Latent Implicit Visual Reasoning

Kelvin Li[1*] Chuyi Shang[1*] Leonid Karlinsky[2] Rogerio Feris[3] Trevor Darrell[1] Roei Herzig[1,3]
[1]University of California, Berkeley [2] Xero [3]MIT-IBM Watson AI Lab

Abstract

While Large Multimodal Models (LMMs) have made significant progress, they remain largely text-centric, relying on language as their core reasoning modality. As a result, they are limited in their ability to handle reasoning tasks that are predominantly visual. Recent approaches have sought to address this by supervising intermediate visual steps with helper images, depth maps, or image crops. However, these strategies impose restrictive priors on what "useful" visual abstractions look like, add heavy annotation costs, and struggle to generalize across tasks. To address this critical limitation, we propose a task-agnostic mechanism that trains LMMs to discover and use ***visual reasoning tokens*** *without explicit supervision. These tokens attend globally and re-encode the image in a task-adaptive way, enabling the model to extract relevant visual information without hand-crafted supervision. Our approach outperforms direct fine-tuning and achieves state-of-the-art results on a diverse range of vision-centric tasks – including those where intermediate abstractions are hard to specify – while also generalizing to multi-task instruction tuning.*

1. Introduction

In recent years, Large Multimodal Models (LMMs) have demonstrated great progress in visual understanding. However, they still struggle with vision-centric tasks that require heavy visual processing. This limitation stems from several factors. Firstly, most modern LMMs follow a LLaVA [18] style architecture, where visual inputs are projected into a language model that is trained to output text only. This introduces significant language bias, forcing the LMM to reason about visual information through text alone. Text-based representations may inherently lack the expressivity required to form the sophisticated visual abstractions needed for complex reasoning tasks. For example, humans can visualize objects from different angles, solve jigsaw puzzles, or identify visual patterns through mental imagery alone, without relying on language. Attempting to solve such tasks using language alone, however, may be extremely difficult. Moreover, recent LMM progress has largely focused on tasks requiring limited visual reasoning, such as document understanding or mathematical problem solving, where most of the reasoning occurs in the text space after initial visual information extraction.

Given these limitations, many works have attempted to train LMMs to be more "visual" through explicit supervision. However, this approach faces several challenges. First, it requires large amounts of task-specific supervised data, which both incurs substantial annotation costs and embeds human biases about what constitutes "useful" visual reasoning. For example, models are often trained to predict intermediate visual steps, such as bounding boxes and image crops, even though the intermediate steps that are intuitive for human reasoning may not be the most effective for the model to learn. Second, such supervision is difficult to specify for tasks that require complex or abstract visual structure, and the resulting models often generalize poorly beyond the supervision regimes they were designed for. As a result, this data-dependent approach does not scale well to a diverse range of vision-centric reasoning tasks.

Consider the task in Figure 1, where the model is given a reference image and must select the most visually similar image from a set of choices. Describing the relationship between the sets of images using only text can be challenging and ambiguous. Training the model with explicit supervision is difficult as well since it is not clear what intermediate visual representations would be helpful to provide to the model. Even if we could identify useful intermediate steps, we would need to create large amounts of task-specific data, which is impractical to scale across different tasks.

Our proposed approach, Latent Implicit Visual Reasoning (LIVR), enables models to autonomously discover useful intermediate visual representations without explicit supervision. LIVR augments the LMM with latent tokens that are learned implicitly through a novel visual bottlenecking approach, requiring no task-specific supervision.

To summarize, our main contributions are as follows: (i) We introduce LIVR, a new method for visual reasoning that allows the model to implicitly learn useful visual information through latent tokens, without the need for additional data or explicit supervision. (ii) We show that

