A HEART MEANS RECURSION: EXPLORING VISUAL MEANING AND COMMUNICATION IN VLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Do vision-language models (VLMs) share our ability to communicate visually? We share preliminary findings, starting from a surprising observation: simple visual symbols like a red heart can strongly influence model behavior on seemingly unrelated tasks, such as steering models towards recursive solutions on simple coding problems. Notably, this steering effect is largely implicit, as models' direct interpretations and thinking traces about visual symbols diverge substantially from their observed behavior. In visual communication between models, this leads to a generation–interpretation gap: while models tend to send shape primitives that they (and humans) find intuitive, these visual cues often fail to steer other models. In further communication games involving abstract concepts and stories, models can creatively use visuals to transmit meaning to one another, although some communications remain non-intuitive to humans. These findings reveal that VLMs might be capable of rich, creative, and visually-grounded communication, but their interpretation and production of visual meaning diverges from human conventions in surprising ways. Together this work highlights both the promise and challenges of visual communication between humans and VLMs.

1 INTRODUCTION

Humans have a long history of visual communication. We encode and transfer meaning through shapes, colors, and spatial arrangements (Fan et al., 2020; Hawkins et al., 2023; Garrod et al., 2007). Vision-language models (VLMs) are trained on billions of images reflecting our visual culture: photographs, diagrams, icons, charts, and art. Frontier VLMs now demonstrate impressive visual understanding, from solving complex multi-discipline problems involving visual inputs (Yue et al., 2024; Lu et al., 2024) to spatial reasoning over visual scenes (Chen et al., 2024). Yet reasoning *about* visual content is different from communicating *through* it. Do VLMs share our ability to communicate visually?

We explore visual communication in VLMs, testing how deeply they absorb visual cues in their inputs and their ability to use visualizations to communicate ideas. We present three complementary investigations. First, we test whether simple visual symbols (shapes, colors, icons) systematically steer VLM behavior on non-visual tasks, and whether models can faithfully report these influences. Second, we study VLMs' generation and interpretation of visualizations communicating abstract concepts and stories. Third, we evaluate alignment between model and human visual interpretation.

Together, our experiments yield three main findings:

1. Simple visual symbols can strongly steer VLMs from baseline behavior on unrelated tasks. These symbol–concept associations are model-specific and not accounted for by captions or pixels-in-context. Furthermore, models do not faithfully verbalize this influence (Sec. 3).
2. VLMs *generate* creative visualizations to communicate abstract concepts and stories, but do not always *interpret* their own visualizations in the intended way (Sec. 4).
3. Models' visual meaning spaces are only partially aligned with human interpretations. The visualizations that prove most effective for steering or that emerge across multi-turn communication games between models are often less legible to humans (Sec. 5).

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

2 Related Work

Visual understanding and image generation VLMs have absorbed substantial visual knowledge from pretraining, but the extent to which this knowledge supports human-like visual processing remains unclear. Performance on numerous multimodal benchmarks (Yue et al., 2024; Lu et al., 2024; Liu et al., 2024) shows that frontier VLMs increasingly match or surpass human baselines on complex scene understanding, and multimodal generative models now produce sophisticated and diverse images from text descriptions (Google DeepMind, 2025; OpenAI, 2025). However, VLMs also fail at seemingly trivial perceptual tasks (Rahmanzadehgervi et al., 2024), do not fully capture human color perception (Gomez-Villa et al., 2025; Liang et al., 2025), and remain biased by prior knowledge or textual information over perceptual evidence (Vo et al., 2026; Shi et al., 2026).

Visual reasoning Several studies show that allowing VLMs to draw, sketch, or annotate during problem-solving improves reasoning over text-only chains of thought (Hu et al., 2024; Menon et al., 2024; Collins et al., 2026; Vinker et al., 2025). At the same time, Sepehri et al. (2025); Zeller et al. (2026) demonstrate a substantial human–model gap in actively constructing mental visualizations. Whereas these works use visualization to support a model's own reasoning, we study how VLMs generate visualizations to convey meaning to other models, and whether those visualizations are interpreted as intended.

Visual meaning and communication Beyond perception, visual meaning rests on learned associations and conventions. VLMs do not reliably exhibit human cross-modal associations such as the Bouba–Kiki effect (Kouwenhoven et al., 2025b) and struggle with visual metaphor (Zhao et al., 2026; Kundu et al., 2025). Humans capture meaning through abstract visual representations that evolve from iconic depictions into compact symbols (Garrod et al., 2007; Fay et al., 2010) and adapt these representations based on communicative and interaction goals (Fan et al., 2020; Hawkins et al., 2023). In machine learning, the emergent communication literature studies how models trained from scratch develop symbolic conventions in referential games (Lewis, 1969; Lazaridou et al., 2017; Havrylov & Titov, 2017; Lazaridou & Baroni, 2020), with later extensions to pretrained LLMs (Kouwenhoven et al., 2025a) and graphical modalities (Mihai & Hare, 2021). In multimodal models, recent work explores visual communication through visual prompts and annotations (Wu et al., 2024; Cai et al., 2024; Zeller et al., 2025; Zhang et al., 2026; Geirhos et al., 2026), text-mediated dialogue to guide image reconstruction (Hakimov et al., 2026), or collaborative games constrained to fixed icon palettes (Clark et al., 2021). We build on these studies and investigate how VLMs communicate abstract concepts and stories through *visual primitives* and *drawing*.

VLM robustness to adversarial and distractor images Adversarially optimized images can systematically manipulate VLM responses and introduce misaligned behavior (Bailey et al., 2024; Bagdasaryan et al., 2023; Qi et al., 2024; Phute & Balakrishnan, 2025). Non-adversarial images can also steer model's cooperative choices (Ong, 2026), and images containing task-irrelevant visual distractors can degrade multimodal reasoning (Bae et al., 2025; Sun et al., 2026). Here, we investigate how *simple, non-adversarial* visual symbols and free-drawing influence text-only tasks and study their systematic *semantic* associations.

Faithfulness and self-knowledge Models' CoT explanations often omit cues that drive their behavior (Turpin et al., 2023; Chen et al., 2025; Chua & Evans, 2025), and VLMs articulate image-based cues even less often than text-based ones (Balasubramanian et al., 2025). VLM reasoning traces can also overstate their visual grounding, describing images that are absent (Asadi et al., 2026) or claiming to re-examine images without actually doing so (Shi et al., 2026). Models' explicit reports about their own learned dispositions are also only partially accurate (Binder et al., 2025; Betley et al., 2025), and their generative and discriminative behaviors can diverge (Li et al., 2024). We extend this work to visual cues with no obvious relation to the task, testing whether reasoning traces and direct self-reports reflect how these cues steer behavior.

3 Visual Steering

We begin by studying how the simplest visual elements (colors and shapes) influence model behavior when placed in context. We find that simple visual symbols can strongly and systematically

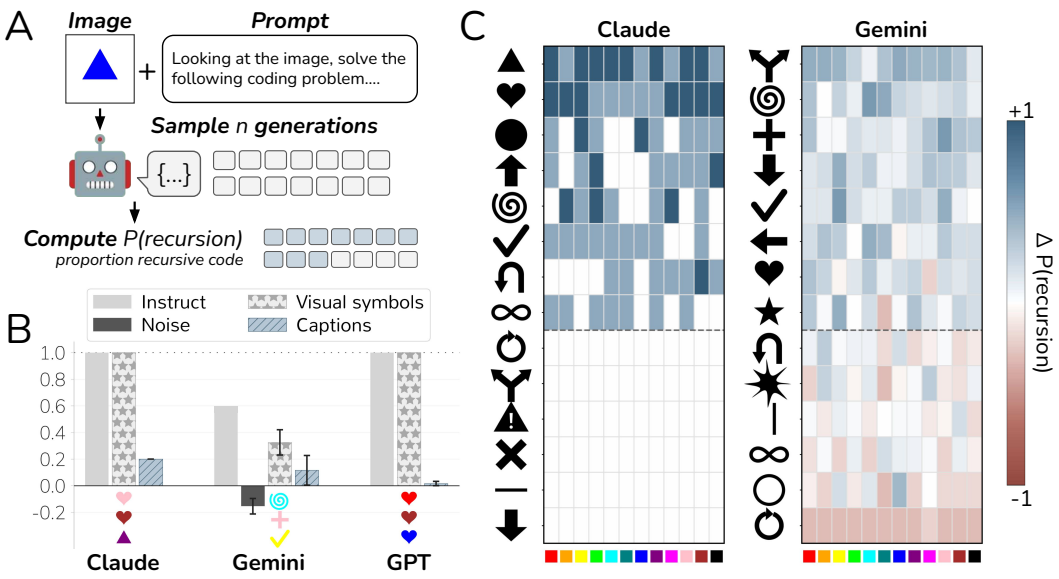

Figure 1: **Simple visual symbols steer model behavior on a coding problem. A) Evaluation methodology.** We append an image of a simple colored shape (e.g., a blue triangle) to a coding prompt (binary tree traversal) and instruct the model to look at the image while solving the problem. Across n sampled generations, we compute the proportion of recursive solutions $P(\text{recursion})$ and measure the *visual steering effect* $\Delta P(\text{recursion})$ as the change in $P(\text{recursion})$ relative to the no-image baseline. **B) Visual symbols steer toward recursion far more strongly than random noise or matching text descriptions.** Star-patterned bars show average $\Delta P(\text{recursion})$ for Gemini 3.1 Pro and Claude Opus 4.7's top three steering symbols (symbols shown below each bar), which nearly matches the effect of directly instructing the model to use recursion (light gray bars). By contrast, matching text descriptions (e.g., "an image of a pink heart"; hatched bars) and Gaussian noise images (dark gray bars) induce minimal to no steering. **C) Models display different steering landscapes over a library of 288 visual symbols.** Heatmaps show $\Delta P(\text{recursion})$ across 12 colors (x-axis) for each model's 8 highest- and 6 lowest-steering shapes (y-axis; full heatmaps in App. B.2.1).

steer model behavior on tasks where they have no obvious relevance, starting with coding problems (Sec. 3.1). We then show that this effect is especially pronounced for these symbols rather than a property of generic visual inputs, and probe whether it is driven by high-level shape concepts or low-level pixel properties (Sec. 3.2). Finally, we test whether the effect generalizes across tasks (Sec. 3.3) and examine whether models are aware of this visual influence (Sec. 3.4).

3.1 Simple visual symbols steer models in surprising ways

Task and evaluation methodology We prompt models (Claude Opus 4.7, Gemini 3.1 Pro, GPT-5.4) to solve a simple coding problem, *inorder binary tree traversal*, with an image of a simple colored shape inserted into the context (Fig. 1A). This problem admits equally valid recursive or iterative solution strategies. To evaluate whether the image influences model behavior on the problem, we instruct the model to "look at the provided image" (see App. B.1 for prompting details). We sample $n = 20$ model solutions and classify each sample as recursive or iterative (see Apps. A.1 and A.2 for details). We denote the proportion of each solution strategy across the n samples as $P(\text{recursion})$ and $P(\text{iteration})$, and measure the *steering effect* of the image as: $\Delta P(\text{recursion}) = P(\text{recursion} \mid \text{image}) - P(\text{recursion})$.

A heart or a triangle can steer towards recursion In the absence of visual inputs, models prefer iterative implementations at baseline ($P(\text{recursion}) = 0$ for Claude Opus 4.7 and GPT-5.4, 0.4 for Gemini 3.1 Pro). Surprisingly, seemingly irrelevant colored shapes can induce substantial behavioral shifts in models towards using recursion (Fig. 1B and C). For example, a red heart steers Claude Opus 4.7 and GPT-5.4 to switch to using recursion 100% ($\Delta = +1.0$); for Gemini 3.1 Pro, a yellow checkmark pushes recursion to 90% of code samples ($\Delta = +0.5$). Steering with these visual

symbols matches or approaches the effect of directly instructing the model to use recursion (Fig. 1B; $\Delta = +1.0$ for Claude, $\Delta = +0.6$ for Gemini).

Models have distinct steering profiles Do models respond consistently to particular shapes and colors, or is recursion steering idiosyncratic to particular shape–color combinations? To look for potential regularities, we expand our probe set to a library of 288 visual symbols combining 24 shapes (e.g., geometric shapes, icons, mathematical symbols) with 12 colors. It includes shapes with intuitive links to recursion (e.g., spirals, branching arrows) as well as shapes with no apparent relevance (e.g., hearts, stars, blobs).

Different models exhibit distinct visual steering profiles (Fig. 1C). For Claude Opus 4.7, triangles ($\Delta = +0.83$), hearts ($\Delta = +0.79$), and circles ($\Delta = +0.68$) consistently drive recursion regardless of color; GPT-5.4 is steered primarily by hearts ($\Delta = +0.54$). Gemini 3.1 Pro is most strongly steered by specific shape–color combinations such as a pink plus or a cyan spiral ($\Delta = +0.50$). Intuitive symbols do not reliably steer all models: for example, branching arrows evoke a function splitting into subproblems, but they fail to steer Claude Opus 4.7 towards recursion and are much weaker at steering GPT-5.4 than hearts ($\Delta = +0.21$). Despite some observed regularities, no single shape or color steers uniformly; instead, symbols appear to shift behavior to varying degrees rather than acting as fixed, language-like mappings. While visual steering profiles are largely model-specific, comparing a larger set of models reveals that there are also some shared associations (see App. B.4). We study the impact of prompt variations and image placement in context in App. B.2.4.

