

Thinking with Visual Primitives

Ruijie Lu^{1,2,*}, Yiyang Ma^{1,*}, Xiaokang Chen^{1*,‡}, Lingxiao Luo^{1,3,*}, Zhiyu Wu^{1,*}, Zizheng Pan^{1,*},
Xingchao Liu^{1,*}, Yutong Lin¹, Hao Li¹, Wen Liu¹, Zhewen Hao¹, Xi Gao¹, Shaoheng Nie¹,
Yixuan Wei¹, Zhenda Xie¹, Ting Chen³, Gang Zeng²

¹DeepSeek-AI ²Peking University ³Tsinghua University
*: Core contributors ‡: Project lead

Abstract

Despite the remarkable progress in Multimodal Large Language Models (MLLMs), the prevailing Chain-of-Thought (CoT) paradigms remain predominantly confined to the linguistic space. While recent advancements have focused on bridging the **Perception Gap** through high-resolution cropping (e.g., Thinking with Images), they overlook a more fundamental bottleneck: the **Reference Gap**. The inherent ambiguity of natural language often fails to provide precise, unambiguous pointers to complex spatial layouts, leading to logical collapse in tasks requiring rigorous grounding. In this work, we introduce **Thinking with Visual Primitives**, a novel reasoning framework that elevates spatial markers—such as points and bounding boxes—to “minimal units of thought”. By interleaving these visual primitives directly into the thinking process, our model can “point” while it “reasons”, effectively grounding its cognitive trajectory in the physical coordinates of the image. Notably, our framework is built on a highly optimized architecture with extreme visual token efficiency. Despite its compact model scale and significantly lower image-token budget, our model achieves frontier-competitive performance on a focused suite of challenging visual QA tasks, matching or exceeding models such as GPT-5.4, Claude-Sonnet-4.6, and Gemini-3-Flash. This demonstrates a path toward more efficient and scalable System-2-like multimodal intelligence.

1. Introduction

The convergence of Large Language Models (LLMs) and computer vision has ushered in an era of Multimodal Large Language Models (MLLMs) capable of sophisticated scene understanding. However, as we push these models toward complex reasoning, often conceptualized as Daniel Kahneman’s “System 2” thinking [23], a fundamental limitation of the current paradigm emerges. While the internal reasoning of these models, typically manifested as Chain-of-Thought (CoT), has become increasingly robust in the linguistic domain, it remains largely disjointed from the visual domain.

Recent efforts to enhance multimodal reasoning, such as the visual-scaling strategies seen in frontier models [8, 12, 21, 24, 26, 34], have primarily addressed the Perception Gap. By employing high-resolution cropping and dynamic patching, these models ensure they “see” the fine-grained details of an image. Yet, “seeing” is not “reasoning”. Even with perfect perception, MLLMs frequently suffer from logical collapse in tasks involving complex spatial layouts or dense object interactions. We identify this failure as the Reference Gap: the inherent inability of

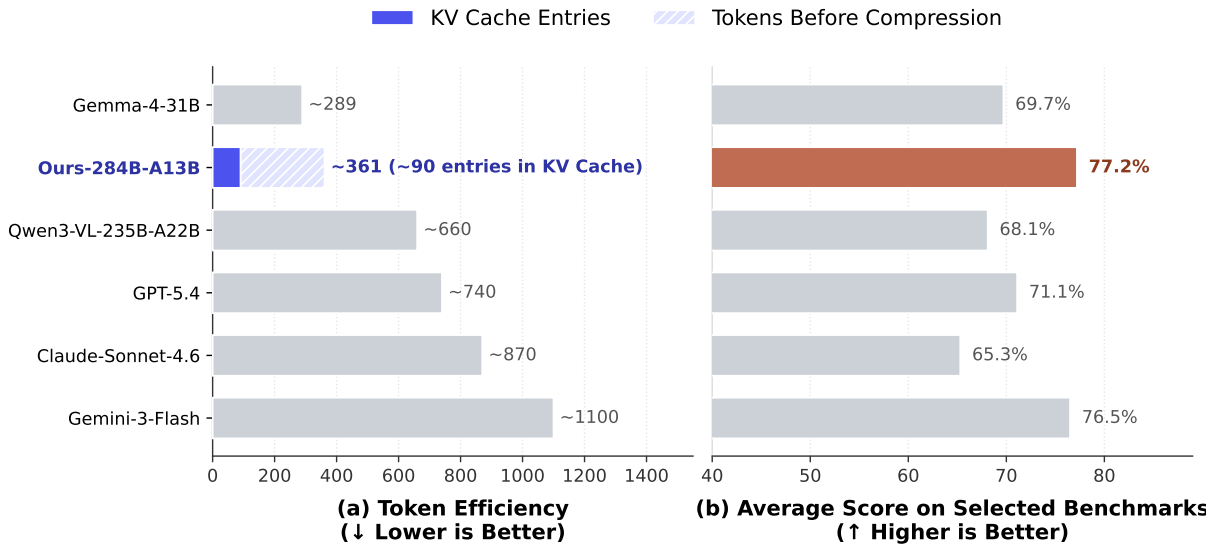


Figure 1 | (a) Token consumption across various models for an 800×800 resolution image. (b) Average performance across 7 benchmarks—including counting and spatial reasoning—with in-house benchmarks excluded. We note that the reported scores cover only a subset of evaluation dimensions that are directly relevant to the research focus of this paper, and are therefore not indicative of the models’ overall capabilities. For an 800×800 input, our model retains only approximately 90 entries in the KV cache, delivering competitive performance through a highly efficient compression strategy. Best viewed on screen.

natural language to serve as a precise, unambiguous pointer within a continuous visual space. In scenarios like dense counting or multi-step spatial deduction, the model’s linguistic “thoughts” lose track of the visual entities they intend to reference, leading to cascading hallucinations. While recent works [6, 27, 32] have explored integrating bounding boxes into the chain-of-thought process, they primarily treat grounding as a post-hoc verification mechanism to enhance perception-heavy tasks. These approaches are often confined to high-resolution benchmarks where the challenge is “seeing” rather than “reasoning”, and their reliance on labor-intensive supervision further limits scalability. More importantly, they fail to address the Reference Gap in complex structural reasoning—such as topological navigation—where visual markers must function as the intrinsic medium of thought rather than merely verifiable evidence.

In this work, we propose a paradigm shift: **Thinking with Visual Primitives**. We move beyond treating visual grounding as a secondary task or a final output. We elevate spatial markers—points and bounding boxes—to “minimal units of thought” that are interleaved directly into the model’s reasoning trajectory. This mechanism draws inspiration from human cognitive processes. When navigating a complex maze or counting a dense collection of objects, humans naturally employ deictic pointers—such as finger gestures—to reduce cognitive load and maintain logical consistency. By interleaving visual primitives into the thinking process, our model mimics this “point-to-reason” synergy, effectively anchoring abstract linguistic thoughts onto concrete spatial coordinates.

Furthermore, our framework is built upon an architecturally efficient foundation [3] designed for high-throughput, long-context multimodal interactions. Unlike traditional approaches that rely on massive visual token sequences to compensate for visual deficiencies, our model leverages Compressed Sparse Attention [3] that compress the Key-Value (KV) cache of every m

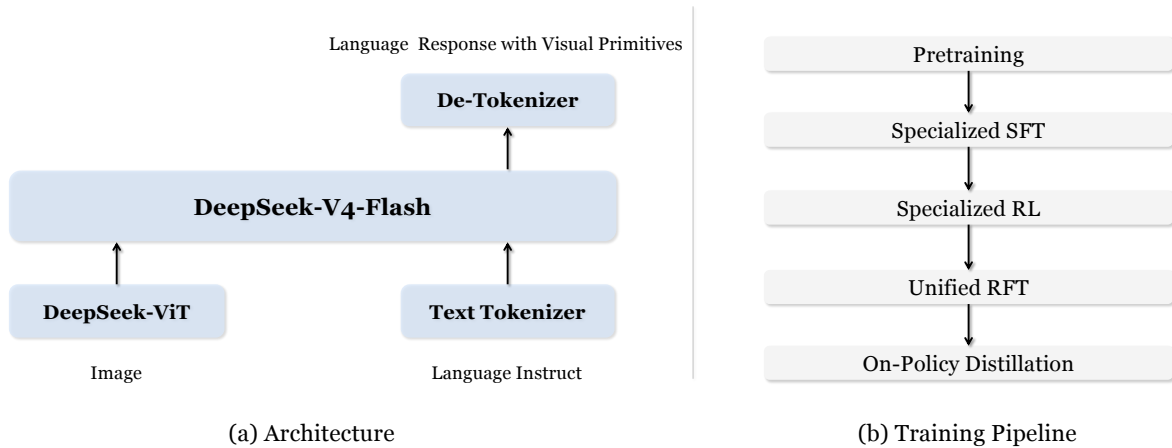


Figure 2 | **Model architecture and training pipeline.** Developed upon the DeepSeek-V4-Flash [3], our model acquires foundational visual primitive generation capabilities during the pretraining phase. This is followed by a post-training stage employing an expert-wise specialization and consolidation paradigm. Best viewed on screen.

visual token into one entry. This design allows the model to operate with only a fraction of the visual tokens used by other frontier systems, while maintaining comparable cognitive depth.

Through extensive benchmarking, we demonstrate that Thinking with Visual Primitives delivers a significant leap in reasoning accuracy. Our model achieves competitive performance, standing on par with or surpassing the latest iterations of GPT, Claude, and Gemini across a wide spectrum of challenging spatial reasoning and visual QA tasks (seeing Fig. 1). Our findings suggest that the future of multimodal intelligence lies not just in seeing more pixels, but in developing more precise and less ambiguous referential mechanisms that bridges the gap between language and the visual world.

2. Method

2.1. Overview

This section first introduces the model architecture. Next, we elaborate on the training pipeline, as illustrated in Fig. 2, and describe the corresponding data used across the pretraining and post-training phases.

2.2. Architecture

Our model adopts a standard architecture similar to LLaVA [18, 19]. Specifically, input images are processed by a Vision Transformer (ViT) to extract visual features, which are then concatenated with language instructions to form an interleaved sequence of vision-language tokens. This sequence is subsequently fed into the Large Language Model (LLM) to generate responses. The language backbone is instantiated with DeepSeek-V4-Flash [3], a Mixture-of-Experts (MoE) model comprising 284B total parameters and 13B active parameters during inference.

