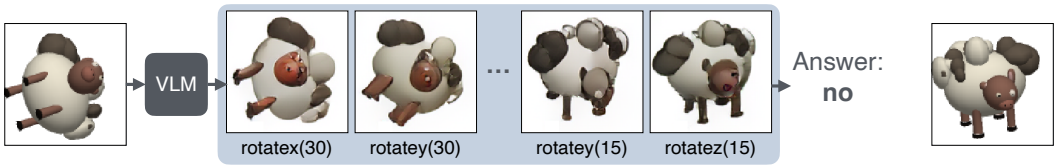
Do multimodal models imagine electric sheep?

Santhosh Kumar Ramakrishnan* **Carl Vondrick*** **Raja Giryes***
Philipp Krähenbühl **Vladlen Koltun**
Apple

Abstract

Yes. We find that large multimodal models develop mental imagery when solving spatial puzzles, and they do imagine sheep when solving sheep puzzles. We fine-tune a Qwen3.5 VLM to solve twelve diverse visual reasoning tasks – including tangram, jigsaw, sokoban, 3D mental rotation, and rush hour – that require understanding geometry, spatial relationships, and the consequences of actions. By supervising the model to predict the open-loop sequence of actions to solve a puzzle from an initial state, we show that the model's activations after each action encode meaningful visual information about the intermediate state. This finding suggests that an imperfect visual world model begins to form as a byproduct of learning to select correct actions, in the absence of any explicit visual supervision. Building on this observation, we study two techniques to sharpen and use the mental images formed by the model. We find that integrating as few as sixteen visual tokens per step into the chain of thought improves the average solve rate from 83% to 89%, with particularly strong gains on reasoning-heavy tasks such as jigsaw and 3D mental rotation. Code is available at `https://github.com/apple-aiml-research/ml-mental-imagery`.



(a) Left sheep (b) Model's imagination (c) Right sheep

Figure 1: **Are the left and right sheep identical?** We show that VLMs develop mental imagery when trained to solve such spatial puzzles, even in the absence of explicit visual supervision. The model takes the left (a) and right (c) images to predict a series of actions that equalize their pose before making a decision on whether the two sheep are identical. The center images (b) visualize the model's internal representations as read out by a separate visual decoder.

1 Introduction

Mental imagery is a powerful underpinning for spatial cognition, allowing people to visualize and transform objects and environments in their head to answer questions and accomplish goals [1–5]. For example, can you determine whether the sheep in Fig. 1 are the same? To answer this question, many people will mentally rotate the sheep in their head [1].

In contrast to humans, frontier models often perform poorly on spatial reasoning tasks, struggling to understand geometric relationships, track objects through transformations, and reason about the

*Equal contribution.

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

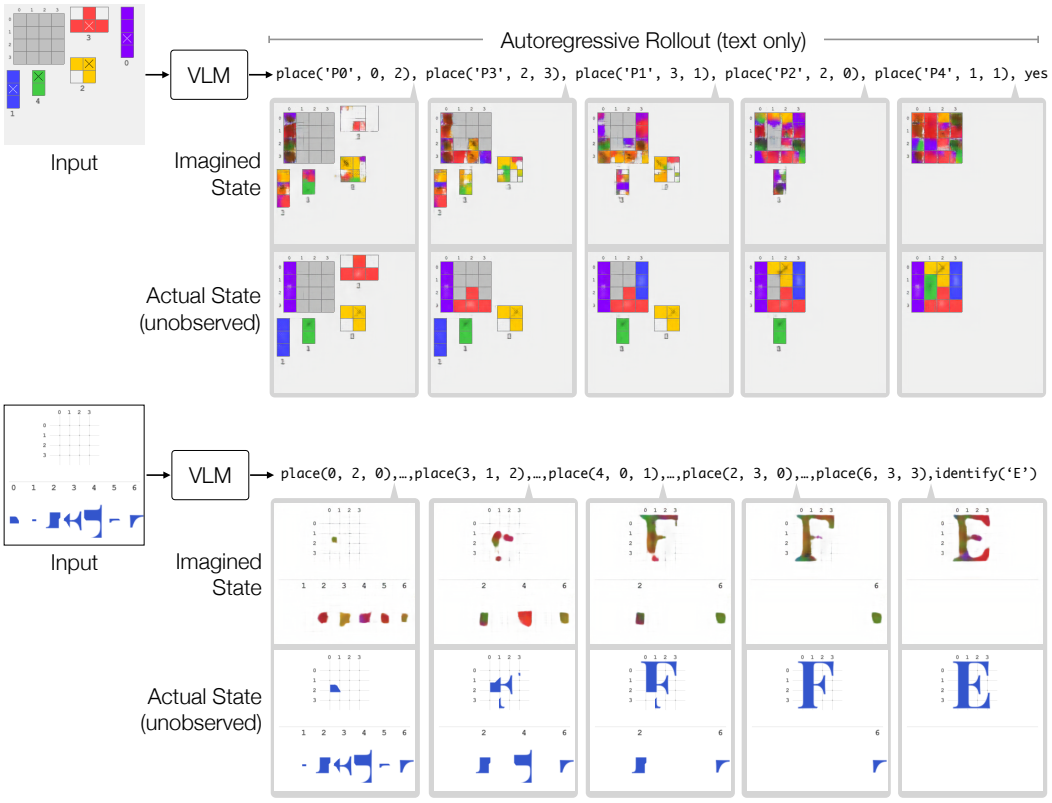


Figure 2: **Reconstructing world state from VLM activations.** We freeze a VLM that has been trained to autoregressively predict the actions needed to solve puzzles, attach a visual decoder on the hidden activations at the action boundaries, and train it to predict the visual state. Although the VLM is trained open-loop to predict actions without observing intermediate steps, the decoder is able to reconstruct the state from the VLM's internal representation. The colors are wrong, but the reconstruction of the tangram game matches the geometry well (top), and the identified character emerges in the visual representation (bottom). This suggests that mental imagery emerges in the model.

effects of actions [6–9]. This gap is not limited to any particular model family or scale, and exposes a fundamental limitation in how current large models represent and reason about visual structure.

We study to what extent mental imagery emerges in VLMs and how to develop it further for spatial tasks. We use a suite of twelve puzzle classes spanning mental rotation, space filling, jigsaw, and others to analyze the internal representations of multimodal language models. Working with such puzzles allows us to focus on the core challenge of spatial reasoning in controlled settings.

We use open-loop behavior cloning, where the model must learn to predict the full action sequence from the first frame without observing intermediate feedback. Our base model is the recent Qwen3.5 9B, which can only solve a small fraction of the puzzles out of the box (13% average success rate). Behavior cloning successfully teaches the model to solve some of the puzzles, reaching an 83% average success rate. More interestingly, we find that mental imagery emerges in the course of training, even though there is no direct visual supervision. Open-loop behavior cloning creates a supervisory signal that indirectly elicits an internal representation of the evolving world state. With enough data, mental imagery emerges. We visualize the model's mental imagery of sheep in Fig. 1 and of other objects in Fig. 2, showing that the internal representations are granular enough to reconstruct the visual state, recovering object positions, viewpoints, layouts, and spatial relationships that the model was never explicitly asked to represent. This echoes emergent world representations observed in prior work [10–12], now in visually complex environments.

Since multimodal models appear to spontaneously develop mental imagery, we explore how to shape these mechanisms through additional supervision, and how mental imagery can further enhance test-time reasoning. We develop two approaches. The first adds an auxiliary loss for direct prediction

of the visual state after each action. The second explicitly adds tokens representing the visual states after each action into the chain of thought, training the model to autoregressively predict interleaved text and state tokens. We study multiple tokenizations of the visual state: image tokens, bounding boxes, and json-based text descriptions, and find that they have complementary benefits: image tokens are best overall, achieving an 88.9% average success rate across puzzles, 6 percentage points above action supervision. With as few as sixteen image tokens per action, we see substantial gains on visually complex puzzles such as jigsaw and 3D mental rotation, where it is difficult to describe the state of the environment in words. On the other hand, the text and bounding box tokenizations do well in puzzles such as tangram, bloxorz, sokoban, and rush hour, where the visual state is easily discretizable. Our results suggest that shaping and using internal visual representations may advance the spatial cognition of frontier models.

2 Related Work

Spatial reasoning in vision-language models. A large body of recent work has documented the limited spatial cognition of existing vision-language models (VLMs). Benchmarks such as SPACE [6], OmniSpatial [7], SpatialViz-Bench [13], 11Plus-Bench [9], and Mind the Gap [14] have demonstrated the limitations of VLMs on mental rotation, spatial visualization, and geometric reasoning. STARE [15] and MIRA [16] probe multi-step spatial simulation and find that VLMs exhibit inconsistent performance even with oracle visual chain-of-thought guidance during evaluation. BabyVision shows that the core visual reasoning abilities that develop in humans before language remain out of reach for current models [8], while MOCHI demonstrates large gaps between humans and models on 3D shape inference from multiple viewpoints [17]. Budny et al. [18] trace these failures to deficits in visually-grounded serial processing. Liu and Lee [19] show that VLMs cannot reliably track identical objects through spatial transformations. Fu et al. [20] find that VLMs substantially underperform their own visual encoders on perception tasks. Asadi et al. [21] expose "mirage reasoning" where models produce seemingly correct outputs without grounding in the image.

Visual chain-of-thought. Several methods aim to augment VLMs with visual reasoning capabilities. Lotfi et al. [22] propose chain-of-sketch, decomposing visual tasks into intermediate sketch steps for improved out-of-distribution generalization. Surís et al. [23] use a code-generation model to compose vision-and-language modules into Python programs that are executed to answer visual queries. Menon et al. [24] use code generation to allow LLMs to draw before answering. Hu et al. [25] give multimodal models a visual sketchpad, using code and vision specialists to draw lines, boxes and masks on the input image, and reason over the resulting sketches. Wu et al. [26] propose visualization-of-thought, where LLMs generate intermediate visualizations for spatial reasoning; however, their approach relies on code-based rendering in text-only models rather than learning visual representations end-to-end. Bigverdi et al. [27] introduce perception tokens, which are tokenized visual representations such as depth maps and bounding boxes, to improve spatial understanding without external tools. Fan et al. [28] train VLMs with reinforcement learning to interleave natural-language reasoning with bounding-box coordinates that point to regions of the input image. Ray et al. [29] propose mull-tokens, modality-agnostic latent tokens for multimodal reasoning. Qin et al. [30] propose chain-of-visual-thought, in which the VLM autoregressively predicts a small budget of continuous visual tokens distilled from vision experts (depth, segmentation, edges, DINO features) to improve dense perception. Other methods re-inject content from the input image into the reasoning chain, either adaptively when the reasoning requires it [31] or into latent think tokens at test time [32].

Closest to our visual tokens approach are methods that generate intermediate visual states. Qin et al. [33] and Zhang et al. [34] train unified image understanding-and-generation models to generate intermediate images as part of their chain of thought, targeting image generation and editing. In the embodied domain, Li et al. [35] engineer explicit visual chain-of-thought by fine-tuning on ground-truth intermediate frames for navigation. Zhao et al. [36] show that predicting future image frames as intermediate visual goals improves robotic manipulation. Liang et al. [37] fine-tune a video diffusion model on human demonstrations to imagine the execution of a task in a new scene and use the generated video to control a robot, and Liang et al. [38] jointly train video and action generation, improving robustness and sample efficiency over behavior cloning. In autonomous driving, Tan et al. [39] replace natural-language chain-of-thought with a latent one that interleaves action proposals with world-model tokens that express the future outcomes of these actions. However, Zeller et al. [40] find with MentisOculi that visual strategies, from latent tokens to explicitly generated imagery,

generally fail to improve reasoning due to compounding generation errors. Cheng et al. [41] analyze multimodal chain-of-thought and find that its gains come from visual thoughts that carry image information into the reasoning, whether expressed in text or in images, with larger gains for clearer and more concise expressions. Our work shows that visual imagination emerges spontaneously from action prediction, and that it can be further sharpened to aid multi-step planning through auxiliary visual losses or compact visual tokens of the world state while mitigating drift.

Emergent world models. A growing body of work shows that sequence models trained via next-token prediction develop rich internal representations without explicit supervision. Nanda [10] demonstrates that a GPT trained on Othello gameplay learns a linear, causally relevant board-state representation in its activations. Gurnee and Tegmark [42] discover that large language models encode linear representations of space and time at multiple scales, with identifiable spatial neurons. Jin and Rinard [11] show that transformers trained on synthetic programs develop representations of unobserved intermediate program states. Liu et al. [43] find that visual information is encoded in the internal key-value tokens of multimodal language models, though it often fails to surface in outputs. We add to this body of work by demonstrating the emergence of visual imagination in multimodal models trained via imitation learning.

World models and unified multimodal architectures. World models that learn to predict future states from actions [44, 45] have seen renewed interest in the context of large generative models. Sudhakar et al. [46] learn an action-conditioned image generator from videos of hands interacting with objects, which predicts the visual outcome of a desired interaction. Maes et al. [47] present LeWorldModel, a joint-embedding predictive architecture that achieves stable end-to-end training from pixels. Wiedemer et al. [48] demonstrate emergent zero-shot reasoning in video models including maze solving and physical reasoning. Wang et al. [49] reveal that reasoning in video diffusion models emerges along denoising steps through a "chain-of-steps" process. Wu et al. [50] use video generation models to augment multimodal models with geometric cues. Magne et al. [51] train a vision-action foundation model on diverse gameplay via behavior cloning. Evaluations by Cai et al. [52] and Guo et al. [53] show that generative models achieve moderate physical reasoning but fail at abstract spatial tasks. Wang et al. [54] introduce a large-scale video reasoning benchmark that reveals early signs of emergent generalization. Heek et al. [55] propose unified latent representations regularized by diffusion for images and video. On the architecture side, unified models that combine visual understanding and generation – such as BAGEL [56] – enable both perception and generation within a single model. We use BAGEL as a baseline, and study a state chain-of-thought approach inspired by Mitra et al. [57].

3 Methods

We train VLMs to solve spatial puzzles. The model must predict a sequence of actions to solve the game or answer the question in the prompt.

3.1 Problem Setup

Given the initial image s_0 corresponding to a puzzle and a text instruction t described in natural language, the VLM $\mathcal{V}$ must produce a sequence of actions that solve the puzzle:

$$a_1, a_2, \cdots, a_N, r = \mathcal{V}(s_0, t), \tag{1}$$

where a_i is the action to be executed at state s_i to solve it. Some puzzles additionally require the model to predict a natural language answer to the puzzle (e.g., yes / no), denoted by r. The VLM produces the entire action sequence from the initial image alone, without observing intermediate states that result from its own actions. Since errors cannot be corrected from feedback [58], the model must internally simulate the consequences of each action to choose the next one. Open-loop supervision therefore creates pressure to maintain a faithful internal representation of the evolving state, the exact mechanism we hypothesize gives rise to mental imagery.