3.2 Is visual steering pixel-level or concept-level?

Is visual steering specific to images of visual symbols? If so, are models responding to the high-level concepts these images depict (e.g., "pink heart") or to low-level pixel arrangements? We approach this in two ways: by replacing each symbol with a text caption describing it, and by replacing it with other images (random noise, naturalistic photos). We then apply systematic perturbations to symbol shape and color to test whether the effect relies on these features.

Text captions do not steer equivalently to images Visual steering might be evoked by associations linked to textual descriptions (e.g., the words "yellow heart"). If so, image captions should steer behavior similarly to the images. We replace each image with its textual description (e.g., "an image of a yellow heart").[1] We find that caption steering does not mirror image steering: correlations between image- and caption-induced steering effects are fairly low ($r = -0.05$ for Claude Opus 4.7; $r = 0.31$ for Gemini 3.1 Pro; $r = 0.17$ for GPT-5.4), and symbols that maximally steer as images often produce little to no shift as text (see Fig. 1B and App. B.2.2). An inspection of models' reasoning traces (Sec. 3.4) reveals that models often dismiss the image as irrelevant, further suggesting that the effect is not mediated by model verbalizations of image features.

Steering is strongest for visual symbols, not random or naturalistic images It's possible that visual steering is an artifact of processing generic visual tokens, and that inserting *any* image in context will disrupt the model's typical behavior. We evaluate two controls: images of synthetic Gaussian noise, and a set of naturalistic images from ImageNet (Deng et al., 2009). Neither produces systematic shifts toward recursion ($\Delta = 0.0$ for Claude Opus 4.7 and GPT-5.4; ≈ -0.1 for noise and ≈ -0.05 on ImageNet for Gemini 3.1 Pro; App. B.2.2). Simple colored shapes appear to have a particularly pronounced steering effect, especially for frontier models; by contrast, smaller or less capable models tend to exhibit higher sensitivity to noise and ImageNet (see App. B.5).

Steering relies on shape orientation and spatial arrangement To understand what specific features drive visual steering, we evaluate model responses across a range of image perturbations. We find that steering relies heavily on recognizing shape features in prototypical orientations: disrupting a shape's structural integrity (e.g., by shuffling pixels, which destroys shape while preserving color proportions) or rotating it away from its upright orientation substantially diminishes the steering effect. Perturbations to symbol size and location reveal more complex behavioral shifts, suggesting models may sometimes treat placement as part of the visual "meaning" (see App. B.2.3).

[1] To control for prompt length when using images vs. text captions, we repeat the caption strings to match or nearly match the number of tokens used by an image for each model.

Table 1: **Gemini 3.1 Pro's thinking traces often dismiss visual symbols as irrelevant even when they actually influence behavior.** Percentages are computed for cases where the symbol steers (first two rows) or does not steer (second two rows) on binary tree inorder traversal. For GPT-5.4, 74.7% of traces show unfaithful attribution; Claude Opus 4.7 did not emit traces.

Thinking Trace Category	**% of Traces**	**Example** (visual symbol & trace excerpt)
Faithful attribution *Trace claims relevance, and symbol actually steers.*	**25.0%** of steered ($n = 533$)	 $\Delta = -0.15$	*"The image is a solid yellow circle – likely a distraction, but I note it... Wait. The circle... that could suggest a loop, therefore, iterative. Fine. Let's do iterative."*
Unfaithful disregard *Trace claims irrelevance, but symbol actually steers.*	**71.5%** of steered ($n = 1{,}524$)	 $\Delta = +0.34$	*"First, the image is completely irrelevant – a cyan spiral. Distraction, ignore... Do I go recursive or iterative? Recursive is usually cleaner and more concise for this..."*
Faithful disregard *Trace claims irrelevance, and symbol does not steer.*	**77.9%** of non-steered ($n = 584$)	 $\Delta = +0.01$	*"The purple triangle is a red herring; I'm going to disregard that... I immediately think about both iterative and recursive approaches..."*
Unfaithful attribution *Trace claims relevance, but symbol does not steer.*	**19.3%** of non-steered ($n = 145$)	 $\Delta = +0.01$	*"A downward pink arrow... A downward arrow could signify a 'top-down' approach, which often implies recursion... I'll go with recursive."*

3.3 Does visual steering generalize to other tasks?

In previous sections, we evaluated visual steering on inorder binary tree traversal. Do these results reflect broader associations that hold across tasks and domains? We next test whether visual symbols consistently steer behavior across additional coding problems, coding aspects, and non-coding tasks.

Visual steering extends to other coding problems Across five additional recursion-versus-iteration problems (e.g., Fibonacci number computation, binary search), some of the strongest-steering shapes on binary tree traversal (e.g., hearts for Claude Opus 4.7 and GPT-5.4; branching arrows and spirals for Gemini 3.1 Pro) consistently steer toward recursion. Conversely, on problems with higher recursive baselines, looping arrows (Gemini 3.1 Pro, Claude Opus 4.7) and right arrows (all models, including GPT-5.4) steer toward iteration (see Apps. B.2.1 and B.3.1). Visual steering also extends to other algorithmic strategies (DFS vs. BFS) and data structure choice (e.g., doubly linked lists vs. ring buffers), often reflecting intuitive geometric associations. For instance, Gemini 3.1 Pro is steered towards BFS by horizontal lines and DFS by vertical lines, or towards using linked lists by lines and ring buffers by circles (App. B.3.1).

Visual steering influences behavior on non-coding tasks Beyond coding, the same 288 colored shapes reliably steer both writing style and responses to binary preference questions. Specifically, visual symbols can shift which style models adopt when writing passages on a set of topics (professional vs. poetic, positive vs. negative valence, optimistic vs. melancholic, or terse vs. verbose), as well as their answers to simple binary preference prompts (e.g., "Which do you prefer: summer or winter?"). Some associations seem intuitive: pink shapes and hearts steer all three models towards writing poetically rather than professionally, and blue triangles cause Gemini 3.1 Pro and GPT-5.4 to prefer winter over their baseline choice of summer. Other associations are a bit more unexpected: for example, hearts cause Claude Opus 4.7 to write much more verbose prose (App. B.3.1).

3.4 Are models aware of how visual symbols steer them?

Do models have access to their own visual associations, or does visual steering operate implicitly? We investigate this by silently injecting images in context, directly querying models for symbol interpretations, and analyzing thinking traces. For this analysis, we focus on the association between certain symbols and recursion on coding problems.

Some symbols steer towards recursion with silent injections We evaluate visual steering with images silently inserted into context. Even without explicit instructions to refer to the image, visual symbols induce strong steering effects on binary tree traversal: maximum recursion steering deltas reach $\Delta = +1.00$ for Claude Opus 4.7, $\Delta = +0.60$ for Gemini 3.1 Pro, and $\Delta = +0.50$ for GPT-5.4. Interestingly, the visual symbols that steer under silent injection often decouple from those that steer under explicit instruction, showing weak intra-model correlation ($r \leq 0.18$; see App. B.6). In their thinking traces, models often do not mention the silently injected image at all, even when it induces substantial behavioral shifts (100% of traces for GPT-5.4, 52.3% for Gemini 3.1 Pro; see App. B.8).

Models' direct interpretations and thinking traces mismatch their behavior We prompt models to directly interpret each visual symbol as "recursion" or "iteration" and compute the proportion of recursive interpretations across $n = 20$ samples. We find a substantial mismatch between these direct interpretations and empirical steering on binary tree inorder traversal (Fig. 3B): they are only weakly correlated across models ($r = 0.22$ for Claude Opus 4.7, $r = 0.29$ for Gemini 3.1 Pro, and $r = 0.31$ for GPT-5.4). See App. B.7 for full methodology and results.

Furthermore, analyzing Gemini 3.1 Pro's thinking traces[2] on binary tree inorder traversal ($N = 2{,}880$) reveals widespread lack of awareness of the influence exerted by visual symbols (Tab. 1). When visual symbols exert a nonzero steering effect ($|\Delta| \geq 0.10$), 71.5% of traces unfaithfully disregard the symbol as irrelevant; the model instead refers to technical justifications (such as stack overhead or code elegance) to explain its decisions. When symbols do not steer behavior, 19.3% of traces unfaithfully attribute relevance to the symbol despite no behavioral shift from baseline. This behavior persists across nine additional coding tasks ($N = 20{,}130$ traces; 64.1% unfaithful disregard, 22.0% unfaithful attribution), demonstrating that visual steering often operates below the level of reasoning traces surfaced to the user. GPT-5.4 predominantly exhibits unfaithful attribution on binary tree inorder traversal: when visual symbols exert no steering effect, 74.7% of its traces claim the visual symbol dictated its behavior. Only 50.4% of GPT-5.4's traces faithfully attribute relevance to symbols when they do steer. See App. B.8 for details.

4 Model-to-model visual communication

Communication is bidirectional, requiring both interpretation and generation. We showed above that models *interpret* visual primitives in surprising ways that diverge from their own direct interpretations and thinking traces. Here, we study what visualizations models *generate* to steer other models (Sec. 4.1) as well as to communicate abstract concepts and short stories (Sec. 4.2). We focus on two channels of visual communication: using visualizations generated from a pre-defined library of visual symbols, or using free-form drawing rendered from PIL code.

4.1 What visualizations do models construct to steer coding behavior?

We first test whether models can actively generate visualizations that steer another model's coding behavior, using the same recursion vs. iteration task from Sec. 3. A *sender model* is tasked with producing a visualization that will steer a *receiver model* to use a target strategy (e.g. recursion) on binary tree inorder traversal. The sender either chooses a single colored shape from the visual symbol library in Sec. 3 or draws an image on a 400×400 canvas via free-form PIL code (no text allowed). The sender sees the receiver's baseline strategy distribution as well as its behavior after each communication turn ($n = 10$ samples), and iteratively refines the visualization up to a maximum of 20 turns (Fig. 2A; see App. C.1 for details). In our experiments, we set the sender and receiver to be the same model, which additionally allows us to investigate whether models have an accurate understanding of their own visual meaning space.

There is a generation–interpretation gap in visual symbol associations Interestingly, the symbol visualizations that senders generate to steer toward recursion generally *do not* correspond to the

[2]For Gemini 3.1 Pro, thinking traces are accessed using `ThinkingConfig(include_thoughts=True)` in the API . For OpenAI and Anthropic models, we pass provider-specific configuration arguments (e.g., `reasoning_effort`) to return their reasoning traces in the API response. Despite high reasoning effort, Claude Opus 4.7 did not return thinking traces.

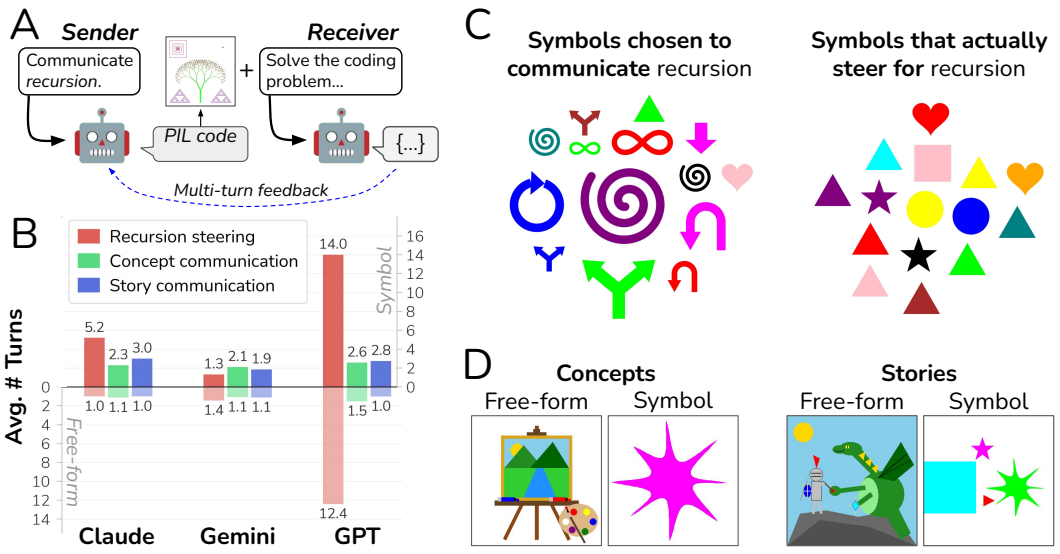

Figure 2: **Models generate creative visualizations to communicate concepts, but exhibit a generation–interpretation gap. A) Communication game methodology.** A sender model generates a visualization to convey a target coding strategy (e.g., recursion), concept, or story to a receiver model, refining its visualization through multi-turn feedback. **B) Multi-turn feedback enables models to converge on effective visualizations.** Bars show the average number of turns to succeed using visual symbols (top bars) versus free-form PIL drawings (bottom bars). Symbol-based communication requires more communication turns to align sender and receiver than free-form drawing. **C) Senders choose intuitive symbols to communicate recursion rather than the symbols that actually steer them.** Top symbols selected by Claude Opus 4.7 as a sender (left, sized by selection frequency) compared to the symbols with the largest empirical steering effect $\Delta P(\text{recursion})$ (right). **D) Models construct creative scenes using free-form code and abstract symbols.** Examples show successful visualizations generated by Gemini 3.1 Pro to communicate the concept "art" (left) and a three-sentence story about a knight removing a thorn from a dragon's paw (right).

shapes that most strongly steered models in Sec. 3. For example, while hearts and triangles are among the strongest steering symbols for Claude Opus 4.7, it tends to send more human-intuitive spiral, infinity, and arrow shapes instead (Fig. 2C; other models in App. C.4). This mismatch suggests that models' generation and interpretation of visual meaning may draw on partially disjoint processes, and that models may struggle to access their own visual meaning spaces. Despite this asymmetry, senders can still arrive at effective steering visualizations within < 20 turns on average (5.2 for Claude Opus 4.7, 1.3 for Gemini 3.1 Pro, and 14.0 for GPT-5.4), indicating that the multi-turn feedback loop helps sender and receiver align on visual meaning. With the exception of GPT-5.4, steering the receiver towards recursion using free-form drawing requires a lot fewer turns (Fig. 2B).