*Equal contribution

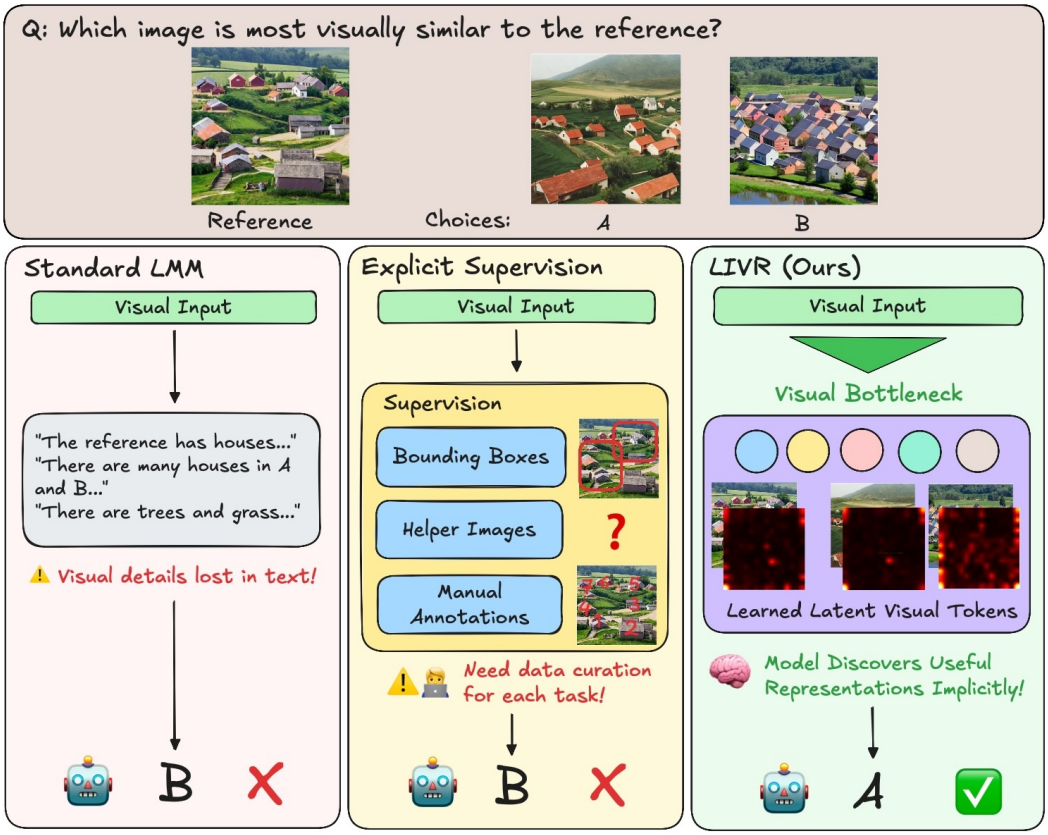

Figure 1. The model is asked to determine which image option is most similar to the reference image. Standard LMMs can only output text, which cannot capture all visual information and may introduce ambiguity. While methods using explicit supervision can train models to output intermediate reasoning steps, these approaches may fail when the reasoning steps themselves are unclear. Our approach allows the model to learn useful representations implicitly. Visualizing the attention maps of the latent tokens shows that the model has learned to recognize underlying visual structures relevant to answering the question that would have been hard for humans to design supervision for.

our approach outperforms direct supervised fine-tuning and achieves state-of-the-art results on multiple single-task fine-tuning setups. (iii) We demonstrate strong generalization capabilities by outperforming supervised fine-tuning on a general, multi-task fine-tuning setup.

2. Related Work

Text-Based Visual Reasoning. Chain-of-thought (CoT) prompting has shown that explicitly generating intermediate text steps can substantially improve LLM performance on complex reasoning tasks [27, 34, 38]. Recent works extend CoT into the multimodal regime by training the model to describe its visual understanding in text before producing an answer [31, 35, 37]. For example, LLaVA-CoT fine-tunes an LMM to generate structured textual rationales that describe the image before concluding with an answer [35]. More recent works like Visual-RFT, Vision-R1, R1-VL and PAPO use RL-based post-training to encourage long, step-by-step textual explanations to help answer questions [11, 19, 33, 39]. In all of these approaches, the entire intermediate reasoning process is represented with text. Thus, it can be difficult for these methods to form rich, spatially structured visual abstractions that go beyond what can be easily verbalized.

Interleaved Multimodal Reasoning Text-only reasoning often struggles on visual tasks, motivating recent work that interleaves visual representations into the reasoning process itself. We group these methods into two main classes: visual token recycling and visual intermediates.

Visual Token Recycling. Visual CoT [26], Argus [20], VGR [32], ViGoRL[24] and Pixel Reasoner [29] predict bounding boxes and reintegrate the selected visual regions into the reasoning chain, usually by cropping and resampling. Other works like UV-COT [40] avoid manual bounding-box annotations by using learned rewards to guide where the model should look. However, these methods can limit expressivity, since the model can only reuse tokens from the original input. Moreover, these methods depend on explicit supervision and hand-designed crops, potentially introducing suboptimal human biases.

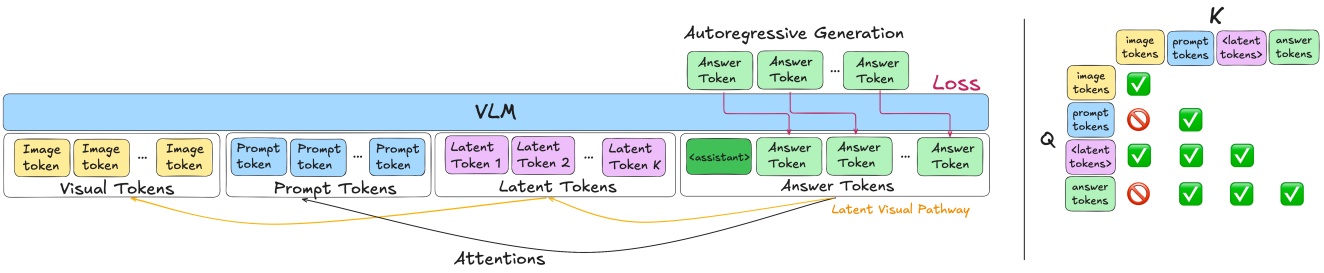

Figure 2. An illustration of our method and bottleneck attention masking. Latent tokens are appended to the prompt and losses are computed on the answer tokens. In our bottleneck attention masking, answers and prompt tokens cannot attend to image tokens.

Visual Intermediates. Another approach generates visual representations of intermediate reasoning steps. Some methods do this multimodally: MVoT [15] and CoT-VLA [41] explicitly render intermediate images or future frames as visual chain-of-thought. Others instead inject these intermediate visual representations into the language backbone of the LMM: Aurora [4] learns discrete perception tokens for targets like depth maps and bounding boxes, while Mirage [36] introduces latent tokens that are trained to reconstruct intermediate embeddings. However, these methods have inherent limitations: these visual intermediates need to be explicitly supervised which incurs large annotation costs, many tasks may lack well-defined visual intermediate targets, and human-designed abstractions may not be optimal for the model to learn. Our approach bypasses these issues by learning implicit visual representations in latent space, without explicit intermediate targets or additional data.