To study this problem, we implement twelve spatial puzzles shown in Fig. 3. Our suite comprises well-studied puzzles involving object rotation [6, 13, 17], arranging pieces to form shapes [16, 59, 60], sliding objects to reach a target [61–63], and text manipulation [64, 65]. Prior work has primarily focused on question-answering versions of these puzzles with small sample sizes, making it hard to perform systematic studies. In contrast, we implement them as game engines (written in Rust

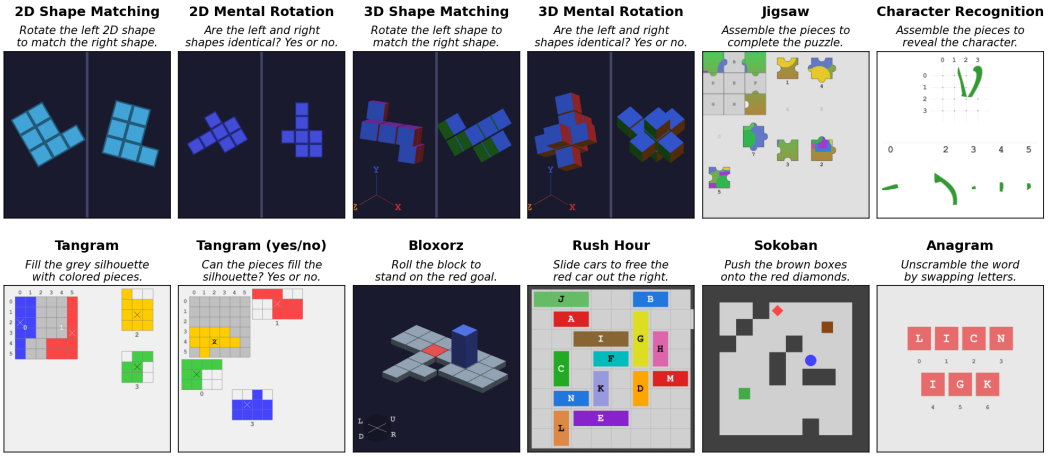


Figure 3: **Spatial reasoning puzzles.** We use twelve puzzles that require a variety of skills such as shape perception, visualization, and planning. We implement optimal solvers to rapidly sample training data on the fly, and use them to study spatial cognition in VLMs.

for speed) that can rapidly sample a variety of puzzle instances, along with expert solvers (such as breadth-first search, Algorithm X, or closed-form solutions) that can provide supervision via behavior cloning. Please see Section 4.1 for more details.

3.2 Action Supervision

We train a Qwen3.5 9B model with behavior cloning to predict the sequence of actions needed to reach the goal or answer the question. The model sees the initial state s_0, receives the puzzle instruction t, and generates a sequence of actions $(a_1, \cdots, a_N)$ supervised by the expert solver (and optionally a response r for puzzles that require a final answer). The model receives no visual supervision beyond the initial image (see Fig. 4 (top) for an illustration). For each puzzle, we run the appropriate expert solver and do full fine-tuning of the base model to predict the next token with teacher forcing.

What makes visual puzzles difficult is that they are sequential: each action changes the world, and the next action must account for that change. Solving a tangram puzzle with five pieces requires the model to mentally track how each placement and rotation reshapes the available space. Many of the puzzles require sequences of over twenty steps to solve, and the state space is inherently visual and geometric. Pieces have shapes, positions, and orientations, and moves have spatial preconditions and spatial effects. Although the model is trained without any visual supervision (only visual input), the model must learn to maintain an internal representation of the world to plan effectively, and that representation must be rich enough to capture the spatial relationships that determine which actions are legal and which lead toward the goal.

The model must learn to represent these states to minimize an open-loop behavior cloning objective. To probe for emergent mental imagery, we build a visual decoder that is trained separately. We use a four-layer transformer based on masked autoencoders [66] (see Appendix C). We condition only on the hidden features of the last token of each action, and train to predict the visual state after applying this action. Since the decoder only sees the representation for one token (and not even the initial image), it allows us to understand how well the VLM is tracking the visual state through long-horizons.

3.3 Visual Supervision

Action-only supervision requires mental images to emerge indirectly. By removing the stop gradient on the visual probe and letting the reconstruction loss shape the hidden states, we can strengthen and refine these emergent representations, encouraging the model to form sharper mental images. During training, we attach the same auxiliary visual prediction head from above that reads the model's hidden states at each action boundary, but now allow the gradients to flow between the two models in order to jointly learn behavior cloning with imagination of the visual state (see Fig. 4 (middle)).

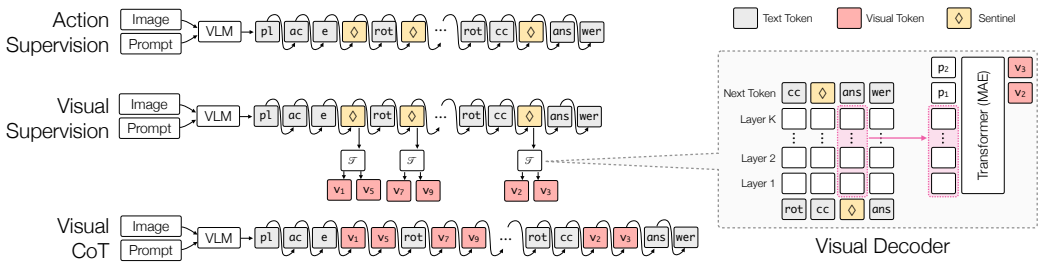


Figure 4: **Architectures. Top:** baseline post-trained model, which receives the input image and prompt and directly predicts the sequence of actions needed to solve the puzzle (Section 3.2). **Middle:** visual supervision, which jointly trains a visual decoder to predict the visual consequence of each action, thereby encouraging mental imagery (Section 3.3). **Bottom:** next-state chain-of-thought, illustrated with Visual CoT; State CoT and Boxes CoT write text tokens that describe the state in their place (Section 3.4).

The training objective for the visual head combines the standard language modeling objective with the visual prediction loss:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \, \mathcal{L}_{\text{visual}}, \tag{2}$$

where $\mathcal{L}_{\text{LM}}$ is the next-token prediction cross-entropy over action tokens, $\mathcal{L}_{\text{visual}}$ is the cross-entropy between predicted and ground-truth FSQ codes averaged over all action boundaries in the batch, and λ is a weighting hyperparameter. The gradient of $\mathcal{L}_{\text{visual}}$ flows through the cross-attention back into the language model, encouraging the hidden states at action boundaries to encode the visual state. At inference time, the visual head is removed and the model generates actions identically to action supervision, with no additional cost.

3.4 Next-State Chain-of-Thought

We also study chain-of-thought methods [67] in which the model generates discrete tokens that describe what the next state looks like as part of its output sequence. After each action, the model writes the resulting state as a block of state tokens z enclosed in special `<board>` `</board>` delimiters:

$$\texttt{<action>}\; a_{i_1} \cdots a_{i_M}\; \texttt{</action>}\; \texttt{<board>}\; z_{i_1} \cdots z_{i_K}\; \texttt{</board>}\; \texttt{<action>} \cdots \texttt{</action>} \cdots$$

All variants share the same training and inference procedure. The state blocks are teacher-forced to the ground-truth next state produced by the game engine, and the training loss decouples the state tokens from the other tokens to allow independent weighting:

$$\mathcal{L}_{\text{cot}} = w \; \text{CE}(H_{\text{state}}) + (1 - w) \; \text{CE}(H_{\text{txt}}), \tag{3}$$

where H_{state} and H_{txt} partition the supervised token positions into the tokens inside the state blocks and all other tokens (actions, delimiters, and answers), and w is a loss weight hyperparameter (values in Section 3.5). At inference time, the model generates its own state blocks and conditions on them when choosing the next action; the state blocks are not scored, and only the actions and the final answer are evaluated. The variants differ only in the type of tokens that describe the state.

Visual CoT: The state is an image. We resize the rendered 256×256 image and encode it with a frozen VQ-VAE [68, 69]: resizing to 64×64 yields a 4×4 code grid (16 tokens per state, Visual CoT 64×64), and resizing to 128×128 yields an 8×8 grid (64 tokens, Visual CoT 128×128). The model's vocabulary is extended with $|\mathcal{C}|$ new tokens, one for each entry in codebook $\mathcal{C}$, whose embeddings are initialized by aligning the VQ-VAE tokens to the VLM's visual encoder embeddings (Appendix C); the VLM and the new embeddings are trained jointly. The generated tokens can be decoded as images and inspected, which makes the model's visual reasoning explicit and interpretable.

State CoT: Inspired by compositional chain-of-thought [57], the state is a compact JSON description written with the model's existing text vocabulary. A per-game schema encodes the minimum information needed to reconstruct the state, such as the board as a string of cell characters in sokoban or the rotation angles of the shape in mental rotation, plus a few cheaply derivable quantities. The schemas are calibrated to at most about 64 tokens per state on average for every game, matching Visual CoT 128×128. See Appendix E for illustrations of the schema for each game.

Boxes CoT: The state is the integer location of every task-relevant object in the image, written as JSON. The location can be image-space or 3D-space bounding boxes normalized to $[0, 100]$, grid

cells or slots for discrete games like anagram and sokoban. This representation is visually grounded but not pixel-based. It is related to grounded reasoning with bounding boxes [28], but our boxes describe the imagined next state rather than regions of the input image. The boxes encode where objects are but not their full state (e.g., a rotated jigsaw piece keeps its box), and use 57 tokens per state on average. See Appendix F for illustrations of the bounding boxes for each game.

3.5 Implementation Details

The visual prediction head and Visual CoT both use discrete visual tokens to represent the world state. We use FSQ [70] for the visual supervision head and LlamaGen [69] for the visual tokens in the chain of thought (see ablation in Appendix H.4). The visual supervision loss weight λ (Section 3.3) is 0.3. The chain-of-thought loss weight w (Section 3.4) is 0.05 and 0.1 for Visual CoT 64×64 and 128×128, and 0.3 for State CoT and Boxes CoT. The visual prediction head follows an architecture based on masked autoencoders. All methods fine-tune the same model, Qwen3.5 9B (see Appendix H.1 for other models). We train each model on one node with 8 B200 GPUs for up to 72 hours. Each model is evaluated with greedy autoregressive decoding using vLLM [71]. See Appendix C for more details.

4 Experiments

The goals of our experiments are a) to understand the extent to which mental images emerge in VLMs, b) to assess when they are helpful for visual reasoning, and c) to demonstrate how to sharpen them during learning. We present results across a variety of spatial puzzles, and compare against baselines.

4.1 Puzzle Suite

We evaluate on nine puzzle types, three with certificate variants, totaling twelve tasks.

- **Spatial rotation [6, 7, 13, 17, 72]:** 3D Shape Matching requires rotating a 3D polycube (7–11 cubes) around three axes at 15° increments to match a target orientation; 2D Shape Matching is its planar counterpart, requiring rotation of a 2D polyomino into alignment.
- **Reconstruction from pieces [15, 16, 40, 59, 60, 73]:** Jigsaw requires placing and rotating puzzle pieces (90° CW/CCW) to reconstruct an image on a 2×2 to 3×3 grid; Tangram requires tiling a target silhouette with the seven classical tangram pieces.
- **Sliding and pushing [40, 61–63, 74–78]:** Sokoban requires pushing boxes onto goal cells with four-directional moves (boxes can only be pushed, not pulled); Bloxorz requires rolling a $1 \times 1 \times 2$ block to a goal cell, where some tiles are fragile and break after the block rolls off; Rush Hour requires sliding vehicles on a 6×6 grid to free the target car to the exit.
- **Text [64, 65]:** Anagram requires unscrambling letters (6–10 characters, at least 5 swaps from the solution) into a valid word; Character Recognition requires rotating scattered glyph fragments into labeled grid slots and identifying the assembled character.

We include certificate variants of tangram and shape matching, where half of the task instantiations are unsolvable and the model must answer 'yes' or 'no' after optionally trying to construct the solution. For shape matching, we denote the certificate variants as 3D and 2D mental rotation. Each game is parameterized to control difficulty. We ensure train / test separation by holding out 200 puzzle instances in each puzzle for evaluation. Please see Table 2 in the appendix for the task prompts.

4.2 Emergence of Mental Imagery

We evaluate models on the puzzle suite, comparing their success rates in Fig. 5. While the off-the-shelf Qwen3.5 model achieves very low performance (13% average success rate), the base behavior-cloning policy (action supervision, Section 3.2) achieves 83% average success rate. This is notable and even slightly surprising because the model is trained with open-loop behavior cloning and teacher forcing. During learning, the model needs to plan all the actions to solve the game from a single image, sometimes over a long horizon (20+ steps) – a situation where errors can compound. What is the model learning in order to achieve this?

We train a visual decoder on the frozen behavior-cloned LLM, which allows us to visualize and inspect what is encoded at each step in the autoregressive rollout. Although no gradients flow from the visual head to the LLM, and there is no visual supervision on the LLM, we are able to reconstruct

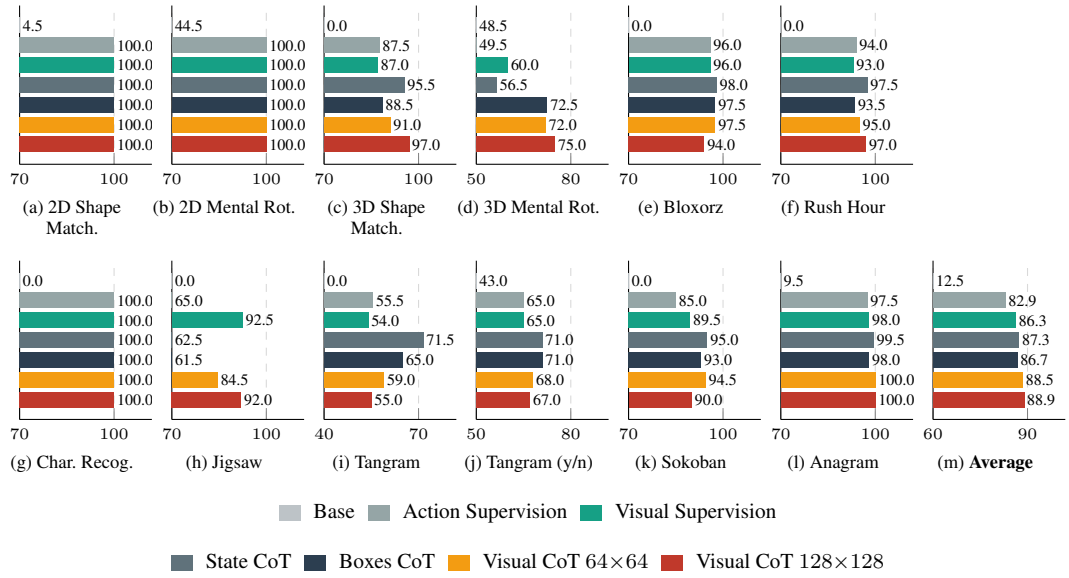


Figure 5: **Solve rates on visual reasoning tasks.** Performance of the stock Qwen3.5 ('Base', without any training on our part), the baseline behavior cloning policy ('Action Supervision'), and various forms of supervision. Visual CoT performs best across the puzzle suite.

significant detail about the intermediate states during the LLM rollout. Fig. 2 visualizes two examples, where although the colors are off, the geometry of the tangram grid is reproduced well (top) and the correct shape emerges as the VLM imagines piecing the character together (bottom). Furthermore, we plot the correlation between solve rate and mental image quality in Fig. 6. We can see that action supervision elicits higher-fidelity mental imagery.

This raises a question: if the model already encodes partial visual state in its representations, can we improve performance by explicitly encouraging this behavior? The imperfect quality of the reconstructions suggests that the emergent encoding, while useful, leaves room for sharper internal representations. Fig. 6 shows that, for games with initially low solve rate, removing the stop gradient on the visual probe, and thus explicitly supervising the internal representations to be predictive of the visual state, can lead to dramatic gains. This can be seen in the sharp pivot to the right in Fig. 6 for 3D mental rotation, jigsaw, and sokoban, where the model's solve rate improves with the fidelity of the mental images. Anagram is an exception, where the base model has near-zero solve rates but better reconstructions than post-trained models. Upon inspection, we found that the base model produces less informative reconstructions that are quantitatively better since the image contains very few foreground pixels (see Appendix H.5 for more detail).

4.3 Game Playing and Question Answering

We compare six methods that differ in how intermediate state information is incorporated during training, with the stock Qwen3.5 9B model ('Base') as a reference: action-only supervision from Section 3.2; visual supervision from Section 3.3, which further shapes the internal representations to be predictive of the world state; and the four next-state chain-of-thought variants from Section 3.4, namely Visual CoT 64×64 and 128×128, which generate 64×64 or 128×128 images of the state after each action, State CoT, which writes a JSON description of the state, and Boxes CoT, which writes the locations of the objects of interest. All methods fine-tune the same Qwen3.5 9B model.