For visual encoding, we employ DeepSeek-ViT, an in-house ViT trained from scratch that supports arbitrary-resolution inputs. It first partitions the input image using a 14×14 patch size to generate patch tokens. Subsequently, at the ViT output, we apply a 3×3 spatial token

compression (compressing every 9 adjacent patch tokens into a single token along the channel dimension). Furthermore, leveraging the Compressed Sparse Attention (CSA) mechanism integrated within the base LLM, the visual tokens stored in the Key-Value (KV) cache are further compressed by a factor of 4.

To illustrate this pipeline, consider an input image of 756×756 resolution comprising 571,536 pixels. The patch embedding layer processes this into 2,916 image patch tokens for the ViT. Following the 3×3 compression, only 324^1 visual tokens are fed into the LLM during the prefilling stage. Ultimately, the CSA mechanism reduces this to a mere 81 visual KV entries in the KV cache. Throughout this entire process, from raw pixels to the final KV cache entries, the system achieves an overall compression ratio of $7,056\times$.

2.3. Pretraining

2.3.1. Definition of Visual Primitives

During the pretraining phase, our objective is to equip the model with the fundamental capability to output “visual primitives”. We identify two standard output formats in computer vision as primitives: bounding boxes and points. Both representations fulfill the crucial role of spatial referencing. However, they exhibit distinct functional advantages: bounding boxes are adept at capturing the exact location and scale of specific objects, while points are more appropriate for abstract visual referencing, such as tracking motion trajectories or solving topological reasoning problems.

2.3.2. Motivation for Large-Scale Data Curation

While existing public datasets, such as COCO [17] and Pixmo-Points [4], provide relatively accurate box or point annotations, they suffer from insufficient scale and a notable lack of diversity. To ensure the generalizability of our “Thinking with Visual Primitives” paradigm, it is imperative to curate large-scale web data with rich semantics and high diversity. We prioritize the extensive scaling of bounding box data for the following reasons:

- **Determinism of Annotations:** A bounding box tightly encloses an object, making its annotation relatively deterministic. Conversely, point annotations are highly ambiguous; any coordinate within the object’s boundaries can serve as a valid reference, leading to the absence of a strict ground truth. In extreme scenarios involving occlusion, a point intended for a background object might fall onto a foreground occluder, resulting in significant ambiguity.
- **Task Generalizability:** A model trained to output bounding boxes can effortlessly generalize to point-based formats. Since a bounding box can be defined by two points (the top-left and bottom-right coordinates), it inherently encompasses the point representation.
- **Information Richness:** Bounding boxes support a broader range of downstream tasks compared to points. While a point merely provides spatial localization, a bounding box encapsulates detailed geometric information (e.g., width and height). This additional context enables the model to perform more complex reasoning within the “Thinking with Visual Primitives” framework.

¹To balance performance and computational cost, we restrict the number of visual tokens output by the ViT to a range of 81 to 384. Images with resolutions falling outside this range are resized while preserving their original aspect ratio.

2.3.3. Large-Scale Web Data Construction

Raw Data Acquisition. We acquire a massive volume of internet data related to box grounding by conducting large-scale web scraping across multiple websites. Taking Huggingface as an example, we utilize its official API to filter task data tagged with “Object Detection” or “Grounding”. We perform an initial screening based on popularity metrics (e.g., rankings by likes and downloads) and rigorously exclude all validation and test splits to prevent potential data contamination (i.e., data leakage) during model evaluation. Furthermore, we employ an LLM-based agent to parse the README.md files of these repositories, automatically converting the diverse dataset structures into our predefined, unified storage format. Following extensive crawling and deduplication across these website, we ultimately curate 97,984 box-grounding-related data sources. Manual inspection of the sampled data reveals a highly diverse range of object categories, spanning from common targets (e.g., people, faces) to domain-specific entities (e.g., lesion regions in CT scans or specific anime characters). However, these raw box annotations still suffer from various issues, such as semantic ambiguity and geometric inaccuracy, necessitating further rigorous filtering. We design a two-step filtering pipeline, as illustrated below.

Step I: Semantic-based Review. Given that the directly crawled datasets are replete with noisy labels unsuitable for vision-language alignment training, we introduce an automated, MLLM-driven semantic review mechanism. While traditional data filtering primarily focuses on the geometric accuracy of bounding boxes, this stage aims to ensure the validity of the semantic label. Specifically, this review process focuses on eliminating three categories of fatal semantic defects:

- **Meaningless Machine Codes and Gibberish:** Many raw datasets retain internal development codes (e.g., pure numeric classes like “0” or “1”). Since these labels lack human-readable natural language semantics, forcing the model to learn such mappings would severely degrade its language generation capabilities. Thus, they are directly discarded.
- **Ungeneralizable Private Entities:** Certain datasets employ specific personal pronouns (e.g., “MyRoommate”) or private identifiers (e.g., “ID_Card_1”). Because an MLLM cannot generalize the visual features of a non-public figure from isolated samples (i.e., the visual features of “someone” cannot generalize to a universal concept), such data are strictly filtered out. Conversely, widely recognized celebrities or public figures are retained.
- **Ambiguous Abbreviations and Subjective Evaluations:** Labels common in specific domains (e.g., industrial inspection), such as “OK” or “NG” (Not Good), often lack concrete visual descriptiveness. For instance, a bare “OK” label introduces extreme semantic ambiguity, as an “intact apple” and an “intact circuit board” share absolutely no visual correlation.

For each dataset, we sample three images and prompt the model to calculate a quality score (ranging from 0 to 10) based on the aforementioned criteria. The model then outputs a definitive “KEEP” or “DISCARD” decision, accompanied by a clear justification. This review stage retains 43,141 out of the initial 97,984 data sources, which are subsequently advanced to the next filtering phase.

Step II: Visual-Geometric Quality Review. We further evaluate the geometric quality and annotation completeness of the bounding boxes to ensure the model learns precise region-text alignments. This process specifically targets three types of structural annotation defects:

- **Severe Missing Annotations (Low Recall):** This refers to scenarios where multiple in-

stances corresponding to the given label exist in the image, but only a few are annotated. If a large-scale missing annotation issue (e.g., a miss rate >50%) is detected during sampling, the dataset is immediately discarded.

- **Severe Truncation and Offset:** This occurs when bounding boxes fail to reasonably enclose the target objects. In practice, we adopt a differential tolerance strategy: slightly loose boxes (incorporating minor background noise) are deemed acceptable; however, severe truncations that slice through critical visual features of the object (e.g., cutting off a head or wheels) are strictly unacceptable.
- **Mega Boxes Issue:** If a bounding box meaninglessly covers over 90% of the image area, it is typically indicative of image classification data being forcibly converted into detection data. If this occurs only occasionally within a sampled batch, it is treated as acceptable noise. However, if such global boxes consistently appear across all three sampled images, the dataset is considered to lack meaningful localization information and is discarded.

This review stage further retained 31,701 out of the 43,141 remaining data sources. To achieve dataset balance, we design a category-based sampling strategy. For each category within every dataset, we randomly sample N images associated with that class (if the total number of available images for a category is less than N , all are retained). Since a single image may simultaneously belong to multiple categories, we perform global deduplication on the aggregated set after the per-category selection. In practice, we set $N = 1,000$, ultimately yielding over 40 million high-quality samples.

2.3.4. Unified Pretraining

For general multimodal data, we predominantly utilize large-scale web-crawled data rather than synthetic data generated via model distillation (e.g., synthetic image caption). The raw data undergoes careful curation, and we refrain from utilizing LLMs to rewrite the data content. Regarding the specialized data designed to equip the model with foundational capabilities to output visual primitives, in addition to the aforementioned web crawling and filtering, we also incorporate several high-quality public datasets, such as [4, 15, 17, 25, 29, 33]. We establish a unified formatting standard for both box grounding and point data. For box grounding tasks, we devise several prompt templates, such as “*Locate TARGET in this image and report its bounding box coordinates.*”, where TARGET serves as a placeholder for the queried object. The corresponding response format is formulated as follows: `<|ref|>TARGET</ref|><|box|>[[x1,y1,x2,y2],[x3,y3,x4,y4]...]</box|>`, where `<|ref|>`, `</ref|>`, `<|box|>`, and `</box|>` are special tokens within the vocabulary. $x1, y1$ and $x2, y2$ denote the top-left and bottom-right coordinates of the bounding box. These coordinates are normalized to discrete integers ranging from 0 to 999. In scenarios with multiple instances, the bounding boxes are ordered from left to right. Similarly, for point tasks, we design prompt templates such as “*Help me find TARGET. Give me the center point for each instance.*” The expected response format is defined as: `<|point|>[[x1,y1],[x2,y2]...]</point|>`, where `<|point|>` and `</point|>` are special tokens, and $x1, y1$ denotes the point coordinates. Notably, in contrast to the box grounding format, the response paradigm for point tasks does not require outputting the object name. This design choice aims to extend point-based representations to more abstract concepts, such as utilizing a sequence of points to denote a trajectory. Ultimately, the whole pretraining phase consumes trillions of multimodal tokens.

2.4. Task Design & Cold-Start Data

Cold-Start Data for Post-Training. While pretraining equips the model with general multimodal priors and basic visual primitive capabilities, post-training (Specialized SFT/RL and the subsequent unified RFT) needs a small but high-precision cold-start dataset to bootstrap instruction following and reward learning under our visual primitive output interface. Concretely, we construct cold-start data with (i) explicit supervision targets derived from annotations (e.g., boxes/points) or programmatically generated, (ii) automatic verifiers (e.g., rule-based checkers) whenever possible to reduce label noise. We selected representative tasks that benefit from visual primitive-based reasoning (via boxes or points), and designed our cold-start data across four key dimensions: counting, spatial reasoning & general visual QA, maze navigation, and path tracing.