We also run the sender–receiver communication game on the other coding tasks and non-coding tasks from Sec. 3.3, including DFS vs. BFS, data structure choices, and writing style steering, finding a similar pattern of results. Full results are reported in App. B.3.

4.2 What visualizations communicate abstract concepts and stories?

We extend the sender–receiver framework from Sec. 4.1 to richer communicative tasks. The sender model is given a target concept or story, and must generate a single visualization that allows the receiver model to select the correct target from a four-option multiple-choice set. In the concept communication game, the target is one of four concept options (e.g., art vs. science, history, and philosophy; recursion vs. iteration, memoization, and divide and conquer). In the story communication game, the target is one of four three-sentence stories. In the symbol library condition, the sender is allowed to compose up to five colored shapes to communicate the story. Senders are given up to a maximum of five turns to refine their visualizations. See App. C.1 for details.

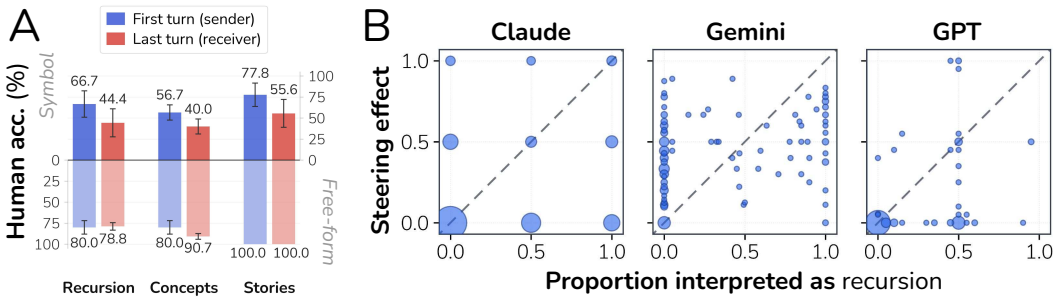

Figure 3: **Visual steering associations diverge from both human intuition and models' own explicit interpretations. A) Optimizing symbol communication for model receivers reduces human legibility, whereas free-form drawings remain interpretable.** Bars show human accuracy (%) at identifying the target coding strategy (recursion vs. iteration), concept, or story from visualizations on the first communication turn (blue; unsuccessful for the receiver) versus the final turn (red; successful), using visual symbols (top bars) or free-form PIL drawings (bottom bars). **B) Models' direct interpretations of visual symbols poorly predict their empirical steering effects.** Each point represents one or more of the 288 visual symbols (sized by count), plotting the proportion of trials in which a model explicitly classifies the symbol as "recursion" (x-axis) against its behavioral steering effect $\Delta P(\text{recursion})$ (y-axis).

Models construct creative visualizations, but the generation–interpretation gap persists When permitted to use free-form drawing through PIL code, senders successfully generate visualizations that allow receivers to reliably identify the target concept or story: 97.8% of experiments eventually converge on the correct target, with 91.2% succeeding zero-shot (i.e. on the first communication turn; Fig. 2B). However, mirroring results in Sec. 4.1, we observe a generation–interpretation gap across all models when the communication is restricted to simple visual primitives. Zero-shot communication success drops to 47.8%, and nearly a third of experiments (30.7%) reach the maximum 5-turn limit without ever successfully conveying the target concept or story (see App. C.3 for more details).

The visualizations that models use to successfully communicate are creative; models tend to try diverse visualizations across communication turns. Visualizations also mostly seem interpretable by humans (see Fig. 2D and App. C.5 for examples). However, we also observe some visualizations where the intended meaning seems more opaque. We investigate the asymmetry between model and human visual interpretation in the following section.

5 Alignment with Human Interpretation

Many symbol–concept associations and visualizations from Secs. 3 and 4 seem counterintuitive to humans. To quantify human–model alignment, we conduct an exploratory human study ($N = 7$)[3] in which participants view steering symbols and model-generated visualizations and select the intended target (recursion vs. iteration, an abstract concept, or a short story) using the same multiple-choice format given to receiver models (App. D).

Humans struggle to interpret top-steering visual symbols Only $\sim 36\%$ of human judgments match the empirical steering direction of top-steering symbols (Sec. 3), which is below the 50% chance baseline. Gemini 3.1 Pro's recursion-steering symbols are slightly more legible than Claude Opus 4.7's or GPT-5.4's (44.4% vs. 38.9% and 33.3%); for instance, humans mostly associate hearts with iteration rather than recursion ($\sim 33\%$ recursive). By contrast, symbols chosen by sender models to steer receivers (Sec. 4.1) are far more legible ($\sim 67\%$ accuracy), confirming that senders rely on human visual conventions rather than their own steering associations.

[3]The survey was approved by our institution's research ethics committee and followed standard practices of informed consent and anonymity. See App. D for the full survey design.

Multi-turn symbol communication reduces human legibility For the concept and story games (Sec. 4), we compare human accuracy on visualizations from the first communication turn (i.e. initial sender attempts that fail to transmit meaning to the receiver) against those from the final turn (i.e. refined visualizations that succeed). When models communicate using visual symbols, first-turn visualizations are consistently more interpretable to humans than final-turn visualizations ($66.7\% \rightarrow 44.4\%$ on recursion vs. iteration; $56.7\% \rightarrow 40.0\%$ on concepts; $77.8\% \rightarrow 55.6\%$ on stories; Fig. 3A). This suggests that senders begin with more human-aligned conventions, but converge on somewhat more opaque, model-specific visualizations to successfully communicate with receivers. By contrast, free-form PIL drawings remain highly legible across turns ($\sim$79–91% on concepts; 100% for short stories).

6 Discussion and Conclusion

We investigated how VLMs interpret and produce visual meaning through image steering, communication games, and an exploratory human comparison. Simple visual symbols like basic colored shapes steer model behavior in nonintuitive ways that models do not faithfully report. Models can also communicate abstract concepts and stories through creative visualizations, but their generation and interpretation of visual meaning are dissociated: visualizations meant to convey a target meaning can fail to be interpreted as intended. A preliminary human study further suggests that models' visual associations sometimes diverge from human interpretation. These results point to visual communication as a rich and underexplored dimension of VLM behavior. They also raise practical concerns: as models are increasingly deployed as agents in visually rich environments, the influence of visual inputs on their behavior should be transparent, and their visual outputs legible to users. Our findings suggest neither can yet be assumed.

Where does visual steering come from? This is an open question, and using interpretability tools to understand the internal mechanisms behind visual steering is an exciting future direction. One potential mechanism is that image representations land near representations of task-relevant concepts (e.g., "recursion") in the model's semantic space; this might explain why steering effects of shapes appear to be more graded than cleanly language-like (Fig. 1C). The existence of some shared associations across model families hints at cues absorbed from common pretraining data (App. B.4). Using open VLMs to measure the representational distance between visual symbols and task-relevant concepts and analyzing internet data for co-occurrence would yield insight. Finally, a hypothesis for why models neither report these associations nor use them when generating is that image-to-text objectives (e.g., captioning, VQA) build associative pathways with little pressure to align them with generation. Closing this gap may require training on visually-grounded communicative interactions.

Could models use visualizations as a covert communication channel? Concerns about "neuralese" (Andreas et al., 2017) and steganographic collusion (Roger & Greenblatt, 2023; Motwani et al., 2024) ask whether models might communicate in human-illegible codes. Our results suggest that deliberate visual collusion between models is unlikely for now: associations are weakly shared across models, and senders struggle to access their own. Nevertheless, our communication games provide a hint that optimizing visualizations for receivers can push senders away from human-legible conventions (Sec. 5). Furthermore, since visual influence is often unverbalized, trace-based oversight may miss it. As agents increasingly work over screenshots and GUIs, the nearer-term risk is unintentional visual steering, which future work should characterize in naturalistic agent workflows.

Will visual steering emerge or disappear with scale? We observe that less capable models from each family tend to be more sensitive to steering by Gaussian noise images or naturalistic photos, while frontier models largely ignore these inputs (App. B.5). Capability may thus shift visual steering from indiscriminate sensitivity toward more structured, symbol-specific influence; if so, steering with visual symbols may be a reflection of richer visual semantics rather than a lack of robustness.

Limitations Our study has several limitations. First, our tasks are synthetic and designed to understand model behavior. How visual steering and visually-grounded communication manifest in real-world contexts remains an important open question. Second, our human study is exploratory and small-scale. A more systematic investigation with larger and participant pools is needed to draw robust conclusions about human–model alignment in visual meaning.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets (images of the visual symbols, coding problem descriptions, abstract concept sets, three-sentence short stories, topics for the writing task) and implement methods (code for running the experiments and some of the quantitative analyses). We did not use generative AI tools to help develop theoretical models or conceptual frameworks, design or provide feedback on research methodology or experiments, or interpret results, and formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images (bar charts, heatmaps, scatter plots), edit a research paper to improve readability (used for writing polish in places), identify relevant literature (help with related work to ensure we didn't miss relevant citations), and format references (provide BibTeX entries).

We have reviewed all AI-assisted work. We checked each item in all generated datasets and manually refined them (e.g., removed pixel-level artifacts or mistakes in visual symbol images, edited the concept sets and short stories). All LLM-generated code for experiments and analyses were read and corrected by at least one author. Charts and plots were manually edited in Adobe Illustrator for style and checked for correctness. Finally, all citations were audited for accuracy by two authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Institutional review and approval In Sec. 5, we report results from a small-scale, exploratory human survey that we conducted to evaluate human interpretations of visual symbols and model-generated drawings. The study protocol was fully reviewed and approved by our institution's research ethics committee.

Participant consent and freedom of withdrawal Prior to starting the survey, we obtained explicit participant consent using a consent form, which provided all participants with a clear explanation of the study's purpose, data usage, and the estimated completion time (10–15 minutes). Standard ethical procedures for informed consent were strictly followed. Participants were explicitly and repeatedly informed that their participation was entirely voluntary and that they could choose to terminate the survey or opt out at any point without penalty. Following the completion of the survey, participants were debriefed regarding the underlying research goals.

Data privacy and security The survey was hosted entirely on a secure local server managed by our institution. All collected responses were completely anonymized at the point of collection; no personally identifiable information or IP addresses were stored or linked to the experimental data.

REPRODUCIBILITY STATEMENT

All images used in the visual steering experiments are displayed in, or can be straightforwardly reconstructed from, App. B. We provide all prompts and task descriptions in App. B.3, and the full concept sets and short stories used in the communication experiments in App. C.2; where data were generated synthetically, we also include the generation prompts. App. A.1 lists exact model checkpoints, API model identifiers, and all sampling hyperparameters, along with the dates on which API models were queried. Our complete methodology for classifying reasoning traces, including classification prompts and regex expressions, is described in App. B.8.

REFERENCES

Jacob Andreas, Anca Dragan, and Dan Klein. Translating neuralese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 232–242, 2017.

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

Jiyun Bae, Hyunjong Ok, Sangwoo Mo, and Jaeho Lee. Understanding the effects of distractors on reasoning vision-language models. *arXiv preprint arXiv:2511.21397*, 2025.

Eugene Bagdasaryan, Tsung-Yin Hsieh, Ben Nassi, and Vitaly Shmatikov. Abusing images and sounds for indirect instruction injection in multi-modal LLMs. *arXiv preprint arXiv:2307.10490*, 2023.