Visual supervision refines mental imagery. Mental imagery emerges naturally without direct visual supervision, and action supervision alone already solves at least 90% of the puzzle instances in six of the twelve puzzles (2D shape matching, 2D mental rotation, character recognition, anagram, bloxorz, and rush hour; Fig. 5). Further shaping this ability through visual supervision improves the fidelity of mental images (Fig. 6) and improves the average solve rate by 3.4 percentage points (Fig. 5).

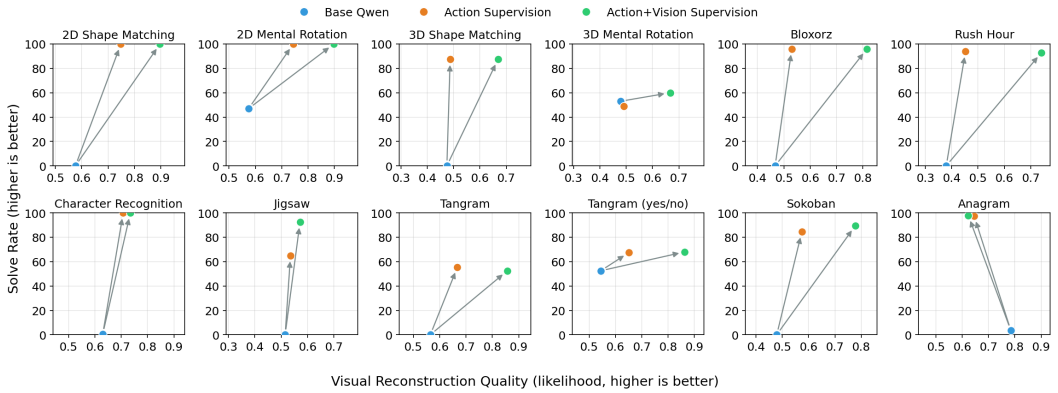


Figure 6: **Solve rate versus mental image quality.** We train visual decoders on the base Qwen3.5 9B model (blue) and our post-trained models (orange, green). We plot their success rates (vertical axis) and mental image fidelity (horizontal). When an arrow points toward the top right, higher visual fidelity is accompanied by a higher success rate – a trend we see for ten out of the twelve games.

Table 1: **Token-replacement test.** Average solve rate of the Visual CoT 128×128 model when its generated visual tokens are replaced at inference time. The token count, positions, and sequence length are unchanged. Parentheses show the change in points relative to the generated tokens.

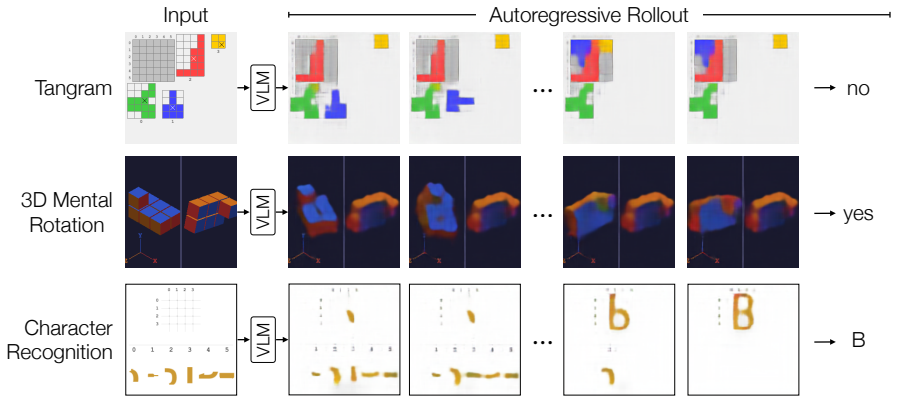
Visual token content	Generated	Ground truth	Black image	Uniform random	Random earlier frame	Initial frame
Solve rate (%)	88.9	90.5 (+1.6)	64.3 (−24.6)	59.2 (−29.7)	56.5 (−32.4)	51.1 (−37.8)

Visual CoT is effective. As shown in Fig. 5, explicitly generating visual tokens achieves an average solve rate of 88.9% (Visual CoT 128×128). It outperforms State CoT, Boxes CoT, visual supervision, and action supervision by 1.6, 2.2, 2.6, and 6.0 percentage points on average, respectively. It provides the most benefit on puzzles where the visual state is not easily expressed in language. Compared to action supervision, it improves the solve rate on 3D mental rotation from 49.5% to 75.0% (+25.5 points), on jigsaw from 65.0% to 92.0% (+27.0 points), and on 3D shape matching from 87.5% to 97.0% (+9.5 points). See Fig. 7 for illustrations of the visual tokens generated, and Fig. 27 in the appendix for a qualitative analysis of the attention weights over the generated visual tokens.

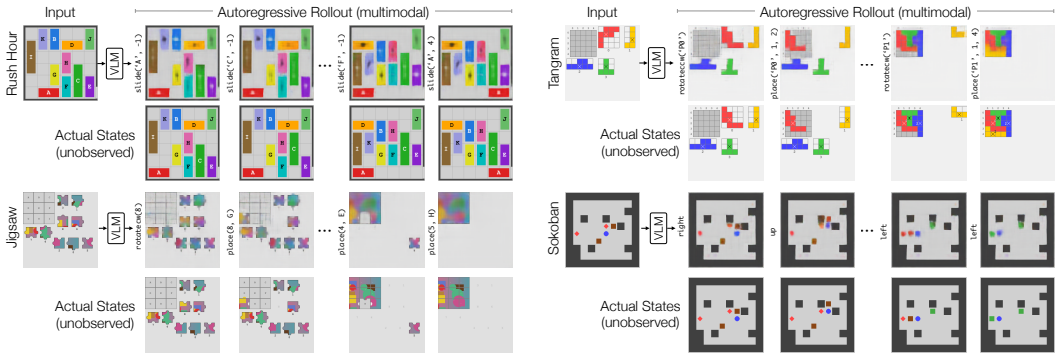
The model uses the content of its visual tokens. To test whether the gains come from the content of the generated visual tokens rather than from the additional context or a larger "thinking" budget, we perform the token-replacement test of Zhang et al. [79]. During rollouts of the Visual CoT 128×128 model, we replace the content of each generated visual token block at inference time while holding everything else fixed (the number of tokens, their positions, and the sequence length). We replace the generated tokens with the ground-truth visual tokens of the true state, the tokens of a black image, uniformly random tokens, a randomly selected earlier frame, or the initial frame. As shown in Table 1, corrupting the visual tokens reduces the average solve rate by 24 to 38 points, indicating that the gains do not come from the added sequence length alone. Moreover, the model's own generated tokens perform only 1.6 points below the ground-truth tokens (88.9% vs. 90.5%), so the model's imagery is nearly as useful for solving the puzzles as the true visual state.

Visual state expressed in text is effective. State CoT and Boxes CoT improve over action supervision on average (87.3% and 86.7% vs. 82.9%). They are especially strong on puzzles whose state is clearly discretizable (tangram, sokoban, bloxorz, and rush hour), where State CoT is the best method on all four; on tangram, it reaches 71.5%, against 55.5% for action supervision. However, both underperform where the visual state is not easily expressed in text: on jigsaw, they reach 62.5% and 61.5%, against 92.0% for Visual CoT 128×128. Boxes CoT is the exception on 3D mental rotation (72.5%, vs. 56.5% for State CoT): it writes the axis-aligned 3D bounding boxes of the current and target shapes (Fig. 23), and how the box of the current shape changes with each rotation largely conveys the shape transformation and gives cues about whether the two shapes match.

Compact tokenization is effective. The two resolutions for visual tokens produce complementary per-puzzle profiles. The 64×64 variant, with only 16 tokens per state, is better on long-horizon puzzles like sokoban, tangram, and bloxorz, while the 128×128 variant with 64 tokens per state is



(a) QA puzzles



(b) Gameplay

Figure 7: **Visual CoT rollouts.** The visual thinking tokens that Visual CoT 128×128 generates after each action, decoded as images. (a) On QA puzzles, the model forms mental images of the intermediate states to decide whether a puzzle is solvable, whether two shapes match, or which character is assembled. (b) During gameplay, it imagines the consequences of its actions as it plans its moves (top rows), then executes them; the bottom rows show the actual states, which the model never observes.

better on puzzles like jigsaw, 3D shape matching, and 3D mental rotation, where higher resolution for intermediate states proves beneficial. Despite these differences, their overall scores are close (88.9% vs. 88.5%), suggesting that even highly compressed visual representations carry enough spatial information to improve action selection on balance. Visual CoT 64×64 (88.5%) also outperforms State CoT (87.3%) and Boxes CoT (86.7%) while writing far fewer tokens per state: 16, against 46.9 and 56.7 on average over the twelve puzzles (Tables 4 and 5 in the appendix). The compactness of the 64×64 representation is particularly notable: 16 tokens per state adds minimal sequence length overhead while still capturing sufficient spatial structure to improve planning.

Additional studies. We evaluated fine-tuning BAGEL [56], a unified understanding and generation model, but it did not reach reasonable performance (Appendix G). Our findings generalize to other models (Appendix H.1). We visualize the reconstructions of the visual decoder in Fig. 38 in the appendix. Please see the appendix for further analyses.

5 Conclusion

We studied the emergence of mental imagery in vision-language models trained to solve multi-step spatial puzzles. We showed that models trained with open-loop behavior cloning learn implicit visual representations of the evolving world state. Sharpening mental imagery via additional supervision further improved performance on spatial puzzles with no test-time overhead. Introducing chain-of-thought reasoning that incorporates visual representations increased the model's success even more. We hope that this work contributes to a better understanding of spatial cognition in frontier models.

References

[1] Roger N Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972), 1971.

[2] Priti Shah and Akira Miyake. *The Cambridge Handbook of Visuospatial Thinking*. Cambridge University Press, 2005.

[3] Fred W. Mast and Lutz Jäncke. *Spatial Processing in Navigation, Imagery and Perception*. Springer, 2007.

[4] Valérie Gyselinck and Francesca Pazzaglia. *From Mental Imagery to Spatial Cognition and Language*. Psychology Press, 2012.

[5] Bence Nanay. *Mental Imagery: Philosophy, Psychology, Neuroscience*. Oxford University Press, 2023.

[6] Santhosh Kumar Ramakrishnan, Erik Wijmans, Philipp Krähenbühl, and Vladlen Koltun. Does spatial cognition emerge in frontier models? In *ICLR*, 2025.

[7] Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. OmniSpatial: Towards comprehensive spatial reasoning benchmark for vision language models. In *ICLR*, 2026.

[8] Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Y. Charles, Yiping Bao, Yuantao Fan, Guopeng Li, Haiyang Shen, Xuanzhong Chen, Wendong Xu, Shuzheng Si, Zefan Cai, Wenhao Chai, Ziqi Huang, Fangfu Liu, Tianyu Liu, Baobao Chang, Xiaobo Hu, Kaiyuan Chen, Yixin Ren, Yang Liu, Yuan Gong, and Kuan Li. BabyVision: Visual reasoning beyond language. *arXiv:2601.06521*, 2026.

[9] Chengzu Li, Wenshan Wu, Huanyu Zhang, Qingtao Li, Zeyu Gao, Yan Xia, José Hernández-Orallo, Ivan Vulic, and Furu Wei. 11Plus-Bench: Demystifying multimodal LLM spatial reasoning with cognitive-inspired analysis. *arXiv:2508.20068*, 2025.

[10] Neel Nanda. Actually, Othello-GPT has a linear emergent world model, Mar 2023. URL `https://neelnanda.io/mechanistic-interpretability/othello`.

[11] Charles Jin and Martin C. Rinard. Emergent representations of program semantics in language models trained on programs. In *ICML*, 2024.

[12] Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *ICLR*, 2023.

[13] Siting Wang, Minnan Pei, Luoyang Sun, Cheng Deng, Yuchen Li, Kun Shao, Zheng Tian, Haifeng Zhang, and Jun Wang. SpatialViz-Bench: A cognitively-grounded benchmark for diagnosing spatial visualization in MLLMs. In *ICLR*, 2026.

[14] Ilias Stogiannidis, Steven McDonagh, and Sotirios A. Tsaftaris. Mind the gap: Benchmarking spatial reasoning in vision-language models. *arXiv:2503.19707*, 2025.

[15] Linjie Li, Mahtab Bigverdi, Jiawei Gu, Zixian Ma, Yinuo Yang, Ziang Li, Yejin Choi, and Ranjay Krishna. Unfolding spatial cognition: Evaluating multimodal models on visual simulations. In *ICLR*, 2026.

[16] Yiyang Zhou, Haoqin Tu, Zijun Wang, Zeyu Wang, Niklas Muennighoff, Fan Nie, Chaorui Deng, Shen Yan, Haoqi Fan, Yejin Choi, James Zou, Cihang Xie, Huaxiu Yao, and Qinghao Ye. When visualizing is the first step to reasoning: MIRA, a benchmark for visual chain-of-thought. In *CVPR*, 2026.

[17] Tyler Bonnen, Stephanie Fu, Yutong Bai, Thomas P. O'Connell, Yoni Friedman, Nancy Kanwisher, Josh Tenenbaum, and Alexei A. Efros. Evaluating multiview object consistency in humans and image models. In *NeurIPS*, 2024.

[18] Nicholas Budny, Kia Ghods, Declan Campbell, Raja Marjieh, Amogh Joshi, Sreejan Kumar, Jonathan D. Cohen, Taylor W. Webb, and Thomas L. Griffiths. Visual serial processing deficits explain divergences in human and VLM reasoning. *arXiv:2509.25142*, 2025.

[19] Tiedong Liu and Wee Sun Lee. Can vision-language models solve the shell game? *arXiv:2603.08436*, 2026.

[20] Stephanie Fu, tyler bonnen, Devin Guillory, and Trevor Darrell. Hidden in plain sight: VLMs overlook their visual representations. In *Conference on Language Modeling (COLM)*, 2025.

[21] Mohammad Asadi, Jack W. O'Sullivan, Fang Cao, Tahoura Nedaee, Kamyar Rajabalifardi, Fei-Fei Li, Ehsan Adeli-Mosabbeb, and Euan A. Ashley. MIRAGE: The illusion of visual understanding. *arXiv:2603.21687*, 2026.

[22] Aryo Lotfi, Enrico Fini, Samy Bengio, Moin Nabi, and Emmanuel Abbe. Chain-of-sketch: Enabling global visual reasoning. *arXiv:2410.08165*, 2024.

[23] Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. In *ICCV*, 2023.

[24] Sachit Menon, Richard S. Zemel, and Carl Vondrick. Whiteboard-of-thought: Thinking step-by-step across modalities. In *EMNLP*, 2024.

[25] Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *NeurIPS*, 2024.

[26] Wenshan Wu, Shaoguang Mao, Yadong Zhang, Yan Xia, Li Dong, Lei Cui, and Furu Wei. Mind's eye of LLMs: Visualization-of-thought elicits spatial reasoning in large language models. In *NeurIPS*, 2024.

[27] 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 *CVPR*, 2025.

[28] Yue Fan, Xuehai He, Diji Yang, Kaizhi Zheng, Ching-Chen Kuo, Yuting Zheng, Xinze Guan, and Xin Wang. GRIT: Teaching MLLMs to think with images. In *NeurIPS*, 2025.

[29] Arijit Ray, Ahmed Abdelkader, Chengzhi Mao, Bryan A. Plummer, Kate Saenko, Ranjay Krishna, Leonidas J. Guibas, and Wen-Sheng Chu. Mull-tokens: Modality-agnostic latent thinking. *arXiv:2512.10941*, 2025.

[30] Yiming Qin, Bomin Wei, Jiaxin Ge, Konstantinos Kallidromitis, Stephanie Fu, Trevor Darrell, and Xudong Wang. Chain-of-visual-thought: Teaching VLMs to see and think better with continuous visual tokens. In *ECCV*, 2026.

