AR Long-CoT	14.6	1.00$\times$	40.78	40.30	39.10	44.90	48.80	37.90	33.70
	DoT	12.6	0.86$\times$	41.05	40.70	39.30	45.00	49.00	38.40	**33.90**
	DiffCoT	11.8	0.81$\times$	41.20	**40.90**	**39.40**	**45.10**	49.10	38.80	**33.90**
	DiffThink ($D = 32$)	**607**	**41.58**$\times$	38.25	34.40	27.90	43.90	48.70	**42.50**	32.10
	DiffThink ($D = 32$) + R$\times$8	534	4.57$\times$	**41.45**	40.80	38.20	**45.10**	**49.30**	42.10	33.20
InternVL3-8B	AR Answer-only	55.4	2.45$\times$	33.88	33.00	31.40	38.80	43.70	30.10	26.30
	AR Long-CoT	53.2	1.00$\times$	37.17	36.30	35.00	41.80	46.20	35.20	28.50
	DoT	46.7	0.88$\times$	37.52	36.60	35.30	42.00	46.40	35.70	28.70
	DiffCoT	43.9	0.83$\times$	**37.73**	**36.80**	**35.40**	**42.10**	**46.60**	36.00	**28.90**
	DiffThink ($D = 32$)	**2167**	**40.73**$\times$	34.72	30.10	25.10	40.60	44.70	**40.20**	27.60
	DiffThink ($D = 32$) + R$\times$8	1907	4.48$\times$	36.28	35.00	32.50	41.30	45.10	36.40	27.40
InternVL3-38B	AR Answer-only	13.3	2.39$\times$	38.85	38.20	36.70	43.10	48.00	35.40	31.70
	AR Long-CoT	12.7	1.00$\times$	41.27	41.20	39.80	45.20	49.60	38.90	33.00
	DoT	11.0	0.87$\times$	41.88	41.50	40.00	45.50	49.90	39.70	33.30
	DiffCoT	10.3	0.81$\times$	42.02	**41.70**	**40.10**	45.60	50.00	40.00	33.40
	DiffThink ($D = 32$)	**531**	**41.81**$\times$	39.02	35.10	28.70	44.60	49.00	**44.10**	32.60
	DiffThink ($D = 32$) + R$\times$8	467	4.60$\times$	**42.38**	41.60	39.30	**45.80**	**50.40**	43.40	**33.80**

recovers part of the accuracy lost under block-wise denoising. On larger backbones, it can surpass the AR Long-CoT operating point: Qwen2.5-VL-32B reaches 41.45 Avg. with 4.57$\times$ speedup, and InternVL3-38B reaches 42.38 Avg. with 4.60$\times$ speedup. This supports our main claim that Diffusion Thinking creates latency budget and Diffusion Rethinking reallocates that budget to self-correction.

Beyond answer-only decoding. DiffThink($D = 32$) improves the operating point beyond the trained AR Answer-only control: across all five backbones, it achieves higher Avg. score than AR Answer-only while being approximately 15.1–17.5$\times$ faster in end-to-end latency. With R$\times$8, DiffThink further improves accuracy by 2.40–3.63 Avg. points over AR Answer-only, while still remaining 1.66–1.93$\times$ faster than AR Answer-only and 4.20–4.60$\times$ faster than AR Long-CoT. Thus, the key result is not merely that Long-CoT can be more accurate than answer-only prediction, but that Diffusion Thinking and Rethinking provide a more favorable speed–accuracy frontier.

Correction-style baselines improve quality but remain slow. DoT and DiffCoT slightly improve Avg. over AR Long-CoT on all five backbones. However, they remain slower than AR Long-CoT, with effective speedups below 1$\times$, because they still rely on token-level or iterative corrective generation. Thus, these methods occupy a high-quality but low-speed region, whereas DiffThink+Rethinking provides a more favorable speed–accuracy trade-off.

Faster than AR acceleration baselines. Table 2 controls for a strong autoregressive acceleration baseline on Qwen3.5-27B with native multi-token prediction support. Native MTP-1 improves throughput from 18.8 to 31.0 token/s, corresponding to 1.65$\times$ speedup, but remains close to the autoregressive low-speed regime. Lossless DFlash speculative decoding [Chen et al., 2026], run with its official implementation, preserves AR Long-CoT accuracy at 3.50$\times$ speedup. In contrast, DiffThink($D = 32$) reaches 41.76$\times$ speedup by replacing token-level autoregressive decoding with block-wise parallel refinement. After Rethink$\times$8, DiffThink retains 4.67$\times$ speedup and achieves the best Avg. score in the table, while remaining 1.34$\times$/1.40$\times$ faster than DFlash on H20/H200 (Appendix H). This comparison shows that our gain is not merely due to using a faster AR decoder; the main advantage comes from changing the inference-time compute structure.

Table 2: Comparison with Native Multi-token Prediction (MTP) and DFlash based on Qwen3.5-27B.

Method	Speed	Eff.Speedup	Avg.	Abs. Dist.	Obj. Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order
AR Long-CoT	18.8	1.00×	42.10	41.70	**40.30**	46.00	49.70	39.50	35.40
Native MTP-1	31.0	1.65×	41.82	41.10	39.60	45.70	49.40	39.10	35.00
DFlash	65.8	3.50×	42.10	41.70	**40.30**	46.00	49.70	39.50	35.40
DiffThink ($D = 32$)	**785**	**41.76**×	39.55	36.30	30.90	45.10	50.20	**43.90**	30.80
DiffThink ($D = 32$) + R×8	702	4.67×	**42.48**	**42.10**	39.90	**46.30**	**50.40**	43.40	**35.60**

Table 3: Deployment profiling (batch/concurrency 1) and zero-shot MMSI-Bench transfer. Pairs report H20/H200; latency is mean end-to-end seconds; speedups are over same-GPU AR Long-CoT. DT denotes DiffThink ($D = 32$).

	Six-task test set (H20/H200)					Zero-shot MMSI-Bench (speedups H20/H200)		
Backbone	AR (s)	DT (s)	DT Speedup	R×8 (s)	R×8 Speedup	ACC AR/DT/R×8	DT Speedup	R×8 Speedup
Qwen2.5-VL-32B	62.877/50.216	1.512/0.977	41.58/51.40×	13.758/10.254	4.57/4.90×	30.2/28.8/30.9	32.22/43.30×	4.58/4.97×
InternVL3-38B	72.913/57.996	1.744/1.115	41.81/52.00×	15.850/11.722	4.60/4.95×	28.9/27.5/29.7	32.38/43.78×	4.60/5.01×
Qwen3.5-27B	49.574/39.576	1.187/0.759	41.76/52.16×	10.615/7.901	4.67/5.01×	31.8/30.4/32.6	30.51/41.85×	4.63/5.04×

4.3 Deployment Profiling, Objective Isolation, and Transfer

Absolute latency on two GPU classes. Table 3 reports single-request end-to-end latency from image-tensor-ready input to answer extraction (batch/concurrency 1, bf16, deterministic decoding, clean-prefix KV caching; details in Appendix H); we do not claim batch>1 throughput. For $L = 918$ (the matched AR/DT mean length on Qwen2.5-VL-32B), $D = 32$, and $S = 8$, the sequential depth falls from 918 AR calls to $8\lceil 918/32 \rceil = 232$ ($\rho = 3.96$); the remaining gain comes from cheaper block-parallel invocations (κ in Eq. (4)) and is hardware dependent: H200 accelerates AR by $\sim 1.25\times$ but DT by ~ 1.55–$1.56\times$, raising the DT speedup from $\sim 42\times$ to $\sim 52\times$.

Objective isolation and transfer. FMB-8, which changes only the objective of DT to fixed-mask iterative CE (same Qwen2.5-VL-32B backbone, data, budget, $D = 32$ mask, and eight passes per block), reaches 36.32 Avg. at 40.57× versus 38.25 at 41.58× for DT, with DT better on every task (Appendix I); the gain is thus not explained by block attention or post-training alone. On MMSI-Bench [Yang et al., 2026c] (1,000 human-designed multi-image questions, never used for training, tuning, checkpoint selection, or prompt optimization; official deterministic parser), DT is 1.4 points below AR at ~ 30–$44\times$ speedup, and R×8 is numerically +0.7/+0.8/+0.8 above AR at ~ 4.6–$5.0\times$; we scope this to spatial and multi-image reasoning on the three tested backbone lines.