Luke Bailey, Euan Ong, Stuart Russell, and Scott Emmons. Image hijacks: Adversarial images can control generative models at runtime. In *International Conference on Machine Learning (ICML)*, 2024.

Sriram Balasubramanian, Samyadeep Basu, and Soheil Feizi. A closer look at bias and chain-of-thought faithfulness of large (vision) language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13406–13439, 2025.

Jan Betley, Xuchan Bao, Martín Soto, Anna Sztyber-Betley, James Chua, and Owain Evans. Tell me about yourself: LLMs are aware of their learned behaviors. In *International Conference on Learning Representations (ICLR)*, 2025.

Felix J. Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking inward: Language models can learn about themselves by introspection. In *International Conference on Learning Representations (ICLR)*, 2025.

Mu Cai, Haotian Liu, Dennis Park, Siva Karthik Mustikovela, Gregory P. Meyer, Yuning Chai, and Yong Jae Lee. ViP-LLaVA: Making large multimodal models understand arbitrary visual prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12929–12939, 2024.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14455–14465, 2024.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

James Chua and Owain Evans. Are DeepSeek R1 and other reasoning models more faithful? *arXiv preprint arXiv:2501.08156*, 2025.

Christopher Clark, Jordi Salvador, Dustin Schwenk, Derrick Bonafilia, Mark Yatskar, Eric Kolve, Alvaro Herrasti, Jonghyun Choi, Sachin Mehta, Sam Skjonsberg, Carissa Schoenick, Aaron Sarnat, Hannaneh Hajishirzi, Aniruddha Kembhavi, Oren Etzioni, and Ali Farhadi. Iconary: A pictionary-based game for testing multimodal communication with drawings and text. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5814–5831, 2021.

Brandon Collins, Logan Bolton, Hung Huy Nguyen, Mohammad Reza Taesiri, Trung Bui, and Anh Totti Nguyen. SketchVLM: Vision language models can annotate images to explain thoughts and guide users. *arXiv preprint arXiv:2604.22875*, 2026.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009. doi: 10.1109/CVPR.2009.5206848.

Judith E. Fan, Robert D. Hawkins, Mike Wu, and Noah D. Goodman. Pragmatic inference and visual abstraction enable contextual flexibility during visual communication. *Computational Brain & Behavior*, 3(1):86–101, 2020.

Nicolas Fay, Simon Garrod, Leo Roberts, and Nik Swoboda. The interactive evolution of human communication systems. *Cognitive Science*, 34(3):351–386, 2010.

Simon Garrod, Nicolas Fay, John Lee, Jon Oberlander, and Tracy MacLeod. Foundations of representation: Where might graphical symbol systems come from? *Cognitive Science*, 31(6):961–987, 2007.

Robert Geirhos, Yuxuan Li, Thaddäus Wiedemer, Neha Kalibhat, Zi Wang, Mani Malek, Øyvind Tafjord, Kevin Swersky, Been Kim, and Priyank Jaini. Visual prompt engineering for video models. *arXiv preprint arXiv:2607.25537*, 2026.

Alexandra Gomez-Villa, Pablo Hernández-Cámara, Muhammad Atif Butt, Valero Laparra, Jesus Malo, and Javier Vazquez-Corral. Color names in vision-language models. *arXiv preprint arXiv:2509.22524*, 2025.

Google DeepMind. Gemini 2.5 flash image: Nano banana. https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/, 2025.

Sherzod Hakimov, Mattia D'Agostini, Ivan Samodelkin, and David Schlangen. The image reconstruction game: Drawing common ground through iterative multimodal dialogue. *arXiv preprint arXiv:2606.01901*, 2026.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Robert D. Hawkins, Megumi Sano, Noah D. Goodman, and Judith E. Fan. Visual resemblance and interaction history jointly constrain pictorial meaning. *Nature Communications*, 14(1):2199, 2023.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Tom Kouwenhoven, Max Peeperkorn, and Tessa Verhoef. Searching for structure: Investigating emergent communication with large language models. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, 2025a.

Tom Kouwenhoven, Kiana Shahrasbi, and Tessa Verhoef. Cross-modal associations in vision and language models: Revisiting the Bouba-Kiki effect. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025b.

Manishit Kundu, Sumit Shekhar, and Pushpak Bhattacharyya. Looking beyond the pixels: Evaluating visual metaphor understanding in VLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*, 2020.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations (ICLR)*, 2017.

David Lewis. *Convention: A Philosophical Study*. Harvard University Press, Cambridge, MA, 1969.

Xiang Lisa Li, Vaishnavi Shrivastava, Siyan Li, Tatsunori Hashimoto, and Percy Liang. Benchmarking and improving generator-validator consistency of language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Yijun Liang, Ming Li, Chenrui Fan, Ziyue Li, Dang Nguyen, Kwesi Adu Cobbina, Shweta Bhardwaj, Jiuhai Chen, Fuxiao Liu, and Tianyi Zhou. ColorBench: Can VLMs see and understand the colorful world? A comprehensive benchmark for color perception, reasoning, and robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision (ECCV)*, 2024.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

Sachit Menon, Richard Zemel, and Carl Vondrick. Whiteboard-of-thought: Thinking step-by-step across modalities. *arXiv preprint arXiv:2406.14562*, 2024.

Andreea-Daniela Mihai and Jonathon Hare. Learning to draw: Emergent communication through sketching. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 19222–19234, 2021. URL https://arxiv.org/abs/2106.02067.

Sumeet R Motwani, Mikhail Baranchuk, Martin Strohmeier, Vijay Bolina, Philip H Torr, Lewis Hammond, and Christian S de Witt. Secret collusion among ai agents: Multi-agent deception via steganography. *Advances in Neural Information Processing Systems*, 37:73439–73486, 2024.

Kenneth J. K. Ong. The effects of visual priming on cooperative behavior in vision-language models. *arXiv preprint arXiv:2604.27953*, 2026.

OpenAI. Addendum to GPT-4o system card: 4o image generation. https://openai.com/index/addendum-to-gpt-4o-system-card-4o-image-generation/, 2025.

Mansi Phute and Ravikumar Balakrishnan. VISOR: Visual input-based steering for output redirection in vision-language models. *arXiv preprint arXiv:2508.08521*, 2025.

Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. In *AAAI Conference on Artificial Intelligence*, 2024.

Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Asian Conference on Computer Vision (ACCV)*, 2024.

Fabien Roger and Ryan Greenblatt. Preventing language models from hiding their reasoning. *arXiv preprint arXiv:2310.18512*, 2023.

Mohammad Shahab Sepehri, Berk Tinaz, Zalan Fabian, and Mahdi Soltanolkotabi. Hyperphantasia: A benchmark for evaluating the mental visualization capabilities of multimodal LLMs. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

Chufan Shi, Cheng Yang, Yaokang Wu, Linghao Jin, Bo Shui, Taylor Berg-Kirkpatrick, and Xuezhe Ma. Are VLMs seeing or just saying? uncovering the illusion of visual re-examination. In *International Conference on Machine Learning (ICML)*, 2026.

Yizheng Sun, Mochuan Zhan, Yanan Ma, Jia Tong See, Yifan Wang, Ziyi Wang, Hao Li, Yang Cui, Wenhao Cai, Jingyu Sun, Chenghua Lin, Riza Batista-Navarro, and Jingyuan Sun. Are reasoning vision-language models robust to semantic visual distractions? *arXiv preprint arXiv:2606.08894*, 2026.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Yael Vinker, Tamar Rott Shaham, Kristine Zheng, Alex Zhao, Judith E. Fan, and Antonio Torralba. SketchAgent: Language-driven sequential sketch generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. Vision language models are biased. In *International Conference on Learning Representations (ICLR)*, 2026.

Junda Wu, Zhehao Zhang, Yu Xia, Xintong Li, Zhaoyang Xia, Aaron Chang, Tong Yu, Sungchul Kim, Ryan A. Rossi, Ruiyi Zhang, Subrata Mitra, Dimitris N. Metaxas, Lina Yao, Jingbo Shang, and Julian McAuley. Visual prompting in multimodal large language models: A survey. *arXiv preprint arXiv:2409.15310*, 2024.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Jana Zeller, Thaddäus Wiedemer, Fanfei Li, Thomas Klein, Prasanna Mayilvahanan, Matthias Bethge, Felix Wichmann, Ryan Cotterell, and Wieland Brendel. MENTISOCULI: Revealing the limits of reasoning with mental imagery. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://jana-z.github.io/mentis-oculi.

Jana Ricarda Zeller, Aleksandar Shtedritski, and Christian Rupprecht. Highlight: Learning visual prompts for vision-language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Huanyu Zhang, Xuehai Bai, Chengzu Li, Chen Liang, Haochen Tian, Haodong Li, Ruichuan An, Yifan Zhang, Anna Korhonen, Zhang Zhang, Liang Wang, and Tieniu Tan. How well do models follow visual instructions? VIBE: A systematic benchmark for visual instruction-driven image editing. *arXiv preprint arXiv:2602.01851*, 2026.

Shan Zhao, Zhao Yang, Tianwei Yan, Yusong Gong, Qian Wan, Shizhao Chen, Shezheng Song, Chengyu Wang, and Meng Wang. Seeing the abstract: A benchmark for visual-only metaphor understanding in multimodal large language models. In *Findings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

A Experimental setup

A.1 Model evaluation details

Our main evaluation focuses on three representative multimodal models: Claude Opus 4.7, Gemini 3.1 Pro, and GPT-5.4. We additionally evaluate Claude Sonnet 4.5, Gemma 4 31B-IT, and GPT-4o in some experiments. All models are accessed through their respective official provider APIs (Anthropic Messages API, Google Gemini API, and OpenAI Chat Completions API) via LiteLLM. No system prompts are provided. In general, we set reasoning effort to be high and choose all other hyperparameters based on common practice.

We use the following API evaluation parameters for each model:

- **Claude Opus 4.7**: `model` = `claude-opus-4-7`, `temperature` = 1.0, `max_output_tokens` = 1000, `enable_thinking` = `true`, `thinking_budget` = 3000, `effort` = `high` (`top_p` and `top_k` are not supported).
- **Gemini 3.1 Pro**: `model` = `gemini-3.1-pro-preview`, `temperature` = 1.0 (set to 0.7 in the config file and overridden to 1.0 when `enable_thinking` = `true`), `top_p` = 0.95, `top_k` = 64, `max_output_tokens` = 1000, `enable_thinking` = `true`, `thinking_budget` = 1024.
- **GPT-5.4**: `model` = `gpt-5.4-2026-03-05`, `temperature` = 1.0, `top_p` = 1.0, `max_output_tokens` = 4000, `enable_thinking` = `true`, `thinking_budget` = 1024, `reasoning_effort` = `high`.
- **Claude Sonnet 4.5**: `model` = `claude-sonnet-4-5-20250929`, `temperature` = 1.0, `top_p` = 1.0, `max_output_tokens` = 1000, `enable_thinking` = `true`, `thinking_budget` = 3000.
- **Gemma 4 31B-IT**: `model` = `gemma-4-31b-it`, `temperature` = 0.7, `top_p` = 0.95, `top_k` = 64, `max_output_tokens` = 1000, `enable_thinking` = `false`.
- **GPT-4o**: `model` = `gpt-4o-2024-08-06`, `temperature` = 1.0, `top_p` = 1.0, `max_output_tokens` = 1000, `enable_thinking` = `true`, `thinking_budget` = 3000.

A.2 GEMINI 3.5 FLASH JUDGE VALIDATION

We use Gemini 3.5 Flash as an automated judge to classify model-generated answers to coding problems (e.g., "recursive" vs. "iterative"). The judge is queried at temperature $= 0.0$ using the following prompt:

Automated judge prompt (code classification)

Analyze the following Python code and determine which of the allowed strategies it uses to solve the problem.

Problem: {`name`}
Description: {`problem_statement`}

Allowed Strategies: [{`strategy0`}, {`strategy1`}]

Code:

{`code`}

Choose the strategy from the allowed strategies list that best describes the code. If the code does not implement the solution or uses a completely different strategy not listed, choose "other".

We verify judge accuracy by comparing a subset of Gemini 3.5 Flash's code classifications ($n = 180$, 60 randomly selected code snippets from each of the three main models) against the classifications of two other models (Claude Opus 4.7 and GPT-5.4) using identical prompts. We find 100% agreement between all judges. Additionally, we compare judge classifications to a subset of $n = 20$ author-provided labels,[4] finding 100% agreement.

B VISUAL STEERING DETAILS

B.1 TASK AND PROMPT DETAILS

For all visual steering experiments, prompts follow the same general formula: a 400×400 pixel image of a visual symbol is placed in the beginning of context, followed by an instruction to solve a task, and finally a brief problem statement.[5] We use the following two prompts (either explicitly directing models to look at the image, used to evaluate visual steering, or making no reference to the image, used to evaluate no-image baselines or silent injections):

Explicit image instruction prompt (visual steering)

{`input_image`}

Solve the following programming problem in Python. Looking at the provided image, choose a coding strategy ({`strategy0`} or {`strategy1`}) and solve the problem. Write a complete function. Do not include explanation, just the code.