[31] Xu Liu, Yongheng Zhang, Qiguang Chen, Yao Li, Sheng Wang, and Libo Qin. Let's think with images efficiently! an interleaved-modal chain-of-thought reasoning framework with dynamic and precise visual thoughts. In *AAAI*, 2026.

[32] Chengzhi Liu, Yuzhe Yang, Yue Fan, Qingyue Wei, Sheng Liu, and Xin Eric Wang. Reasoning within the mind: Dynamic multimodal interleaving in latent space. *arXiv:2512.12623*, 2025.

[33] Luozheng Qin, Jia Gong, Yuqing Sun, Tianjiao Li, Haoyu Pan, Mengping Yang, Xiaomeng Yang, Chao Qu, Zhiyu Tan, and Hao Li. Uni-CoT: Towards unified chain-of-thought reasoning across text and vision. In *ICLR*, 2026.

[34] Lei Zhang, Junjiao Tian, Zhipeng Fan, Kunpeng Li, Jialiang Wang, Weifeng Chen, Markos Georgopoulos, Felix Juefei-Xu, Manling Li, Julian J. McAuley, and Zecheng He. Think in strokes, not pixels: Process-driven image generation via interleaved reasoning. In *ECCV*, 2026.

[35] Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulic, and Furu Wei. Imagine while reasoning in space: Multimodal visualization-of-thought. In *ICML*, 2025.

[36] 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 *CVPR*, 2025.

[37] Junbang Liang, Ruoshi Liu, Ege Ozguroglu, Sruthi Sudhakar, Achal Dave, Pavel Tokmakov, Shuran Song, and Carl Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation. In *CoRL*, 2024.

[38] Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv:2508.00795*, 2025.

[39] Shuhan Tan, Kashyap Chitta, Yuxiao Chen, Ran Tian, Yurong You, Yan Wang, Wenjie Luo, Yulong Cao, Philipp Krähenbühl, Marco Pavone, and Boris Ivanovic. Latent chain-of-thought world modeling for end-to-end driving. *arXiv:2512.10226*, 2025.

[40] Jana Zeller, Thaddäus Wiedemer, Fanfei Li, Thomas Klein, Prasanna Mayilvahanan, Matthias Bethge, Felix A. Wichmann, Ryan Cotterell, and Wieland Brendel. MentisOculi: Revealing the limits of reasoning with mental imagery. *arXiv:2602.02465*, 2026.

[41] Zihui Cheng, Qiguang Chen, Xiao Xu, Jiaqi Wang, Weiyun Wang, Hao Fei, Yidong Wang, Alex Jinpeng Wang, Zhi Chen, Wanxiang Che, and Libo Qin. Visual thoughts: A unified perspective of understanding multimodal chain-of-thought. In *NeurIPS*, 2025.

[42] Wes Gurnee and Max Tegmark. Language models represent space and time. In *ICLR*, 2024.

[43] Benlin Liu, Amita Kamath, Madeleine Grunde-McLaughlin, Winson Han, and Ranjay Krishna. Visual representations inside the language model. *arXiv:2510.04819*, 2025.

[44] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *NeurIPS*, 2018.

[45] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *arXiv:2301.04104*, 2023.

[46] Sruthi Sudhakar, Ruoshi Liu, Basile Van Hoorick, Carl Vondrick, and Richard S. Zemel. Controlling the world by sleight of hand. In *ECCV*, 2024.

[47] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv:2603.19312*, 2026.

[48] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv:2509.20328*, 2025.

[49] Ruisi Wang, Zhongang Cai, Fanyi Pu, Junxiang Xu, Wanqi Yin, Maijunxian Wang, Ran Ji, Chenyang Gu, Bo Li, Ziqi Huang, Hokin Deng, Dahua Lin, Ziwei Liu, and Lei Yang. Demystifying video reasoning. *arXiv:2603.16870*, 2026.

[50] Xianjin Wu, Dingkang Liang, Tianrui Feng, Kui Xia, Yumeng Zhang, Xiaofan Li, Xiao Tan, and Xiang Bai. Generation models know space: Unleashing implicit 3D priors for scene understanding. In *ECCV*, 2026.

[51] Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi Fan. NitroGen: An open foundation model for generalist gaming agents. In *CVPR*, 2026.

[52] Zefan Cai, Haoyi Qiu, Tianyi Ma, Haozhe Zhao, Gengze Zhou, Kung-Hsiang Huang, Parisa Kordjamshidi, Minjia Zhang, Wen Xiao, Jiuxiang Gu, Nanyun Peng, and Junjie Hu. MMGR: Multi-modal generative reasoning. *arXiv:2512.14691*, 2025.

[53] Ziyu Guo, Xinyan Chen, Renrui Zhang, Ruichuan An, Yu Qi, Dongzhi Jiang, Xiangtai Li, Manyuan Zhang, Hongsheng Li, and Pheng-Ann Heng. Are video models ready as zero-shot reasoners? An empirical study with the MME-CoF benchmark. *arXiv:2510.26802*, 2025.

[54] Maijunxian Wang, Ruisi Wang, Juyi Lin, Ran Ji, Thaddäus Wiedemer, Qingying Gao, Dezhi Luo, Yaoyao Qian, Lianyu Huang, Zelong Hong, Jiahui Ge, Qianli Ma, Hang He, Yifan Zhou, Lingzi Guo, Lantao Mei, Jiachen Li, Hanwen Xing, Tianqi Zhao, Fengyuan Yu, Weihang Xiao, Yizheng Jiao, Jianheng Hou, Danyang Zhang, Pengcheng Xu, Boyang Zhong, Zehong Zhao, Gaoyun Fang, John Kitaoka, Yile Xu, Hua Xu, Kenton Blacutt, Tin Nguyen, Siyuan Song, Haoran Sun, Shaoyue Wen, Linyang He, Runming Wang, Yanzhi Wang, Mengyue Yang, Ziqiao Ma, Raphaël Millière, Freda Shi, Nuno Vasconcelos, Daniel Khashabi, Alan L. Yuille, Yilun Du, Ziming Liu, Bo Li, Dahua Lin, Ziwei Liu, Vikash Kumar, Yijiang Li, Lei Yang, Zhongang Cai, and Hokin Deng. A very big video reasoning suite. *arXiv:2602.20159*, 2026.

[55] Jonathan Heek, Emiel Hoogeboom, Thomas Mensink, and Tim Salimans. Unified latents (UL): How to train your latents. *arXiv:2602.17270*, 2026.

[56] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Shi Guang, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv:2505.14683*, 2025.

[57] Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain-of-thought prompting for large multimodal models. In *CVPR*, 2024.

[58] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[59] Zesen Lyu, Dandan Zhang, Wei Ye, Fangdi Li, Zhihang Jiang, and Yao Yang. Jigsaw-puzzles: From seeing to understanding to reasoning in vision-language models. In *EMNLP*, 2025.

[60] Yikun Zong and Cheston Tan. TangramSR: Can vision-language models reason in continuous geometric space? *arXiv:2602.05570*, 2026.

[61] Tom C. van der Zanden and Hans L. Bodlaender. PSPACE-completeness of Bloxorz and of games with 2-buttons. In *International Conference on Algorithms and Complexity*, 2015.

[62] Ami Hauptman, Achiya Elyasaf, Moshe Sipper, and Assaf Karmon. GP-rush: using genetic programming to evolve solvers for the rush hour puzzle. In *Genetic and Evolutionary Computation Conference*, 2009.

[63] Zhao Yang, Mike Preuss, and Aske Plaat. Transfer learning and curriculum learning in Sokoban. In *Benelux Conference on Artificial Intelligence*, 2021.

[64] Jingmiao Zhao and Carolyn Jane Anderson. Solving and generating NPR Sunday puzzles with large language models. In *International Conference on Computational Creativity*, 2023.

[65] Andrew Shin and Kunitake Kaneko. Large language models lack understanding of character composition of words. *arXiv:2405.11357*, 2024.

[66] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[67] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[68] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

[69] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv:2406.06525*, 2024.

[70] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *ICLR*, 2024.

[71] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Symposium on Operating Systems Principles*, 2023.

[72] Yuyou Zhang, Radu Corcodel, Chiori Hori, Anoop Cherian, and Ding Zhao. SpinBench: Perspective and rotation as a lens on spatial reasoning in VLMs. In *ICLR*, 2026.

[73] Gur Elkin, Ofir Itzhak Shahar, and Ohad Ben-Shahar. Seq2Seq models reconstruct visual jigsaw puzzles without seeing them. *arXiv:2511.06315*, 2025.

[74] Yaron Shoham and Gal Elidan. Solving Sokoban with forward-backward reinforcement learning. In *International Symposium on Combinatorial Search*, 2021.

[75] Mohammad Taufeeque, Philip Quirke, Maximilian Li, Chris Cundy, Aaron David Tucker, Adam Gleave, and Adrià Garriga-Alonso. Planning in a recurrent neural network that plays Sokoban. In *ICLR*, 2025.

[76] Jialong Wu, Xiaoying Zhang, Hongyi Yuan, Xiangcheng Zhang, Tianhao Huang, Changjing He, Chaoyi Deng, Renrui Zhang, Youbin Wu, and Mingsheng Long. Visual generation unlocks human-like reasoning through multimodal world models. *arXiv:2601.19834*, 2026.

[77] Tahani Q. Alhassan, Shefaa S. Omar, and Lamiaa A. Elrefaei. Game of Bloxorz solving agent using informed and uninformed search strategies. *Procedia Computer Science*, 2019.

[78] Lorenzo Cian, Talissa Dreossi, and Agostino Dovier. Modeling and solving the rush hour puzzle. In *Italian Conference on Computational Logic*, 2022.

[79] Tianyi Zhang, Mahtab Bigverdi, and Ranjay Krishna. Ablate-to-validate: Are vision-language models really using continuous thought tokens? *arXiv:2605.21642*, 2026.

[80] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Appendix Contents

A Limitations **17**

B Broader impacts **17**

C Implementation Details **17**

D Dataset splits **19**

E State Chain-of-Thought Baseline Details **20**

F Boxes Chain-of-Thought Baseline Details **24**

G BAGEL Baseline Details **28**

H Additional Results **29**

H.1 Generalization across model sizes and families 29

H.2 Multi-seed results 32

H.3 Solve rate by puzzle difficulty 33

H.4 Comparing LlamaGen vs. FSQ tokenizers for visual token generation 33

H.5 Anagram Analysis 34

A Limitations

Current experiments are exclusively synthetic, in digital environments with rendered graphics. Our current approach of training games with expert solvers is unlikely to be feasible in the real world. Our results mostly focus on in-domain generalization (within a game or puzzle). Scaling this to out-of-domain games likely requires much larger scale of training environments and compute.

B Broader impacts

The paper provides early evidence that mental imagery can emerge in VLMs when they are trained to solve certain tasks in interactive environments. This could highlight a path towards training multimodal models with some cognitive abilities that have hitherto not been integrated in frontier artificial systems and robotics. Potential negative impacts are far in the future. As mentioned in the limitations, current models struggle to generalize beyond their training environment and are unlikely to negatively impact the real world any time soon.

C Implementation Details

Visual tokens. We use discrete visual tokens to represent images in visual supervision and in Visual CoT. We experiment with two encodings. Firstly, we use an FSQ autoencoder with $D = 6$ dimensions with $L = 5$ levels each, yielding a codebook of $5^6 = 15{,}625$ entries. It maps 256×256 images to a 16×16 grid of codes via a convolutional encoder with four stride-2 stages. Secondly, we also explore LlamaGen VQ-VAE [69] which performs $16\times$ spatial downsampling and a codebook of 16,384 entries. We found that LlamaGen tokens worked best for autoregressive prediction and FSQ worked best for visual supervision. See Section H.4 for supporting experiments.

Visual prediction head. We follow an architecture based on masked autoencoders [66]. The decoder operates at dimension $d = 256$ with 8 attention heads. Hidden states from all N_{LLM} LLM layers at the visual prediction position are projected from the LLM hidden dimension H (for Qwen3.5 9B, $H = 4096$ and $N_{\text{LLM}} = 32$) down to d, yielding one memory token per LLM layer that the decoder attends to. The self-attention stack consists of 4 pre-norm transformer layers with GELU activations, feed-forward dimension $4d = 1024$, and no dropout. The code prediction head is a single linear layer mapping $d \to D \times L = 30$ logits per patch. In total, the visual head has approximately 36.9M trainable parameters for the 9B model. The frozen FSQ autoencoder adds no trainable parameters.

Base model. All methods fine-tune the same base model, Qwen3.5 9B, which is cast to `bfloat16` with FlashAttention-2. Training uses FSDP with full sharding and activation checkpointing.

Optimization. All methods use AdamW [80] with $\beta = (0.9, 0.999)$, weight decay of 0.01, gradient clipping at 1.0, and a cosine learning rate schedule with linear warmup and minimum LR ratio 0.1, for 25,000 optimizer steps. The base model and visual head variants use a learning rate of 1×10^{-5}; the chain-of-thought variants (Visual CoT, State CoT, and Boxes CoT) use 3×10^{-5}, which for Visual CoT accommodates learning the additional token embeddings. For the visual head, its parameters use a $100\times$ learning rate multiplier. When an episode contains more than 32 action boundaries, we randomly subsample visual frames to bound the per-step cost of the visual head.

Evaluation. An episode is capped at $\lceil 1.5 \times$ optimal solution length$\rceil$ actions, and an episode that reaches the cap without solving the puzzle counts as a failure. Invalid or malformed actions leave the state unchanged but count toward the cap. In the yes/no puzzles, the episode ends at the model's answer and counts as solved if the answer is correct.

Embedding initialization for visual tokens. As discussed in Section 3.4, we add $|\mathcal{C}| = 16,384$ VQ-VAE tokens from LlamaGen to the model vocabulary. We observed that initializing the vocabulary embeddings to Gaussian noise resulted in slow convergence, and would result in the model being stuck at nearly 0% solve rates on most games. We identified the root cause of this was the visual tokens in the vocabulary were disconnected from the model's visual embedding space. Specifically, whenever the model encountered visual tokens in its context, it would not be represented in the same way as an image patch would by the visual encoder, but rather as a learnable and randomly initialized embedding. To fix this, we collected a dataset of images sampled from the puzzles via the expert solver. On each image, we obtained the discrete token representation using the VQ-VAE as well as

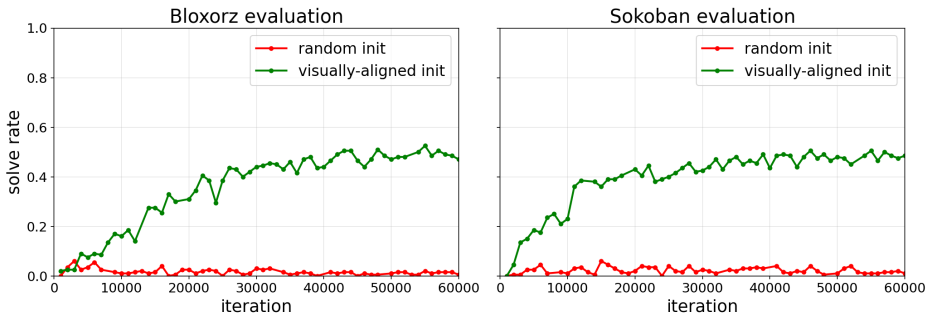


Figure 8: **Embedding initialization for visual tokens.** We compare the solve rates versus training iterations for two embedding initialization schemes when visual tokens are added to the model's vocabulary. When the embeddings for the tokens are randomly initialized, the model fails to train on a representative set of sokoban and bloxorz games. After our proposed initialization by aligning the VQ-VAE's visual tokens with the Qwen3.5 VLM's visual encoder embeddings, the model starts to learn puzzle solving rapidly.