Reliability of Rethinking and trace quality. R×8 yields 204/12 and 215/13 fixes/breaks relative to the initial DT answers on Qwen2.5-VL-32B and InternVL3-38B (two-sided exact McNemar $p < 4 \times 10^{-11}$), and R4 already captures 75.7–79.7% of the final gain (Appendix J); on Qwen2.5-VL-32B, Margin-Stable adaptive stopping with thresholds frozen on held-out training data reaches 41.31 Avg. at 10.11× H20 speedup, versus 41.45 at 4.57× for fixed R×8. A paired human audit (Appendix K) shows that the initial trace-quality gap of DT is concentrated in metric tasks and that R×8 largely restores faithfulness at some cost in concision.

4.4 Speed–Accuracy Frontier

Impact of block size and rethinking rounds. Figure 2 visualizes the speed–accuracy trade-off for each backbone separately. We avoid macro-averaging because the effect of Rethinking depends on model scale. The blue curves show the block-size frontier of Diffusion Thinking: larger D generally improves speed but can reduce accuracy, especially for numerical estimation tasks. The red dashed curves show Rethinking at $D = 32$: additional rounds move the operating point back toward the high-accuracy region while preserving multi-fold speedup.

For Qwen2.5-VL-7B and InternVL3-8B, Rethinking mainly recovers part of the accuracy loss introduced by block-wise diffusion. For Qwen2.5-VL-32B and InternVL3-38B, Rethinking is more effective and can exceed AR long-CoT accuracy. This model-scale-dependent behavior suggests that Rethinking is most useful when the backbone has sufficient latent reasoning capacity and the remaining errors are repairable by self-correction rather than capacity-limited. The full numerical sweep over $D \in \{8, 16, 32, 64\}$ and $R \in \{2, 4, 8\}$ is reported in Appendix B.

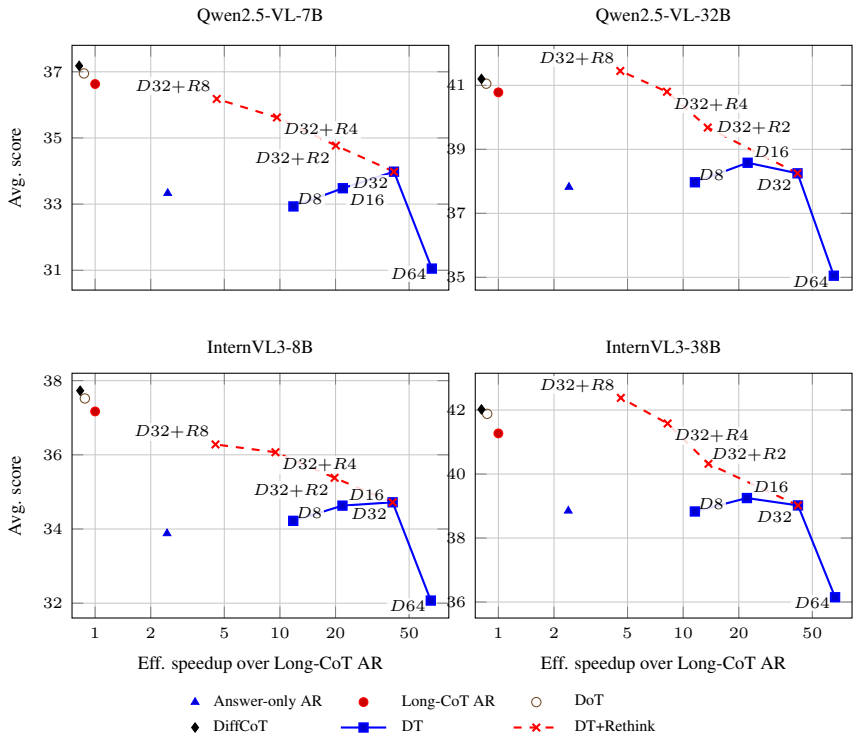

Figure 2: Speed–accuracy Pareto frontiers. The x-axis reports effective end-to-end speedup over Long-CoT AR on a log scale, and the y-axis reports Avg. score. Blue curves connect Diffusion Thinking with different block sizes $D \in \{8, 16, 32, 64\}$, with each point directly annotated by its block size. Red dashed curves start from DT($D = 32$) and show Rethinking with $R \in \{2, 4, 8\}$, with each point annotated as $D32{+}Rk$.

Table 4: Diagnostic analysis of R$\times$8. Fix/Break are measured as percentages on the full test set; A/B/C statistics are measured on initial DiffThink errors. Repair Rate denotes the fraction of repairable A/B/C errors that are corrected.

Model	Fix W→C	Break C→W	Repairable A/B/C	Repair Rate
Qwen2.5-VL-7B	2.6	0.4	34.8	11.3
Qwen2.5-VL-32B	**3.4**	**0.2**	**43.6**	**12.6**

Rethinking as error repair. To understand why Rethinking benefits larger backbones more, we analyze how answers change from DiffThink ($D = 32$) to DiffThink ($D = 32$)+R$\times$8. The transition statistics measure whether Rethinking fixes initially wrong predictions or breaks initially correct ones. We further categorize initial DiffThink errors into locally repairable errors A/B/C and capacity-limited errors D: A denotes final-answer extraction or formatting errors, B denotes local reasoning slips, C denotes missing evidence or incomplete reasoning, and D denotes global misunderstanding. As shown in Table 4, Rethinking changes only a small fraction of predictions and maintains a low break rate for both backbones. Compared with Qwen2.5-VL-7B, Qwen2.5-VL-32B fixes more initially wrong answers while breaking fewer initially correct ones, leading to a larger positive fix–break margin. The larger backbone also has a moderately higher fraction of repairable A/B/C errors and a slightly higher repair rate among them. This supports our interpretation that Rethinking primarily acts as a local verification and correction mechanism, and is most effective when the base model already has sufficient latent reasoning capacity and the remaining errors are local, verifiable, or partially specified rather than global misunderstandings.

Dynamic block-size routing is not automatically Pareto-improving. We also test a preliminary dynamic block-size variant on Qwen2.5-VL-32B, where a lightweight MLP router selects $D \in$

$\{8, 16, 32, 64\}$ for each instance. Figure 3(a) shows that the dynamic policy reaches 37.98 Avg. with 14.64× speedup, placing it between fixed $D = 8$ and $D = 16$ but failing to improve the fixed-block Pareto frontier. In particular, fixed $D = 16$ and $D = 32$ both achieve higher accuracy with larger speedup. Figure 3(b) explains why: the router selects $D = 8$ for 85.2% of all test instances, with an average selected block size of 9.29, and this small-block preference is consistent across all task types. Thus, an accuracy-oriented dynamic router collapses toward the conservative small-block regime rather than learning a latency-efficient adaptive policy.

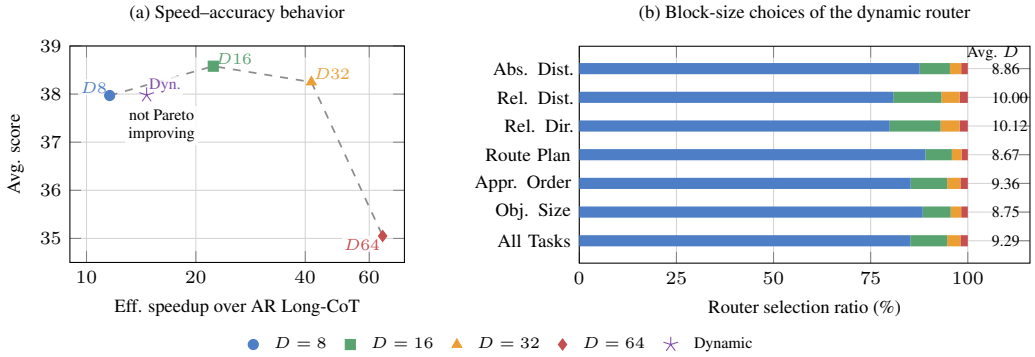

Figure 3: Dynamic block-size routing on Qwen2.5-VL-32B. (a) The dynamic router reaches 37.98 Avg. with 14.64× speedup, placing it near the small-block regime rather than improving the fixed-block Pareto frontier. (b) The selection distribution explains this behavior: the accuracy-oriented router overwhelmingly selects $D = 8$ across task types, with 85.2% $D = 8$ selections overall and an average selected block size of 9.29.

Impact of denoising step. Figure 4 ablates the denoising step number S on Qwen2.5-VL-32B with block size $D = 32$. The result shows an asymmetric speed–accuracy trade-off. Reducing S from 8 to 4 or 2 greatly improves speed, but the generated reasoning becomes under-denoised and accuracy drops sharply. For DiffThink, Avg. decreases from 38.25 at $S = 8$ to 28.69 at $S = 4$ and 21.04 at $S = 2$. The same pattern holds with Rethink×8, where Avg. decreases from 41.45 at $S = 8$ to 31.09 at $S = 4$ and 22.80 at $S = 2$. Increasing S from 8 to 16 yields only marginal quality gains while substantially reducing speed. For DiffThink, Avg. improves from 38.25 to 38.56, but effective speedup drops from 41.58× to 20.82×. For DiffThink+Rethink×8, Avg. improves from 41.45 to 41.68, but speedup drops from 4.57× to 2.29×. This confirms that $S = 8$ is a practical knee point of the speed–accuracy curve.