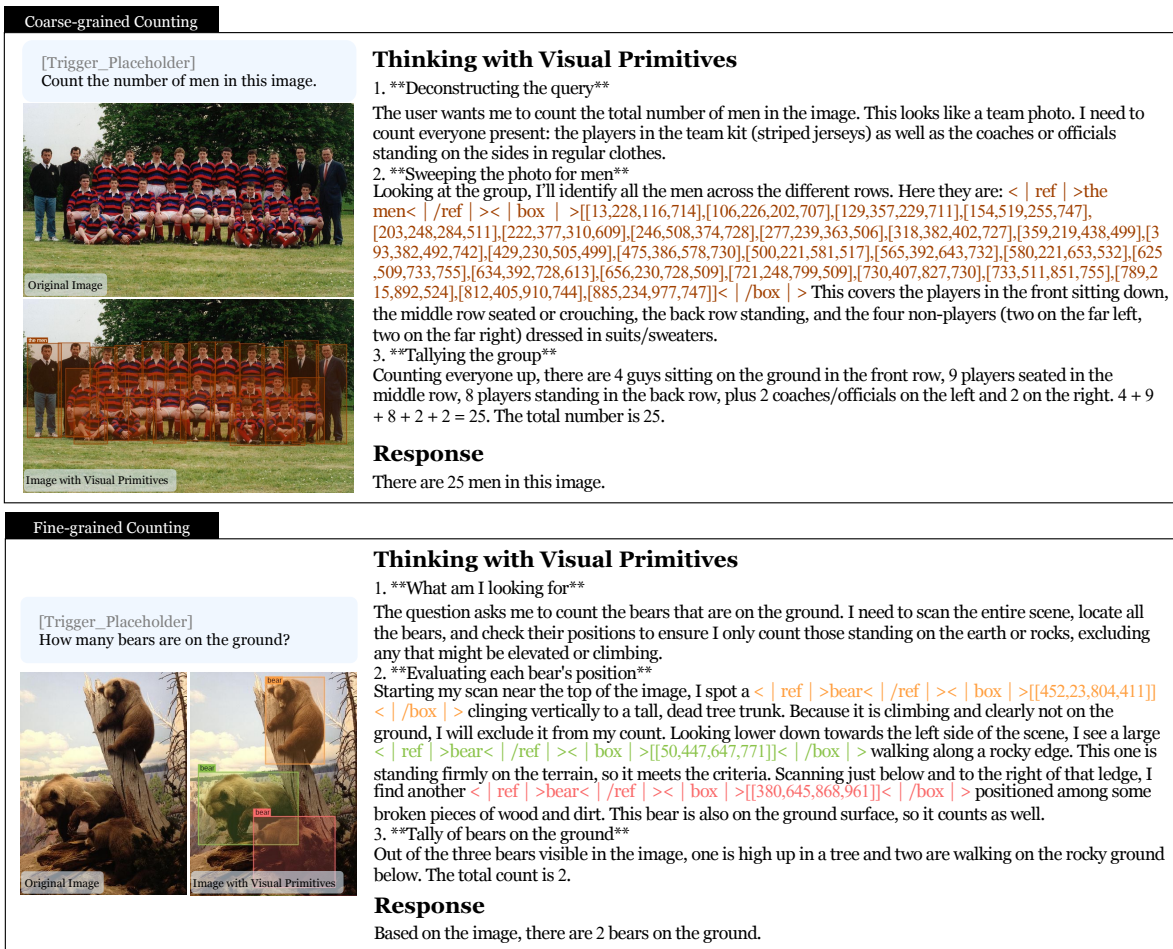
2.4.1. Counting

Multimodal Large Language Models consistently struggle with accurate counting, particularly in dense scenes. Unlike humans, who typically employ a systematic scanning-and-accumulation strategy, language-based models often fail to establish precise object correspondences when the object count is high. We address this fundamental bottleneck by employing bounding boxes as visual primitives to provide explicit referential anchors.

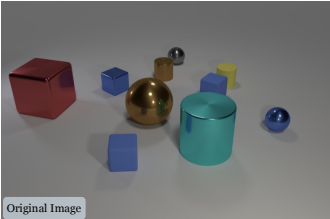
Task Decomposition. We categorize counting tasks into two types: *Coarse-grained Counting* and *Fine-grained Counting*. The former focuses on counting general categories (e.g., “dogs”), while the latter requires distinguishing objects based on specific attributes or spatial constraints (e.g., “white dogs” or “the dog on the left”).

Coarse-Grained Counting. We aggregate data from multiple dense detection datasets, including [2, 9, 14, 22, 28, 29, 35]. To ensure data quality, we implement a filtering process based on three primary criteria: avoiding excessive object density, ensuring bounding boxes are sufficiently large for clear identification, and maintaining a high recall rate for ground-truth box annotations. For the filtered samples, we prompt an MLLM to generate thinking content and concise final response based on the images and box annotations. The thinking content generation follows a structured three-step protocol: (1) *Intent Analysis*, where the model identifies the target category; (2) *Batch Grounding*, where the model utilizes visual primitives to locate all candidate objects simultaneously (we found that batch grounding is more efficient for coarse-grained tasks as it leverages the model’s inherent localization strengths while preventing repetitive enumeration); and (3) *Statistical Summation*, based on the visual primitives. To eliminate noise during cold-start training, we implement a strict verification mechanism to ensure that all box visual primitives in the thinking content strictly align with the metadata coordinates, follow the predefined syntax, and match the final numerical count.

Fine-Grained Counting. Due to the scarcity of publicly available datasets specifically for fine-grained counting, we developed a specialized data construction pipeline. (1) *Question Generation*: Leveraging the images and scene-graph metadata from GQA [10], we prompt an MLLM to curate informative fine-grained counting questions. Samples that fail to yield meaningful questions are discarded. For each valid sample, we record the ground-truth object IDs, the IDs of excluded negative candidates, and the underlying rationale for the QA pair construction. (2) *Thinking Content Synthesis*: Using the image, scene graph, and the previously generated question (along with its associated IDs and rationale) as input, we guide the MLLM to synthesize a reasoning chain integrated with visual primitives. While the overall thinking structure mirrors that of coarse-grained counting, the model is explicitly instructed to perform a



[Trigger_Placeholder]
Is there a purple rubber object that has the same size as the gray metallic object?



Original Image

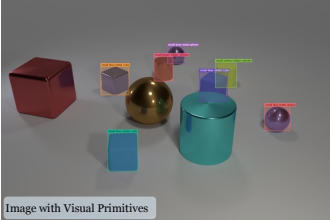


Image with Visual Primitives

Thinking with Visual Primitives

1. **Analyzing the request**
The user is asking to verify if there is a "purple rubber object" that shares the same size as the "gray metallic object". This requires a multi-step attribute verification and comparison: first, I must locate the gray metallic object to determine its relative size in the scene. Second, I need to scan the scene for any purple rubber objects that match this specific size.

2. **Locating the gray metallic object**
Scanning the objects in the scene, I spot a gray, highly reflective spherical object in the background towards the upper center-right. This is the `< | ref | >small gray metal sphere< | /ref | >> | box | >[[502,209,555,288]]< | /box | >`. Comparing it to the large cube, large sphere, and large cylinder, it is clearly categorized as a small object.

3. **Searching for a matching purple rubber object**
Since the gray metallic object is small, I need to look for a small purple rubber object. I will systematically check the other small objects in the scene:

- Right next to the gray sphere, there is a `< | ref | >small brown metal cylinder< | /ref | >> | box | >[[462,255,519,367]]< | /box | >`. It is brown and metallic.
- To the left, there is a `< | ref | >small blue metal cube< | /ref | >> | box | >[[301,300,386,429]]< | /box | >`. It is blue and metallic.
- Further right, in the mid-ground, there is a matte `< | ref | >small blue rubber cube< | /ref | >> | box | >[[599,325,684,458]]< | /box | >`. It is rubber, but its color is blue.
- Right beside that blue cube, there is a `< | ref | >small yellow rubber cylinder< | /ref | >> | box | >[[647,281,711,397]]< | /box | >`. It is yellow.
- On the far right edge of the cluster, there is a `< | ref | >small blue metal sphere< | /ref | >> | box | >[[796,481,873,591]]< | /box | >`. It is blue.
- Finally, in the lower left foreground, there is another matte `< | ref | >small blue rubber cube< | /ref | >> | box | >[[321,600,413,773]]< | /box | >`. This is also blue.

4. **Conclusion**
After examining all the small objects to match the size of the small gray metal sphere, I found blue, brown, and yellow objects, but absolutely no purple ones. Therefore, no purple rubber object of the same size exists in the scene.

Response
False, there is no purple rubber object that matches the size of the gray metallic object.

Figure 4 | **Illustrative cold-start data for spatial reasoning.** The model performs intent decomposition and utilizes visual primitives to anchor all pertinent entities, facilitating sophisticated multi-hop logical inference. Best viewed on screen with zoom.

we prompt an MLLM to design questions centered on spatial relations and object interactions, along with corresponding thinking content. The generated thinking content follows a structured process, including intent analysis, object grounding, and relational inference. To resolve potential ambiguities in crowded scenes, the model is instructed to select distinctive objects and apply multi-attribute constraints (e.g., combining actions and properties) to uniquely specify the target. However, due to the relatively simple relational structure in GQA, it remains challenging to generate complex, multi-hop reasoning samples at scale. To overcome this limitation and fully unlock the model’s potential, we further incorporate complex synthetic data.

Data Construction in Synthetic Scenes. We leverage the CLEVR [13] toolchain to generate multi-hop reasoning data. This framework supports controllable scene generation with varying object densities, along with question generation and programmatic execution traces that map each reasoning step to object-level references (e.g., specific object IDs). To supervise the generation of visual primitives, we project 3D object coordinates onto 2D bounding boxes based on the official toolchain. Given the rendered images, scene graphs, questions, answers, and execution traces, we prompt the MLLM to synthesize “Thinking with Visual Primitives” chains, which include intent analysis, task decomposition, and multi-hop grounded reasoning. An example could be seen in Fig. 4.

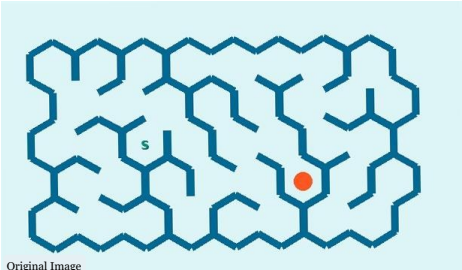
Negative Sample Augmentation. To enhance the model’s reliability, we construct negative training samples where the queried objects or relationships do not exist. In such cases, the model is trained to provide a “faithful refusal” based on the visual evidence rather than generating fabricated responses.