Game	**Task Prompt**
Tangram	Complete the tangram. Fill the grey silhouette using the colored pieces.
Tangram (certificate)	Determine if the pieces can fill the silhouette. Say 'yes' if they can, 'no' if they can't.
Shape Matching (3D)	Rotate the left shape to match the right shape.
Mental Rotation (3D)	Determine if the left and right shapes are the same. Rotate to check, then say 'yes' if they match or 'no' if they don't.
Shape Matching (2D)	Rotate the left 2D shape to match the right shape.
Mental Rotation (2D)	Determine if the left and right 2D shapes are the same (possibly rotated). Rotate to check, then say 'yes' if they match or 'no' if they don't.
Jigsaw	Reconstruct the original image by rotating each piece to the correct orientation and placing it at its correct grid position.
Anagram	The letters of a word have been scrambled. Swap letters to unscramble the word, then submit with `identify(word)`.
Character Recognition	A character has been broken into pieces that are scattered and rotated. Rotate each numbered piece to the correct orientation and place it into the matching lettered slot in the assembly area. Once all pieces are correctly assembled, identify the character.
Sokoban	Solve this Sokoban puzzle. Push the brown boxes onto the red diamonds.
Bloxorz	Solve this Bloxorz puzzle. Roll the block so it stands on the red goal.†
Rush Hour	Slide vehicles to free the red car (A) and move it to the right edge exit.

Table 2: **Puzzle task prompts.** Task prompts shown to the model for each game in our puzzle suite. † When fragile tiles are present, the prompt appends: "Striped tiles are fragile and will break after the block leaves them."

the visual embeddings from the Qwen3.5 visual encoder. Since the tokens and the embeddings are spatially aligned, we can associate Qwen3.5 visual embeddings to corresponding VQ-VAE tokens. We averaged the visual embeddings corresponding to each VQ-VAE token over the collection of images, and used the average visual embedding per token to initialize the corresponding entry in the vocabulary embedding matrix. This ensured that whenever the Qwen3.5 LLM encountered a visual token, it would be similar to the visual embeddings it would see from an image, and made a massive difference to learning. See Fig. 8 for a comparison.

Text instructions for puzzles. We display the text prompts t for each puzzle (see Section 4.1) in Table 2.

D Dataset splits

Every puzzle instance is sampled procedurally by our Rust game engines and labeled by an exact optimal solver, so we can generate a near-unlimited stream of non-repeating training data. We train on over 1M unique episodes per game (12.7M in total) and evaluate on 200 held-out episodes per game (2,400 in total). Training and test puzzles come from the same generator and difficulty parameters, so they are drawn from the same distribution. We guarantee that there is no train/test overlap by randomly generating puzzle instances with different seeds, hashing the state variables of each instance, and reserving 200 instances per puzzle type as the test games. Difficulty within a task is summarized by the optimal solution length, which ranges from 2 to 45 steps (median 8 steps) across the 2,400 test puzzles. The per-game ranges are given in Table 3.

To ensure diversity, each game exposes several generation parameters that are drawn uniformly per episode and control the difficulty of the instance. Table 3 describes how the instances of each game are generated.

Table 3: **Procedural generation of the puzzle instances.** Generation parameters are drawn uniformly per episode. Optimal solution lengths (in steps) are computed over the 200 test puzzles of each game.

		Optimal length (steps)	
Game	Instance generation	Range	Median
2D Shape Matching	An asymmetric, chiral 7-square polyomino is grown at random, colored with a random hue, and shown beside a copy rotated in 15° steps.	3–6	5
2D Mental Rotation	Same as 2D Shape Matching, except that in half the instances the second shape is mirrored as well as rotated, so no rotation can match it.	3–6	4
3D Shape Matching	A 7–11 cube polycube is grown by attaching cubes at random free faces and shown beside a copy rotated in 15° steps about each of the three axes.	4–18	9
3D Mental Rotation	Same as 3D Shape Matching, except that in half the instances the second shape is mirrored, or has a single cube added or removed, so no rotation can match it.	4–17	10
Jigsaw	A random picture (a two-color gradient with 6–14 rectangles and 4–10 circles) is cut into 4 or 9 interlocking pieces with random tab-and-socket edges, then rotated and shuffled into a tray.	5–17	8
Character Recognition	A randomly sampled character (drawn from 59 Latin and Greek letters and digits) is rendered in one of 32 valid fonts, rasterized, and cut into 6–7 random fragments that are rotated and translated.	9–19	14
Tangram	A 4×4 to 7×7 canvas is cut into 4–6 randomly shaped and sized pieces, then randomly re-oriented and re-colored.	4–11	7
Tangram (yes/no)	Same as Tangram, except that in half the instances one or two boundary cells of a single piece are moved so that no tiling exists, making the answer "no".	2–11	5
Bloxorz	A solid region covering 20–50% of an 8×8 to 12×12 board is grown by randomized flood fill from a random goal cell, with the rolling block started as far from the goal as it can reach.	10–23	15
Rush Hour	A target car and 8–14 blocking vehicles of length 2 or 3 are placed at random valid positions on a 6×6 to 8×8 board with a single exit on a random row.	4–19	8
Sokoban	Walls are scattered over roughly 20% of an 8×8 to 12×12 board, then the player, 2–5 boxes, and their goals are dropped onto random free cells.	6–45	23
Anagram	A word is drawn uniformly from a 34,569-word dictionary spanning 6–10 letters, and its letters are shuffled.	5–10	7

E State Chain-of-Thought Baseline Details

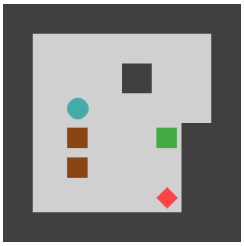
We describe the state chain-of-thought (State CoT) baseline in detail. It is named for what it writes after every action: a description of the game state. Inspired by the compositional scene-graph prompting framework of Mitra et al. [57], we replace the visual token channel with a structured JSON description of the state emitted between `<board>` and `</board>` delimiters at each action boundary. The training pipeline is identical to the Visual CoT variants: the model is teacher-forced to reproduce the ground-truth JSON state description at every step, using the decoupled loss with weight $w = 0.3$. The only difference between State CoT and Visual CoT training is what sits between the `<board>` delimiters – JSON text versus discrete visual codebook indices.

We design a per-game JSON schema that encodes the minimum information needed to reconstruct the state plus a small number of cheaply derivable quantities (counts, distances, flags). The schemas are calibrated to roughly 64 tokens per state on average (mean ≤ 65 for every game) when tokenized with the Qwen3.5 tokenizer, matching the 64 tokens per state of Visual CoT with the larger image of size 128×128. We tried longer state descriptions, but they did not work better than the current setup. For some games, fewer tokens sufficed given the simpler state description. Table 4 reports the measured token statistics across all game types, computed over 100 episodes per game along optimal solution paths. The QA variants use the same schema as their base game and have nearly identical token statistics.

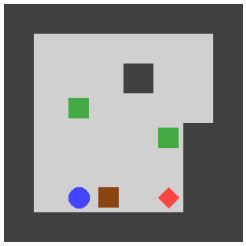
Table 4: **State CoT token counts per description.** Statistics measured over 100 episodes per game along optimal solution paths, tokenized with the Qwen3.5 tokenizer.

Game	Mean	Median	Min	Max
3D Shape Matching / Mental Rotation	64.7	65	58	71
2D Shape Matching / Mental Rotation	31.0	31	29	33
Jigsaw	59.5	65	35	80
Tangram	47.5	49	28	68
Character Recognition	39.8	39	34	49
Sokoban	42.7	41	27	64
Bloxorz	48.1	48	38	61
Rush Hour	54.7	54	44	68
Anagram	31.9	32	28	37

Conventions. Each state description is one compact JSON object. Grids are written row by row as a single string without row separators; the grid width is not part of the description (it is visible in the input image). Positions are `[column, row]` with the origin at the top-left cell. Angles are in degrees in $[0, 360)$, and a delta is the signed shortest rotation from the current to the target angle, in $[-180, 180)$. Every description ends with `moves`, the number of actions taken so far. The yes/no variants use the schema of their base game. Figures 9 to 17 describe the schema of each game and show two example states with their descriptions.

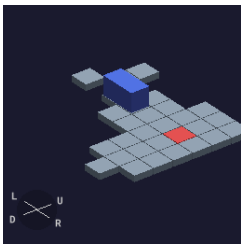


```
{"g":"########
      #......#
      #...#..#
      #.+....#
      #.O..*##
      #.O...##
      #....X##
      ########",
 "moves":9}
```

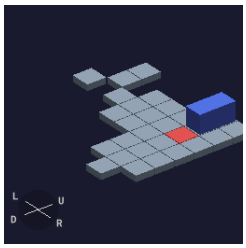


```
{"g":"########
      #......#
      #...#..#
      #.*....#
      #....*##
      #.....##
      #.@O.X##
      ########",
 "moves":18}
```

Figure 9: **State CoT for Sokoban:** `g` is the board, one character per cell: `#` (wall), `.` (floor), `@` (player), `O` (box), `X` (goal), `*` (box on goal) and `+` (player on goal). Left and right show the state after 9 and 18 actions of the same expert episode. Line breaks and indentation are added for readability, and the board string is broken into one line per row; the model writes the description as one compact line.

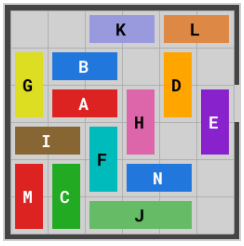



```
{"g":" .   ...
       . .....
      . =-..G.
         .....
           ...
           ...
            .
              ",
 "b":[2,2,"X"],
 "goal":[6,2],
 "broken":1,
 "distance":4,
 "moves":2}
```

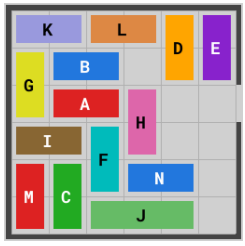




```
{"g":" .   .H.
       . ...|.
      . ....G.
         .....
           ...
           ...
            .
              ",
 "b":[6,0,"Y"],
 "goal":[6,2],
 "broken":1,
 "distance":2,
 "moves":7}
```


Figure 10: **State CoT for Bloxorz:** `g` is the board: `.` (floor), space (void), `F` (fragile tile), `G` (goal), and the block's cells: `S` (standing), `=` - (lying along x) or `H` | (lying along y). `b` is the block's first cell (its left or top cell when lying) and its orientation (`S`, `X` or `Y`); `goal` is the goal cell; `broken` is the number of fragile tiles broken so far; `distance` is the Manhattan distance from the block's first cell to the goal. Left and right show the state after 2 and 7 actions of the same expert episode. Line breaks and indentation are added for readability, and the board string is broken into one line per row; the model writes the description as one compact line.

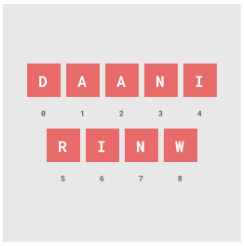



```
{"g":"..KKLL
      GBB.D.
      GAAHDE
      IIFH.E
      MCFNN.
      MCJJJ.",
 "exit_row":2,
 "exit_col":5,
 "n_vehicles":14,
 "blocks_A":["H","D","E"],
 "moves":2}
```

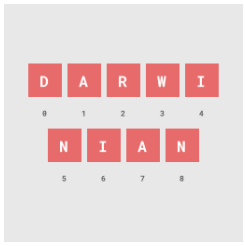




```
{"g":"KKLLDE
      GBB.DE
      GAAH..
      IIFH..
      MCFNN.
      MCJJJ.",
 "exit_row":2,
 "exit_col":5,
 "n_vehicles":14,
 "blocks_A":["H"],
 "moves":6}
```


Figure 11: **State CoT for Rush Hour:** `g` is the board, with each vehicle's letter at the cells it occupies (`A` is the red target car) and `.` for empty cells. `exit_row` and `exit_col` give the exit on the right edge; `n_vehicles` is the number of vehicles; `blocks_A` lists the vehicles between car A's front and the exit, nearest first. Left and right show the state after 2 and 6 actions of the same expert episode. Line breaks and indentation are added for readability, and the board string is broken into one line per row; the model writes the description as one compact line.

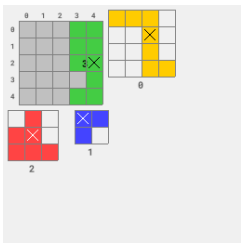



```
{"letters":"daanirinw",
 "sorted":"aadiinnrw",
 "len":9,
 "id":false,
 "n_valid":1,
 "moves":2}
```

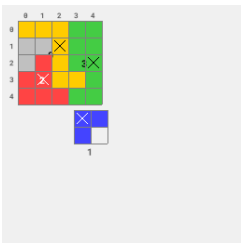




```
{"letters":"darwinian",
 "sorted":"aadiinnrw",
 "len":9,
 "id":true,
 "n_valid":1,
 "moves":6}
```


Figure 12: **State CoT for Anagram:** `letters` is the current arrangement of the tiles; `sorted` is the same letters in alphabetical order; `len` is the word length; `id` is whether an answer has been submitted with `identify`; `n_valid` is the number of dictionary words that use exactly these letters. Left and right show the state after 2 and 6 actions of the same expert episode. Line breaks are added for readability; the model writes the description as one compact line.

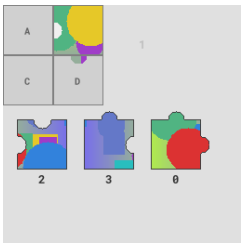



```
{"c":"TTT33
     TTT33
     TTT33
     TTTT3
     TTT33",
 "q_rot":[["P0",90],
          ["P1",0],
          ["P2",0]],
 "moves":2}
```

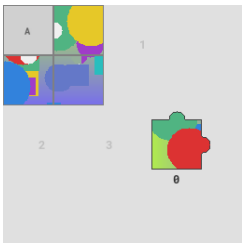




```
{"c":"00033
     TT033
     T2033
     22003
     22233",
 "q_rot":[["P1",0]],
 "moves":4}
```


Figure 13: **State CoT for Tangram:** `c` is the canvas: each cell covered by a placed piece holds the piece's number (`0` for `P0`), an uncovered cell of the silhouette is `T`, and a cell outside the silhouette is `.`. `q_rot` lists every piece still in the tray with its rotation in degrees, in tray order. The rotation of a placed piece is not listed; it is implied by the cells it covers. Left and right show the state after 2 and 4 actions of the same expert episode. Line breaks and indentation are added for readability, the board string is broken into one line per row, and long arrays are broken after a comma; the model writes the description as one compact line.

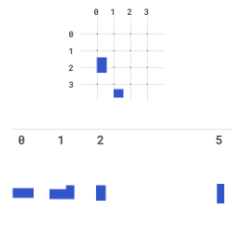



```
{"g":[null,1,null,null],
 "r":[270,0,270,90],
 "unplaced":[0,2,3],
 "n_placed":1,
 "moves":2}
```

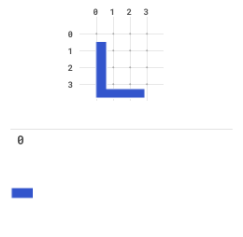




```
{"g":[null,1,2,3],
 "r":[270,0,0,0],
 "unplaced":[0],
 "n_placed":3,
 "moves":6}
```


Figure 14: **State CoT for Jigsaw:** `g` gives the piece in each grid cell, with the cells in reading order (`A`, `B`, `C`, ...) and `null` for an empty cell. `r` gives each piece's rotation in degrees (0, 90, 180 or 270), indexed by piece number, whether the piece is placed or not. `unplaced` lists the pieces in the tray, and `n_placed` is the number of placed pieces. Left and right show the state after 2 and 6 actions of the same expert episode. Line breaks are added for readability; the model writes the description as one compact line.