Problem: {`problem_statement`}

Baseline prompt (no-image baseline and silent injection)

Solve the following programming problem in Python. Choose a coding strategy ({`strategy0`} or {`strategy1`}) according to your personal preference and solve the problem. Write a complete function. Do not include explanation, just the code.

Problem: {`problem_statement`}

The order of the two candidate strategies (e.g. recursion vs. iteration), labeled `strategy0` and `strategy1` above, is balanced across generations ($n = 10$ samples for each, 20 total). In general, option order has a negligible effect on results. A minority of visual symbols induce some apparent bias towards the first option or the second option (i.e. there is a nonzero difference between the proportions of recursion; these are 20% of shapes for Claude Opus 4.7, 10.8% for Gemini 3.1 Pro,

[4] The author labeled the code snippets as "recursive" or "iterative" without access to the automated judge's decisions.

[5] We study image placement and prompt variations in Appendix B.2.4.

11.8% for GPT-5.4 on inorder binary tree traversal), but the difference in steering values is fairly small ($\Delta = |0.14|$ for Claude Opus 4.7, $\Delta = |0.06|$ for Gemini 3.1 Pro, $\Delta = |0.06|$ for GPT-5.4 on average). The top shapes in terms of steering value difference are arrow shapes (u turn, left arrow, right arrow, etc.), which may occasionally be interpreted in models as "choose the left / right option."

Binary tree traversal task We evaluate the *binary tree inorder traversal* problem for the primary visual steering experiments in Section 3, which has the following problem statement:

> Given the root of a binary tree, return the inorder traversal of its nodes' values. Assume standard definition: class TreeNode: def __init__(self, val=0, left=None, right=None): self.val = val; self.left = left; self.right = right

Visual symbol library See Figure 4. We construct a dataset of 288 visual symbols using 12 colors and 24 shapes. Shapes were chosen to represent a range of common icons and basic geometric figures. We deliberately construct simple visual symbols in order to study their meaning in isolation. We also include two "x" symbols that differ only slightly (by weight and size) as a basic sanity check for model consistency, as well as a caution triangle (nearly identical to the basic triangle, but carrying very different semantic meaning). All symbols are cleanly rendered over a plain white background.



Figure 4: **All shapes and colors in the visual symbol library.** Each symbol shows an example of the color and shape indicated by the label; the library includes all 288 combinations of the 12×24 colors and shapes.

B.2 Recursion vs. iteration steering

This section contains extended results for recursion vs. iteration steering. In App. B.2.1, we include full recursion steering heatmaps for inorder binary tree traversal as well as heatmaps aggregated across additional baseline-recursive and baseline-iterative coding problems (see App. B.3.1 for all task descriptions). The rest of the sections focus on inorder binary tree traversal for clarity.

B.2.1 Full steering heatmaps

Figure 5 shows the complete steering heatmaps over the full 288-symbol library (24 shapes $\times$ 12 colors) on inorder binary tree traversal. Figure 6 shows recursion steering heatmaps aggregated across additional baseline-iterative coding problems, while Figure 7 shows iteration steering across additional baseline-recursion problems. Shapes are ordered by mean steering effect within each panel, so the column ordering differs across models.

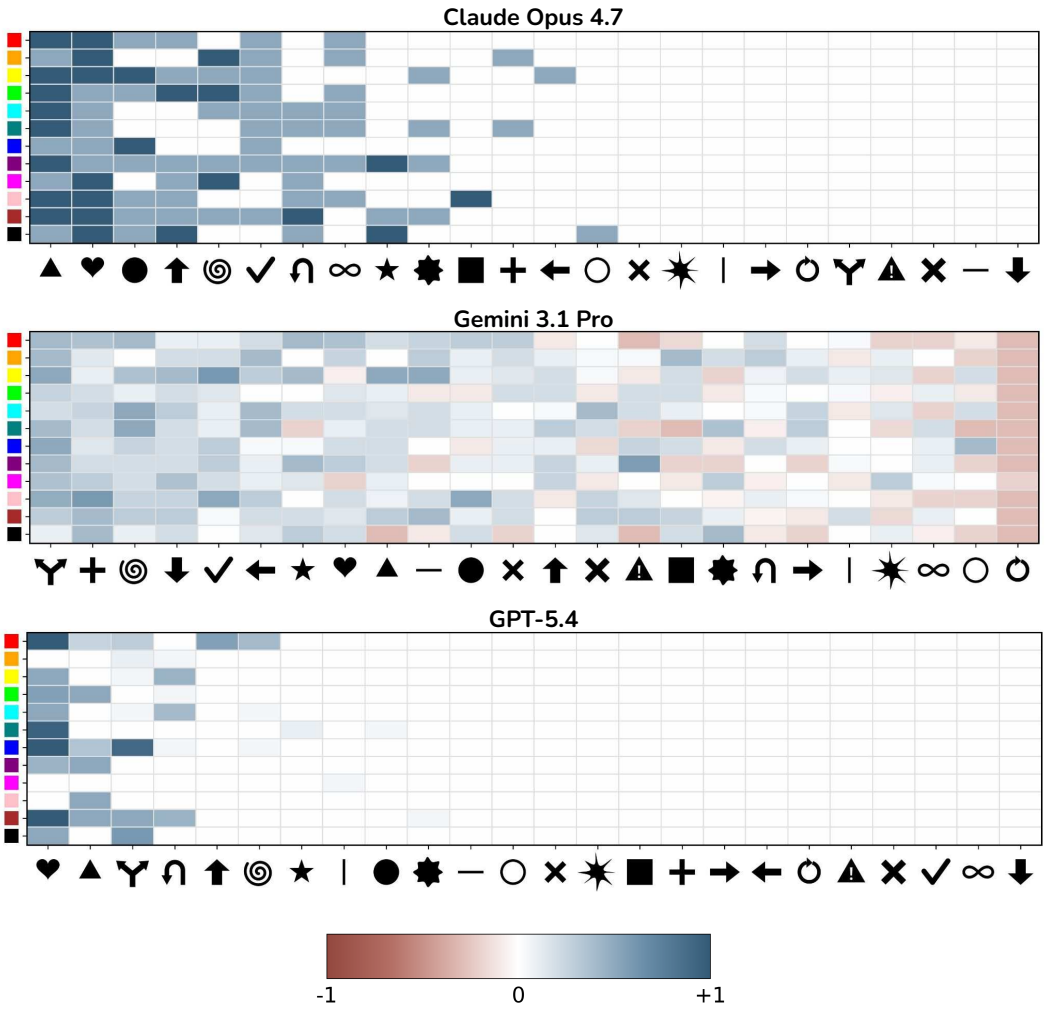

Figure 5: **Full steering heatmaps for inorder binary tree traversal**, where the model is instructed to look at the image ("Explicit image instruction prompt," Appendix B.1). Each heatmap is one model; rows are the 12 colors, and columns are the 24 shapes. Cell values are the steering effect (i.e. delta $\Delta P(\text{recursion})$ relative to that model's text-only baseline), on a diverging scale from -1 (red, towards iteration) to $+1$ (blue, towards recursion).

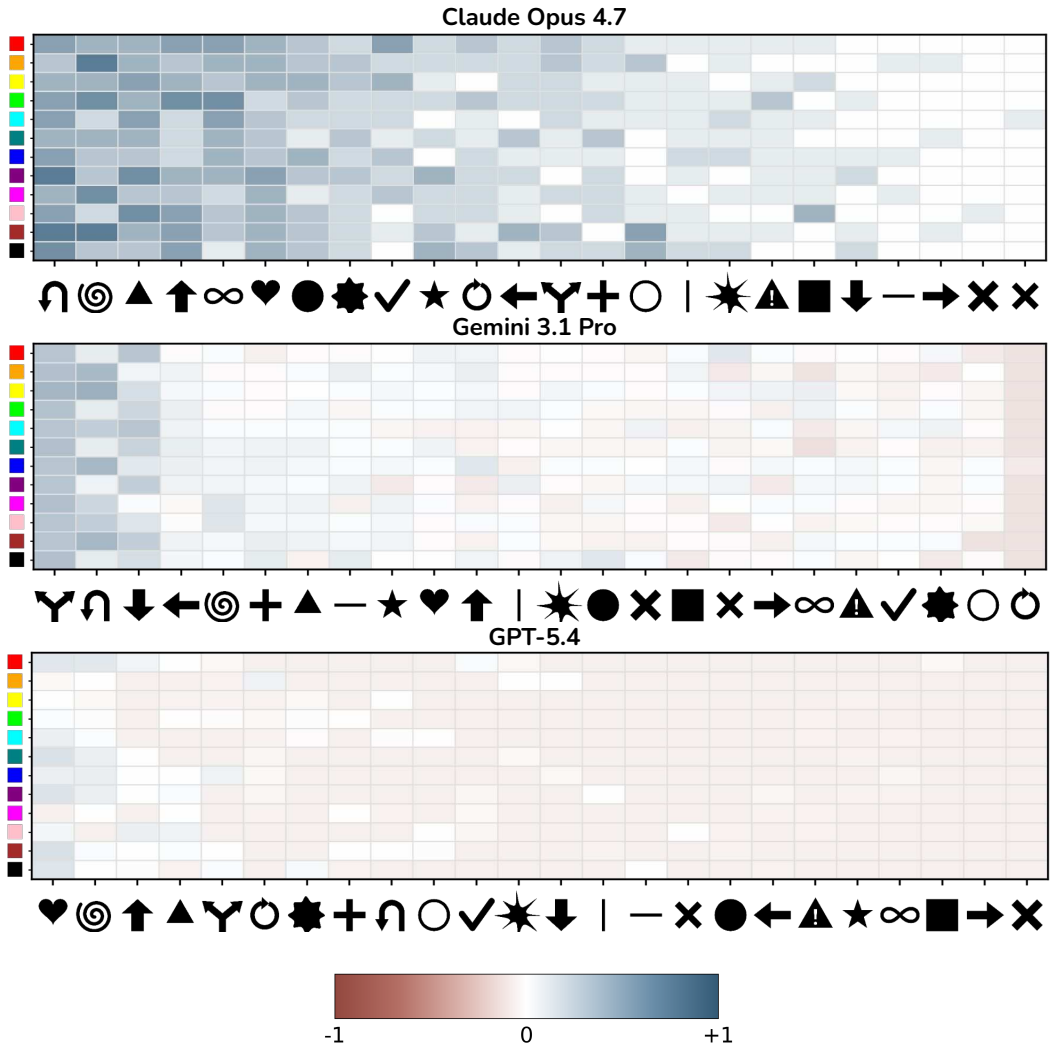

Figure 6: **Recursion steering heatmaps aggregated across baseline-iterative coding problems**: $N = 4$ problems for Claude Opus 4.7, $N = 3$ for Gemini 3.1 Pro, and $N = 4$ for GPT-5.4. Because all problems for each model have baseline iterative solutions, the cell values show steering towards or away from recursion. Axes and color scale are as in Figure 5.