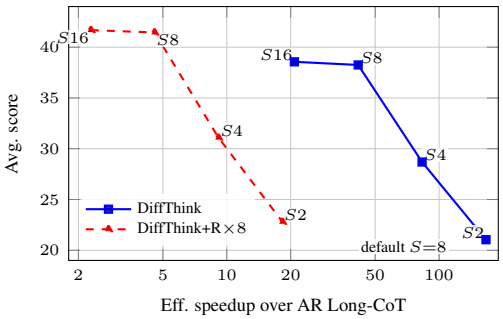

Figure 4: Denoising-step ablation on Qwen2.5-VL-32B with block size $D = 32$.

5 Conclusion

We target fast spatial reasoning under strict latency budgets and show that the key bottleneck of long-CoT VLMs is the sequential critical path of autoregressive decoding. Guided by this insight, we introduce a post-training Diffusion Thinking paradigm that directly converts pretrained AR-VLMs into diffusion-style VLMs for long-CoT spatial reasoning, enabling block-wise parallel refinement with KV-cache reuse and achieving up to $\sim 41\times$ higher throughput on a single H20 GPU. To recover accuracy lost from aggressive within-block parallelism, we further propose a test-time scaling via rethinking mechanism by allocating the saved latency budget to additional self-correction rounds.

Acknowledgments and Disclosure of Funding

This work was supported by Ministry of Education Tier 2 grant, Singapore (MOE-000762-01) and Ministry of Education Tier 2 grant, Singapore (MOE-000935-01). The authors declare no competing interests.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL `https://arxiv.org/abs/2511.21631`.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *Advances in Neural Information Processing Systems*, 38:55981–56007, 2026.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023a.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. Pmlr, 2023b.

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Mengfei Du, Binhao Wu, Zejun Li, Xuan-Jing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 346–355, 2024.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *Advances in neural information processing systems*, 38:52870–52888, 2026.

Dingming Li, Hongxing Li, Zixuan Wang, Yuchen Yan, Hang Zhang, Siqi Chen, Guiyang Hou, Shengpei Jiang, Wenqi Zhang, Yongliang Shen, et al. Viewspatial-bench: Evaluating multi-perspective spatial localization in vision-language models. In *European Conference on Computer Vision*, pages 95–111. Springer, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Ye Tian, Ling Yang, Jiongfan Yang, Anran Wang, Yu Tian, Haochen Wang, Zhiyang Teng, Zhuochen Wang, Yinjie Wang, Yunhai Tong, et al. Mmada-parallel: Multimodal large diffusion language models for thinking-aware editing and generation. In *International Conference on Learning Representations*, volume 2026, pages 36089–36123, 2026.

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. *arXiv preprint arXiv:2203.11171*, 2022a.

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. *arXiv preprint arXiv:2203.11171*, 2022b.

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.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10632–10643. IEEE, 2025.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *Advances in Neural Information Processing Systems*, 38:138867–138907, 2026a.

Rui Yang, Ziyu Zhu, Yanwei Li, Jingjia Huang, Shen Yan, Siyuan Zhou, Zhe Liu, Xiangtai Li, Shuangye Li, Wenqian Wang, et al. Visual spatial tuning. In *European Conference on Computer Vision*, pages 192–211. Springer, 2026b.

Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, et al. Mmsi-bench: A benchmark for multi-image spatial intelligence. In *International Conference on Learning Representations*, volume 2026, pages 157051–157088, 2026c.

A Dataset Visualization

Figure 5 visualizes representative examples from the Long-CoT Spatial Reasoning dataset. The examples cover all six task types used in our benchmark: Relative Direction, Relative Distance, Appearance Order, Object Size, Absolute Distance, and Route Plan. Each example contains the visual input, the question, the verified final answer, and the generated long-form reasoning trace structured into visual decomposition, spatial relationship mapping, logical deduction, and verification. This visualization illustrates that the generated CoT traces are used as dense reasoning supervision, while correctness is always determined by the verified final answer.

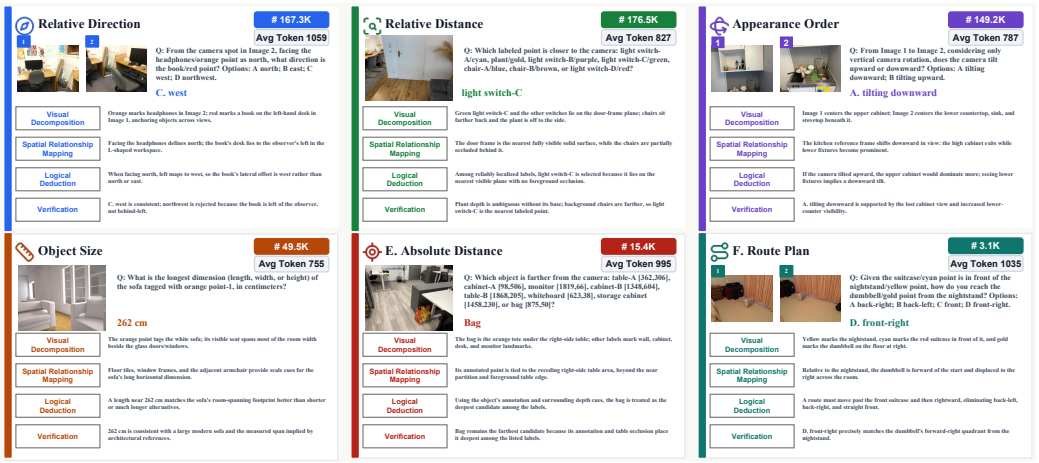

Figure 5: Visualization of representative Long-CoT Spatial Reasoning examples. The dataset covers six spatial task types: Relative Direction, Relative Distance, Appearance Order, Object Size, Absolute Distance, and Route Plan. Each example includes the visual input, spatial question, verified final answer, and a generated reasoning trace organized into visual decomposition, spatial relationship mapping, logical deduction, and verification. The bottom statistics report the number of instances and the average CoT length for each task type.

B Full Block-size and Rethinking Sweep

Tables 5 and 6 report the full numerical results for the block-size sweep and Rethinking sweep. The main text uses Table 1 for representative operating points and Figure 2 for the speed–accuracy frontier.

C Uniform vs. Non-Uniform Corruption

Our default training uses block-uniform corruption: all positions inside the same corrupted block share the same noise level. One may expect non-uniform corruption to better preserve critical reasoning steps. To test this, we exploit the structure of our generated long-CoT, which contain four stages: Observation, Analysis, Deduction, and Verification. We treat Deduction as the most critical stage for spatial inference and vary only its corruption weight while keeping the other stages fixed at 1.0.

Table 10 reports the results on Qwen2.5-VL-32B with $D = 32$. Increasing the Deduction noise to 1.2 hurts performance, especially on Route Plan, Absolute Distance, and Object Size. Reducing the Deduction noise to 0.8 improves some fine-grained categories such as Absolute Distance, Object Size, and Route Plan, but does not improve the overall average over uniform corruption. Thus, stage-aware corruption is an interesting direction, but our current evidence suggests that the simple uniform schedule is the most stable and competitive default.

Table 5: Full block-size and Rethinking sweep on Qwen2.5-VL. DT denotes Diffusion Thinking, D denotes block size, and R denotes the number of rethinking rounds.