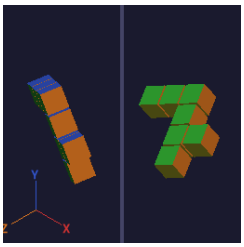



```
{"r":[3,3,2,0,0,3],
 "p":[null,null,null,13,8,
      null],
 "n_placed":2,
 "moves":4}
```

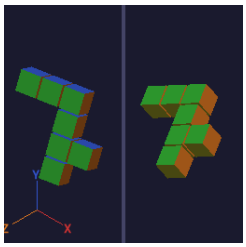




```
{"r":[3,0,0,0,0,0],
 "p":[null,12,4,13,8,14],
 "n_placed":5,
 "moves":11}
```


Figure 15: **State CoT for Character Recognition:** `r` gives each piece's rotation in quarter turns (0–3), indexed by piece number. `p` gives the slot each piece occupies in the 4×4 assembly grid, as the index $4\,\text{row} + \text{column}$, or `null` if the piece is still in the palette. `n_placed` is the number of placed pieces. Left and right show the state after 4 and 11 actions of the same expert episode. Line breaks and indentation are added for readability, and long arrays are broken after a comma; the model writes the description as one compact line.

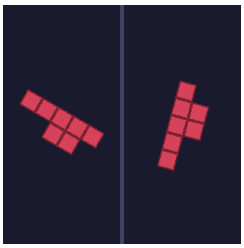



```
{"rx":0,
 "ry":60,
 "rz":30,
 "tx":75,
 "ty":150,
 "tz":105,
 "dx":75,
 "dy":90,
 "dz":75,
 "inc":15,
 "n_voxels":7,
 "moves":3}
```

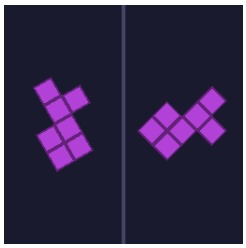




```
{"rx":60,
 "ry":120,
 "rz":60,
 "tx":75,
 "ty":150,
 "tz":105,
 "dx":15,
 "dy":30,
 "dz":45,
 "inc":15,
 "n_voxels":7,
 "moves":8}
```


Figure 16: **State CoT for 3D Mental Rotation:** `rx`, `ry`, `rz` are the current rotation of the left shape as Euler angles (the rotation is $R_z R_y R_x$); `tx`, `ty`, `tz` are the target angles that match the right shape; `dx`, `dy`, `dz` are the per-axis signed shortest deltas from current to target; `inc` is the rotation increment; `n_voxels` is the number of cubes in the shape. Left and right show the state after 3 and 8 actions of the same expert episode. Line breaks are added for readability; the model writes the description as one compact line.




```
{"r":300,
 "target":195,
 "delta":-105,
 "inc":15,
 "n_cells":7,
 "moves":2}
```


```
{"r":60,
 "target":135,
 "delta":75,
 "inc":15,
 "n_cells":7,
 "moves":2}
```


Figure 17: **State CoT for 2D Mental Rotation:** `r` is the current rotation of the left shape, `target` the rotation that matches the right shape, `delta` the signed shortest rotation from `r` to `target`, `inc` the rotation increment, and `n_cells` the number of cells in the shape. Left and right show two different puzzles, each after 2 actions. Line breaks are added for readability; the model writes the description as one compact line.

F Boxes Chain-of-Thought Baseline Details

We describe the Boxes CoT baseline, which uses an intermediate representation that is visually grounded rather than symbolic. It is related to grounded reasoning with bounding boxes [28]. However, whereas their boxes point to regions of the input image, ours predict where objects will be after each action, i.e., in the imagined future state. Boxes CoT uses identical losses to the State CoT baseline (Section E) and is comparable to the Visual CoT variants in training infrastructure (identical optimizer steps, batch sizes, learning rate, and game suite). Unlike State CoT, which emits a per-game state description, boxes CoT emits the spatial extent of every task-relevant object at each action boundary.

Box representation. At every action boundary, the model writes a JSON object that maps each task-relevant object to a short list of integers. All values are normalized to $[0, 100]$; image coordinates have their origin at the top left of the rendered image, with x to the right and y down. Unlike State CoT, there is no `moves` field. The objects and the number of values per object depend on the game: Figures 18 to 26 list them for each game and show two example states with their boxes and descriptions. The yes/no variants use the representation of their base game. The boxes describe where objects are, not their full state: they do not encode piece rotations in jigsaw (a rotated piece keeps its box), and encode rotations only indirectly, through the change in box shape, in tangram, character recognition and 2D shape matching.

Token budget. Table 5 reports the token statistics, measured the same way as Table 4. Unlike the state description, the box representation is not calibrated to a fixed budget: its length is set by the number of objects to localize. Averaged over the twelve tasks, boxes CoT uses 56.7 tokens per state against 46.9 for State CoT ($1.21\times$). It is longer where many objects must each be localized (character recognition, jigsaw, rush hour) and shorter where a box replaces a verbose symbolic encoding (3D tasks).

Table 5: **Boxes CoT token counts per description.** Statistics measured over 100 episodes per game along optimal solution paths, tokenized with the Qwen3.5 tokenizer.

Game	Mean	Median	Min	Max
3D Shape Matching / Mental Rotation	39.7	40	36	41
2D Shape Matching / Mental Rotation	28.3	28	28	30
Jigsaw	103.7	130	57	134
Tangram	68.5	62	60	95
Character Recognition	96.1	103	29	107
Sokoban	43.4	39	39	71
Bloxorz	40.9	29	28	82
Rush Hour	82.0	84	62	92
Anagram	41.3	41	25	57

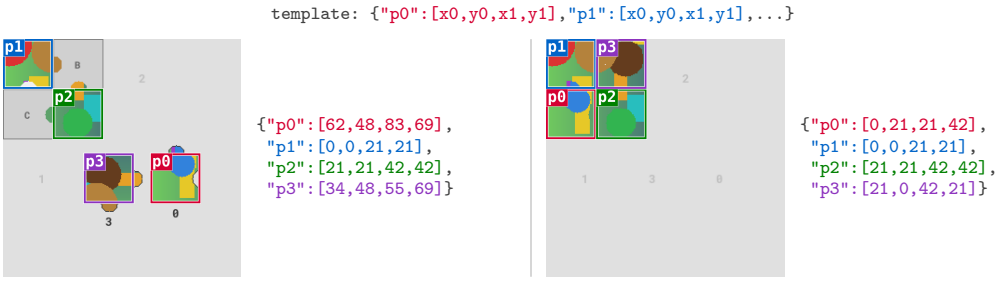


Figure 18: **Boxes CoT for Jigsaw:** One entry per piece, keyed by the number drawn on it (`p0`, `p1`, ...), with the image-space box $[x_0, y_0, x_1, y_1]$ of the piece's square (tabs excluded): its grid cell once placed, otherwise its slot in the tray. Left and right show the state after 3 and 6 actions of the same expert episode. Each image shows the box of every listed object (outline and key tag in the object's color). The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

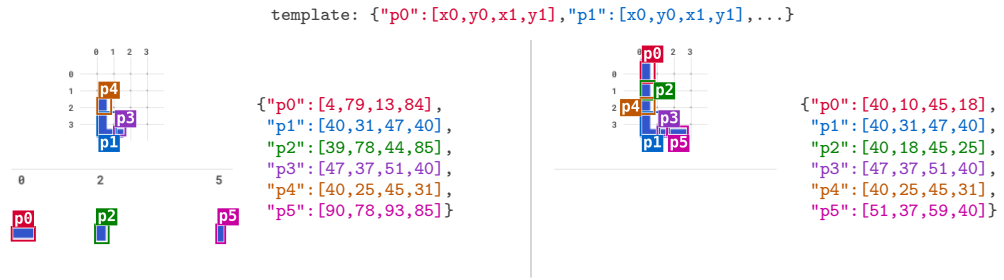


Figure 19: **Boxes CoT for Character Recognition:** One entry per piece (`p0`, `p1`, ...), with the image-space box of the piece where it is drawn: in the assembly grid (top half) once placed, otherwise in the palette (bottom half). Left and right show the state after 6 and 14 actions of the same expert episode. Each image shows the box of every listed object (outline and key tag in the object's color). The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

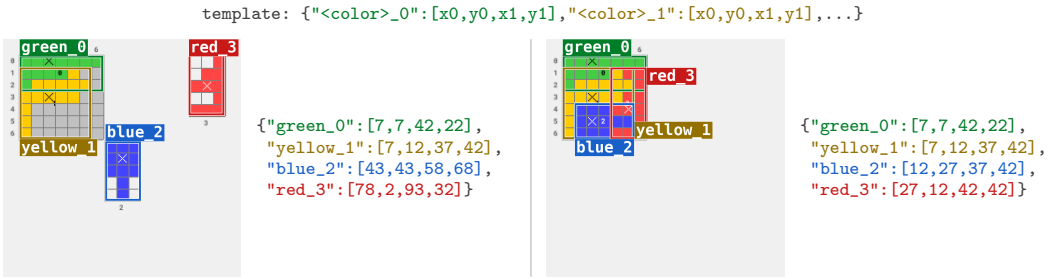


Figure 20: **Boxes CoT for Tangram:** One entry per piece, keyed by its color and number (`purple_0` is piece `P0`), with the image-space box of the piece: on the canvas once placed, otherwise in the tray. Left and right show the state after 3 and 6 actions of the same expert episode. Each image shows the box of every listed object (outline and key tag in the object's color). The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

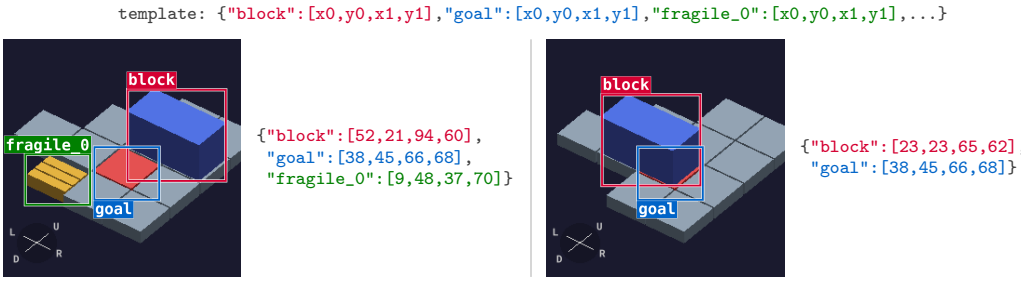


Figure 21: **Boxes CoT for Bloxorz:** The `block`, the `goal` tile and each fragile tile not yet broken (`fragile_0`, ...), each with the image-space box of its projected 3D geometry (the board is drawn isometrically). The block's box changes shape with its orientation; a broken tile is removed from the list. Left and right show the state after 2 and 9 actions of the same expert episode. Each image shows the box of every listed object (outline and key tag in the object's color); the fragile tile breaks at action 5 and leaves the list. The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

template: {"current":[x0,y0,x1,y1],"target":[x0,y0,x1,y1]}

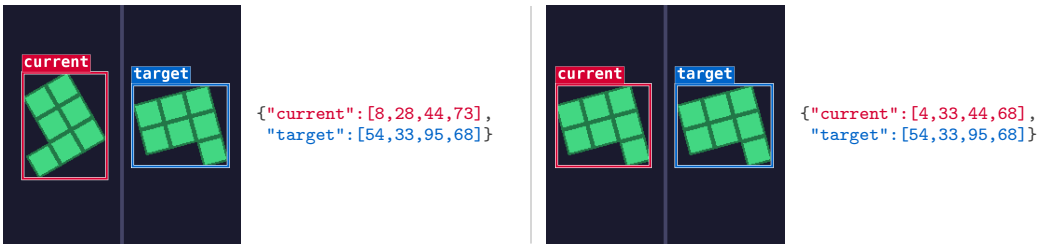


Figure 22: **Boxes CoT for 2D Shape Matching / Mental Rotation:** The image-space box of the `current` shape (left panel) and of the `target` shape (right panel). Left and right show the state after 1 and 4 actions of the same expert episode. Each image shows the box of every listed object (outline and key tag in the object's color). The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

template: {"current":[x0,y0,z0,x1,y1,z1],"target":[x0,y0,z0,x1,y1,z1]}

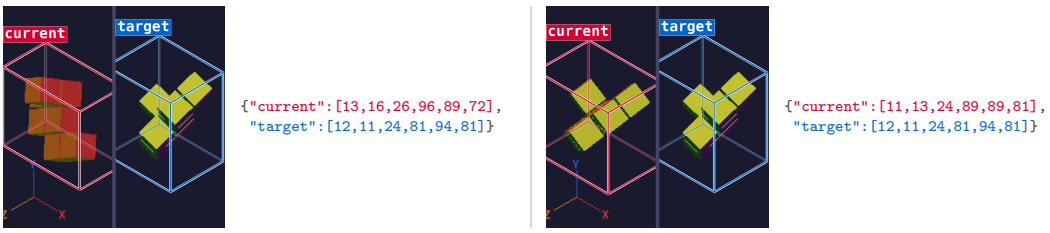


Figure 23: **Boxes CoT for 3D Shape Matching / Mental Rotation:** For the `current` and `target` shapes, a 6-number box $[x_0, y_0, z_0, x_1, y_1, z_1]$: the axis-aligned extent of the rotated polycube in a fixed world frame, normalized by the polycube's bounding radius, so that the chain-of-thought carries pose rather than silhouette. Left and right show the state after 2 and 7 actions of the same expert episode. Each image shows each box as a wireframe in its shape's 3D scene, projected with the renderer's camera and clipped to the shape's panel (hidden edges fainter); the `current` box turns with the shape, the `target` box stays fixed. The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

template: {"player":[col,row],"box_0":[col,row],...,"goal_0":[col,row],...}

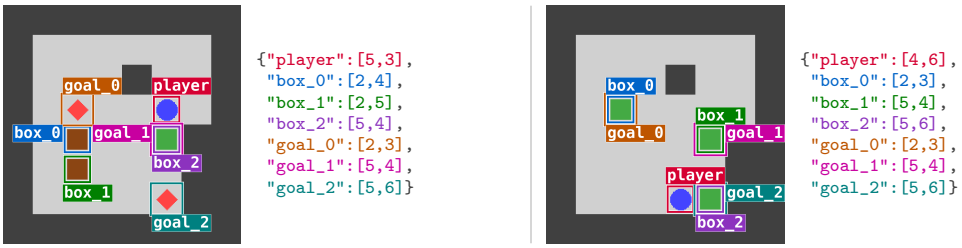


Figure 24: **Boxes CoT for Sokoban:** The grid cell [column, row] of the `player`, of each box (`box_0`, ...) and of each goal (`goal_0`, ...). Left and right show the state after 6 and 20 actions of the same expert episode. Each image outlines the listed cells in the object's color (goal cells at the cell border, player and box cells inside it). The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

template: {"red":[col,row],"blue":[col,row],"green":[col,row],...}

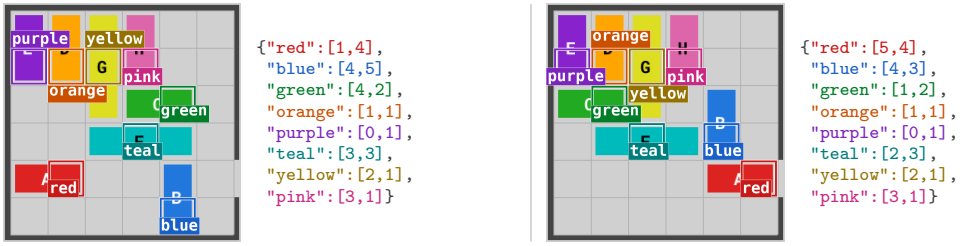


Figure 25: **Boxes CoT for Rush Hour:** One entry per vehicle, keyed by its color (`red` is the target car A; vehicles `A`, `B`, `C`, ... are `red`, `blue`, `green`, ...; past 12 vehicles the colors repeat with a suffix: `red_1`, `blue_1`, ...), with the grid cell [column, row] at the center of the vehicle. Left and right show the state after 3 and 10 actions of the same expert episode. Each image outlines each vehicle's center cell and labels it with its key, in the key's color. The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

template: {"<letter>":[slot],...,"<letter>_0":[slot],"<letter>_1":[slot],...}

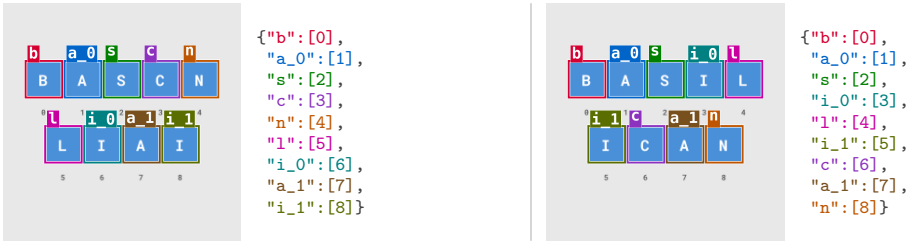


Figure 26: **Boxes CoT for Anagram:** One entry per letter tile, keyed by its letter, with the slot $[i]$ the tile is in. Repeated letters are numbered in slot order (`a_0` is the `a` in the lowest-numbered slot), so a tile's key can change when it moves. Left and right show the state after 3 and 7 actions of the same expert episode. Each image outlines the slot of each listed tile in the tile's color (the slot index is drawn under the tile). The template of the description is at the top; right of each image is its description, each object in the color of its overlay. Line breaks and indentation are added for readability; the model writes the description as one compact line.