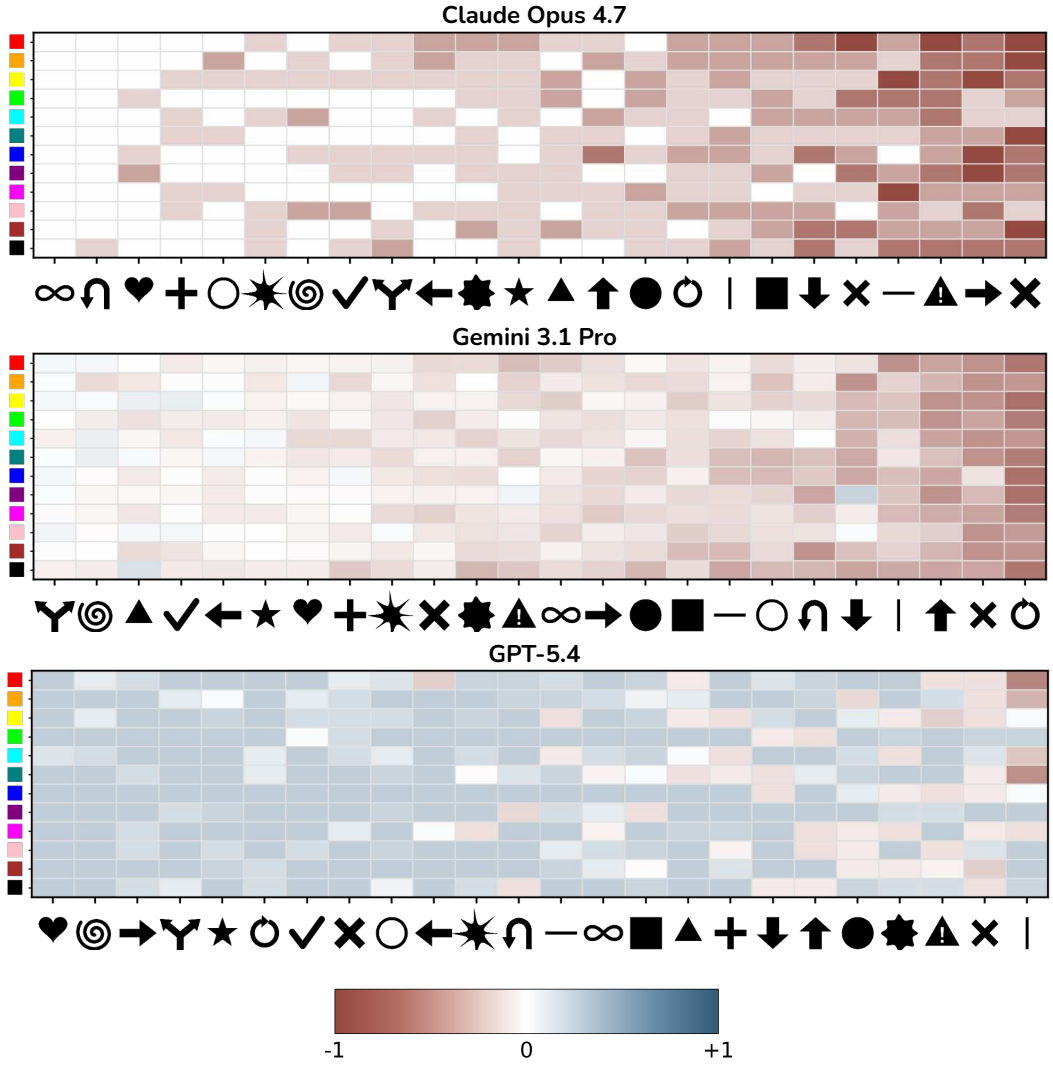

Figure 7: **Iteration steering heatmaps aggregated across baseline-recursive coding problems**: $N = 2$ problems for Claude Opus 4.7, $N = 3$ for Gemini 3.1 Pro, and $N = 2$ for GPT-5.4. Because all problems for each model have baseline recursive solutions, the cell values show steering towards or away from iteration. Axes and color scale are as in Figure 5.

B.2.2 ADDITIONAL STEERING BASELINES

Mirroring how we measure visual steering (Section 3), steering effects for all baselines are computed as the change in $P(\text{recursion})$ relative to the no-image baseline for a given model. For all text-based baselines (text instruction, captions), we repeat the steering token(s) (e.g. "recursion" or "An image of a pink heart") until the length of the insertion matches the number of image tokens for a given model.[6] This controls for prompt length between image-based and text-based prompts.

Text instruction steering We measure whether models can follow direct instructions specified in text (e.g. to use recursion). We evaluate two text instruction conditions, in each case substituting in the target `strategy` word (e.g. "recursive" or "iterative") repeated enough times to match or nearly match the token count an image would occupy for a given model:

Explicit text instruction prompt

{strategy_repeated}

Solve the following programming problem in Python. Solve it using a {strategy} approach. Write a complete function. Do not include explanation, just the code.

Problem: {problem_statement}

Silent text injection prompt

{strategy_repeated}

Solve the following programming problem in Python. Choose a coding strategy ({strategy0} or {strategy1}) according to your personal preference and solve the problem. Write a complete function. Do not include explanation, just the code.

Problem: {problem_statement}

These two prompts mirror those used for visual steering and silent image injection respectively (Appendix B.1).

We find that text instruction and silent text injection both steer all models towards the target strategy 100% of the time, indicating that models are capable of solving problems according to direct instructions and providing a ceiling for the other steering effects we observe.

Caption steering We test whether the visual steering effect is specific to the visual channel (i.e. pixel inputs processed by a vision encoder) by replacing each image with a short text caption. Captions are generated deterministically using a template: "`An image of a {color} {shape}`". We repeat the caption string n times to roughly match the number of tokens the corresponding image consumes for that model, where n is computed per model and per caption from measured image token counts. We run this image caption control experiment for all visual symbols in the 288-symbol library on inorder binary tree traversal. As in the visual steering experiments, we use an explicit instruction prompt ("Explicit image instruction prompt," Appendix B.1) and a silent injection prompt ("Baseline prompt" with the caption string at the beginning, Appendix B.1). We use an identical explicit instruction prompt to the visual steering experiments (i.e., the caption is referred to as an "image" rather than a caption).[7]

Caption steering results for the top-three steering symbols are reported alongside image steering in Figure 1B. Fig. 8A shows the top three steering captions for each model on inorder binary tree traversal; Fig. 8B compares the actual steering effect of visual symbols (y-axis) to the steering effect of their matching caption (x-axis). We find that Pearson-r values between the steering effect of images of visual symbols and their captions is fairly low: $r = -0.05$ ($p = 0.43$) for Claude Opus 4.7; $r = 0.31$ ($p = 1.4 \times 10^{-5}$) for Gemini 3.1 Pro; $r = 0.17$ ($p = 0.005$) for GPT-5.4. We computed Pearson-r and two-sided p-values across all paired symbols in the 288-symbol library between each symbol's order-balanced recursive solution proportion $P(\text{recursion})$ under explicit

[6] Image and phrase token counts are obtainable through model APIs.

[7] We tried a variant of the explicit instruction prompt where we told models to instead look at the "image caption," but results were largely unchanged.

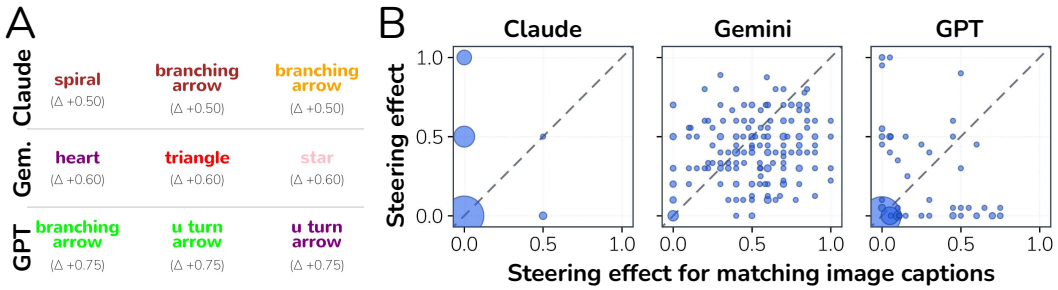

Figure 8: **The steering effect of text captions largely does not match that of images. A) Top three steering captions for each model.** The colors of the text show the corresponding color. Captions are formatted as, e.g., "An image of a brown spiral." **B) The steering effect of captions decouples from the steering effect of images of visual symbols.**

image steering and under its matching text caption ($N = 288$ for Claude Opus 4.7 and GPT-5.4; $N = 190$ for Gemini 3.1 Pro after excluding API errors).

Gemini 3.1 Pro shows the highest image-caption correlation amongst the three models. This may suggest that this model engages in somewhat more "visual thinking" when given text descriptions of visual elements, although further work is needed.

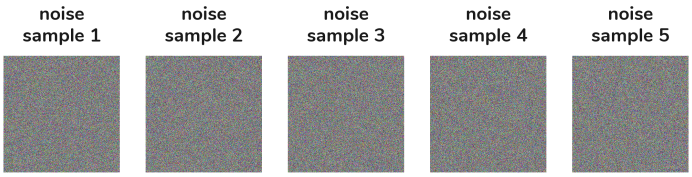

Figure 9: **Images of Gaussian noise** ($N = 5$) **used to measure the noise image steering baseline.**

Gaussian noise image steering To test whether visual steering is a byproduct of inserting generic visual tokens into the context, we construct a control dataset of five synthetic noise images. Each is a 400×400 RGB image (matching the size of the symbol images) whose pixels are drawn IID from $\mathcal{N}(128, 64)$ and clipped to $[0, 255]$ (see Figure 9). They are run through the identical evaluation pipeline as the symbol library (Section 3).



Figure 10: **Naturalistic animal images** ($N = 20$) **used to measure the ImageNet steering baseline.**

ImageNet steering While random noise fails to steer similarly to visual symbols (or at all), those images are very far out of distribution. What about generic naturalistic images? We test whether models are steered by $N = 20$ images representing 20 different ImageNet animal classes, providing another control for the effect of generic (but within-distribution) visual tokens. The animal classes

and images were both randomly drawn from the validation set of ImageNet, assuring a wide variety of colors, poses, backgrounds, and image compositions (see Figure 10).

B.2.3 VISUAL PERTURBATION EXPERIMENTS

Pixel shuffling perturbations To test whether visual steering is driven by global shape structure or low-level color statistics, we progressively shuffle a fraction of pixels (0% to 100% in 10% increments; Fig. 11A) across five representative symbols (pink heart, green triangle, blue triangle, blue circle, and red square) on inorder binary tree traversal ($n = 10$ samples per condition). This preserves the proportion of colored pixels but disrupts shape features. For Claude Opus 4.7 and Gemini 3.1 Pro models, disrupting shape geometry sharply diminishes recursion steering (Fig. 11B). Gemini 3.1 Pro appears to be more sensitive to color and lower level noise, especially in the middle stages of pixel shuffling.

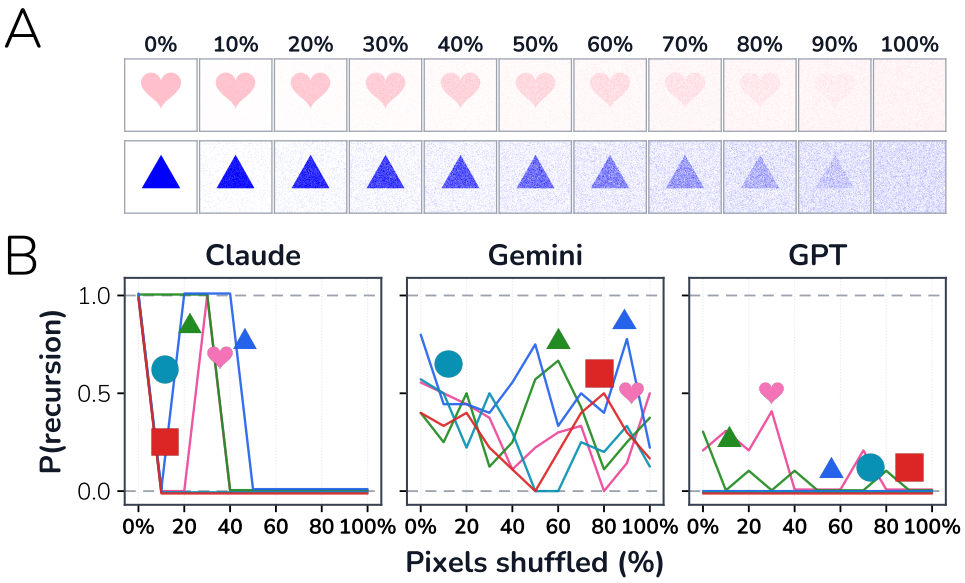

Figure 11: **Disrupting shape structure via pixel shuffling disrupts visual steering at high enough percentages. A) Example pixel-shuffled stimuli.** Progressive pixel shuffling (0% to 100%) preserves global color proportions while progressively destroying shape geometry and adding low-level noise (shown for the pink heart and blue triangle). **B) Recursion proportion P(recursion) across pixel shuffle levels.** Lines show P(recursion) on inorder binary tree traversal across five symbols (visualized next to their lines) for Claude Opus 4.7, Gemini 3.1 Pro, and GPT-5.4.

Rotation perturbations To test whether visual steering is rotation invariant (purely shape based) or instead is sensitive to symbol orientation, we rotate the three representative symbols (pink heart, green triangle, and blue triangle) from 0° to 315° in 45° increments ($n = 10$ samples per condition; Fig. 12A). Steering appears to be highly sensitive to orientation (Fig. 12B), mostly peaking near canonical orientations (like upwards or downwards triangle) while dropping sharply at other orientations. However, steering is not perfectly aligned with canonical orientations (especially for Gemini 3.1 Pro and GPT-5.4); the full effect dimensions like orientation require further study.

Position and size perturbations We evaluate sensitivity to spatial layout by sweeping canvas position (top left, center, bottom right) and symbol size (small, medium, large) across the same five symbols (Fig. 13A). Models exhibit distinct spatial sensitivities (Fig. 13B, Fig. 14): Claude Opus 4.7 steers most strongly for centered (0.83 mean P(recursion)) and large (0.79) symbols, whereas placing small or medium symbols in the bottom-right corner suppresses steering (0.00 and 0.38). GPT-5.4 exhibits a subtle top-to-bottom positional gradient, steering most strongly at the top left (0.20 mean, peaking at 0.70 for the medium green triangle) and nearly vanishing at the bottom right.