In total, we generated 9,000 cold-start samples for the spatial reasoning and general VQA

Maze Navigation

[Trigger_Placeholder]

Is there a feasible way to get from the lime text label to the tangerine circle?
Please draw the route if any. Display `\boxed{True}` at the end if there is a path, else display `\boxed{False}`.



Original Image

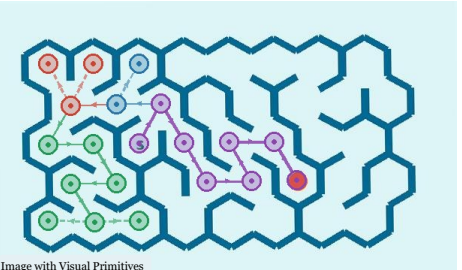


Image with Visual Primitives

Thinking with Visual Primitives

I'll use a trial-and-error strategy to explore this maze. First locate the starting point: `< | point | >[[309,512]]< | /point | >`, and the destination: `< | point | >[[652,644]]< | /point | >`.

****Start Exploring**:**

****Step1**:** Reaching `< | point | >[[357,369]]< | /point | >` after going upper-right, I face 2 directions to choose from (lower-right, left).lower-right direction might also be correct, but I'll explore left direction first.

****Step2**:** Moving left: `< | point | >[[357,369],[260,372]]< | /point | >`...reaching `< | point | >[[260,372]]< | /point | >`, I see 2 forks here. Observing that upper-right direction is a straight dead end at `< | point | >[[310,226]]< | /point | >`, can see wall directly. Still need to explore left direction.

****Step3**:** Walking left: `< | point | >[[260,372],[161,375]]< | /point | >`...arriving at `< | point | >[[161,375]]< | /point | >`, this is another junction with 3 paths. Observing that upper-right direction is a straight dead end at `< | point | >[[211,226]]< | /point | >`, can see wall directly; upper-left direction to `< | point | >[[113,226]]< | /point | >` is obviously a dead end. Still need to explore lower-left direction.

****Step4**:** Moving lower-left: `< | point | >[[161,375],[113,518],[211,515],[261,656],[162,658],[213,796]]< | /point | >`...reaching `< | point | >[[213,796]]< | /point | >`, I see 2 forks here. Observing that right direction is a straight dead end at `< | point | >[[310,791]]< | /point | >`, can see wall directly; left direction is a straight dead end at `< | point | >[[113,791]]< | /point | >`, can see wall directly. All directions are obviously dead ends, need to backtrack.

****Step5**:** This path is blocked, backtracking to the previous junction `< | point | >[[260,372]]< | /point | >`, trying another direction.

****Step6**:** All paths in this branch are explored, all dead ends. Going back to main junction `< | point | >[[357,369]]< | /point | >`, taking lower-right direction.

****Step7**:** Reaching `< | point | >[[357,369]]< | /point | >` after going upper-right, I face 2 directions to choose from (lower-right, left).left direction was already explored, it's a dead end.

****Step8**:** Heading towards lower-right: `< | point | >[[357,369],[408,510],[457,650]]< | /point | >`

****Step9**:** The path is clear, going directly to the exit: `< | point | >[[457,650],[554,648],[506,507],[603,507],[652,645]]< | /point | >`

****Step10**:** Up ahead `< | point | >[[652,645]]< | /point | >` is the destination! Through systematic exploration, I find the correct path.

****Final Path**:** After exploration, the correct route is:
`< | point | >[[310,513],[357,369],[408,510],[457,650],[554,648],[506,507],[603,507],[652,645]]< | /point | >`

Successfully reaching the destination: `< | point | >[[652,644]]< | /point | >`!

Response

The maze is solvable. The verified path is:
`< | point | >[[310,513],[357,369],[408,510],[457,650],[554,648],[506,507],[603,507],[652,645]]< | /point | >`
`[\boxed{True}]`

Figure 5 | **Example of cold-start data for the maze navigation task.** The model first identifies the start and end points, then explores possible paths in a DFS manner. Best viewed on screen with zoom.

domain.

2.4.3. Maze Navigation

While MLLMs have shown proficiency in solving advanced scientific problems, a robust paradigm for topological reasoning remains elusive. Purely linguistic CoT struggles to accurately describe trajectories of irregular shapes. To address this gap, Thinking with Visual Primitives, which could employ points as cognitive units, is uniquely suited for such challenges. We first introduce a maze navigation task that requires the model to determine the solvability of a maze—a process that demands a fundamental understanding of spatial connectivity and reachability. We construct our cold-start data through synthetic data generation, the details of which are provided below.

Design Methodology. We use Depth-First Search (DFS), Prim, and Kruskal algorithms to produce solvable and non-trivial mazes. All three algorithms generate challenging mazes where

only few paths exist between any two cells, ensuring solutions that cannot be trivially guessed. We design three maze topologies: rectangular grids, circular mazes composed of concentric rings with angular sectors, and hexagonal (honeycomb) lattices. To enhance model robustness, we additionally designed a series of unsolvable mazes. We first generate a solvable maze and obtain the solution paths, then deliberately place a few walls around the middle of that path—avoiding areas too close to the start or end. This breaks the connectivity in a less obvious way, making the maze appear solvable at first glance, but actually requiring a full search to confirm that no valid path exists. We apply diverse visual styles including gradient and extra-thick walls, varied background patterns, multiple marker types, and random small-angle rotations to prevent overfitting to specific visual patterns. Image resolutions are randomized, and aspect ratios are continuously sampled, with grid dimensions adjusted proportionally.

Difficulty Control. The difficulty of maze navigation largely depends on how many visual reasoning steps the model needs to chain together. We control this by changing the grid size. As the grid becomes larger, the model has to parse more cells, track connectivity over longer distances, and deal with more dead ends that require backtracking. Each of these adds to the overall reasoning complexity. Concretely, easy mazes require the model to chain only a handful of local connectivity checks, while nightmare-level mazes demand sustained, long-range composition of hundreds of such primitive operations without losing track of previously explored regions. We enforce minimum resolution thresholds at each difficulty level to ensure that the visual primitives remain perceptible, even in the hardest configurations. This ensures that task difficulty stems from reasoning complexity rather than visual ambiguity.

Thinking Content Synthesis. We design several natural language formats and templates to produce descriptions of the DFS-based exploration process, including forward exploration and backtracking. Each exploration step is grounded to the image via pointing coordinates, explicitly converting visual primitive operations—checking wall connectivity at a cell, advancing to an adjacent cell, or retreating from a dead end—into verbalized reasoning chains. This serves as the cold start supervision for teaching the model to think with visual primitives rather than merely perceive them. The final output indicates whether the maze is solvable and, if so, provides a verified solution path.

In total, we generate 460,000 cold-start samples with various difficulties for the task of Maze Navigation. An example is shown in Fig. 5.

2.4.4. Path Tracing

In addition to the maze navigation task, we further design a path tracing task to enhance the model’s ability to leverage visual primitives for reasoning across diverse scenarios. The task asks the model to follow a specified curve through a tangle of overlapping lines to identify the endpoint it reaches. We instantiate this task as line tracing through procedurally generated images of entangled curves, where each line connects a uniquely labeled start point to an endpoint.

Design Methodology. We generate images which consist of multiple Bézier curves, each connecting a labeled start point to a labeled endpoint. The central challenge lies in intersection disambiguation: wherever two lines cross, the model must invoke a local geometric-continuity primitive to decide which branch continues the target curve. To ensure this primitive is genuinely tested, we carefully prevent any endpoint from overlapping with or being crossed by an unrelated line, discarding and regenerating configurations that violate these constraints. We further include a uniform-style mode in which every line shares the same color and stroke

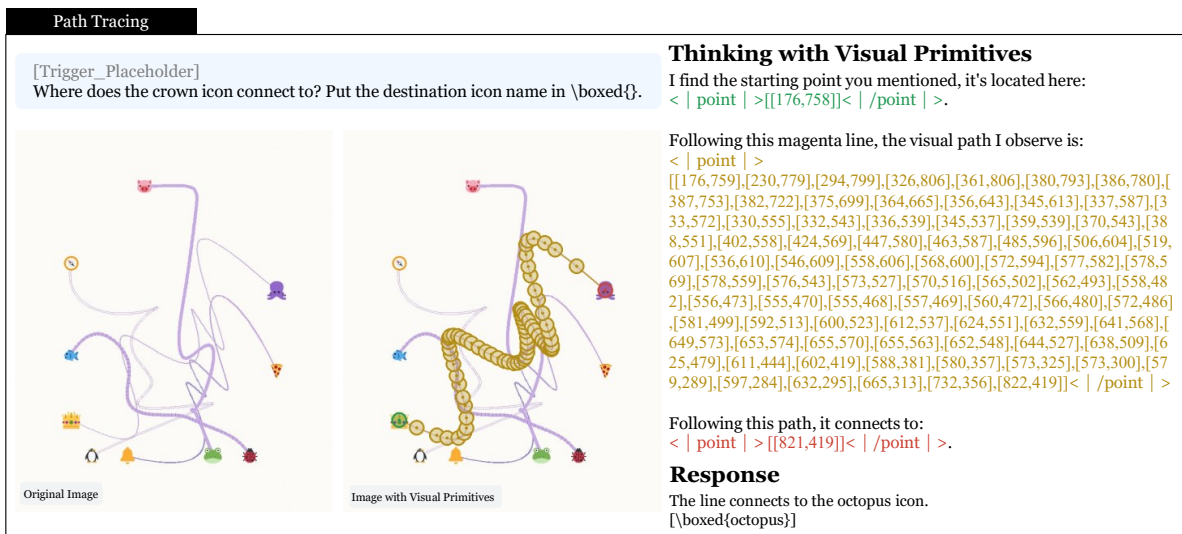


Figure 6 | Example of cold-start data for the path tracing task. The model identifies the start and end points, then traces the line using visual primitives. Best viewed on screen with zoom.

width, stripping away color-based shortcuts and forcing the model to rely solely on curvature continuity at crossings—a direct test of whether the path-tracing primitive has been internalized rather than approximated by color matching. Difficulty scales naturally with the number of lines and their curvature amplitude: easy instances present a few gently curving lines with sparse crossings, while harder instances pack many tightly wound curves into the canvas, multiplying the intersection points at which the figure-ground primitive must be applied. Image resolutions, aspect ratios, and visual styles (color palettes, line styles, endpoint markers, backgrounds) are all randomized to prevent superficial pattern matching.