Latent Reasoning. A separate line of work explores allocating additional computation in latent space. Coconut treats hidden states as continuous "thoughts" that are iteratively fed back into the model [9], while Think Before You Speak uses pause tokens to trigger extra forward passes without emitting visible tokens [8]. Together, these works suggest that latent representations provide a more flexible internal representation space for reasoning than natural language, and that extra compute in latent space can be beneficial. Decoupling internal computation from external tokens lets the model refine its internal state solely to optimize task performance, rather than being constrained by what can be explicitly verbalized. Recent approaches have begun to explore latent-space reasoning in LMMs, but their latent variables are still trained with explicit intermediate supervision [14, 36]. In contrast, we study latent reasoning in an LMM without explicit supervision on intermediate solutions: dedicated latent tokens operate on joint visual–text states and are trained end-to-end from task objectives, allowing the model to learn implicit, task-specific visual abstractions.

3. Method

We begin by describing some background on the LMM architectures (Section 3.1), then introduce our method (Section 3.2) and implementation details (Section 3.3). An illustration of our method is shown in Figure 2.

3.1. Preliminaries

Large Multimodal Models. LMMs are generative models that process both visual and textual inputs to perform various tasks. They typically consist of three parts, a visual encoder, a language model decoder, and a projector that projects outputs from the visual encoder into the embedding space of the language model. To be more precise, given a text prompt Q and visual input I, the prompt Q is first encoded by a language encoder l. The image I is encoded using a visual encoder v, then projected into the language model's embedding space via a projector p. Finally, the language model M processes these embeddings to output a textual response R:

$$R = M\left(p(v(I)),\ l(Q)\right)$$

Visual Question Answering. In Visual Question Answering (VQA), the LMM is provided with a set of images and tasked with answer questions about the images. Typically, the model is evaluated through *top-1* accuracy. In the open-ended setting, the model is provided prompt Q and visual inputs I, and is asked to output the best answer a. In the multiple choice setting, the model is additionally provided with a set of possible choices $a_1, a_2, \cdots, a_n$ and tasked with selecting the best a_i. Next, we introduce our method.

3.2. Latent Implicit Visual Reasoning

Current LMMs are trained to autoregressively generate text tokens for visual tasks. Visual information is projected once into the language space at the beginning of the inputs, after which the LMM reasons in the text space. We hypothesize that LMMs' abilities can be improved by providing extra visual compute space, allowing them to implicitly learn how to best utilize visual information. We do this by equipping

the LMM with latent tokens and training the model to use them through a novel visual bottlenecking approach.

Latent Tokens. To provide the LMM more expressivity to reason beyond the discrete text space, we equip it with latent tokens. Specifically, we introduce K new special tokens, $L = \{l_1, l_2, \cdots, l_K\}$, to the model's existing vocabulary, V. The new vocabulary becomes $V \cup L$, with a total size of $|V| + K$. During training, we append these latent tokens to the input. Thus, given an original prompt Q, the new prompt Q' becomes $Q \ + L$. While these tokens are randomly initialized, their corresponding rows in the embedding table remain unfrozen during training. Crucially, the model does not need to learn how to generate these latent tokens. Instead, it only needs to learn how to use them to represent important visual information.

Visual Bottlenecking. In order to train our latent tokens, we introduce a bottlenecking approach where we force visual information to pass through the latent tokens. We do this by modifying the attention mask so that the answer tokens can only attend to the prompt tokens Q and the latent tokens L, but cannot attend to the visual inputs I. To avoid residual visual leakage to the answer tokens, we also prevent the prompt tokens Q from attending to the visual inputs I. In this setup, the model can only "see" visual information through the latent tokens, which serve as the bottleneck.

This bottlenecking may help for a few reasons. Firstly, it forces the latents to carry visual information, providing extra "visual computation" that may be more expressive than pre-trained text tokens. Secondly, the model must focus on these visual latents to answer the question correctly, which may reduce existing language biases for the model.

Multi-Stage Training. We utilize a 2-stage approach to train our model. In Stage 1, we apply the masking described above and train the model using the standard negative log likelihood (NLL) objective:

$$L = -\frac{1}{|x|} \sum_{i=1}^{|x|} \log P(x_i | x_{<i})$$

where we compute the loss only on the answer tokens. By doing so, our objective directly optimizes the latent tokens to capture the *most useful visual information for solving the question*. Moreover, this approach allows the model to implicitly discover optimal ways to use latent tokens without requiring explicit supervision or additional data.

After the latent tokens are trained to contain useful visual information in Stage 1, we revert to a standard attention mask that allows the answer tokens to attend to both the original image tokens and the latent tokens. The loss remains the same and is still computed only on the answer tokens. The goal of this stage is to train the model to jointly use both the original image tokens and the now-enriched latent tokens to answer the question.

3.3. Implementation Details

We fine-tune the language backbone using LoRA (applied to attention and MLP blocks) [10, 25], while keeping the vision encoder and projector frozen. In addition to LoRA parameters, we unfreeze only the embedding rows corresponding to the K new latent tokens. Full optimization and schedule details are provided in the Appendix.

4. Evaluation

We evaluate our method on the tasks described in Section 4.1, and compare it to baselines in Section 4.2. Finally, results and ablations are in Section 4.3 and Section 4.4, and visualizations are in Section 4.5.