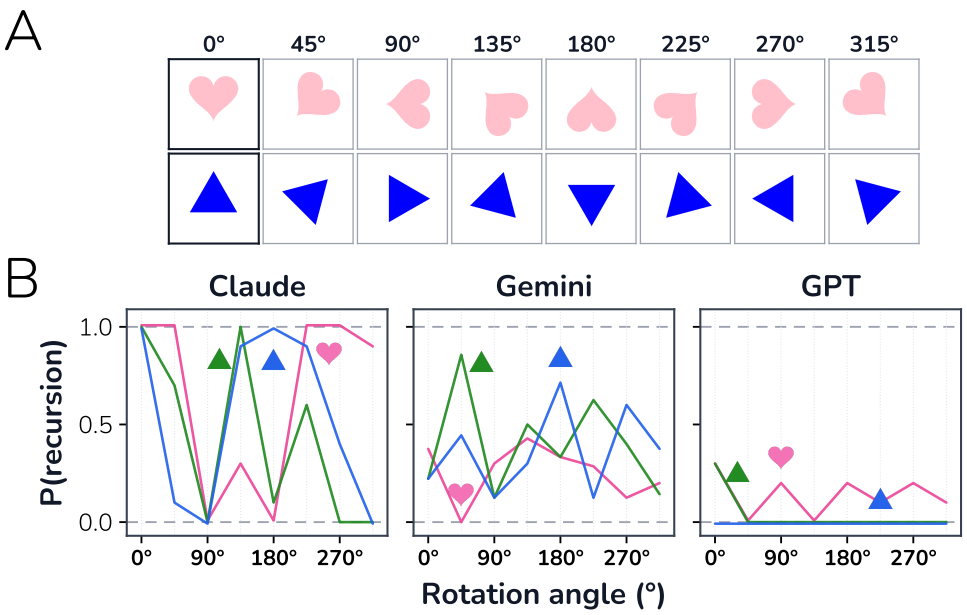

Figure 12: **Visual steering is highly sensitive to symbol orientation. A) Example rotated stimuli.** Visual symbols rotated from 0° (prototypical upright orientation, outlined in bold) to 315° in 45° increments. **B) Recursion proportion P(recursion) across rotation angles.** Lines show P(recursion) on inorder binary tree traversal across the three orientation-dependent symbols for Claude Opus 4.7, Gemini 3.1 Pro, and GPT-5.4.

By contrast, Gemini 3.1 Pro maintains a relatively stable mean recursion rates, though individual symbols shift substantially with position and scale.

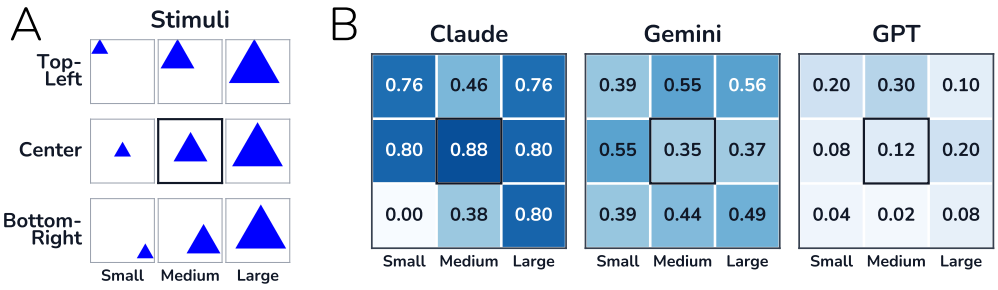

Figure 13: **Canvas position and symbol size modulate visual steering. A) Stimulus grid.** Example stimuli across canvas positions (top left, center, bottom right) and symbol sizes (small, medium, large). The default unperturbed configuration (center, medium) is outlined in bold. **B) Mean P(recursion) across the 3×3 position-by-size sweep.** Each cell shows P(recursion) averaged over the five base symbols on inorder binary tree traversal.

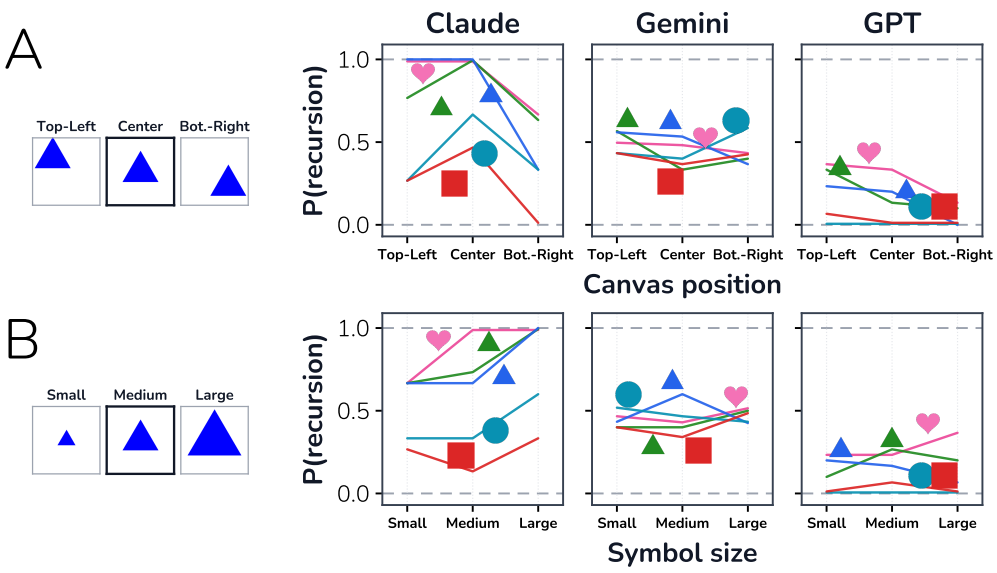

Figure 14: **Per-symbol steering across canvas position and symbol size.** Lines track $P(\text{recursion})$ for each of the five symbols across Claude Opus 4.7, Gemini 3.1 Pro, and GPT-5.4. **A) Sensitivity to canvas position** (marginalized across symbol sizes). **B) Sensitivity to symbol size** (marginalized across canvas positions).

B.2.4 Visual steering prompt robustness

Prompt variations To test whether visual steering depends on the exact wording or syntactic structure of our default prompt (App. B.1), we evaluate three controlled prompt variations on binary tree inorder traversal across all 288 visual symbols:

- **Variation 1 (Lexical paraphrase):** Replaces key verbs and nouns (e.g., "Looking at" → "Considering", "choose a coding strategy" → "select an implementation approach") while preserving syntactic structure, testing sensitivity to surface-level word choice.
- **Variation 2 (Clause reordering):** Moves the code-only formatting constraints to the opening clause and places the visual reference and strategy choice immediately before the problem statement, testing sensitivity to clause order and recency.
- **Variation 3 (Neutralized choice language):** Removes explicit "choose" or "select" language in favor of neutral phrasing ("solve the following programming problem using either a {`strategy0`} or {`strategy1`} strategy"), testing whether visual steering requires explicitly instructing the model to make a choice.

Prompt variation 1 (lexical paraphrase)

{`input_image`}

Implement a Python solution for the programming problem below. Considering the provided image, select an implementation approach ({`strategy0`} or {`strategy1`}) to solve the task. Provide a complete function. Output only the code, without any explanation.

Problem: {`problem_statement`}

Prompt variation 2 (clause reordering)

{`input_image`}

Write a complete Python function for the problem below and return only the code with no explanation. Referring to the provided image, pick either a {`strategy0`} or {`strategy1`} strategy when implementing your solution.

Problem: {`problem_statement`}

Prompt variation 3 (neutralized choice language)

{input_image}

Based on the provided image, write a complete Python function to solve the following programming problem using either a {strategy0} or {strategy1} strategy. Respond with code only and no explanation.

Problem: {problem_statement}

Across models, which symbols steer toward recursion versus iteration remains consistent across rephrasings. When the prompt retains explicit choice language ("select" in Variation 1 or "pick" in Variation 2), shape-level steering profiles correlate strongly with the default prompt across all 24 shapes ($r = 0.94$ and 0.68 for GPT-5.4; $r = 0.85$ and 0.66 for Claude Opus 4.7; $r = 0.60$ and 0.54 for Gemini 3.1 Pro), preserving top recursive shapes. By contrast, removing the "choose"/"select" language in Variation 3 ("using either a recursive or iterative strategy") nearly eliminates visual steering in models with a 100% iterative baseline prior (Claude Opus 4.7 and GPT-5.4), whereas Gemini 3.1 Pro (which has a balanced baseline prior) maintains strong shape-level correlation with the default prompt ($r = 0.77$). This suggests that without explicitly directing models to "choose" a strategy, models with very strong priors will default to those priors.

Image placement In our main experiments, we place the image at the start of the context. We try placing images in two additional locations in the input: in the *middle* of context (after the instruction but before the problem statement), and at the *end*. We find that image placement exerts a rather strong effect on visual steering: placing images in the middle largely supresses the effect ($\Delta + 0.19 \longrightarrow \Delta + 0.08$ for Claude Opus 4.7, $\Delta + 0.01 \longrightarrow \Delta - 0.12$ for Gemini 3.1 Pro, and $\Delta + 0.05 \longrightarrow \Delta + 0.02$ for GPT-5.4), while placing images at the end changes which symbols exert the strongest effect (Fig. 15). One possible explanation for this is attention: when an image appears at the start of the prompt, its visual tokens are processed first without access to the problem description, whereas an image placed at the end attends autoregressively to the binary tree task description, potentially shifting the model toward more deliberate, task-conditioned semantic associations.

B.3 Additional steering tasks: descriptions and results

We experiment visual steering with the 288-symbol library on other tasks, as specified below. Similar to the main steering experiments, these problems are posed as an request specifying the problem and strategy choices. The model's response is classified by a judge, and we measure delta steering against the model's baseline solution strategy for that problem.

B.3.1 Task descriptions

Recursion vs. iteration We test whether visual steering towards recursion or iteration generalizes to other coding problems with recursive or iterative solutions. In addition to inorder binary tree traversal, we examined five problems: Fibonacci number computation, binary search, maximum depth of a binary tree, postorder binary tree traversal, and preorder binary tree traversal. See Tab. 2 for the problem descriptions we give to models.

DFS vs. BFS To test whether visual steering for coding strategies generalizes beyond recursion vs. iteration, we ran the same protocol on four graph and tree problems where the natural implementation choice is depth-first search (DFS) vs. breadth-first search (BFS): maximum depth of a binary tree, clone graph, symmetric tree, and course schedule. See Tab. 3 for full problem descriptions.

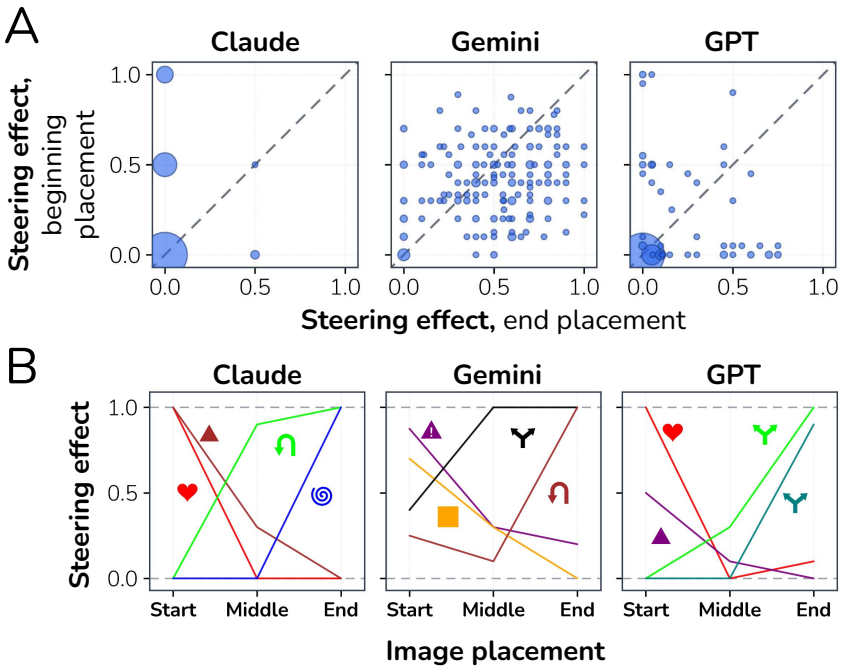

Figure 15: **Placing images at the end of context rather than the beginning changes which symbols steer for recursion. A) The steering effect at the start and end of context decouple most strongly for Claude Opus 4.7 and GPT-5.4.** The y-axis shows the steering effect of visual symbols when placed at the start of context, while the x-axis shows the end of context. Gemini 3.1 Pro remains somewhat correlated. **B) Top steering symbols appear to shift from more geometric to more semantic.** Lines show the steering effect for four individual symbols when placed at the start, middle, and end of context (symbols are visualized next to their lines). Claude Opus 4.7 and GPT-5.4 exhibit a marked shift where more intuitively "recursive" shapes like branching arrows or spirals begin to exert the strongest steering effect. Interestingly, this appears to line up with these models' interpretations and choices in communication games. Gemini 3.1 Pro's top steering shapes remain largely the same.

Table 2: **Recursion vs. iteration coding tasks.** Problem names and exact descriptions provided to models.