G BAGEL Baseline Details

This appendix describes our adaptation of BAGEL-7B-MoT [56] to the puzzle suite via behavioral cloning and our analysis of why it failed to converge when being trained with its visual head.

Setup. BAGEL is a 7B-parameter Mixture-of-Transformers model that natively handles both image understanding (SigLIP ViT + connector) and image generation (VAE + flow-matching head) within a single architecture. We adapt it to our puzzle suite by constructing interleaved trajectories: the user turn contains ViT-encoded patches of the initial image plus the puzzle prompt, and the assistant turn alternates between blocks of 196 VAE latent tokens, encoding the world state after each action as a 14×14 continuous latent grid, with discrete action tokens enclosed in custom `<action>`/`</action>` delimiters. Images are rendered at 128 px by the same game engine used for all other methods and encoded by BAGEL's frozen VAE.

Loss design. Each trajectory position carries one of two losses: (i) standard shifted next-token cross-entropy on the action-token spans, or (ii) flow-matching MSE on the VAE velocity-field head at the 196 latent positions of each generated frame, trained under a Rectified-Flow schedule. The total loss is $\mathcal{L} = \alpha\,\text{mean}(\text{CE}) + \beta\,\text{mean}(\text{MSE})$. Because the MSE averages over $\sim$196 tokens per frame while the CE averages over $\sim$5–10 action tokens per step, the per-episode gradient contribution is strongly tilted toward reconstruction even at nominally equal weights. We train with LoRA adapters (rank 16) on all attention and MLP layers including the vision connector and the MoT generation head, using AdamW with bf16 and FlashAttention-2.

Training pathology. The naive approach feeds the full interleaved sequence causally. Each action attends to every prior VAE-latent block and every prior action. This collapsed training for two compounding reasons. First, effective context explodes: with 196 VAE tokens per frame, a medium-difficulty episode consumes thousands of context tokens, and self-attention becomes dominated by cross-frame latent-to-latent interactions that carry minimal task signal. Second, error accumulates at evaluation: under teacher forcing the model sees ground-truth VAE latents, but at rollout it sees its own noisy generations. Because each action conditions on the full visual history, errors at frame t corrupt the context for all subsequent steps. In practice, multi-step solve rate stagnated near zero despite reasonable single-step accuracy – a classic sign of overfitting to the teacher-forced visual stream rather than learning a coherent policy. We modified BAGEL to just generate the visual state without keeping it in the attention KV-cache or attend only to one past state. This improved the training but still fell well short of the baseline.

H Additional Results

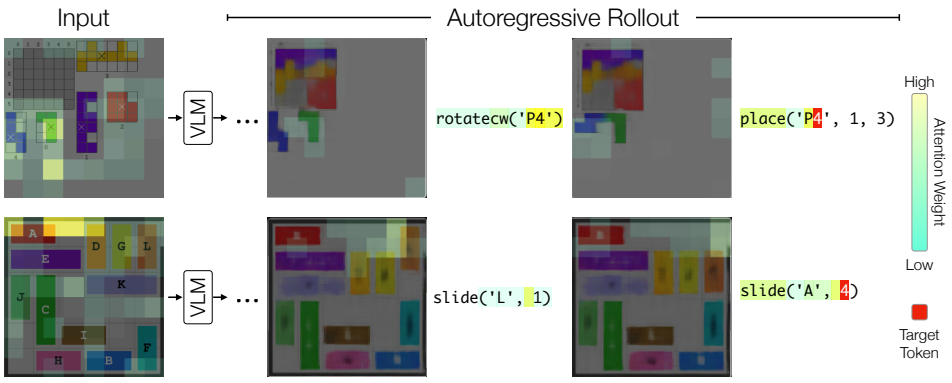


Figure 27: **Attention weight on Visual CoT.** We visualize the attention weights for a target token (in red). The VLM with Visual CoT is able to attend to its own mental images to help plan actions. The top row shows the model attending to the rotated piece to decide its location, and the bottom row shows the model attending to the free space to decide how to win the rush hour game.

H.1 Generalization across model sizes and families

We study how our findings generalize across model sizes and model families. Figures 28 to 31 show the puzzle solve rates of fine-tuned Qwen3.5 0.8B, 2B, 27B, and 35B-A3B (a mixture-of-experts model), and Figs. 32 and 33 show the solve rates of Gemma 4 E2B and InternVL3.5 1B, which use the same training recipe, data, and token budget as the Qwen3.5 models. For the smaller Qwen3.5 models, the action-only method achieves solve rates of 82% and 84% (0.8B and 2B, respectively), and using Visual CoT further enhances the solve rates on the most complex puzzles. This shows that our primary findings on the Qwen3.5 9B model hold for smaller models as well.

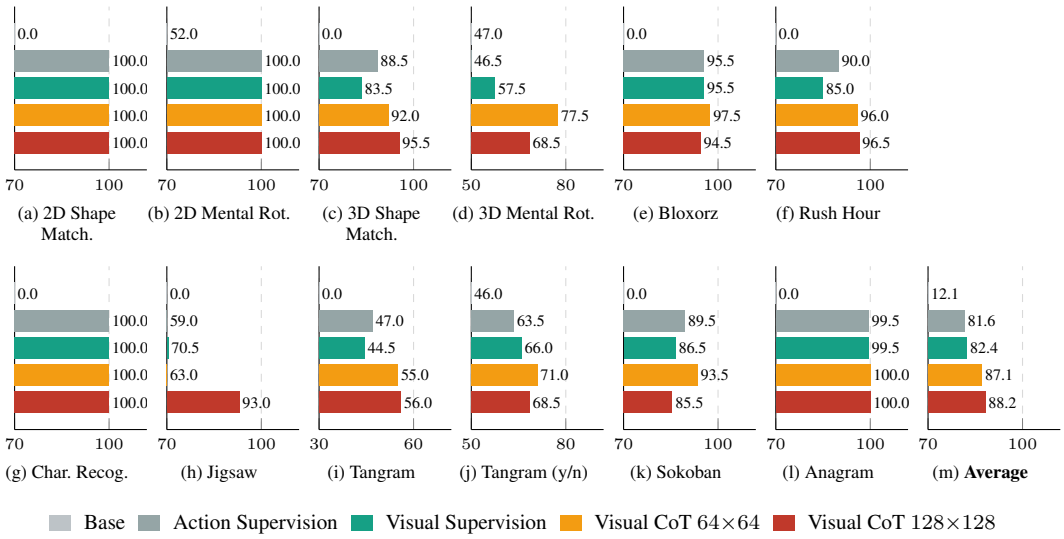


Figure 28: **Solve rates on visual reasoning tasks (Qwen 3.5 0.8B).** Each chart shows the results of the stock Qwen3.5 ('Base', without any training on our part), action supervision and various forms of visual supervision. Visual CoT 64×64 improves over action supervision on every non-saturated game; the largest gains come on 3D Mental Rot. (+31 pts for Visual CoT 64×64) and Jigsaw (+34 pts for Visual CoT 128×128).

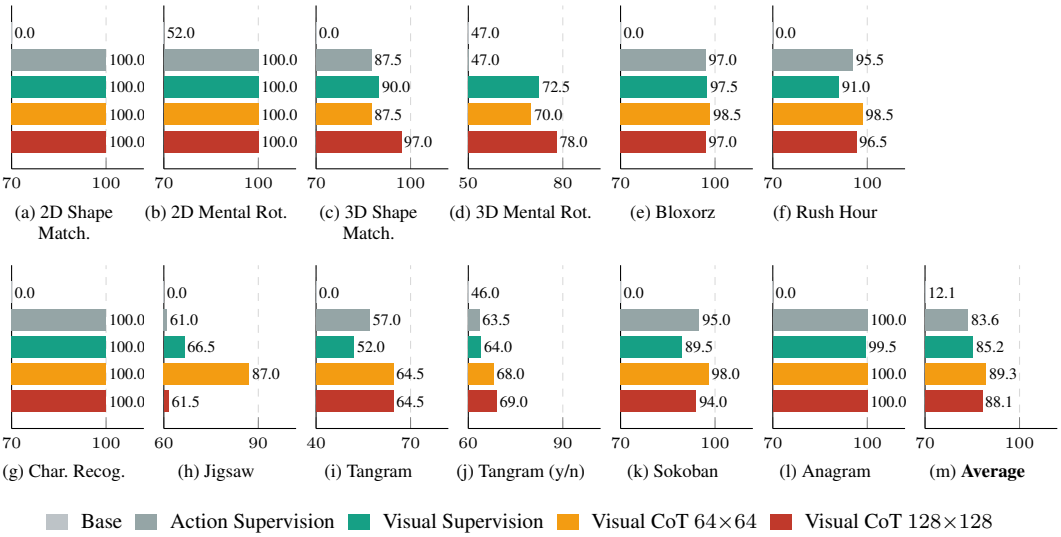


Figure 29: **Solve rates on visual reasoning tasks (Qwen 3.5 2B).** Each chart shows the results of the stock Qwen3.5 ('Base', without any training on our part), action supervision and various forms of supervision. Visual CoT 64×64 leads the average (89.3%); the largest gains over action supervision come on 3D Mental Rot. (+31 pts for Visual CoT 128×128) and Jigsaw (+26 pts for Visual CoT 64×64).

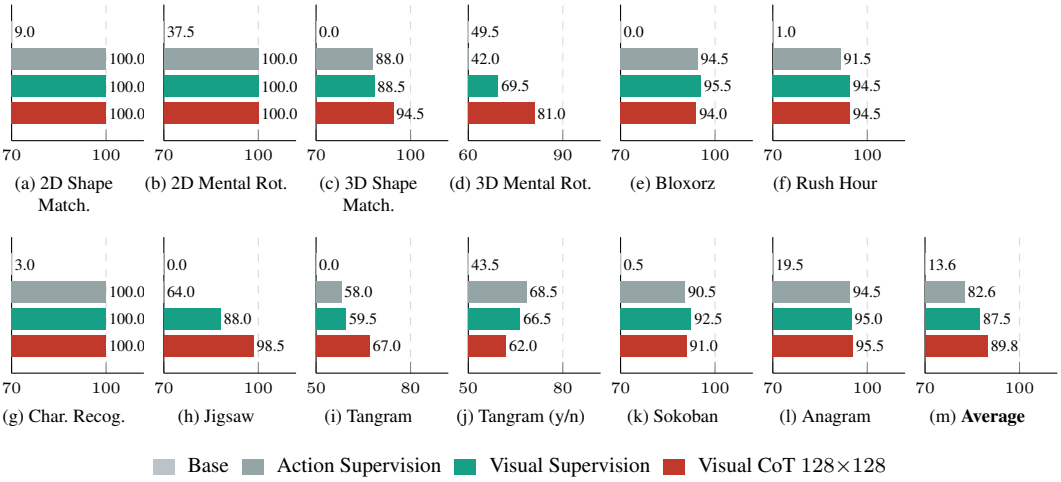


Figure 30: **Solve rates on visual reasoning tasks (Qwen 3.5 27B).** Action supervision, visual supervision, and Visual CoT 128×128, all evaluated at 25,000 optimizer steps. 'Base' is the stock Qwen3.5 27B without any training on our part, evaluated zero-shot.

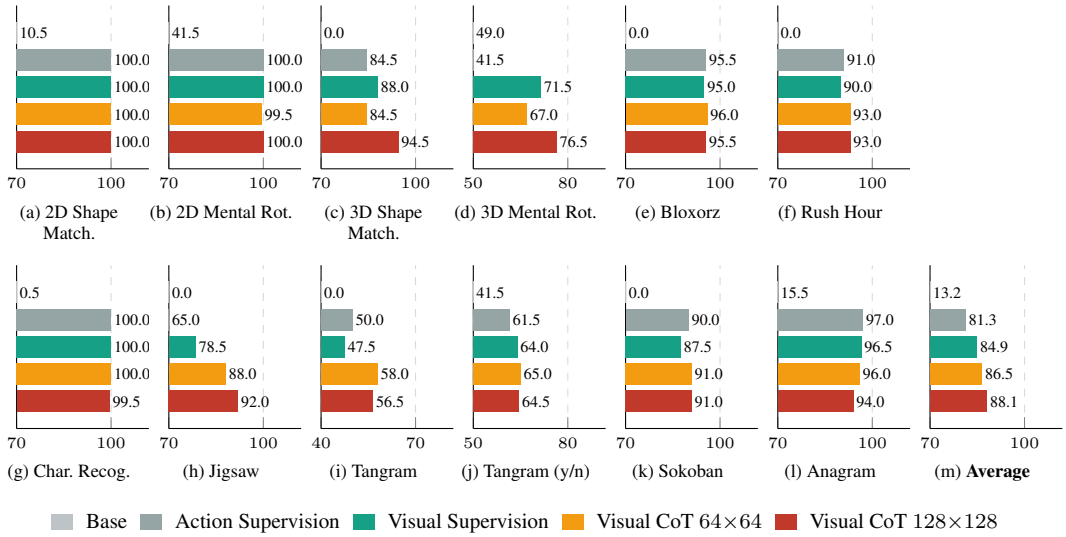


Figure 31: **Solve rates on visual reasoning tasks (Qwen 3.5 35B-A3B).** Action supervision, visual supervision, Visual CoT 64×64, and Visual CoT 128×128, all evaluated at 25,000 optimizer steps. 'Base' is the stock Qwen3.5 35B-A3B without any training on our part, evaluated zero-shot.

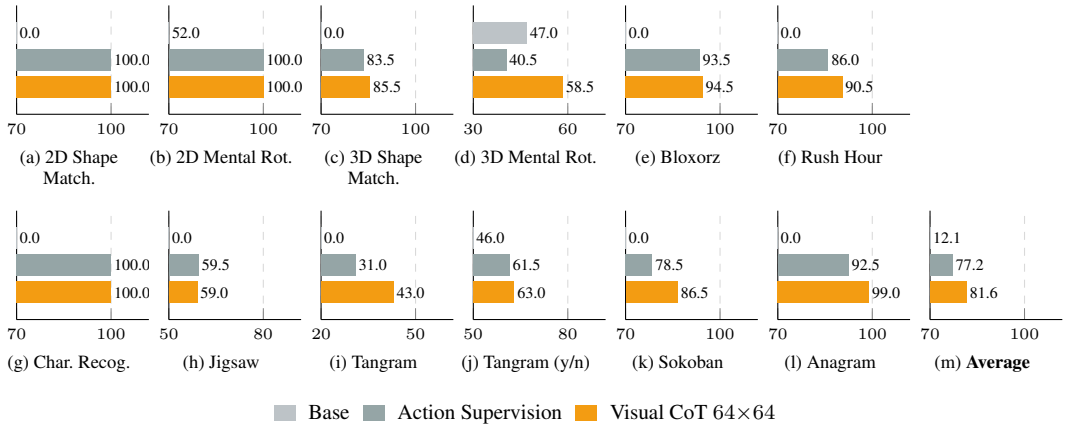


Figure 32: **Solve rates on visual reasoning tasks (Gemma 4 E2B).** Action supervision and Visual CoT 64×64, both evaluated at 25,000 optimizer steps. 'Base' is the stock Gemma 4 E2B without any training on our part, evaluated zero-shot.