4.1. Tasks and Datasets

We evaluate our method on nine perception-heavy tasks adapted from the BLINK benchmark [7]: counting, jigsaw, object localization, visual correspondence, art style classification, semantic correspondence, functional correspondence, relative reflectance, and visual similarity. We choose these tasks because they require a strong degree of visual reasoning and abstraction. However, BLINK and most other challenging visual-centric datasets are designed for evaluation only, and there is a lack of readily available VQA-style training data. As such, we create our own training data sets from popular vision datasets. We note that all data we generate consists only of direct question-answer pairs, without any additional chains-of-thought or visual intermediate steps. All tasks except for counting are framed as BLINK-style multiple-choice VQA using top-1 accuracy as the evaluation metric; counting is evaluated in the standard open-ended setting with exact-match accuracy.

Counting uses the official PixMo-Count splits [5]. We adopt PixMo-Count to evaluate a more challenging open-ended counting setting, where the model must generate the count rather than choose from discrete options. For the remaining tasks, we build training/validation splits from COCO [17] (Jigsaw, Localization), ArtBench-10 [16] (art style), SPair-71k [21] (semantic correspondence), HPatches [3] (visual correspondence), FunKPoint [12] (functional correspondence), MID [22] (relative reflectance), and DreamSim [6] (visual similarity). We test on the official BLINK validation sets for Jigsaw, Object Localization, Art Style, Semantic Correspondence, Relative Reflectance, and Visual Similarity. For Visual Correspondence and Functional Correspondence, we evaluate on held-out HPatches and FunKPoint splits (rather than BLINK) due to the small size of these source datasets. For all tasks, we de-duplicate custom train/validation data against their corresponding test sets. Full construction details and prompt templates are provided in the Appendix.

Table 1. Single-task fine-tuning accuracy.

Method	Counting	Jigsaw	Local.	Vis. Corr.	Art Style	Sem. Corr.	Func. Corr.	Rel. Refl.	Vis. Sim.	Mean
Random Choice	11.11	50.00	50.00	25.00	50.00	25.00	25.00	33.33	50.00	35.49
Qwen2.5-VL-3B-Instruct										
Zero-shot	46.78	49.33	56.56	29.86	55.56	32.37	26.71	45.52	50.37	43.67
Direct SFT	60.04	53.33	75.41	88.00	83.76	41.01	18.49	44.78	89.63	61.61
Ours	**63.64**	**65.33**	**79.51**	**90.43**	**87.18**	**46.76**	**31.51**	**51.49**	**94.82**	**67.85**
Δ vs SFT	(+3.60)	(+12.00)	(+4.10)	(+2.43)	(+3.42)	(+5.75)	(+13.02)	(+6.71)	(+5.19)	(+6.24)
Qwen3-VL-4B-Instruct										
Zero-shot	58.52	84.67	59.02	55.43	77.78	39.57	31.51	47.76	82.22	59.61
Direct SFT	66.86	83.33	79.51	90.86	78.63	61.15	58.90	56.72	91.11	74.12
Ours	**66.67**	**85.33**	**83.61**	**93.29**	**81.20**	**64.75**	**67.81**	**62.69**	**92.59**	**77.55**
Δ vs SFT	(-0.19)	(+2.00)	(+4.10)	(+2.43)	(+2.57)	(+3.60)	(+8.91)	(+5.97)	(+1.48)	(+3.43)
LLaVA-OneVision-1.5-4B-Instruct										
Zero-shot	53.98	56.00	56.56	36.86	56.41	29.50	21.92	35.82	51.11	44.24
Direct SFT	60.42	65.33	68.85	86.86	76.92	46.76	23.29	52.24	92.59	63.70
Ours	**63.64**	**70.67**	**72.95**	**88.71**	**80.34**	**51.08**	**50.69**	**53.73**	**91.85**	**69.30**
Δ vs SFT	(+3.22)	(+5.34)	(+4.10)	(+1.85)	(+3.42)	(+4.32)	(+27.40)	(+1.49)	(-0.74)	(+5.60)

4.2. Baselines and Models

We experiment with three recent open-source LMMs of similar scale: Qwen2.5-VL-3B-Instruct [2], Qwen3-VL-4B-Instruct [30], and LLaVA-OneVision-1.5-4B-Instruct [1]. These models are competitive on a broad range of vision-language benchmarks, providing strong and comparable backbones for our study. For each task and backbone, we consider three settings: (i) **Zero-shot**, the pre-trained instruct model evaluated without any task-specific training; (ii) **Direct SFT**, standard supervised fine-tuning on our task training set; and (iii) **LIVR**, our proposed training method, run with the same task data and training setup as Direct SFT. Direct SFT serves as our main baseline, as it uses identical task supervision but no intermediate supervision, enabling a clean comparison to LIVR. We also compare against text-based CoT methods, visual token recycling, latent reasoning approaches, and RL based methods. Specifically, we compare against Mirage [36] and LVR [14], latent reasoning approaches that explicitly train latents to represent helper images and image crops. We also compare against ViGoRL [24], a text-based CoT approach that grounds reasoning with explicit coordinates, in addition to other RL-based methods like PixelReasoner [29], Vision-R1 [11], and PAPO [33].

4.3. Experiments

Single-Task Fine-Tuning. For single-task experiments, we use 1k training examples per task. Direct supervised fine-tuning runs for 10 epochs. LIVR uses a two-stage schedule: 4 epochs of Stage 1 (visual bottlenecking) followed by 6 epochs of Stage 2 (standard masking) with $K = 16$ latent tokens. These hyperparameters were determined through ablation studies on 3 tasks (Section 4.4.3) and kept fixed across all tasks, though we hypothesize that task-specific tuning could further improve results. For all runs, we select checkpoints by highest validation accuracy.