Thinking Content Synthesis. We explicitly represent the path-tracing process as a sequence of coordinates sampled along the target curve, which reflects how the model attends to and follows the path across the image. The process starts by locating the queried start point, then follows the curve through a series of intermediate waypoints, and finally identifies the endpoint reached. Importantly, the density of these waypoints adapts to the local geometry of the curve. Straightforward segments are represented with fewer points, while highly curved regions or dense intersections are described with finer-grained coordinates, mirroring how a human would slow down and pay closer attention in visually complex regions.

In total, we generate 125,000 cold-start samples across different difficulty levels for the task of Path Tracing. An example is shown in Fig. 6.

2.5. Post-Training Pipeline

To maximize the learning efficiency of the model for both box and point visual primitives, our post-training pipeline adopts a “train specialists—then—merge” strategy, which is detailed below.

2.5.1. Specialized SFT

In the Specialized SFT phase, the overall training data consists of 70% general multimodal and pure-text data, and 30% specialized “thinking with visual primitives” data. We conduct SFT

separately using the two types of cold-start data constructed in Section 2.3.4: box (*thinking with grounding*) and point (*thinking with pointing*). This separation prevents mode conflict when the volume of specialized data is relatively small. After this training phase, we obtain two specialized models, denoted as F_{TWG} and F_{TwP} .

2.5.2. Specialized RL

Subsequently, we apply Reinforcement Learning (RL) independently to both F_{TWG} and F_{TwP} . Following [3], we utilize the Group Relative Policy Optimization (GRPO) algorithm and follow their hyper-parameters. Given that the visual primitives (e.g., boxes and points) within the thinking content of our cold-start data have been rigorously verified, we do not explicitly supervise the visual primitives generated during the model’s thinking process in the RL phase. This design enhances the scalability of the RL training data. Consequently, we only require images, questions, and final answers when collecting RL data, which significantly broadens the scope of accessible data.

During training, we design several Reward Models (RMs) to provide concurrent supervision for each task from three perspectives: format constraints, quality constraints, and accuracy constraints. The first two constraints are shared across different tasks, while the final accuracy constraint requires specific designs tailored to the task type.

Format RM. This RM evaluates the output based on rules to generate a reward score ranging from 0 to 1. Specifically, it verifies whether the representation format of the visual primitives generated by the model is correct. For *thinking with grounding*, this RM additionally checks for redundancy in the model’s output, such as generating duplicate bounding boxes; this effectively mitigates the issue of the SFT model falling into an infinite loop of box generation.

Quality RM. This is an LLM-based Generative Reward Model (GRM). The Quality RM takes the thinking content and the final response generated by the model as inputs, and evaluates them from the following aspects:

- Whether there is redundancy in the model’s response.
- Whether the model’s thinking content is consistent with its final response.
- Whether there are self-contradictions during the “thinking with visual primitives” process.
- Whether the referred objects are meaningful entities when the model outputs visual primitives in the form of boxes.
- Whether the model exhibits “reward hacking” behaviors, such as forcefully fabricating a fake ground truth identical to its own prediction in the response to deceive the reward model.

Ultimately, the model outputs a score from three discrete tiers [0.0, 0.5, 1.0] and provides a rationale for the given score. Below, we will introduce the Accuracy RM specific to each task.

Accuracy RM for Counting. To provide a smooth and informative learning signal, we design a rule-based counting reward model that captures the degree of deviation between the predicted and ground truth, instead of relying on binary exact-match supervision. Specifically, we apply a smooth exponential decay over the relative error, so that near-correct predictions are only lightly penalized, while larger mistakes receive significantly lower scores. The reward R is given as:

$$R(\hat{y}, y) = \alpha \cdot \exp\left(-\beta \cdot \frac{|\hat{y} - y|}{|y| + 1}\right), \quad (1)$$

where \hat{y} and y denote the predicted and ground-truth counts, respectively. The normalization term $|y| + 1$ makes the reward depend on relative error, allowing small deviations to be more tolerable in scenes with larger object counts. The coefficients α and β control the overall reward scale and decay rate, respectively. In practice, we set $\alpha = 0.7$ and $\beta = 3$, which are empirically chosen to provide stable and smooth learning signals.

Accuracy RM for Spatial Reasoning and General VQA. For these tasks, we design an LLM-based GRM. We feed the model’s thinking content, its final response, the user query, and the ground-truth answer into the GRM to independently evaluate and score the thinking process and the response. The final reward is calculated as the average of the two scores.

Accuracy RM for Maze Navigation. To encourage the model to explore the maze, we design a rule-based RM. The final reward is a weighted combination of the following components.

- **Causal exploration progress.** We process the model’s step-by-step exploration sequentially. Upon encountering the first wall violation (*i.e.*, where the model claims to move between two cells separated by a wall), we truncate all subsequent exploration, as it is causally invalidated. Then, we calculate the shortest distance between the explored regions and the endpoint. The score is 1 minus the fraction of the distance and the length of the ground-truth path. The score measures how much of the maze was legally explored before truncation relative to the optimal reachable region, rewarding thorough and correct exploration. This item is only applied for solvable mazes. For unsolvable mazes, this item is kept as 1.
- **Exploration completeness.** For unsolvable mazes, the model must demonstrate that no path exists by exhaustively exploring the reachable region. The score is the fraction of the number of explored regions and the number of all the achievable regions. This component measures the fraction of truly reachable cells that appear in the model’s (causally valid) exploration trace. This item is only applied for unsolvable mazes. For solvable mazes, this item is kept as 1.
- **Wall violation penalty.** Independent of the causal truncation above, we scan the entire exploration trace to count every wall-violating transition. This score is 1 minus the fraction of the number of wall-violating transitions and the number of all the legal transitions of the maze. This directly penalizes each incorrect connectivity judgment, ensuring that wall violations are never cost-free even if they occur late in the exploration.
- **Final path validity.** When the model claims the maze is solvable, it must output a concrete solution path. We verify both that consecutive cells in this path are legally connected (no wall violations) and that the path forms a continuous route from start to end. This item is a binary score for solvable mazes. For unsolvable mazes, this item is kept as 1.
- **Answer correctness.** A binary score for whether the model’s solvability judgment matches the ground truth.

This decomposition ensures that the reward signal is dense and informative: the model receives credit for each correctly applied visual primitive rather than only for the final binary answer.

Accuracy RM for Path Tracing. To enforce the model to trace follow the lines, we propose a rule-based RM to judge the generated sequences of points. The final reward is a weighted summation of the following items.

- **Trajectory accuracy.** We evaluate the alignment between the predicted trajectory and the ground-truth curve from two complementary directions. In the forward direction, for each predicted point, we compute its minimum distance any segment of the groundtruth

poly-line, then average the distances across all the predicted points. This item penalizes points that deviate from the true path. In the reverse direction, for each ground-truth point, we compute its minimum distance to any segment of the predicted poly-line. This penalizes incomplete coverage where the model skips portions of the curve. The final trajectory score is the average of both directions.

- **Endpoint accuracy.** We separately verify that the model correctly identifies the start and end locations. For each, we compute the distance between the model’s predicted coordinate and the ground-truth bounding box center. The score decays with distance, reaching zero beyond a tolerance threshold.
- **Trajectory continuity penalty.** If the distance between the last point in the model’s trajectory and its predicted endpoint exceeds a threshold, a fixed penalty is applied. This discourages the model from outputting a partial trajectory and then “jumping” to a guessed endpoint without actually tracing the full path.
- **Answer correctness.** A score for whether the endpoint label in the model’s final answer matches the ground truth.

The bidirectional trajectory evaluation is critical. The forward direction alone would allow the model to output only a few safe points near the start, while the reverse direction alone would not penalize hallucinated detours. Together, they incentivize the model to produce a complete and accurate coordinate trace of the target curve.

RL Data. We expand the data pool during the RL phase. Prior to RL training, we use the SFT cold-start model (F_{TWG} or F_{TWP}) to perform rollouts over the data pool, generating N rollouts for each sample. Subsequently, based on the RM scores, we count the number of correct responses among the N rollouts for each sample and categorize the data pool into three difficulty levels:

- Easy-Level: All N rollouts are correct.
- Normal-Level: The number of correct rollouts k satisfies $1 \leq k < N$.
- Hard-Level: All N rollouts are incorrect.

We select samples from the “Normal-Level” category for RL, ensuring that the model receives valuable supervisory signals during the GRPO training process. Following the Specialized RL phase, we obtain two expert models, denoted as E_{TWG} and E_{TWP} .

2.5.3. Unified RFT

Equipped with the robust expert models E_{TWG} and E_{TWP} obtained above, we proceed to integrate the two visual-primitive-based reasoning paradigms—*thinking with grounding* and *thinking with pointing*—into a single unified model. We employ these expert models to perform rollouts over the data pool to generate RFT data. Applying the previously introduced difficulty categorization criteria, we retain all samples classified as “Normal-Level” and randomly sub-sample 5% of the “Easy-Level” data (to prevent catastrophic forgetting in excessively simple scenarios). Leveraging this larger and more diverse RFT dataset, we initialize from the base pretrained model to train an enhanced SFT model. Our RFT training configuration remains identical to that of the SFT cold-start phase (including the training hyper-parameters and the initial checkpoint), with the only difference being the updated training data mixture. Following this procedure, we obtain the unified model F .

2.5.4. On-Policy Distillation

Although the RFT model F demonstrates substantial improvements over the cold-start models F_{TWG} and F_{TWP} in their respective domains, a noticeable performance gap remains when compared to the expert models E_{TWG} and E_{TWP} . To bridge this gap, we follow [3] and employ On-Policy Distillation (OPD) to effectively consolidate the capabilities of the expert models into a single unified model. This distillation process is achieved by enabling the student model to learn the output distribution of the teacher models based on its own generated trajectories. Formally, given a set of N expert models $\{\pi_{E_1}, \pi_{E_2}, \dots, \pi_{E_N}\}$, the OPD objective function is defined as:

$$\mathcal{L}_{\text{OPD}}(\theta) = \sum_{i=1}^N w_i \cdot D_{\text{KL}}(\pi_{\theta} \parallel \pi_{E_i}), \quad (2)$$

where w_i denotes the weight assigned to each expert model, D_{KL} represents the reverse Kullback-Leibler (KL) divergence loss, and π_{θ} denotes the student model. We adopt full-vocabulary logit distillation for our OPD implementation. In practice, we utilize two teacher models, including E_{TWG} and E_{TWP} .