Model	Method	Speed (tok/s)	Eff. Speedup	Avg.	Abs. Dist. (MRA)	Obj. Size (MRA)	Rel. Dist. (ACC)	Rel. Dir. (ACC)	Route Plan (ACC)	Appr. Order (ACC)
Qwen2.5-VL-3B	AR (w/ CoT)	116	1.00×	**30.88**	**28.90**	**27.40**	**35.10**	38.70	29.80	25.40
	DT ($D = 8$)	1204	10.91×	28.72	17.60	21.20	32.30	**40.30**	33.10	27.80
	DT ($D = 16$)	2238	20.00×	29.40	20.50	19.80	34.00	39.90	**34.60**	27.60
	DT ($D = 32$)	4230	38.18×	29.47	23.60	18.10	34.80	38.60	33.80	**27.90**
	DT ($D = 64$)	**6310**	**59.09**×	27.47	19.90	16.70	32.40	36.70	32.60	26.50
	DT ($D = 32$)+R×2	4103	18.52×	29.98	25.90	20.10	34.90	38.90	32.40	27.70
	DT ($D = 32$)+R×4	3934	8.87×	30.47	27.10	22.50	35.00	39.10	31.70	27.40
	DT ($D = 32$)+R×8	3722	4.20×	30.80	27.60	24.80	34.90	39.00	31.20	27.30
Qwen2.5-VL-7B	AR (w/ CoT)	58	1.00×	**36.63**	**35.60**	**34.10**	**41.50**	45.70	34.30	28.60
	DT ($D = 8$)	686	11.83×	32.93	19.10	26.60	35.10	**49.20**	39.70	27.90
	DT ($D = 16$)	1270	21.90×	33.48	22.80	24.90	37.90	46.60	40.30	28.40
	DT ($D = 32$)	2402	41.41×	33.98	26.10	23.40	40.30	44.00	**40.80**	**29.30**
	DT ($D = 64$)	**3850**	**66.38**×	31.05	21.90	21.80	35.80	40.30	39.40	27.10
	DT ($D = 32$)+R×2	2330	20.08×	34.77	29.80	26.40	40.60	44.80	37.90	29.10
	DT ($D = 32$)+R×4	2234	9.63×	35.62	32.10	29.10	41.10	45.10	36.90	**29.40**
	DT ($D = 32$)+R×8	2114	4.55×	36.18	33.50	32.00	41.30	44.90	36.10	**29.30**
Qwen2.5-VL-32B	AR (w/ CoT)	14.6	1.00×	40.78	40.30	**39.10**	44.90	48.80	37.90	**33.70**
	DT ($D = 8$)	169	11.58×	37.97	28.70	32.00	40.30	**53.10**	44.20	29.50
	DT ($D = 16$)	326	22.33×	38.58	31.80	30.30	42.80	51.40	**44.50**	30.70
	DT ($D = 32$)	607	41.58×	38.25	34.40	27.90	43.90	48.70	42.50	32.10
	DT ($D = 64$)	**954**	**65.34**×	35.05	30.30	24.80	40.50	44.30	40.10	30.30
	DT ($D = 32$)+R×2	564	13.58×	39.68	37.80	33.80	44.20	49.00	40.80	32.50
	DT ($D = 32$)+R×4	551	8.17×	40.80	39.60	36.50	44.60	**49.40**	41.70	33.00
	DT ($D = 32$)+R×8	534	4.57×	**41.45**	**40.80**	38.20	**45.10**	49.30	42.10	33.20

Table 6: Full block-size and Rethinking sweep on InternVL3. DT denotes Diffusion Thinking, D denotes block size, and R denotes the number of rethinking rounds.

Model	Method	Speed (tok/s)	Eff. Speedup	Avg.	Abs. Dist. (MRA)	Obj. Size (MRA)	Rel. Dist. (ACC)	Rel. Dir. (ACC)	Route Plan (ACC)	Appr. Order (ACC)
InternVL3-8B	AR (w/ CoT)	53.2	1.00×	**37.17**	**36.30**	**35.00**	**41.80**	46.20	35.20	**28.50**
	DT ($D = 8$)	628	11.80×	34.22	23.50	29.00	36.10	**50.40**	40.60	25.70
	DT ($D = 16$)	1159	21.79×	34.63	26.70	27.00	38.60	47.80	**41.20**	26.50
	DT ($D = 32$)	2167	40.73×	34.72	30.10	25.10	40.60	44.70	40.20	27.60
	DT ($D = 64$)	**3486**	**65.53**×	32.07	27.30	23.00	37.40	41.60	38.10	25.00
	DT ($D = 32$)+R×2	2102	19.76×	35.38	32.60	27.90	41.00	45.20	37.90	27.70
	DT ($D = 32$)+R×4	2015	9.46×	36.07	34.40	31.00	41.20	45.40	37.10	27.30
	DT ($D = 32$)+R×8	1907	4.48×	36.28	35.00	32.50	41.30	45.10	36.40	27.40
InternVL3-38B	AR (w/ CoT)	12.7	1.00×	41.27	41.20	**39.80**	45.20	49.60	38.90	33.00
	DT ($D = 8$)	147	11.57×	38.83	30.10	33.40	40.80	**53.80**	45.20	29.70
	DT ($D = 16$)	281	22.13×	39.25	32.90	31.00	43.40	51.50	**45.70**	31.00
	DT ($D = 32$)	531	41.81×	39.02	35.10	28.70	44.60	49.00	44.10	32.60
	DT ($D = 64$)	**843**	**66.38**×	36.15	31.30	26.00	41.40	45.40	42.30	30.50
	DT ($D = 32$)+R×2	495	13.71×	40.32	38.80	34.60	44.90	49.50	42.00	32.10
	DT ($D = 32$)+R×4	483	8.24×	41.58	40.50	37.40	45.40	50.10	42.80	33.30
	DT ($D = 32$)+R×8	467	4.60×	**42.38**	**41.60**	39.30	**45.80**	**50.40**	43.40	**33.80**

Table 7: Full task-wise denoising-step ablation on Qwen2.5-VL-32B with block size $D = 32$. The main text visualizes Avg. score and effective speedup in Figure 4.

Method	S	Speed (tok/s)	Eff. Speedup	Avg.	Abs. Dist.	Obj. Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order
DiffThink (D=32)	2	**2428**	**166.30**×	21.04	18.92	15.35	24.15	26.79	23.38	17.66
	4	1214	83.15×	28.69	25.80	20.93	32.93	36.53	31.88	24.08
	8	607	41.58×	38.25	34.40	27.90	43.90	48.70	42.50	32.10
	16	304	20.82×	**38.56**	**34.68**	**28.12**	**44.25**	**49.09**	**42.84**	**32.36**
DiffThink (D=32)+R×8	2	**2136**	**18.29**×	22.80	22.44	21.01	24.81	27.12	23.16	18.26
	4	1068	9.14×	31.09	30.60	28.65	33.83	36.98	31.58	24.90
	8	534	4.57×	41.45	40.80	38.20	45.10	49.30	42.10	33.20
	16	267	2.29×	**41.68**	**41.02**	**38.41**	**45.35**	**49.57**	**42.33**	**33.38**

Table 8: Dynamic block-size ablation on Qwen2.5-VL-32B. The dynamic router selects from $D \in \{8, 16, 32, 64\}$ per instance.

Block Size	Speed	Speedup	Avg.	Abs. Dist.	Obj. Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order
$D = 8$	169	11.58×	37.97	28.70	**32.00**	40.30	**53.10**	44.20	29.50
$D = 16$	326	22.33×	**38.58**	31.80	30.30	42.80	51.40	**44.50**	30.70
$D = 32$	607	41.58×	38.25	**34.40**	27.90	**43.90**	48.70	42.50	**32.10**
$D = 64$	**954**	**65.34**×	35.05	30.30	24.80	40.50	44.30	40.10	30.30
Dynamic $\{8, 16, 32, 64\}$	214	14.64×	37.98	29.22	31.56	40.66	52.62	44.09	29.72

Table 9: Block-size selection distribution of the dynamic router. The policy strongly prefers small blocks, indicating that accuracy-oriented routing collapses toward the low-latency-gain regime.

Task	% $D = 8$	% $D = 16$	% $D = 32$	% $D = 64$	Avg. D
Absolute Distance	87.6	7.8	2.9	1.7	8.86
Relative Distance	80.8	12.4	4.7	2.1	10.00
Relative Direction	79.9	13.1	4.9	2.1	10.12
Route Plan	89.1	6.8	2.5	1.6	8.67
Appearance Order	85.3	9.4	3.4	1.9	9.36
Object Size	88.4	7.2	2.7	1.7	8.75
All Tasks	85.2	9.5	3.4	1.9	9.29

Table 10: Uniform versus stage-aware non-uniform corruption on Qwen2.5-VL-32B with $D = 32$. Only the Deduction-stage corruption weight is changed; Observation, Analysis, and Verification remain at 1.0.

Corruption Policy	Avg.	Abs. Dist.	Obj. Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order
Uniform, all stages = 1.0	**38.25**	34.40	27.90	**43.90**	**48.70**	42.50	**32.10**
Higher noise on Deduction, 1.2	37.32	32.90	26.70	43.30	48.20	40.60	30.90
Lower noise on Deduction, 0.8	38.18	**34.90**	**28.40**	43.60	48.50	**42.90**	31.80

D Algorithms and Implementation Details

This appendix provides pseudocode for the two stages of Diffusion Thinking: post-training a pre-trained autoregressive VLM with block-wise denoising, and performing block-wise diffusion inference with optional Rethinking. We use D for block size, $K = \lceil L/D \rceil$ for the number of response blocks, S for the number of denoising steps per block, and N for the number of rethinking rounds.