Table 1 reports single-task accuracy across the nine visual-centric tasks for all three backbones. With Qwen2.5-VL, our method achieves significantly better results across all tasks, outperforming Direct SFT by an average of 6.24%. The improvements are particularly pronounced on challenging tasks that require complex visual abstractions: gains of 12% on Jigsaw and 13.02% on Functional Correspondence demonstrate that our method effectively enhances the LMM's ability to form useful visual abstractions.We also observe gains on tasks such as Art Style, Visual Similarity, and Relative Reflectance, where explicit visual intermediates are difficult to specify; in these settings, LIVR provides a way to learn useful latent visual abstractions when it is hard—even for humans—to define hand-designed intermediate labels. On Qwen3-VL and LLaVA-OneVision-1.5, we also improve results across datasets by an average of 3.43% and 5.60% respectively, demonstrating the generalizability of our approach across multiple models.

Multi-Task Fine-Tuning. To test if our approach generalizes to multi-task setups, we use Qwen3-VL-4B-Instruct, the strongest backbone, and train on a combined dataset of six tasks: Counting, Localization, Visual Correspondence, Semantic Correspondence, Functional Correspondence, and Relative Reflectance, using 1k examples per task (6k total). We omit Jigsaw, Art Style, and Visual Similarity, as single-task baseline accuracies for Qwen3-VL-4B-Instruct on these tasks are already high, making relative improvements harder to interpret. Direct SFT is trained for 5

Table 2. Multi-task fine-tuning accuracy on **Qwen3-VL-4B-Instruct**.

Method	Counting	Local.	Vis. Corr.	Sem. Corr.	Func. Corr.	Rel. Refl.	Mean
Zero-shot	58.52	59.02	55.43	39.57	31.51	47.76	48.64
Direct SFT	66.10	77.87	91.29	62.59	63.01	56.72	69.60
Ours	**67.80**	**81.97**	**92.00**	**67.63**	**64.38**	**60.45**	**72.37**
Δ vs SFT	(+1.70)	(+4.10)	(+0.71)	(+5.04)	(+1.37)	(+3.73)	(+2.77)

Method	**SAT Val**	**BLINK-3**	**RoboSpatial**
Qwen2.5VL-3B	46.1	44.4	54.4
+ SFT direct	58.3	46.4	62.3
+ Vanilla GRPO	50.0	46.5	**69.7**
Text-CoT (SFT+GRPO)	58.7	45.4	–
ViGoRL-3B	62.9	48.5	67.1
LIVR-3B (Ours)	**85.6**	**59.5**	66.7

Table 3. Comparison across spatial reasoning benchmarks. All rows except LIVR-3B (Ours) are reported from ViGoRL.

Method	**MMVP**	**V***	**BLINK-5**
Qwen2.5-VL-7B	66.7	78.5	53.66
PAPO	54.3	36.1	54.81
Vision-R1	46.7	70.2	42.76
PixelReasoner	67.0	80.1	54.52
LVR-7B	71.7	**80.6**	**55.37**
LIVR-7B (Ours)	**75.3**	80.1	54.28

Table 4. Comparison across MMVP, V*, and BLINK benchmarks. All rows except LIVR-7B (Ours) are reported from LVR.

epochs, while LIVR is trained for 2 epochs of Stage 1 and 3 epochs of Stage 2, maintaining the same 2:3 ratio as in single-task experiments and using $K = 16$ latent tokens. We report performance using the final checkpoint.

Table 2 shows results for multi-task training on Qwen3-VL-4B-Instruct across the six perception tasks. LIVR improves over Direct SFT on all tasks, demonstrating that the latent mechanism effective in single-task settings also benefits joint multi-task training. A key advantage of LIVR is its task-agnostic nature: because it trains latent tokens implicitly from the end-task loss without requiring task-specific helper images or intermediate labels, the same method applies directly to multi-task settings. This contrasts with approaches that tie latent tokens to task-specific visual targets (e.g., depth maps, bounding boxes, helper images), which require different supervision per task and are difficult to extend to heterogeneous multi-task setups. This makes our method well-suited as a simple, general-purpose enhancement for perception-heavy multi-task fine-tuning.

Comparison with Other Methods. We compare LIVR with Mirage [36], a latent reasoning approach that trains latents to represent intermediate helper images. We evaluate on the Visual Spatial Planning (VSP) task only, as data for other tasks have not been released. For LIVR, we discard the helper images and set the number of latents to match Mirage at $K = 4$. On Qwen2.5-VL-3B, we reproduce Mirage using their released dataset and helper images; zero-shot accuracy is 6.00, Mirage achieves 46.00, and LIVR reaches 66.00 (+20.00). On Qwen2.5-VL-7B, we use Mirage's reported numbers; LIVR achieves 77.50, outperforming Mirage (76.00), Mirage with text-CoT (58.00), and Mirage with RL (60.00), despite not using any helper images.

In Table 3, we compare LIVR against several baselines: direct SFT, vanilla GRPO [42], text SFT+RL and ViGoRL [24], a method trained with SFT and RL to output textual reasoning and bounding box coordinates as intermediate steps. All methods are initialized from Qwen-2.5-VL-3B and trained on the same SAT-32k dataset [23], taken from ViGoRL. Because SAT is fully synthetic, it serves as a useful test for out-of-distribution generalization to real images like in BLINK [7]. To isolate the gains from latent reasoning, we remove intermediate images and textual CoT from LIVR, while other methods are free to use them. Following ViGoRL, we evaluate on SAT Val, a 3-task subset of BLINK (relative depth, multi-view reasoning, spatial relation), and RoboSpatial [28], which tests spatial understanding in robotics contexts. Despite operating without text-CoT, explicit grounding or RL, LIVR achieves strong results across all benchmarks and demonstrates the ability to generalize to out of distribution tasks.