3. Experiments

3.1. Implementation Details

Our model is trained and evaluated using HAI-LLM [7], a lightweight and efficient distributed training framework built upon PyTorch. During the pre-training stage, we employ a sequence length of 64K and FP8 precision; in the post-training stage, the sequence length is extended to 256K. To maximize the performance of the domain experts, we utilize FP8 precision during the Specialized SFT and Specialized RL phases, and subsequently apply FP4 (MXFP4) quantization in the Unified RFT and OPD phases.

3.2. Evaluation Setup

Our evaluation framework integrates widely adopted public benchmarks with a curated in-house suite. While public benchmarks are vital for standardized comparisons, their constrained evaluation dimensions often fail to capture the full spectrum of a model’s capabilities, e.g., thinking with visual primitives. To bridge this gap, our in-house suite introduces more diverse and challenging axes, serving as a critical complement to the public datasets.

Public Benchmarks. To evaluate counting capabilities, we use two widely used counting benchmarks, including CountQA [30] and Pixmo-Count [4]. We follow the standard evaluation protocols for each dataset, using the official test split for Pixmo-Count. To evaluate spatial reasoning and general VQA, we use benchmarks such as SpatialMQA [20], CV-Bench [31], EmbSpatial [5], OmniSpatial [11], and MIHBench [16].

In-House Benchmarks. To conduct a more granular evaluation of the model’s capacity to solve tasks by “Thinking with Visual Primitives”, we curate a tailored in-house benchmark suite spanning three critical dimensions: fine-grained counting, multi-hop spatial reasoning, and topological reasoning.

Table 1 | **Comparison with frontier models.** To ensure a fair comparison, we evaluated all models via their respective APIs using an identical set of prompts. The best results are highlighted in bold; the second-best results are underlined.

Category	Benchmark (Metric)	Gemini-3-Flash	GPT-5.4	Claude-Sonnet-4.6	Gemma4-31B	Qwen3-VL		Ours
						235B-A22B-Thinking	284B-A13B-Thinking	
Counting	CountQA (EM / RA@10)	66.1/75.1	48.3 / 60.3	34.8 / 46.6	43.2 / 54.6	42.7 / 54.8	<u>64.9/74.1</u>	
	Pixmo-Count (EM)	<u>88.2</u>	76.6	68.7	82.9	77.2	89.2	
	DS_Finegrained_Counting (EM)	79.1	84.2	82.6	79.5	<u>87.2</u>	88.7	
Spatial Reasoning & General VQA	MIHBench (ACC)	83.2	<u>83.5</u>	81.7	82.2	75.1	85.3	
	SpatialMQA (ACC)	<u>67.0</u>	61.9	58.2	60.6	54.5	69.4	
	EmbSpatial (ACC)	<u>82.6</u>	80.9	75.1	82.1	83.7	83.7	
	CV-Bench (ACC)	88.6	87.5	85.1	87.5	88.1	<u>88.4</u>	
	OmniSpatial (ACC)	59.6	58.8	53.2	49.4	55.3	<u>59.5</u>	
	DS_Spatial_Reasoning (ACC)	93.2	81.1	<u>97.2</u>	77.2	96.8	98.7	
Topological Reasoning	DS_Maze_Navigation (ACC)	49.4	<u>50.6</u>	48.9	49.8	49.6	66.9	
	DS_Path_Tracing (ACC)	41.4	<u>46.5</u>	30.6	33.9	24.5	56.7	

- **Fine-grained Counting:** Existing fine-grained counting benchmarks, such as TallyQA [1], often suffer from annotation errors and ambiguities, rendering them suboptimal for rigorously evaluating a model’s fine-grained counting capabilities. To address this, we introduce the DS_Finegrained_Counting evaluation set. Specifically, we prompt an MLLM to generate counting queries constrained by specific attributes or spatial locations, deliberately ensuring the presence of hard negative samples (i.e., objects sharing the same category as the query target but different attributes). Following rigorous manual verification to ensure data quality, we retain a final set of 600 high-quality test cases.
- **Multi-hop Spatial Reasoning:** We sample 1,000 true/false questions and 1,000 open-ended questions from the validation set of CLEVR [13]. To facilitate standardized and automated evaluation, we leverage an MLLM to generate plausible distractor options for the open-ended queries, thereby converting them into a multiple-choice format. This reorganized evaluation suite is denoted as DS_Spatial_Reasoning.
- **Topological Reasoning:** Following methodologies in Section 2.4.3 and Section 2.4.4, we construct two distinct evaluation sets: DS_Maze_Navigation and DS_Path_Tracing, comprising 2,000 instances for both of the two sets.

3.3. Comparison with Frontier Models

For fair comparison, we adopt a unified evaluation protocol across all models. Given that some legacy public benchmarks contain low-resolution images, we apply a preprocessing step to ensure data quality. Specifically, any image with a total pixel count below 640,000 is upscaled to reach this pixel threshold while strictly preserving its original aspect ratio. For frontier models that support configurable reasoning or thinking budgets (e.g., GPT and Gemini-3-Flash), we uniformly set the thinking budget to low for all evaluations to ensure a fair and consistent comparison. For all other benchmarks, we follow the official evaluation protocols and metrics. Results are shown in Table 1. Benefiting from the ability to think with visual primitives, our model achieves competitive performance on these tasks with remarkable token efficiency. Notably, all frontier models exhibit suboptimal performance on topological reasoning tasks, suggesting that substantial room for improvement remains in the reasoning capabilities of multimodal large language models.

3.4. Qualitative Results

3.4.1. Boxes as Visual Primitives

As shown in Figs. 7 to 9, our model demonstrates strong performance on coarse-grained and fine-grained counting tasks through thinking with grounding, while also exhibiting emergent capability synergies. For instance, the model is able to integrate world knowledge for visual question answering, perform counterfactual reasoning, and provide actionable suggestions with spatial coordinates tailored to users’ everyday needs. Although our post-training data about visual primitives does not include any Chinese corpus, the model is capable of thinking and responding in Chinese, benefiting from the multilingual capabilities inherited from the base model.

3.4.2. Points as Visual Primitives

As shown in Fig. 10, our model shows the ability of topological reasoning via thinking with pointing, producing step-by-step exploration traces for mazes and sequential tracking trajectories for path tracing. On in-domain instances, the model has the ability to identify and follow the paths, which is enforced via mitigating the cold start data and rewarding during the specialized RL process.


4. Limitations

Despite these promising results, our current work has certain limitations. First, constrained by input resolution, the model’s performance in fine-grained scenarios remains sub-optimal, leading to occasionally imprecise outputs of visual primitives. This could potentially be addressed by integrating our framework with existing methods targeting the "Perception Gap" to achieve complementary benefits. Second, the current “thinking with visual primitives” capability relies on explicit trigger words for activation. In the future, we aim to enable the model to autonomously determine whether to invoke this mechanism based on the specific context. Third, utilizing points as visual primitives to solve complex topological reasoning problems remains a formidable challenge, and our current model exhibits limited cross-scenario generalization. Exploring ways to broaden the applicability and robustness of this technique constitutes an important direction for future research.

5. Conclusion

To address the inherent “Reference Gap” in Multimodal Large Language Models (MLLMs) during complex reasoning, we introduce “Thinking with Visual Primitives”, a novel reasoning framework. Moving beyond the conventional reliance on simply increasing perceptual resolution, we elevate spatial markers—such as points and bounding boxes—to “minimal units of thought” and interleave them directly into the model’s thinking process. This mechanism endows the model with the ability to “point while it reasons”, precisely anchoring abstract linguistic concepts onto physical image coordinates. Furthermore, leveraging a highly efficient visual token compression architecture, our model achieves performance on par with frontier models across highly challenging tasks, including spatial reasoning, visual QA, and topological reasoning, while significantly reducing image token consumption. Our work demonstrates that the path to System-2 multimodal intelligence lies not merely in “seeing more pixels”, but in constructing a precise, unambiguous referential bridge between language and vision.

[Trigger_Placeholder]
How many pokemons are there in this photo?



Original Image





Image with Visual Primitives

Thinking with Visual Primitives

- Understanding the request**
The user wants me to count the number of Pokémon figures in the image. I need to look for characters from the Pokémon franchise among the various toys displayed on the table.
- Scanning the table for Pokémon**
Looking at the collection of figures, I can identify the Pokémon characters: `< | ref | > pokemons < /ref | >` `> [[215,483,368,711],[330,180,422, 348],[465,438,582,638],[581,403,674,588],[635,195,723,323],[640,244, 814,429]] < /box | >`. Breaking this down visually, I see Pikachu on the left, Psyduck in the back row, Charmander in the center, Bulbasaur in the front row, Jigglypuff in the back right, and Squirtle next to Jigglypuff, partially obscured by the Detective Pikachu figure.
- Distinguishing Pokémon from other figures**
I am making sure not to include characters from other franchises. Specifically, I'm ignoring Doraemon on the left, Luffy, Zoro, and the red-hatted figure from One Piece in the front left, the Crayon Shin-chan figure in the back center, the female figure, the guy in the detective hat, and the boxy character on the far right.
- Final count of Pokémon**
Counting the distinct Pokémon characters found, there are 6 of them.

Response
Based on the image, there are 6 Pokémon figures.

[Trigger_Placeholder]
Which one is heavier?



Original Image





Image with Visual Primitives

Thinking with Visual Primitives

- Understanding the weight comparison task**
The user is asking to determine which of the two objects on the scale is heavier. To answer this, I need to locate each object and evaluate the physical evidence provided by the scale's state (which side is lower and which is higher).
- Locating the left object**
On the left side of the scale, I see a `< | ref | > large gray metal locker < /ref | >` `> [[26,41,396,699]] < /box | >`. It is constructed with solid, rigid materials and mounted securely on the left `< | ref | > metal tray < /ref | >` `> [[20,685,393,759]] < /box | >` of the balance. The locker is rectangular and imposing, suggesting substantial structural mass.
- Locating the right object**
On the right side of the scale, resting on the corresponding `< | ref | > metal tray < /ref | >` `> [[636,747,970,860]] < /box | >`, is a `< | ref | > translucent rainbow bear < /ref | >` `> [[691,407,916,808]] < /box | >`. It looks like a large, vibrant gummy bear made of a translucent, jelly-like substance.
- Evaluating the scale's equilibrium**
The visual evidence from the balance scale is very clear. The `< | ref | > metal tray < /ref | >` `> [[636,747,970,860]] < /box | >` on the right is positioned lower to the ground than the `< | ref | > metal tray < /ref | >` `> [[20,685,393,759]] < /box | >` on the left. The arm holding the right tray is also pointing downward. This means the right side of the scale is heavier.
- Conclusion**
Because the right side containing the gummy bear is lower than the left side containing the locker, the bear must be heavier in this scenario.