E Dataset Construction and Evaluation Details

Data source and task scope. We construct a *Long-CoT Spatial Reasoning* dataset for training long-form spatial reasoning and diffusion-style generation in VLMs. The dataset follows the task taxonomy and evaluation format of VSI-Bench [Yang et al., 2025], while the underlying verified visual-spatial annotations are derived from Visual Spatial Tuning (VST) [Yang et al., 2026b]. Each raw instance contains an image or multi-image observation I, a question prompt P, a task label τ, and a verified final answer y. The verified answer y is the only ground-truth label used for evaluation.

The corpus contains **561K** spatial VQA instances covering both single-image and multi-image settings. We consider six task types: Relative Distance, Relative Direction, Appearance Order, Object Size, Absolute Distance, and Route Plan. Table 11 reports the task distribution and CoT-length statistics.

Generated CoT as training supervision. The original VST annotations provide verified answers but do not include chain-of-thought traces. We therefore generate long-form CoT as *training supervision*, not as evaluation ground truth. For each verified instance (I, P, y, τ), Qwen3-VL-Plus [Bai et al., 2025] first produces an answer-constrained CoT trace that analyzes the visual evidence and spatial relations and concludes with the verified answer y. Qwen3-Max then refines the trace for coherence, step ordering, and logical consistency while preserving the same final answer.

Algorithm 2 Post-training an AR-VLM into a Diffusion-Thinking VLM

Require: Pretrained VLM parameters θ, training set $\mathcal{D}_{\text{train}}$, block size D, noise sampler $q(t)$, masking policy $\pi(\cdot \mid t)$, learning rate η

for each minibatch $\{(I, P, x_0)\} \sim \mathcal{D}_{\text{train}}$ **do**
  Encode multimodal context $c \leftarrow \Phi_\theta(I, P)$
  Pad each response x_0 to length $L' = D\lceil L/D \rceil$
  Split x_0 into $K = \lceil L/D \rceil$ blocks:

$$x_0 = [x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(K)}], \qquad x_0^{(k)} \in \mathcal{V}^D$$

  Initialize minibatch loss $\mathcal{L} \leftarrow 0$
  for $k = 1$ **to** K **do**
    Sample a block noise level $t_k \sim q(t)$
    Sample corrupted positions $\mathcal{M}_k \sim \pi(\cdot \mid t_k)$
    Corrupt current block:

$$\tilde{x}_{t_k}^{(k)} \leftarrow \text{Mask}\Big(x_0^{(k)}, \mathcal{M}_k\Big)$$

    Use clean prefix blocks $x_0^{(<k)}$ as causal context
    Apply the block attention mask: bidirectional attention inside block k, causal attention to c and $x_0^{(<k)}$
    Predict masked tokens:

$$p_\theta\left(\cdot \mid c,\, x_0^{(<k)},\, \tilde{x}_{t_k}^{(k)}\right)$$

    Accumulate cross-entropy only on corrupted positions:

$$\mathcal{L} \leftarrow \mathcal{L} - \frac{\lambda(t_k)}{|\mathcal{M}_k|} \sum_{i \in \mathcal{M}_k} \log p_\theta\left(x_0^{(k)}[i] \mid c,\, x_0^{(<k)},\, \tilde{x}_{t_k}^{(k)}\right)$$

  end for
  Update parameters:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$$

end for
return Post-trained Diffusion-Thinking VLM parameters θ

Algorithm 3 Diffusion Thinking Inference with Optional Rethinking

Require: Multimodal context $c = \Phi_\theta(I, P)$, block size D, denoising steps S, maximum number of blocks $K_{\max}$, rethinking rounds N, rethinking instruction $\mathcal{I}_{\text{rethink}}$

Run one block-wise diffusion pass:

$$y^{(0)} \leftarrow \text{BlockDiffuse}(c, D, S, K_{\max})$$

for $r = 1$ **to** N **do**
  Append a lightweight self-correction instruction:

$$c^{(r)} \leftarrow [\, c;\, y^{(r-1)};\, \mathcal{I}_{\text{rethink}}\,]$$

  Reuse the KV cache for the prefix $[c; y^{(r-1)}; \mathcal{I}_{\text{rethink}}]$
  Generate a correction trace with block-wise diffusion:

$$\Delta y^{(r)} \leftarrow \text{BlockDiffuse}(c^{(r)}, D, S, K_{\max})$$

  Append the correction trace:

$$y^{(r)} \leftarrow [\, y^{(r-1)};\, \mathcal{I}_{\text{rethink}};\, \Delta y^{(r)}\,]$$

end for
Extract the final answer from $y^{(N)}$ using the deterministic VSI-style answer-extraction rule
return Final extracted answer

Algorithm 4 BLOCKDIFFUSE: Block-wise Diffusion Decoding

Require: Context c, block size D, denoising steps S, maximum number of blocks $K_{\max}$

Initialize generated prefix $y \leftarrow \emptyset$

Initialize and store KV cache for context c

for $k = 1$ **to** $K_{\max}$ **do**

  Initialize the current block as fully masked:

$$\tilde{x}_S^{(k)} \leftarrow [\texttt{MASK}]^D$$

  for $s = S$ **to** 1 **do**

    Apply block attention mask: tokens in the current block attend to c, y, and all positions in the current block

    Predict all masked positions in parallel:

$$p_\theta^{(k,s)} = p_\theta\left(\cdot \mid c,\, y,\, \tilde{x}_s^{(k)}\right)$$

    Update the current block using the denoising schedule:

$$\tilde{x}_{s-1}^{(k)} \leftarrow \mathcal{U}_s\left(\tilde{x}_s^{(k)}, p_\theta^{(k,s)}\right)$$

  end for

  Commit the denoised block:

$$\hat{x}^{(k)} \leftarrow \tilde{x}_0^{(k)}$$

  Append committed block to generated prefix:

$$y \leftarrow [y; \hat{x}^{(k)}]$$

  Update KV cache with the committed block $\hat{x}^{(k)}$

  if <eos> appears in $\hat{x}^{(k)}$ **then**

    Remove tokens after <eos>

    break

  end if

end for

return Generated reasoning trace and answer y

Table 11: Task distribution and CoT-length statistics of the Long-CoT Spatial Reasoning dataset. CoT length is measured with the Qwen2.5-VL tokenizer.

Task Type	**# Instances**	**Avg. CoT Length (tokens)**
Relative Distance	176.5K	827
Relative Direction	167.3K	1059
Appearance Order	149.2K	787
Object Size	49.5K	755
Absolute Distance	15.4K	995
Route Plan	3.1K	1035
Total	561K	894

This construction is intentionally conservative. The generated CoT trace is not treated as an independent proof that the answer is correct, and it is never used as the evaluation target. Instead, it provides dense supervision for teaching the model to produce long-form reasoning traces. All reported benchmark scores are computed only from the extracted final answer and the verified source annotation y.

CoT quality control. To reduce noise from generated CoT traces, we apply answer-consistency and format filters before using an instance for training. First, we extract the final answer from the generated/refined CoT using the same answer-extraction rules as evaluation, and retain the instance only if the extracted answer matches the verified label y. For multiple-choice tasks, the extracted

option letter must match exactly; for numerical-answer tasks, the value must be parseable and normalized to the expected answer format. Second, we remove CoT traces with invalid structure, missing final answers, duplicated or degenerate text, or contradictions between the reasoning trace and the final answer. Third, after Qwen3-Max refinement, we run the answer-consistency check again to ensure that refinement does not alter the verified final answer.

This filtering does not guarantee that every generated CoT trace is fully faithful. However, it prevents generated CoT from changing benchmark labels, and it separates the role of CoT supervision from the role of verified answers: CoT traces supervise the reasoning format during training, while verified annotations determine correctness during evaluation.

Evaluation protocol. We evaluate all methods with a deterministic VSI-style protocol rather than LLM-as-a-judge scoring. Following VSI-Bench [Yang et al., 2025], Relative Distance, Relative Direction, Route Plan, and Appearance Order are evaluated by exact-match Accuracy (ACC), while Absolute Distance and Object Size are evaluated by Mean Relative Accuracy (MRA). For every model output, we first extract the final option letter or numerical value using the VSI-style answer-extraction procedure, and then compare the extracted answer against the verified source annotation. The main test set contains **6,000** examples, with 1,000 examples for each of the six task types. The test split is disjoint from the training split at the question-instance level.