Table 4 compares our method with LVR (latent visual intermediate + RL), PixelReasoner (visual intermediate + RL), Vision-R1 and PAPO (text CoT + RL). All use Qwen-2.5-VL-7B. We train LIVR on the Visual-CoT [26] training set used by LVR, and removed intermediate images and text CoT. Following LVR, we evaluate on MMVP, V*, and 5 BLINK subsets (Counting, IQ Test, Jigsaw, Rel. Reflectance, Spatial Rel.). LIVR is competitive or better across all tasks despite using much less data, demonstrating its generalizability and effectiveness.

4.4. Ablations and Additional Experiments

4.4.1. Usefulness of Latent Tokens

We next test whether the model truly relies on latent tokens rather than ignoring them. We compare LIVR against a *latents-only* variant that adds $K = 16$ latent tokens but trains only with Stage 2 (no bottlenecking). This control is designed to match LIVR's added capacity while providing no explicit pressure for latent tokens to carry visual information, creating an "unused-latents" baseline. We report re-

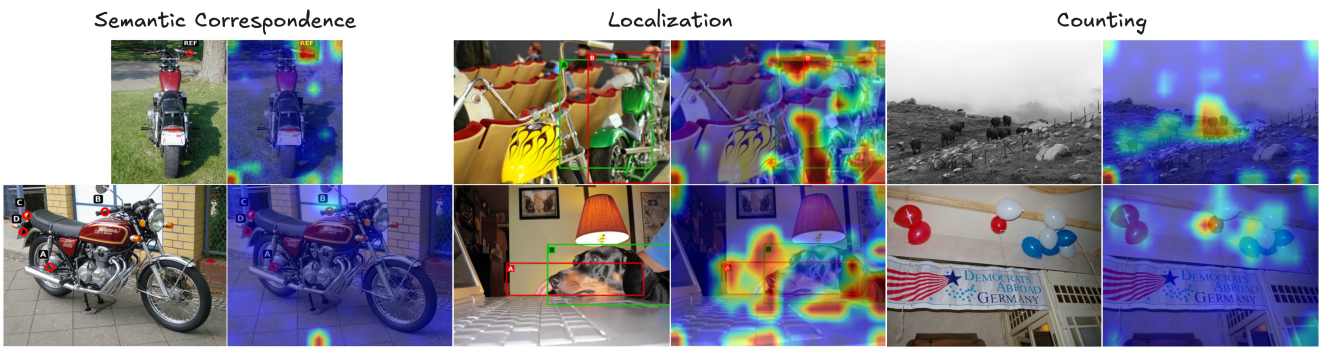

Figure 3. An illustration of latent-to-image attention maps for different tasks. The left columns show the input images, and the right columns show the attention overlays. In the Semantic Correspondence task, the model identifies the option in the second image that aligns with the REF point in the first image. In the Localization task, it selects bounding boxes that best localize the motorcycle and the dog, and in the Counting task, it counts the cows and balloons. We observe that latent-to-image attention concentrates on regions corresponding to the correct answers or the visual evidence needed to resolve each task. Although some attention sinks persist, the dominant patterns align with task-relevant regions, indicating that the latents capture meaningful visual structure without explicit supervision.

Table 5. Design ablations and additional controls.

Method	Local.	Sem. Corr.	Func. Corr.
Baseline	59.02	39.57	31.51
Direct SFT	79.51	61.15	58.90
Ours	**83.61**	**64.75**	**67.81**
Latents only (no mask)	79.51	61.15	58.22
Mask only (no latents)	80.33	61.16	59.59
Input image twice (no mask)	78.69	61.16	58.22
Input image twice (with 2 stage masking)	77.87	61.87	56.85
Prompt tuning	71.31	49.64	36.30

sults on the Localization task using Qwen3-VL-4B-Instruct.

When latent tokens are removed at evaluation, the latents-only model maintains the same accuracy (79.51 $\rightarrow$ 79.51), indicating it has learned to ignore the extra tokens. In contrast, in the standard (unmasked) setting LIVR achieves higher accuracy than the latents-only model (83.61 vs. 79.51) and suffers a clear drop when latents are removed (83.61 $\rightarrow$ 76.23), showing that it depends on them. This is further confirmed by attention patterns: measuring the mean attention from answer tokens to latent tokens (averaged over all heads, layers, and positions), we find much higher scores for LIVR than for the latents-only model (0.076 vs. 0.028). To test whether latents encode useful visual information, we evaluate both models under a bottleneck mask at test time, where the model can only view the image through latent tokens. Under this bottleneck, the latents-only model performs on par with random guessing (43.44), indicating its latents carry no useful visual information, while LIVR retains much higher accuracy (70.49). As a sanity check, if we additionally drop latent tokens under the bottleneck mask, accuracy falls to 43.44, since the image pathway is removed entirely. Together, these results show that the latents in our method are both actually used by the model and encode task-relevant visual information.

4.4.2. Design Ablations for LIVR

We individually test the effectiveness of the two main components of our approach, latent tokens and bottlenecking. We perform these ablations using Qwen3-VL-4B-Instruct as our base model across three challenging tasks: Localization, Semantic Correspondence, and Functional Correspondence. The results are displayed in Table 5.