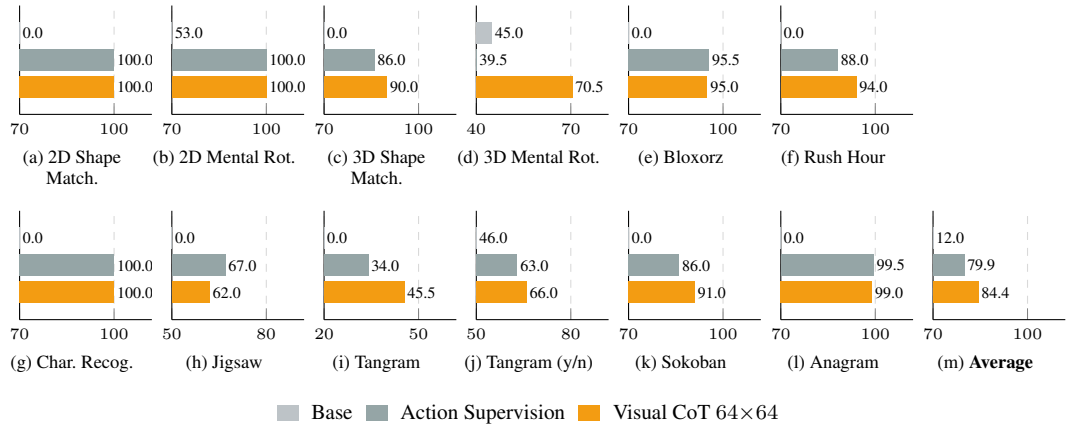


Figure 33: **Solve rates on visual reasoning tasks (InternVL3.5 1B).** Action supervision and Visual CoT 64×64, both evaluated at 50,000 optimizer steps. 'Base' is the stock InternVL3.5 1B without any training on our part, evaluated zero-shot.

H.2 Multi-seed results

Figures 34 and 35 show the per-game solve rates over three training seeds for Qwen3.5 0.8B and 9B. Bars show the mean over seeds and error bars show one sample standard deviation.

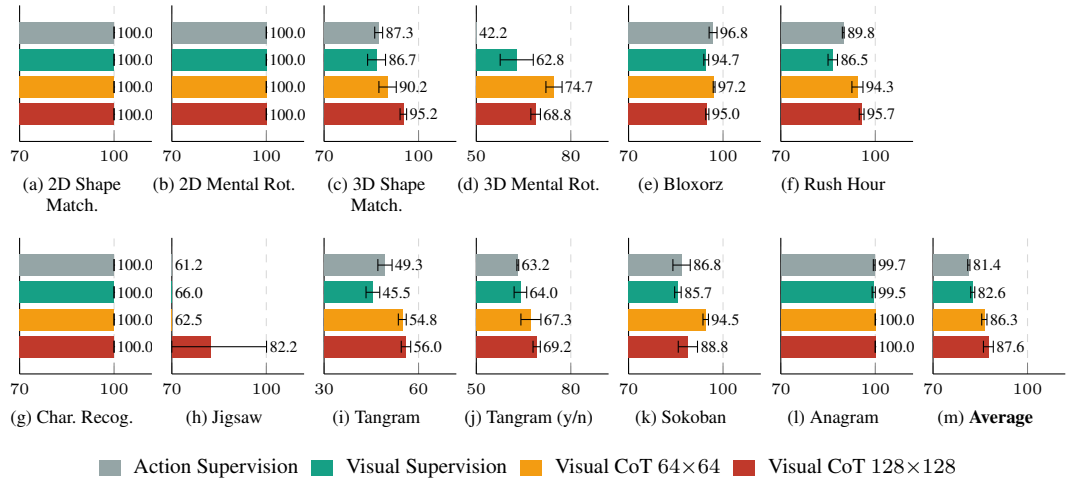


Figure 34: **Multi-seed solve rates (Qwen 3.5 0.8B).** Bars show the mean over three training seeds and error bars show ±1 sample standard deviation. All runs are evaluated at 50,000 optimizer steps.

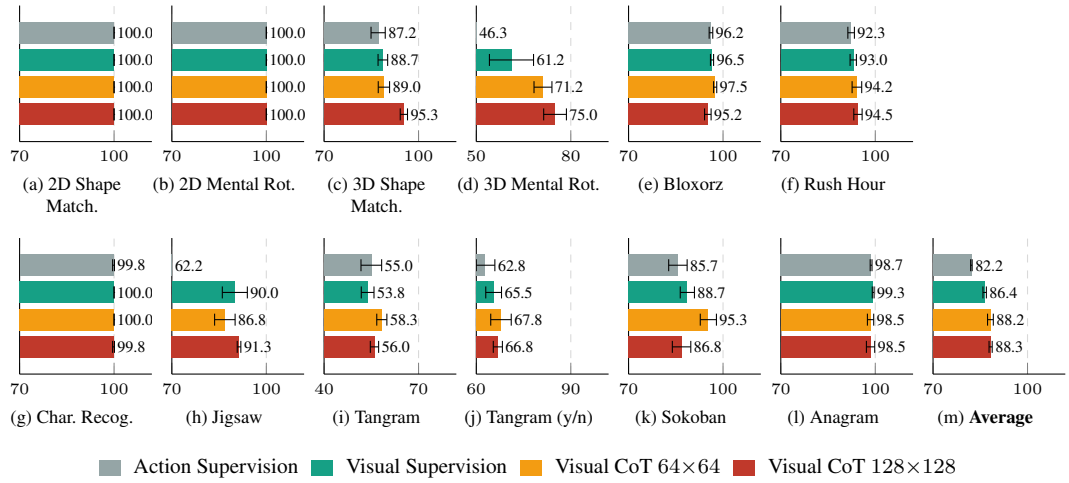


Figure 35: **Multi-seed solve rates (Qwen 3.5 9B).** Bars show the mean over three training seeds and error bars show ± 1 sample standard deviation. All runs are evaluated at 25,000 optimizer steps.

H.3 Solve rate by puzzle difficulty

Figure 36 breaks down the solve rate of the Qwen3.5 9B models by puzzle difficulty. Within each puzzle, we bin the evaluation puzzle instances into five equal-sized groups by optimal solution length (Q1 easiest, Q5 hardest) and average each group over the twelve puzzles. All methods degrade as the solution length grows; for example, action supervision drops from 87.3% on Q1 to 72.9% on Q5, and Visual CoT 128×128 from 94.2% to 78.1%. The gains of next-state reasoning over action supervision are roughly uniform across difficulty levels: Visual CoT 128×128 improves over action supervision by 5.2 to 7.1 points in every group.

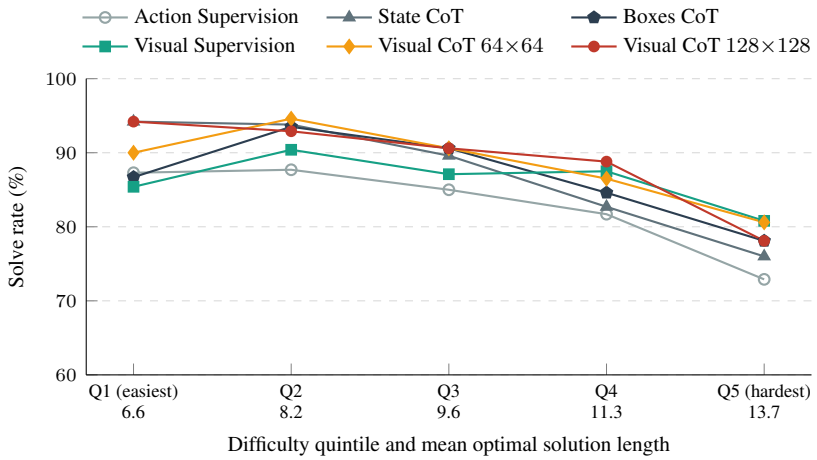


Figure 36: **Solve rate by puzzle difficulty (Qwen 3.5 9B).** Puzzle instances are binned into five equal-sized groups by optimal solution length within each puzzle (Q1 easiest, Q5 hardest); the tick labels give the mean optimal solution length of each group. All methods degrade as the solution length grows, and the gains over action supervision are roughly uniform across groups.

H.4 Comparing LlamaGen vs. FSQ tokenizers for visual token generation

In Fig. 37, we compare the impact of using LlamaGen and FSQ tokens for Visual CoT. On average, we observe that LlamaGen was better than FSQ by 1% on our puzzle suite. Therefore, we use LlamaGen as the default tokenizer for the Visual CoT experiments.

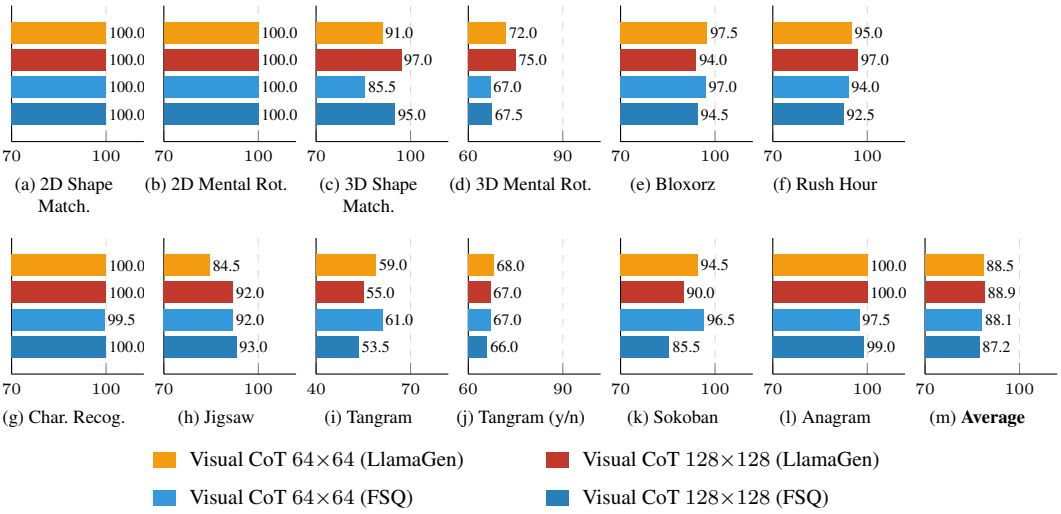
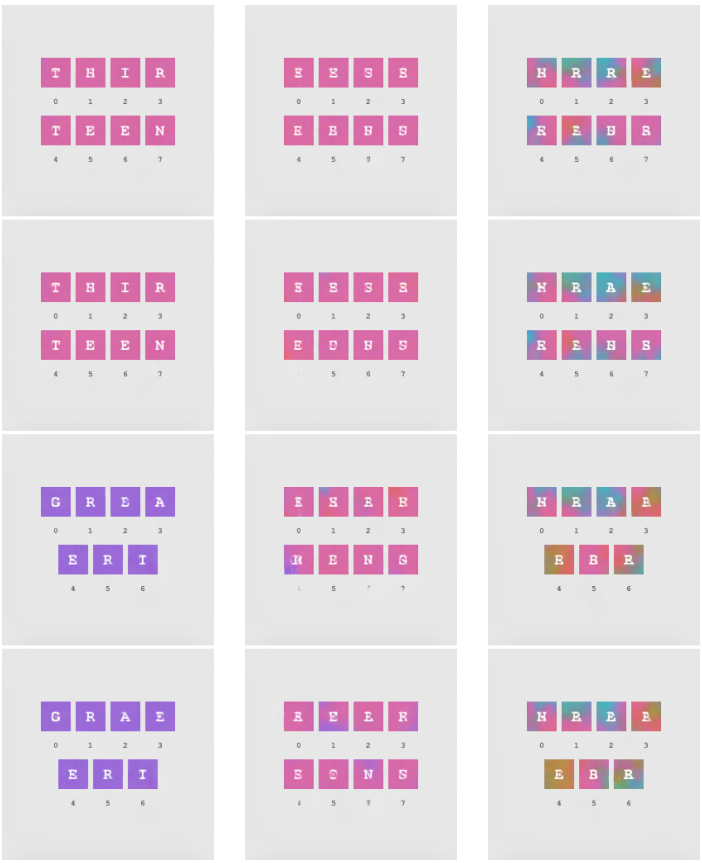


Figure 37: **Comparing Llamagen vs. FSQ tokenizers for visual token prediction.** We observe that Llamagen tokens are on average better for autoregressive generation than the FSQ tokens. Therefore, we use it as our de-facto approach for autoregressive generation.

H.5 Anagram Analysis

As noted in Fig. 6, anagram is the main exception to the general trend where improved reconstruction quality correlates with higher solve rates. However, this apparent discrepancy is explained by examining the nature of the reconstructions themselves. Fig. 38 reveals that the base model achieves lower reconstruction error by producing nearly uniform outputs that vary little across different inputs, approximating a mean image. Because anagram images share the same general structure – colored tiles on a grey background – this mean approximation achieves low pixel-wise error but encodes no task-relevant information about letter positions or identities. In contrast, the action-supervised model produces reconstructions with incorrect tile colors (yielding higher MSE) but faithfully preserves the spatial layout and letter content of each specific input. Thus, the action-supervised model's representations are more informative for solving the task despite their higher reconstruction error, confirming that the correlation between mental image quality and solve rate holds for anagram as well when quality is measured by semantic fidelity rather than pixel-wise distance.



(a) Ground truth (b) Base Qwen (c) Action Only

Figure 38: **Anagram reconstruction comparison.** (a) The ground-truth visual state. (b) Reconstruction from the Base Qwen model, which produces nearly uniform reconstructions that vary little across inputs, achieving low reconstruction error by approximating a mean image that is uninformative. (c) The detached trained visual head for action-supervised model reconstructs images with incorrect colors yielding higher reconstruction error. Yet, it preserves the spatial arrangement and letter identities of the actual input, producing more semantically meaningful representations.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claim of the paper is that VLMs develop mental imagery when learning to solve spatial puzzles. The results in Figs. 5 to 7 show this both qualitatively and quantitatively on select examples.

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Section A.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: There are no theoretical results.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Detailed experimental settings and description of environments is provided in the appendix. Code is available at `https://github.com/apple-aiml-research/ml-mental-imagery`.

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

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

   Answer: [Yes]

   Justification: Code, with instructions to reproduce the main experimental results, is available at `https://github.com/apple-aiml-research/ml-mental-imagery`.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Detailed experimental settings and description of environments is provided in the appendix.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report the mean and sample standard deviation over three training seeds for the main comparison in Section H.2.

8. **Experiments compute resources**

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

Answer: [Yes]

Justification: See Section 3.5

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Experiments are performed purely in simulation on synthetic data. No VLMs were harmed during making of this paper.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: See Section B.

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We do not release datasets or models. We release synthetic puzzle environments to reproduce the experiments.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Base models are cited. All environments are custom. Third-party dependencies are acknowledged in the code release.

13. **New assets**

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

    Answer: [N/A]

    Justification: No assets other than code are released.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

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

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

    Answer: [N/A]

    Justification:The paper does not involve crowdsourcing nor research with human subjects

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components