F Training Targets and Implementation Details

Training targets for baselines. All methods are trained or post-trained on the same Long-CoT Spatial Reasoning training split. The difference is the target format and generation interface. For *AR Answer-only*, we remove the generated CoT trace from each training target and train the autoregressive VLM to directly predict the verified final answer. For *AR Long-CoT*, the model is trained to generate the CoT trace followed by the same verified final answer. DoT and DiffCoT are trained on the same long-CoT targets and use their corresponding correction-style generation procedures. DiffThink uses the same long-CoT targets but replaces next-token prediction with the block-wise denoising objective. Rethinking does not introduce an additional training set; it is a test-time self-correction procedure applied to the trained DiffThink model.

Training hyperparameters. We fine-tune all models for one epoch using AdamW with a cosine learning-rate schedule and linear warmup. Unless otherwise specified, training uses 8×H20 GPUs, bf16 mixed precision, and gradient checkpointing. We set the peak learning rate to 2×10^{-5}, warmup ratio to 3%, weight decay to 0.1, and gradient clipping to 1.0. For block-wise diffusion, each response is padded to a multiple of block size D and partitioned into $K = \lceil L/D \rceil$ blocks. We train fixed block-size variants with $D \in \{8, 16, 32, 64\}$ and use $S = 8$ denoising steps per block by default. Ablations over D, S, and Rethinking rounds are reported in the main experiments and appendix.

Speed measurement. Speed is measured on a single H20 GPU. Effective speedup is computed from end-to-end wall-clock latency relative to AR Long-CoT on the same backbone. For methods with substantially different output lengths, such as AR Answer-only and AR Long-CoT, token throughput alone is not a meaningful speedup proxy. Therefore, token/s is reported as a descriptive throughput statistic, while Eff. Speedup is computed from measured end-to-end latency. For Rethinking, token/s reports the raw decoding throughput of diffusion passes, while Eff. Speedup accounts for all rethinking rounds. Absolute latency distributions, output lengths, peak memory, per-stage timing, and measurements on an additional GPU class (H200) are reported in Sec. 4.3 and Appendix H.

G Design Considerations for Visual Inputs

The multimodal prefix is expensive but reusable: it is encoded once, never corrupted, and cached across blocks and rethinking rounds. Spatial deductions repeatedly reground in the same visual evidence, which motivates clean committed blocks and causal cross-block attention rather than globally denoising the whole rationale. Finally, relational, directional, and route constraints tolerate within-block parallelism better than metric estimation, which motivates a fast visual draft followed by cached verification (Sec. 3.5).

H Deployment Profiling Details

Measurement setup. Latency is measured from image-tensor-ready input through vision encoding, prefill, generation, and final-answer extraction, excluding disk I/O and host-side image preprocessing. All AR, DT, and Rethinking measurements use batch/concurrency 1, bf16, deterministic decoding (temperature 0, top-p 1), 20 warm-up requests, synchronized per-request timing, and clean-prefix KV caching. Both GPU classes (H20 and H200) use Ubuntu 22.04, driver 570.124.06, CUDA 12.8, PyTorch 2.7.1, Transformers 4.53.2, FlashAttention-2 v2.8, and Triton 3.3 with eager execution; `torch.compile`, CUDA Graphs, TensorRT-LLM, xFormers, and JIT compilation are disabled. Every speedup is computed relative to AR Long-CoT on the same GPU. Our latency claim concerns single-request latency, not batch$>$1 throughput serving; DiffThink is architecturally compatible with optimized attention, quantization, tensor parallelism, prefix caching, and continuous batching, but we do not claim unmeasured multiplicative gains from such combinations. Table 12 reports full latency distributions and peak memory, Table 13 reports output-length distributions, and Table 14 breaks the end-to-end latency into stages.

Table 12: Full end-to-end latency distributions (mean/P50/P95 in seconds) at batch/concurrency 1, effective speedup over AR Long-CoT on the same GPU, and peak GPU memory. DT denotes DiffThink ($D = 32$), and R$\times$8 denotes DT with eight rethinking rounds.

Backbone	GPU	AR mean/P50/P95 (s)	DT mean/P50/P95 (s)	DT Eff. Speedup	R$\times$8 mean/P50/P95 (s)	R$\times$8 Eff. Speedup	Peak GB AR/DT/R$\times$8
Qwen2.5-VL-32B	H20	62.877/60.548/88.219	1.512/1.451/2.002	41.58$\times$	13.758/13.315/17.453	4.57$\times$	72.8/75.0/89.7
Qwen2.5-VL-32B	H200	50.216/48.359/70.458	0.977/0.939/1.289	51.40$\times$	10.254/9.926/12.995	4.90$\times$	72.8/75.0/89.7
InternVL3-38B	H20	72.913/70.157/102.362	1.744/1.678/2.320	41.81$\times$	15.850/15.332/20.171	4.60$\times$	83.6/86.2/89.4
InternVL3-38B	H200	57.996/55.807/81.422	1.115/1.075/1.480	52.00$\times$	11.722/11.341/14.902	4.95$\times$	83.6/86.2/89.4
Qwen3.5-27B	H20	49.574/47.713/69.787	1.187/1.141/1.580	41.76$\times$	10.615/10.271/13.533	4.67$\times$	63.5/65.6/81.4
Qwen3.5-27B	H200	39.576/38.092/55.714	0.759/0.731/1.006	52.16$\times$	7.901/7.647/10.062	5.01$\times$	63.5/65.6/81.4

Table 13: Output-length distributions (tokens, mean/P50/P95). R$\times$8 counts the cumulative tokens of the initial trace and eight corrections.

Backbone	AR	DT	R$\times$8
Qwen2.5-VL-32B	918/884/1,288	918/881/1,215	7,347/7,110/9,320
InternVL3-38B	926/891/1,300	926/891/1,232	7,402/7,160/9,420
Qwen3.5-27B	932/897/1,312	932/896/1,240	7,452/7,210/9,500

Table 14: Mean per-stage latency (seconds) at batch/concurrency 1. R$\times$8 corrections denotes the total latency of the eight cached correction rounds.

Backbone	GPU	Vision	Prefill	AR decode	DT denoise	R$\times$8 corrections
Qwen2.5-VL-32B	H20	0.063	0.110	62.704	1.339	12.246
Qwen2.5-VL-32B	H200	0.020	0.033	50.163	0.924	9.277
InternVL3-38B	H20	0.075	0.130	72.708	1.539	14.106
InternVL3-38B	H200	0.023	0.038	57.935	1.054	10.606
Qwen3.5-27B	H20	0.071	0.100	49.403	1.016	9.428
Qwen3.5-27B	H200	0.022	0.031	39.523	0.706	7.143

Systems comparison on two GPU classes. Table 15 extends Table 2 with absolute latency on H20 and H200. AR Long-CoT, Native MTP-1, DFlash, and DT each generate 932 tokens on average, while R$\times$8 generates 7,452 cumulative tokens. Native MTP-1 reaches 1.65$\times$/1.76$\times$ H20/H200 speedup, and lossless DFlash [Chen et al., 2026], run with its official implementation, reaches 3.50$\times$/3.57$\times$. DT is much faster at 41.76$\times$/52.16$\times$ but has a lower initial Avg.; R$\times$8 reaches the highest observed Avg. (42.48, numerically +0.38 over AR) while remaining 4.67$\times$/5.01$\times$ faster than AR and 1.34$\times$/1.40$\times$ faster than DFlash. On Qwen3.5-27B, H200 accelerates AR by $\sim$1.25$\times$ and DT by $\sim$1.56$\times$, increasing the same-GPU DT speedup from 41.76$\times$ to 52.16$\times$ while outputs remain unchanged. Table 16 isolates the training objective, whereas this table establishes the systems frontier.

Table 15: Systems comparison on Qwen3.5-27B with absolute latency (mean/P50/P95, seconds) on H20 and H200. Speedups are relative to AR Long-CoT on the same GPU.