Bottleneck Ablation. We first revisit the *latents-only* variant described in Section 4.4.1, which adds latent tokens but skips Stage 1 bottlenecking. This isolates the effect of added capacity. However, simply introducing extra tokens without bottleneck training significantly underperforms LIVR, showing that capacity alone is insufficient.

Latent Ablation. Second, we test a *mask-only* variant that applies the Stage 1 bottleneck without adding latent tokens. Here, answer tokens cannot attend directly to vision tokens, but prompt tokens can still see the image. The goal is to force existing prompt tokens to act as a visual bottleneck without adding new capacity. This variant also underperforms LIVR. A plausible explanation is that existing text tokens already carry pre-trained semantics, making them harder to repurpose to form abstract visual representations. In contrast, newly introduced latent tokens are free to adapt and can more easily learn to form rich visual abstractions.

Together, these results suggest that both the dedicated latent tokens and the visual bottleneck are necessary for LIVR's full gains. For completeness, we include three additional controls in the same table: duplicating the input image tokens ("input image twice"), where we concatenate two copies of the same image tokens at both training (with and without 2-stage masking) and inference as a generic control for extra visual compute, and prompt tuning [13], a lightweight adaptation baseline.

Table 6. **Ablations of latent-token design choices on Qwen3-VL-4B-Instruct.** All numbers are accuracies (%).

(a) Masking Strategy

Method	Loc.	Sem.	Func.
Ans→Vis only	77.87	60.43	60.27
Ans+Prompt→Vis (ours)	**83.61**	**64.75**	**67.81**
Ours+Latent→Prompt	81.15	62.59	63.01

(b) Stage-1 / Stage-2 Epochs

(S1, S2)	Loc.	Sem.	Func.
0, 10	79.51	61.15	58.22
2, 8	80.33	58.27	65.75
4, 6	**83.61**	**64.75**	**67.81**
6, 4	81.15	61.87	66.44
8, 2	77.87	59.71	60.27

(c) Number of Latents

# Lat.	Loc.	Sem.	Func.
4	81.15	62.59	66.40
8	80.33	63.31	67.12
16	**83.61**	**64.75**	**67.81**
32	80.33	62.59	63.01

4.4.3. Architectural and Training Choices

We ablate design choices of LIVR on Qwen3-VL-4B-Instruct, again focusing on Localization, Semantic Correspondence, and Functional Correspondence. We vary each design choice independently, keeping all others fixed to our defaults: latents placed after the prompt, our default masking scheme (blocking both answer-to-vision and prompt-to-vision attention), unshared latent embeddings, $K = 16$, and a 4-epoch Stage 1, 6-epoch Stage 2 schedule.

Masking strategy. Table 6(a) compares three masking schemes. Our default approach blocks both answer-to-vision and prompt-to-vision attention, forcing all visual information to flow through latents, and achieves the best performance. Blocking only answer-to-vision attention is insufficient: visual information can still reach answer tokens via the prompt, so latents never become a true bottleneck. Conversely, further blocking latents from attending to the prompt is too restrictive, as latents need to see the question to determine what visual information to encode.

Stage-1 / Stage-2 schedule. For our main experiments, we train the model for 4 epochs in Stage 1 and 6 epochs in Stage 2. We experiment with different allocations of Stage 1 and Stage 2 epochs in Table 6(b), while keeping the total number of epochs at at 10. Using only Stage 2 (0,10) corresponds to the latents-only setting from Section 4.4.1 and underperforms LIVR, again highlighting the importance of bottleneck training. Conversely, an (8,2) split also hurts; we hypothesize that in this case the model does not have enough Stage 2 training to learn how to integrate the latent representations with the original image tokens under the standard mask. A balanced schedule with 4 Stage 1 and 6 Stage 2 epochs provides the best trade-off, giving latents enough time to learn visual information while still allowing ample joint training with standard masking.

Shared vs. unshared latent embeddings. In our method, we use different embeddings for each of our K latent tokens. However, we can also insert the same latent token K times, in a configuration we call "shared embeddings". We find that using unshared embeddings (one learnable embedding per latent) yields higher accuracy compared to shared embeddings across all 3 tasks. Specifically, we have scores of (83.61 vs. 81.15), (64.75 vs. 61.87), and (67.81 vs. 63.70) for the Localization, Semantic Correspondence, and Functional Correspondence tasks, respectively. This is consistent with the idea that giving each latent its own embedding increases the expressivity of the latent set.

Number of latents. For our standard experiments, we set $K = 16$, inserting 16 latent tokens per prompt. We experiment with varying K by using values of $4, 8, 16, 32$, which is displayed in Table 6(c). Accuracy generally improves as K increases from 4 to 16, with $K = 16$ (our default) performing best. We hypothesize that 4 and 8 latents do not provide enough capacity, while 16 strikes a good balance between expressivity and learnability. At $K = 32$, performance drops; one possible explanation is that attention becomes more diffuse over a larger latent set, making it harder for the model to learn to use each latent effectively.

4.5. Visualizations

Latent Attention Visualization. We map the latent-to-image attention maps in Figure 3. Our method allows latent tokens to learn useful features across different tasks without explicit supervision. The latent tokens are able to match the handle of the motorcycle in the Semantic Correspondence task, identify the best bounding boxes of the motorcycle and the dog in the Localization task, and focus on all of the objects it needs to count in the Counting task.