Response
Based on the scale's tilt, the rainbow bear is heavier.

[Trigger_Placeholder]
How many Chihuahuas are there in the picture?



Original Image

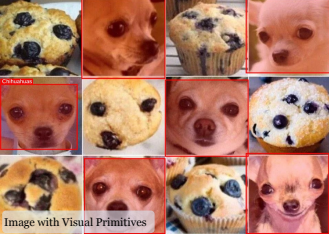


Image with Visual Primitives

Thinking with Visual Primitives


- Defining the target animal**
The question asks me to count the number of Chihuahuas in the image. I need to focus on differentiating the dogs from the blueberry muffins that also appear in the grid. Looking at the overall layout, the grid seems to contain alternating photos of dogs and muffins.
- Scanning the grid for dogs**
I'll search the entire grid to locate every dog. After scanning the rows, I spot the following set of distinct Chihuahuas: `< | ref | > Chihuahuas < /ref | >` `> [[2,366,233,641],[247,0,498,341],[255,680,498,997],[499,347,749,670],[743,0,996,317],[743,658,992,997]] < /box | >`. These dogs appear in the first row (second and fourth columns), the second row (first and third columns), and the third row (second and fourth columns). They are mostly a light tan or brownish-white color, with large dark eyes and large ears.
- Confirming the final count**
Looking at the identified dogs, I have counted 6 Chihuahuas in total.

Response
There are 6 Chihuahuas in the picture.

Figure 7 | Showcases of thinking with grounding. Examples include fine-grained counting and counter-commonsense visual question answering, Best viewed on screen with zoom.

19

[Trigger_Placeholder]
Why is this funny?



Original Image

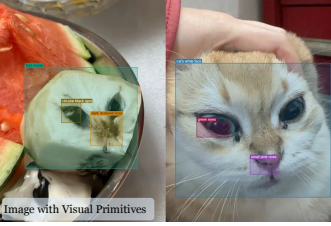


Image with Visual Primitives


Thinking with Visual Primitives

- Initial impression of the image****
This is a side-by-side comparison image. The left side shows a chunk of fruit (maybe a pear or apple) with a face-like indentation and dark marks. The right side shows a close-up of a cat's face. It looks like someone carved the fruit to look exactly like the cat.
- Locating and explaining the resemblance****
I'm looking for the visual similarities. On the left, I see a < | ref | >fruit chunk< /ref | >< | box | >[[75,303,414,758]]< /box | >. Within its pale flesh, there are natural dark spots that look like eyes. Specifically, there is a < | ref | >circular black spot< /ref | >< | box | >[[185,463,244,552]]< /box | > that perfectly mimics a pupil. Additionally, twisted fiber or skin at the bottom of the fruit chunk looks like a < | ref | >dark, textured nose< /ref | >< | box | >[[273,519,369,655]]< /box | >. Now looking at the right, there is a < | ref | >cat's white face< /ref | >< | box | >[[530,284,999,888]]< /box | > filling the frame. I can see a pair of < | ref | >green eyes< /ref | >< | box | >[[594,547,690,617]]< /box | >, looking absolutely devastated or sad. Below the eyes is a < | ref | >small pink nose< /ref | >< | box | >[[755,715,842,786]]< /box | > located between the cheeks.
- Drawing the conclusion****
The humor comes from the 'devastated' expression on the cat's face being perfectly replicated by the randomly shaped marks on the fruit. It looks like the fruit is a raw, unedited version of the cat's soul-piercing sadness. The comparison is spot on, making it funny.

Response

This is funny because the piece of fruit on the left has natural dark spots that form a face looking remarkably similar to the sad, green-eyed cat on the right. The similarities in the "eyes" and "nose" create a hilarious, uncanny resemblance between the two.

[Trigger_Placeholder]
该如何逃出生天呢?



Original Image



Image with Visual Primitives

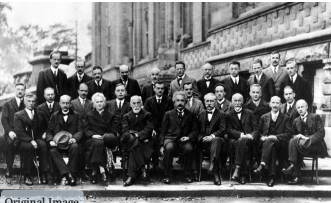
Thinking with Visual Primitives

- 初步观察****
我环顾四周，发现自己身处一个封闭的室内环境。墙面粗糙，左侧有一扇带有铁栅栏的窗户，右侧是一扇关闭的木门。眼前最引人注目的高处架子上是一把金色钥匙。我需要找到逃出去的方法。
- 寻找潜在线索****
我注意到墙面上高悬着 < | ref | >一把发光的金色钥匙< /ref | >< | box | >[[566,110,649,165]]< /box | >，它似乎在暗示我这是解锁的关键。木架上并没有可视的楼梯或踏板。木门下方，我看到 < | ref | >一个带有金属门把手和锁孔的深色木制门< /ref | >< | box | >[[665,231,991,910]]< /box | >，这显然是需要钥匙才能打开的出口。在房间中央的地板上，摆放在我面前的 < | ref | >一把简单的木制椅子< /ref | >< | box | >[[399,554,603,936]]< /box | >。它的位置似乎有些特殊。
- 整合信息与推理****
目前我面临的核心问题是如何拿到高处架子上的钥匙。我只有一把普通高度的木制椅子，直接伸手是够不到钥匙的。观察门把手和椅子的位置，它们都在房间的较低高度，而光线和我的注意力都集中在上方。我意识到，这很可能是一个简单的物理谜题。如果我移动这把椅子，把它推到门边或架子下方，我或许能够踩在上面够到钥匙。
- 得出结论****
综合来看，要逃出这个房间，最直接的策略是：先移动这把低矮的椅子，将其挪到悬挂着金色钥匙的墙壁或架子下方，然后利用椅子作为垫脚石，伸手取下钥匙，最后走到右侧的木门处用钥匙开门。

Response

要逃出生天，你可以把木椅移动到挂着钥匙的架子下方，踩着椅子拿到钥匙，然后去开右侧的门。

[Trigger_Placeholder]
How many people are there?



Original Image




Image with Visual Primitives

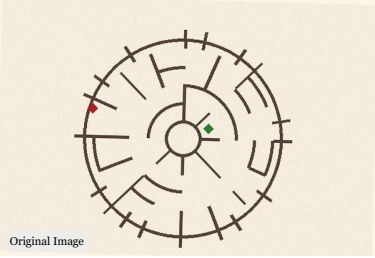
Thinking with Visual Primitives

- Understanding the task****
I need to count the number of people visible in the image. The picture is a large group portrait featuring several rows of individuals, so I need to scan carefully from the back to the front to ensure no one is missed.
- Scanning the group for individuals****
I'll look across the entire frame, identifying heads, faces, and bodies to pick out each individual. Sweeping across the arrangement of people, I find all of them: < | ref | >people< /ref | >< | box | >[[4,459,80,859],[32,506,147,900],[105,477,189,883],[107,335,202,638],[136,502,250,893],[202,362,276,616],[210,463,289,826],[248,499,363,894],[265,386,335,617],[307,455,419,835],[341,383,416,596],[355,510,478,898],[413,392,501,611],[425,447,529,835],[483,489,608,895],[487,374,584,576],[539,450,623,824],[582,374,664,594],[582,498,685,889],[626,471,714,817],[658,371,750,591],[684,508,774,887],[720,469,812,844],[737,365,816,603],[768,510,879,895],[788,320,882,601],[824,470,919,889],[828,370,930,625],[856,529,989,891]]< /box | >. This includes people in the very back standing tall, those standing in the middle slightly hunched forward to be seen, and the individuals seated on chairs in the very front.
- Final count confirmation****
Counting the distinct individuals I just located across the entire group, I find 29 people in total.

Response

There are 29 people in the image.

Figure 9 | Showcases of thinking with grounding. Examples include humor comprehension in images, escape room game guidance, and counting. Best viewed on screen with zoom.



Original Image

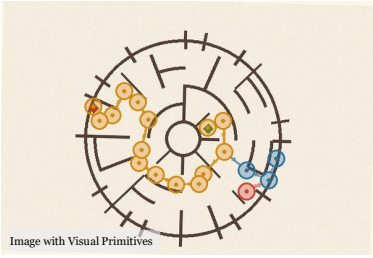


Image with Visual Primitives

[Trigger_Placeholder]
Is it possible to navigate from the green rhombus through the maze to the red label? If so, explain the path. Finally, output `\boxed{True}` or `\boxed{False}` based on solvability.

Thinking with Visual Primitives

Okay, let me analyze the structure of this maze step by step. First locate the starting point: `< | point | >[[555,499]]< | /point | >`, and the destination: `< | point | >[[247,416]]< | /point | >`.

****Start Exploring**:**

****Step1**:** Arriving at `< | point | >[[555,499]]< | /point | >`, I see 2 paths ahead (counter-clockwise, outward). The counter-clockwise direction is also passable, but from the maze layout, outward direction's path looks wider.

****Step2**:** At `< | point | >[[597,472]]< | /point | >`, I notice 2 directions to choose from (clockwise, counter-clockwise). Although counter-clockwise direction is also walkable, after observing path structure, I judge clockwise direction more likely leads to exit.

****Step3**:** Reaching `< | point | >[[600,594]]< | /point | >`, I face 2 forks (clockwise, outward). Let me try outward direction first.

****Step4**:** Moving outward: `< | point | >[[600,594],[660,667],[720,705]]< | /point | >`...reaching `< | point | >[[720,705]]< | /point | >`, I see 2 forks here. Observing that up direction is a straight dead end at `< | point | >[[740,620]]< | /point | >`, can see wall directly. Still need to explore the continuing outward direction.

****Step5**:** Pushing clockwise: `< | point | >[[720,705],[660,750]]< | /point | >`...but hit a wall at `< | point | >[[660,750]]< | /point | >`. It's a dead end, must backtrack to the previous fork `< | point | >[[720,705]]< | /point | >`.