Method	Avg.	H20 mean/P50/P95 (s)	H20 Speedup	H200 mean/P50/P95 (s)	H200 Speedup	Peak GB
AR Long-CoT	42.10	49.574/47.713/69.787	1.00×	39.576/38.092/55.714	1.00×	63.5
Native MTP-1	41.82	30.065/28.935/42.323	1.65×	22.436/21.434/32.556	1.76×	64.8
DFlash	42.10	14.172/13.558/21.791	3.50×	11.086/10.577/16.846	3.57×	68.3
DiffThink ($D = 32$)	39.55	1.187/1.141/1.580	41.76×	0.759/0.731/1.006	52.16×	65.6
DiffThink ($D = 32$) + R×8	42.48	10.615/10.271/13.533	4.67×	7.901/7.647/10.062	5.01×	81.4

I Objective Isolation and MMSI-Bench Details

Objective isolation. FMB-8 in Table 16 matches DT in backbone (Qwen2.5-VL-32B), training data, optimization budget, $D = 32$ block mask, and eight sequential passes per block; only the training objective changes from multi-noise diffusion CE to fixed-mask iterative CE. At matched depth and latency, DT gains +1.93 Avg.: +2.60 on Absolute Distance, +2.10 on Object Size, +2.10 on Relative Distance, +2.20 on Relative Direction, +1.50 on Route Plan, and +1.10 on Appearance Order. DBP-1 uses a single sequential pass per block with a one-shot block CE objective; it is a speed diagnostic rather than the matched control. Because AR Long-CoT, FMB-8, and DT share identical generated targets, this comparison also holds teacher-trace quality constant.

Table 16: Objective isolation on Qwen2.5-VL-32B ($D = 32$, H20). FMB-8 matches DT except for the training objective.

Method	Objective	Passes/ block	Latency (s)	Eff. Speedup	Avg.
AR Long-CoT	AR reference	32	62.877	1.00×	40.78
DBP-1	One-shot block CE (diagnostic)	1	0.275	228.64×	20.00
FMB-8	Fixed-mask iterative CE	8	1.550	40.57×	36.32
DT ($S = 8$)	Multi-noise diffusion CE	8	1.512	41.58×	38.25

MMSI-Bench protocol. MMSI-Bench [Yang et al., 2026c] contains 1,000 human-designed multi-image questions over 1,990 images from eight real-world sources, with 11 categories covering relations, measurement, appearance, motion, and 3–11-step reasoning. We evaluate zero-shot: no MMSI example is used for training, tuning, checkpoint selection, or prompt optimization. Scoring uses the official deterministic parser and exact-match ACC, without an LLM judge. On the multi-step subset, AR/DT/R×8 scores are 29.8/29.3/31.3 on Qwen2.5-VL-32B, 26.8/28.3/29.3 on InternVL3-38B, and 31.3/32.3/33.3 on Qwen3.5-27B.

J Round-wise Rethinking Analysis

Table 17 reports complete R0–R8 trajectories for the three largest backbones. R0 denotes the initial DT pass, and Rr includes that pass plus corrections R1, …, Rr. Latency is the cumulative end-to-end cost through round Rr unless marked as per-round, speedup is relative to AR Long-CoT on the same GPU, and transitions are cumulative relative to the R0 answer. Categorical tasks use exact option match; numerical tasks use per-instance MRA ≥ 0.50 (relative error $< 30\%$) as the primary correctness criterion. Six-task Avg. is the macro-average score over the six tasks.

Front-loaded recovery and correction length. R4 captures 79.7%, 75.7%, and 77.9% of the final R×8 recovery on Qwen2.5-VL-32B, InternVL3-38B, and Qwen3.5-27B, respectively, and R6 captures 93.2%, 92.1%, and 92.7% while retaining 5.85–6.44× speedup. Mean correction length and per-round H20 latency decrease monotonically, and P95 added tokens also decline from R1 to R8 (1,060→985, 1,075→993, and 1,085→997). Thus, R×8 is a high-quality endpoint rather than a required budget, and R4–R6 already provide strong intermediate operating points; the useful fixed stopping region is consistent across model families and both GPUs.

Statistical reliability. For Qwen2.5-VL-32B and InternVL3-38B, all two-sided exact McNemar tests on the paired transitions have $p < 4 \times 10^{-11}$, and R×8 yields 204/12 and 215/13 fixes/breaks,

Table 17: Round-wise Rethinking trajectories with DiffThink ($D = 32$). W→C, C→W, and Changed are cumulative percentages of test examples relative to R0; latency is cumulative end-to-end latency through each round, except the per-round row, which gives the latency of each correction round alone.

Metric	R0	R1	R2	R3	R4	R5	R6	R7	R8
Qwen2.5-VL-32B									
Six-task Avg.	38.25	39.05	39.68	40.30	40.80	41.05	41.23	41.37	41.45
W→C (%)	0.00	0.83	1.52	2.17	2.68	2.97	3.17	3.30	3.40
C→W (%)	0.00	0.03	0.08	0.12	0.13	0.17	0.18	0.18	0.20
Changed (%)	0.00	0.87	1.60	2.28	2.82	3.13	3.35	3.48	3.60
Added tokens per round	–	858	836	822	807	795	783	771	757
H20 per-round latency (s)	–	1.570	1.548	1.540	1.530	1.525	1.519	1.512	1.502
H20 latency (s)	1.512	3.082	4.630	6.171	7.701	9.225	10.744	12.256	13.758
H20 Eff. Speedup	41.58	20.40	13.58	10.19	8.17	6.82	5.85	5.13	4.57
H200 latency (s)	0.977	2.166	3.339	4.506	5.665	6.820	7.971	9.117	10.254
H200 Eff. Speedup	51.40	23.18	15.04	11.14	8.86	7.36	6.30	5.51	4.90
InternVL3-38B									
Six-task Avg.	39.02	39.75	40.32	41.00	41.58	41.89	42.12	42.25	42.38
W→C (%)	0.00	0.78	1.40	2.10	2.72	3.05	3.30	3.45	3.58
C→W (%)	0.00	0.05	0.10	0.12	0.17	0.18	0.20	0.22	0.22
Changed (%)	0.00	0.83	1.50	2.22	2.88	3.23	3.50	3.67	3.80
Added tokens per round	–	865	842	827	812	800	788	777	765
H20 per-round latency (s)	–	1.798	1.775	1.767	1.759	1.756	1.753	1.751	1.746
H20 latency (s)	1.744	3.542	5.317	7.085	8.844	10.600	12.353	14.104	15.850
H20 Eff. Speedup	41.81	20.58	13.71	10.29	8.24	6.88	5.90	5.17	4.60
H200 latency (s)	1.115	2.467	3.802	5.131	6.453	7.774	9.092	10.408	11.722
H200 Eff. Speedup	52.00	23.51	15.26	11.30	8.99	7.46	6.38	5.57	4.95
Qwen3.5-27B									
Six-task Avg.	39.55	40.23	40.77	41.35	41.83	42.08	42.27	42.38	42.48
W→C (%)	0.00	0.72	1.28	1.90	2.42	2.68	2.88	3.02	3.12
C→W (%)	0.00	0.03	0.07	0.10	0.13	0.15	0.17	0.18	0.18
Changed (%)	0.00	0.75	1.35	2.00	2.55	2.83	3.05	3.20	3.30
Added tokens per round	–	870	848	833	817	805	793	783	771
H20 per-round latency (s)	–	1.213	1.195	1.187	1.176	1.171	1.166	1.163	1.157
H20 latency (s)	1.187	2.400	3.595	4.781	5.957	7.129	8.295	9.458	10.615
H20 Eff. Speedup	41.76	20.66	13.79	10.37	8.32	6.95	5.98	5.24	4.67
H200 latency (s)	0.759	1.677	2.582	3.481	4.372	5.260	6.143	7.024	7.901
H200 Eff. Speedup	52.16	23.60	15.32	11.37	9.05	7.52	6.44	5.63	5.01

respectively. For numerical tasks under the primary 30% rule, R×8 corrects 10.8% and 11.1% of the initially wrong numerical answers and breaks 0.65% of the initially correct ones; the repair rate exceeds the break rate under 25%, 30%, and 35% relative-error thresholds (by 9.35/9.65, 10.15/10.45, and 9.95/10.25 points), and the paired continuous ΔMRA is +8.35 [7.50, 9.20] and +8.55 [7.70, 9.40]. At R×8, fixes greatly outnumber breaks on every backbone, and this direction is unchanged at the 25%, 30%, and 35% numerical-error thresholds. This links the aggregate recovery to asymmetric repair of the numerical failure mode rather than to arbitrary answer changes.

Adaptive stopping. Table 18 evaluates adaptive stopping on Qwen2.5-VL-32B. Stable-2 stops after two consecutive stable normalized answers (the same option, or at most 5% relative change for numerical answers), with a minimum of two and a maximum of eight rounds. Margin-Stable additionally requires an option margin of at least 1.0 nat or a numerical answer-span probability of at least 0.80. Thresholds are selected once on 500 held-out training-split examples subject to an H20 speedup of at least 10× and frozen before testing. As Table 18 shows, Margin-Stable retains most of the R×8 quality (41.31 vs. 41.45 Avg.) at 10.11× H20 speedup without test-set tuning. We restrict the adaptive claim to Qwen2.5-VL-32B rather than infer unmeasured results for the other backbones.