5. Conclusion

We introduce LIVR, a method that enables LMMs to perform richer visual reasoning without requiring additional supervision or data. We do this by introducing latent tokens and training them with a novel visual bottlenecking approach, allowing the model to learn useful visual representations implicitly. Across nine perception-heavy tasks, LIVR consistently outperforms direct SFT in single-task training on three LMMs and improves joint multi-task training on Qwen3-VL. LIVR also is competitive with or outperforms other methods that use text CoT, RL, or explicit visual supervision, despite using less data. Through extensive experiments, we demonstrate that LIVR offers a simple, effective, and task-agnostic way to enhance visual reasoning.

References

[1] Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Chunsheng Wu, Huajie Tan, Chunyuan Li, Jing Yang, Jie Yu, Xiyao Wang, Bin Qin, Yumeng Wang, Zizhen Yan, Ziyong Feng, Ziwei Liu, Bo Li, and Jiankang Deng. Llava-onevision-1.5: Fully open framework for democratized multimodal training, 2025. 5

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. 5

[3] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[4] Mahtab Bigverdi, Zelun Luo, Cheng-Yu Hsieh, Ethan Shen, Dongping Chen, Linda G. Shapiro, and Ranjay Krishna. Perception tokens enhance visual reasoning in multimodal language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3836–3845, 2025. 3

[5] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 91–104, 2025. 4

[6] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Advances in Neural Information Processing Systems*, pages 50742–50768, 2023. 4

[7] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *Computer Vision – ECCV 2024*, pages 148–166, Cham, 2025. Springer Nature Switzerland. 4, 6

[8] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens, 2024. 3

[9] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. 3

[10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 4

[11] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models, 2025. 2, 5

[12] Zihang Lai, Senthil Purushwalkam, and Abhinav Gupta. The functional correspondence problem. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15772–15781, 2021. 4

[13] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. 7

[14] Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Hao Chen, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning, 2025. 3, 5

[15] Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. Imagine while reasoning in space: Multimodal visualization-of-thought, 2025. 3

[16] Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The artbench dataset: Benchmarking generative models with artworks, 2022. 4

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 4

[18] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, pages 34892–34916. Curran Associates, Inc., 2023. 1

[19] Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning, 2025. 2

[20] Yunze Man, De-An Huang, Guilin Liu, Shiwei Sheng, Shilong Liu, Liang-Yan Gui, Jan Kautz, Yu-Xiong Wang, and Zhiding Yu. Argus: Vision-centric reasoning with grounded chain-of-thought. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14268–14280, 2025. 2

[21] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence, 2019. 4

[22] Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A dataset of multi-illumination images in the wild.

In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 4

[23] Arijit Ray, Jiafei Duan, Ellis Brown, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Aniruddha Kembhavi, Bryan A. Plummer, Ranjay Krishna, Kuo-Hao Zeng, and Kate Saenko. Sat: Dynamic spatial aptitude training for multimodal language models, 2025. 6

[24] Gabriel Sarch, Snigdha Saha, Naitik Khandelwal, Ayush Jain, Michael J Tarr, Aviral Kumar, and Katerina Fragkiadaki. Grounded reinforcement learning for visual reasoning. 2025. 2, 5, 6

[25] John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. https://thinkingmachines.ai/blog/lora/. 4

[26] Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*, pages 8612–8642. Curran Associates, Inc., 2024. 2, 6

[27] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. 2

[28] Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. RoboSpatial: Teaching spatial understanding to 2D and 3D vision-language models for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. To appear. 6

[29] Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2024. 2, 5

[30] Qwen Team. Qwen3 technical report, 2025. 5

[31] Omkar Thawakar, Dinura Dissanayake, Ketan More, Ritesh Thawkar, Ahmed Heakl, Noor Ahsan, Yuhao Li, Mohammed Zumri, Jean Lahoud, Rao Muhammad Anwer, Hisham Cholakkal, Ivan Laptev, Mubarak Shah, Fahad Shahbaz Khan, and Salman Khan. Llamav-o1: Rethinking step-by-step visual reasoning in llms, 2025. 2

[32] Jiacong Wang, Zijian Kang, Haochen Wang, Haiyong Jiang, Jiawen Li, Bohong Wu, Ya Wang, Jiao Ran, Xiao Liang, Chao Feng, and Jun Xiao. Vgr: Visual grounded reasoning, 2025. 2

[33] Zhenhailong Wang, Xuehang Guo, Sofia Stoica, Haiyang Xu, Hongru Wang, Hyeonjeong Ha, Xiusi Chen, Yangyi Chen, Ming Yan, Fei Huang, et al. Perception-aware policy optimization for multimodal reasoning. *arXiv preprint arXiv:2507.06448*, 2025. 2, 5

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, pages 24824–24837. Curran Associates, Inc., 2022. 2

[35] Guowei Xu, Peng Jin, Ziang Wu, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. Llava-cot: Let vision language models reason step-by-step. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2087–2098, 2025. 2

[36] Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens, 2025. 3, 5, 6

[37] Huanjin Yao, Jiaxing Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and Dacheng Tao. Mulberry: Empowering mllm with o1-like reasoning and reflection via collective monte carlo tree search, 2024. 2

[38] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, pages 11809–11822. Curran Associates, Inc., 2023. 2

[39] Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization, 2025. 2

[40] Kesen Zhao, Beier Zhu, Qianru Sun, and Hanwang Zhang. Unsupervised visual chain-of-thought reasoning via preference optimization, 2025. 2

[41] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Tsung-Yi Lin, Gordon Wetzstein, Ming-Yu Liu, and Donglai Xiang. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1702–1713, 2025. 3

[42] Qihao Zhu Runxin Xu Junxiao Song Mingchuan Zhang Y.K. Li Y. Wu Daya Guo Zhihong Shao, Peiyi Wang. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. 6