****Step6**:** All paths in this branch are explored, all dead ends. Going back to main junction `< | point | >[[600,594]]< | /point | >`, taking clockwise direction.

****Step7**:** Reaching `< | point | >[[544,682]]< | /point | >`, I face 3 forks (inward, outward, outward). Comparing to inward direction, outward direction's passage characteristics are more like a main path.

****Step8**:** Reaching `< | point | >[[532,720]]< | /point | >`, I face 2 forks (clockwise, outward). The outward direction is also passable, but from the maze layout, clockwise direction's path looks wider.

****Step9**:** At `< | point | >[[471,722]]< | /point | >`, I notice 2 directions to choose from (clockwise, inward). Although inward direction is also walkable, after observing path structure, I judge clockwise direction more likely leads to exit.

****Step10**:** Reaching `< | point | >[[415,683]]< | /point | >`, I face 2 forks (clockwise, outward). Although outward direction is also walkable, after observing path structure, I judge clockwise direction more likely leads to exit.

****Step11**:** Reaching `< | point | >[[371,637]]< | /point | >`, I face 2 forks (inward, outward). Comparing to outward direction, inward direction's passage characteristics are more like a main path.

****Step12**:** At `< | point | >[[378,567]]< | /point | >`, I notice 3 directions to choose from (clockwise, inward, outward). The inward, outward direction is also passable, but from the maze layout, clockwise direction's path looks wider.

****Step13**:** At `< | point | >[[396,467]]< | /point | >`, I notice 2 directions to choose from (outward, outward). After eliminating dead ends, outward is the best choice.

****Step14**:** Reaching `< | point | >[[368,397]]< | /point | >`, I face 2 forks (clockwise, outward). The outward direction is also passable, but from the maze layout, clockwise direction's path looks wider.

****Step15**:** At `< | point | >[[330,355]]< | /point | >`, I notice 2 directions to choose from (outward, outward). After eliminating dead ends, clockwise is the best choice.

****Step16**:** Continuing outward: `< | point | >[[330,355],[300,450]]< | /point | >`. Keeping this direction.

****Step17**:** Looking in outward direction, I can directly see the path to the exit, no need to linger at the junction: `< | point | >[[300,450],[265,472],[247,416]]< | /point | >`

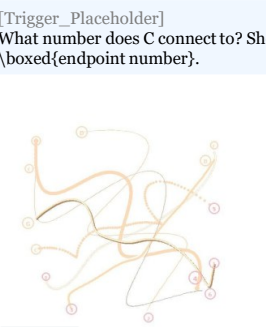
****Step18**:** Arriving at `< | point | >[[247,416]]< | /point | >`, I finally see the exit!

****Final Path**:** After exploration, the correct route is:
`< | point | >[[555,499],[597,472],[600,594],[544,682],[532,720],[471,722],[415,683],[371,637],[378,587],[396,467],[368,397],[330,355],[300,450],[265,472],[247,416]]< | /point | >`

Successfully reaching the destination: `< | point | >[[247,416]]< | /point | >!`

Response

The maze is solvable. The verified path is:
`< | point | >[[555,499],[597,472],[600,594],[544,682],[532,720],[471,722],[415,683],[371,637],[378,587],[396,467],[368,397],[330,355],[300,450],[265,472],[247,416]]< | /point | >`
`\boxed{True}`



Original Image

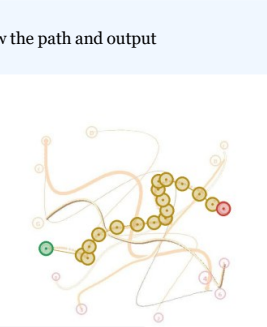


Image with Visual Primitives

[Trigger_Placeholder]
What number does C connect to? Show the path and output `\boxed{endpoint number}`.

Thinking with Visual Primitives

I find the starting point you mentioned, it's located here:
`< | point | >[[195,594]]< | /point | >`.

Following this beige line, the visual path I observe is: `< | point | >[[195,594],[319,617],[338,626],[342,588],[375,547],[438,522],[508,512],[565,505],[604,490],[609,464],[595,427],[579,391],[580,364],[606,358],[661,372],[720,408],[765,442],[803,458]]< | /point | >`

Following this path, it connects to: `< | point | >[[803,457]]< | /point | >`

Response

Based on the visual trace, the endpoint is 5. `\boxed{5}`

Figure 10 | Showcases of thinking with pointing. Examples include maze navigation and path tracing. Best viewed on screen with zoom.

References

- [1] M. Acharya, K. Kafle, and C. Kanan. Tallyqa: Answering complex counting questions. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pages 8076–8084, 2019.
- [2] M. Amgad, L. A. Atteya, H. Hussein, K. H. Mohammed, E. Hafiz, M. A. Elsebaie, A. M. Alhusseiny, M. A. AlMoslemany, A. M. Elmatboly, P. A. Pappalardo, et al. Nucls: A scalable crowdsourcing approach and dataset for nucleus classification and segmentation in breast cancer. GigaScience, 11:giac037, 2022.
- [3] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.
- [4] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 91–104, 2025.
- [5] M. Du, B. Wu, Z. Li, X.-J. Huang, and Z. 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.
- [6] Y. Fan, X. He, D. Yang, K. Zheng, C.-C. Kuo, Y. Zheng, S. J. Narayanaraju, X. Guan, and X. E. Wang. Grit: Teaching mllms to think with images. arXiv preprint arXiv:2505.15879, 2025.
- [7] High-flyer. Hai-llm: Efficient and lightweight training tool for large models, 2023. URL <https://www.high-flyer.cn/en/blog/hai-llm>.
- [8] J. Hong, C. Zhao, C. Zhu, W. Lu, G. Xu, and X. Yu. Deepeyesv2: Toward agentic multimodal model. arXiv preprint arXiv:2511.05271, 2025.
- [9] M.-R. Hsieh, Y.-L. Lin, and W. H. Hsu. Drone-based object counting by spatially regularized regional proposal network. In Proceedings of the IEEE international conference on computer vision, pages 4145–4153, 2017.
- [10] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 6700–6709, 2019.
- [11] M. Jia, Z. Qi, S. Zhang, W. Zhang, X. Yu, J. He, H. Wang, and L. Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models. arXiv preprint arXiv:2506.03135, 2025.
- [12] C. Jiang, Y. Heng, W. Ye, H. Yang, H. Xu, M. Yan, J. Zhang, F. Huang, and S. Zhang. Vlm-r³: Region recognition, reasoning, and refinement for enhanced multimodal chain-of-thought. arXiv preprint arXiv:2505.16192, 2025.
- [13] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2901–2910, 2017.

- [14] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov, T. Duerig, and V. Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.
- [15] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020.
- [16] J. Li, M. Wu, Z. Jin, H. Chen, J. Ji, X. Sun, L. Cao, and R. Ji. Mihbench: Benchmarking and mitigating multi-image hallucinations in multimodal large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 3143–3152, 2025.
- [17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [18] H. Liu, C. Li, Y. Li, and Y. J. Lee. Improved baselines with visual instruction tuning, 2023.
- [19] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023.
- [20] J. Liu, Z. Liu, Z. Cen, Y. Zhou, Y. Zou, W. Zhang, H. Jiang, and T. Ruan. Can multimodal large language models understand spatial relations? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 620–632, 2025.
- [21] Y. Man, D.-A. Huang, G. Liu, S. Sheng, S. Liu, L.-Y. Gui, J. Kautz, Y.-X. Wang, and Z. Yu. Argus: Vision-centric reasoning with grounded chain-of-thought. 2025.
- [22] A. Mondal, S. Nag, X. Zhu, and A. Dutta. Omnicount: Multi-label object counting with semantic-geometric priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19537–19545, 2025.
- [23] D. T. O’Brien. *Thinking, fast and slow* by daniel kahneeman. 2012.
- [24] OpenAI. Introducing o3 and o4-mini. <https://openai.com/index/introducing-o3-and-o4-mini/>, 2025.
- [25] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, and F. Wei. Kosmos-2: Grounding multimodal large language models to the world. *ArXiv*, abs/2306.14824, 2023.
- [26] J. Qi, M. Ding, W. Wang, Y. Bai, Q. Lv, W. Hong, B. Xu, L. Hou, J. Li, Y. Dong, et al. Cogcom: A visual language model with chain-of-manipulations reasoning. *arXiv preprint arXiv:2402.04236*, 2024.
- [27] H. Shao, S. Qian, H. Xiao, G. Song, Z. Zong, L. Wang, Y. Liu, and H. Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 37: 8612–8642, 2024.
- [28] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018.

- [29] S. Shao, Z. Li, T. Zhang, C. Peng, G. Yu, X. Zhang, J. Li, and J. Sun. Objects365: A large-scale, high-quality dataset for object detection. In Proceedings of the IEEE/CVF international conference on computer vision, pages 8430–8439, 2019.
- [30] J. S. Tamarapalli, R. Grover, N. Pande, and S. Yerramilli. Countqa: How well do mllms count in the wild? arXiv preprint arXiv:2508.06585, 2025.
- [31] S. Tong, E. Brown, P. Wu, S. Woo, M. Middepogu, S. C. Akula, J. Yang, S. Yang, A. Iyer, X. Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. Advances in Neural Information Processing Systems, 37:87310–87356, 2024.
- [32] H. Wang, X. Li, Z. Huang, A. Wang, J. Wang, T. Zhang, J. Zheng, S. Bai, Z. Kang, J. Feng, et al. Traceable evidence enhanced visual grounded reasoning: Evaluation and methodology. arXiv preprint arXiv:2507.07999, 2025.
- [33] J. Wang, P. Zhang, T. Chu, Y. Cao, Y. Zhou, T. Wu, B. Wang, C. He, and D. Lin. V3det: Vast vocabulary visual detection dataset. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 19844–19854, 2023.
- [34] J. Wang, Z. Kang, H. Wang, H. Jiang, J. Li, B. Wu, Y. Wang, J. Ran, X. Liang, C. Feng, et al. Vgr: Visual grounded reasoning. arXiv preprint arXiv:2506.11991, 2025.
- [35] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling. Detection and tracking meet drones challenge. IEEE transactions on pattern analysis and machine intelligence, 44(11): 7380–7399, 2021.