Representative outcomes. Verified examples cover all three outcomes. In Route Plan, Rethinking changes `front` to `front-right` after recovering omitted rightward evidence (fix). In Object Size, it moves the estimate from 168 to 182 cm against a 262 cm ground truth because the scale anchor

Table 18: Fixed-round versus adaptive stopping on Qwen2.5-VL-32B with DiffThink ($D = 32$). Latency is cumulative end-to-end latency on H20.

Strategy	Stopping rule	Avg.	H20 latency (s)	H20 Eff. Speedup
Fixed R2	Always stop at R2	39.68	4.630	13.58×
Fixed R4	Always stop at R4	40.80	7.701	8.17×
Fixed R8	Always stop at R8	41.45	13.758	4.57×
Stable-2	Two consecutive stable normalized answers; min 2/max 8	40.96	5.790	10.86×
Margin-Stable	Stable-2 plus frozen confidence threshold	41.31	6.220	10.11×

remains wrong (failure). In Relative Direction, it changes `west` to `northwest` after over-interpreting depth (break).

K Human Audit of Generated Reasoning Traces

Final correctness always uses verified source labels, never teacher traces or an LLM judge, and AR Long-CoT, FMB-8, and DT share identical generated training targets, which holds teacher-trace quality constant in the objective comparison. To assess the generated reasoning itself, we conduct paired 200-example audits on the six-task test set (stratified) and on MMSI-Bench against independent human step-by-step references. Human 1–5 ratings assess support from visual evidence (faithfulness), internal logic (coherence on the six-task set and spatial logic on MMSI), coverage of decisive steps (completeness), fluency, and repetition (non-redundancy); contradiction is the share of traces containing any image-, label-, or internally inconsistent claim. Table 19 reports the results; the MMSI rows are for Qwen3.5-27B, and the same direction holds on Qwen2.5-VL-32B and InternVL3-38B.

Table 19: Human audit of generated reasoning traces on 200 paired examples per audit set. Ratings are on a 1–5 scale (higher is better); Contradiction is the share of traces with any inconsistent claim (lower is better).

Audit set	Method	Faith.	Logic	Comp.	Fluency	Non-red.	Contradiction
Six-task	AR	4.15	4.20	4.05	4.35	4.15	6.5%
Six-task	DT	3.90	3.85	3.80	4.05	3.95	11.0%
Six-task	DT+R×8	4.12	4.00	4.20	3.95	3.30	7.5%
MMSI-human ref.	AR	4.12	4.02	4.05	4.30	4.16	6.0%
MMSI-human ref.	DT	3.88	3.84	3.82	4.08	3.98	10.0%
MMSI-human ref.	DT+R×8	4.20	4.10	4.20	3.96	3.40	6.5%

On the six-task set, the gap is concentrated in numerical and scale tasks: faithfulness is 4.15/3.63/4.05 for AR/DT/DT+R×8 on Absolute Distance and Object Size, versus 4.15/4.05/4.15 on the four relational and global tasks. Initial DT thus has a modest trace-quality gap; R×8 recovers most of the six-task faithfulness gap (4.12 vs. 4.15 for AR) and exceeds AR on MMSI (4.20 vs. 4.12), improves completeness, and brings contradictions close to AR, at the cost of redundancy from appended corrections. Because the MMSI audit uses independent human references, the trend persists outside the teacher's style. Relaxed within-block causality is therefore more harmful for metric and scale accumulation than for global relational constraints, and Rethinking repairs local, verifiable errors on capable backbones while reducing concision.

L Limitations

Our work targets the long-CoT spatial reasoning regime and does not claim that all spatial questions require long-CoT reasoning. For simple perception-oriented queries, direct answering can be more appropriate. The long CoT in our dataset are generated and filtered as training supervision; although verified answers determine evaluation correctness, generated CoT may not always be fully faithful. Our latency results are measured on H20 and H200 GPUs at batch/concurrency 1 and may vary across hardware, kernels, and serving systems; we claim single-request latency, not batch$>$1 throughput serving, and do not claim unmeasured multiplicative gains from combining DiffThink with optimized serving stacks. Rethinking is most effective on larger backbones, while smaller models recover less

accuracy from additional self-correction. We verify transfer across three backbone lines and zero-shot on MMSI-Bench, but do not claim universality across every VLM or across multimodal domains beyond spatial and multi-image reasoning. Initial DiffThink traces are less faithful on numerical tasks, and appended Rethinking corrections reduce concision. Adaptive stopping is evaluated only on Qwen2.5-VL-32B. Finally, our main experiments use fixed block sizes; latency-aware adaptive block routing remains an important direction for future work.

Broader impacts. Diffusion Thinking reduces the latency and compute cost of long-form VLM reasoning, which can make long-form spatial reasoning more practical in latency-sensitive embodied and interactive applications. Faster VLM reasoning may also be deployed in embodied or surveillance-like settings, where incorrect spatial predictions may cause harm.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the scope as fast long-form spatial reasoning under a latency budget, rather than claiming that all spatial tasks require long-CoT reasoning.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in appendix, including the use of generated rationales as training supervision, the fact that not all spatial questions require long-form reasoning, hardware-dependent speedups, weaker Rethinking gains on smaller backbones, and the current use of fixed block sizes.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theorems or proof-based theoretical claims. The method section provides latency accounting and complexity-style analysis with explicitly defined quantities, but these are used as explanatory analysis rather than formal theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the dataset construction, deterministic evaluation protocol, answer-extraction procedure, training hyperparameters, inference settings, baselines, and hardware setup in Sec. 4.1 and the appendix. Pseudocode for post-training, block-wise diffusion inference, and Rethinking is provided in Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We do not provide a fully public code and data release with this paper. We provide detailed reproducibility instructions, pseudocode, hyperparameters, prompts, filtering procedures, and evaluation protocols in the paper and appendix, and plan to release the dataset, code, and generation pipeline publicly.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4.1 specifies the data source, task types, test split, and evaluation metrics, and Appendix F specifies the optimizer, learning rate, warmup, weight decay, gradient clipping, precision, GPU setup, block sizes, and denoising steps. Additional full sweep and ablation details are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: We report deterministic evaluation results on a 6,000-sample test set across five VLM backbones, but we do not report error bars for the main tables. For Rethinking, we report two-sided exact McNemar tests on paired answer transitions and paired ΔMRA intervals (Sec. 4.3 and Appendix J). Running repeated full fine-tuning and inference sweeps for all large VLM backbones would be computationally expensive; instead, we report consistent trends across model families, model scales, and ablation settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F reports the compute setup, including the use of 8$\times$H20 GPUs for training, bf16 mixed precision, gradient checkpointing, and single-GPU inference measurement on H20, and Appendix H reports deployment profiling on H20 and H200, including latency and peak memory. Training wall-clock time is not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. The work studies algorithmic acceleration and evaluation of VLM reasoning, uses existing visual-spatial annotations and generated rationales, and does not involve deception or unsafe deployment; the only human involvement is the rating audit of model-generated reasoning traces in Appendix K.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix L discusses potential positive impacts such as reducing the latency and compute cost of long-form VLM reasoning, as well as risks from deploying faster VLM reasoning in embodied or surveillance-like settings where incorrect spatial predictions may cause harm.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk general-purpose generative model or a scraped dataset. The proposed dataset is derived from existing visual-spatial annotations with generated rationales, and release documentation will include data provenance, filtering procedures, and usage guidance.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: We cite the datasets and benchmarks used in the paper (VSI-Bench, VST, and MMSI-Bench) and the models used to generate CoT supervision (Qwen3-VL-Plus and Qwen3-Max), but the paper does not cite every backbone model family or list the exact license and terms of use for every asset. We will provide a complete asset and license table with the public release of the dataset and code.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper introduces a new Long-CoT Spatial Reasoning dataset, but the full asset is not released with this paper. The paper provides documentation of data construction, filtering, and evaluation, and the asset will be released publicly.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [No]

Justification: Appendix K describes the human rating audit of model-generated reasoning traces: 200 paired examples per audit set, 1–5 ratings of faithfulness, logic, completeness, fluency, and non-redundancy, and the definition of contradiction. The paper does not include the full text of the rating instructions or details about compensation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [No]

Justification: The only human involvement is the rating audit in Appendix K, in which raters score model-generated reasoning traces. The paper does not describe risks to raters or report IRB (or equivalent) review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLM/VLM systems are used as part of the dataset construction pipeline: Qwen3-VL-Plus generates answer-constrained long-form rationales, and Qwen3-Max refines them for coherence and consistency while preserving verified final answers. This usage is described in Sec. 4.1; generated rationales are used only as training supervision, while evaluation uses verified source annotations.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.