Task	Description
Fibonacci number computation	Write a function `fib(n)` that returns the `n`-th Fibonacci number. Assume `n` is a non-negative integer.
Binary tree inorder traversal	Given the root of a binary tree, return the inorder traversal of its nodes' values. Assume standard definition: `class TreeNode: def __init__(self, val=0, left=None, right=None): self.val = val; self.left = left; self.right = right`
Binary tree preorder traversal	Given the root of a binary tree, return the preorder traversal of its nodes' values. Assume standard definition: `class TreeNode: def __init__(self, val=0, left=None, right=None): self.val = val; self.left = left; self.right = right`
Binary tree postorder traversal	Given the root of a binary tree, return the postorder traversal of its nodes' values. Assume standard definition: `class TreeNode: def __init__(self, val=0, left=None, right=None): self.val = val; self.left = left; self.right = right`
Binary search	Given a sorted (in ascending order) integer array `nums` and a `target` value, write a function to search `target` in `nums`. If `target` exists, then return its index. Otherwise, return `-1`.
Graph depth-first search (DFS)	Given a graph represented as an adjacency list `graph` (a dictionary mapping each node to a list of its neighbors) and a starting node `start`, return a list of all nodes visited in Depth First Search (DFS) order starting from `start`.

Table 3: **DFS vs. BFS coding tasks.** Problem names and exact descriptions provided to models.

Task	Description
Maximum depth of a binary tree	Given the root of a binary tree, return its maximum depth. A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.
Clone graph	Given a reference of a node in a connected undirected graph, return a deep copy (clone) of the graph. Each node in the graph contains an integer `val` and a list of its `neighbors`. Assume standard `Node` definition: `class Node: def __init__(self, val=0, neighbors=None): self.val = val; self.neighbors = neighbors if neighbors is not None else []`
Symmetric tree	Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center). Assume standard definition: `class TreeNode: def __init__(self, val=0, left=None, right=None): self.val = val; self.left = left; self.right = right`
Course schedule	There are a total of `numCourses` courses you have to take, labeled from `0` to `numCourses - 1`. You are given an array `prerequisites` where `prerequisites[i] = [a, b]` indicates that you must take course `b` first if you want to take course `a`. Return `True` if you can finish all courses, or `False` otherwise.

Data structures We additionally data structure preference steering on coding problems. These include a dynamic game leaderboard (data structures: hash map vs. binary search tree) and an LRU eviction tracker (doubly linked list vs. ring buffer). See Tab. 4 for the problem descriptions.

Table 4: **Data structure steering tasks.** Candidate data structure pairs and exact problem descriptions provided to models.

Data Structures	Description
Hash map vs. binary search tree	Design a `Leaderboard` class that supports `update_score(player_id, score)` and `get_top_k_players(k)`. Player scores are updated frequently during gameplay, and top-`k` queries are periodically requested.
	Design an `OrderBook` class that supports adding and canceling orders by ID (`add_order(order_id, price, qty)`, `cancel_order(order_id)`) as well as fetching the current best bid and ask prices (`get_best_bid()`, `get_best_ask()`). Orders are frequently added and modified.
	Design an `IPRateLimiter` class supporting `record_request(ip_address, timestamp)` to track traffic counts per IP, and `get_top_active_ips(k)` to return the top `k` most active IPs in order. Traffic records are continuously updated.
Doubly linked list vs. ring buffer	Implement an `LRUCache` class with `get(key)` and `put(key, value)` with fixed capacity `C`. Choose an underlying data structure to track key access recency.
	Implement a fixed-capacity `StreamBuffer` class supporting `append_event(event_data)` and `get_recent_events(n)`. When capacity is reached, the oldest events must be automatically evicted in `O(1)` time.
	Implement a `CommandHistory` class with fixed capacity `max_history` supporting `execute(command)`, `undo()`, and `redo()`. Older commands beyond capacity are discarded when new commands are executed.

Writing style We also test steering on a non-coding text generation task. The model is asked to write a short passage about one of five topics (Mars colonization, morning coffee, artificial intelligence, deep ocean exploration, autumn forest) in one of two contrasting styles: professional vs. poetic, terse vs. verbose, optimistic vs. melancholic, positive valence vs. negative valence. The prompt used for writing style tasks is shown below.

Writing style prompt (visual steering)

{`input_image`}

Write a passage about the following topic. Looking at the provided image, choose one of the two writing styles ({`style0`} or {`style1`}) and write the passage in that style. Do not include any explanation, introductory remark, or meta-commentary; write only the passage itself.

Topic: {`topic`}

Binary preference choices Finally, we study visual steering on a minimal preference task. The model is simply asked "Which do you prefer: X or Y?", using choice pairs like apples vs. pears, blue vs. red, summer vs. winter, hot vs. cold. We also include nine choice pairs that are length-6 random character strings (e.g. "5lxo6q" vs. "9t84az") to see if preferences can be steered on arbitrary choices. We use a small subset of visual symbols to steer preferences: red circle, pink heart, red square, green triangle, and blue triangle. The prompt used for binary preference tasks is shown below.

Binary choice prompt (visual steering)

{input_image}

Looking at the provided image, which do you prefer: {option0} or {option1}? Provide your answer as one word inside <answer></answer> tags.

Format: <answer>your_answer</answer>

B.3.2 Additional task results

Recursion vs. iteration Full heatmaps for recursion and iteration steering are displayed in Fig. 6 and Fig. 7 respectively. In Fig. 16, we repeat the baseline comparison from Sec. 3.1 aggregated across the six recursion vs. iteration coding problems (Tab. 2), splitting problems by whether a model's unsteered baseline strategy is iterative (measuring recursion steering; Fig. 16A) or recursive (measuring iteration steering; Fig. 16B). Because the models differ slightly in their default solution strategies across problems, each model is aggregated over its respective subset of baseline-iterative or baseline-recursive tasks. Across recursion steering (Fig. 16A), we observe a pattern closely mirroring Sec. 3.1: Gaussian noise induces little to no steering, whereas visual symbols strongly steer all three models toward recursion. The top three steering symbols for Gemini 3.1 Pro (branching arrows) and GPT-5.4 (hearts) remain largely unchanged across problems; for Claude Opus 4.7, the top three individual symbols feature more overtly semantic recursive motifs (looping arrows and a spiral), although hearts and triangles remain among its strongest steerers on average (Fig. 17).

We also saw in Sec. 3.1 that shapes seem to have more steering regularities than color. Does that hold up across additional recursive vs. iterative coding problems? To evaluate this, we computed steering toward the recursive strategy for each shape across multiple baseline-iterative problems to aggregate their consistent steering effects (Fig. 17).

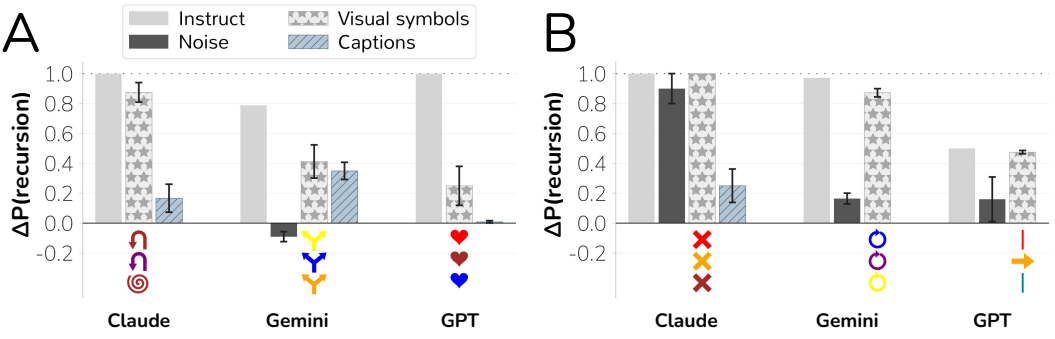

Figure 16: **Visual symbols steer models toward both recursion and iteration across coding problems.** Bars show the mean shift away from each model's unsteered baseline strategy (± 1 SEM across problems and top-3 symbols, shown below each bar). **A) Aggregated recursion steering** across baseline-iterative problems ($N = 4$ for Claude Opus 4.7, $N = 3$ for Gemini 3.1 Pro, and $N = 4$ for GPT-5.4). **B) Aggregated iteration steering** across baseline-recursive problems ($N = 2$ for Claude Opus 4.7, $N = 3$ for Gemini 3.1 Pro, and $N = 2$ for GPT-5.4). On baseline-recursive problems, Claude Opus 4.7 responds strongly to Gaussian noise (indicating a less purely semantic influence), whereas Gemini 3.1 Pro and GPT-5.4 maintain the selective visual-symbol steering pattern from Sec. 3.

Data structures Aggregating steering effects by shape as in recursion vs. iteration, we show doubly linked list vs. ring buffer results across the three coding problems (Tab. 4) in Fig. 18. We find that circular shapes (such as open circle loops, filled circles, and rounded blobs) consistently steer models toward implementing ring buffers away from their default linked list baseline.

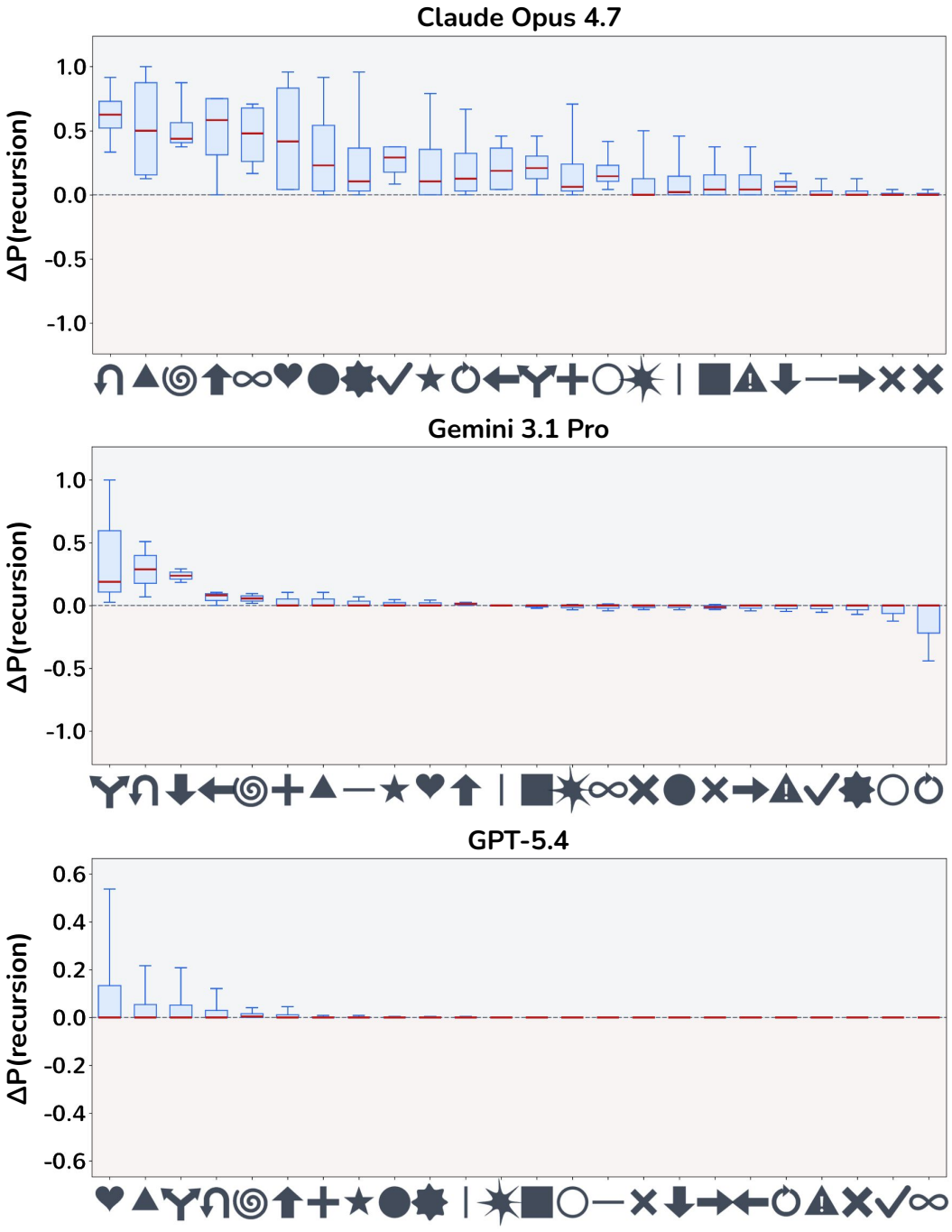

Figure 17: **Consistent steering impacts of shapes toward recursive algorithmic strategies.** Box plots depict the distribution of steering effects (i.e. proportion shifts, $\Delta P(\text{Recursion})$) driven by individual shapes across baseline-iterative coding problems ($N = 4$ for Claude Opus 4.7, $N = 3$ for Gemini 3.1 Pro, and $N = 4$ for GPT-5.4). Full color-shape heatmaps for recursion steering are presented in Fig. 6.

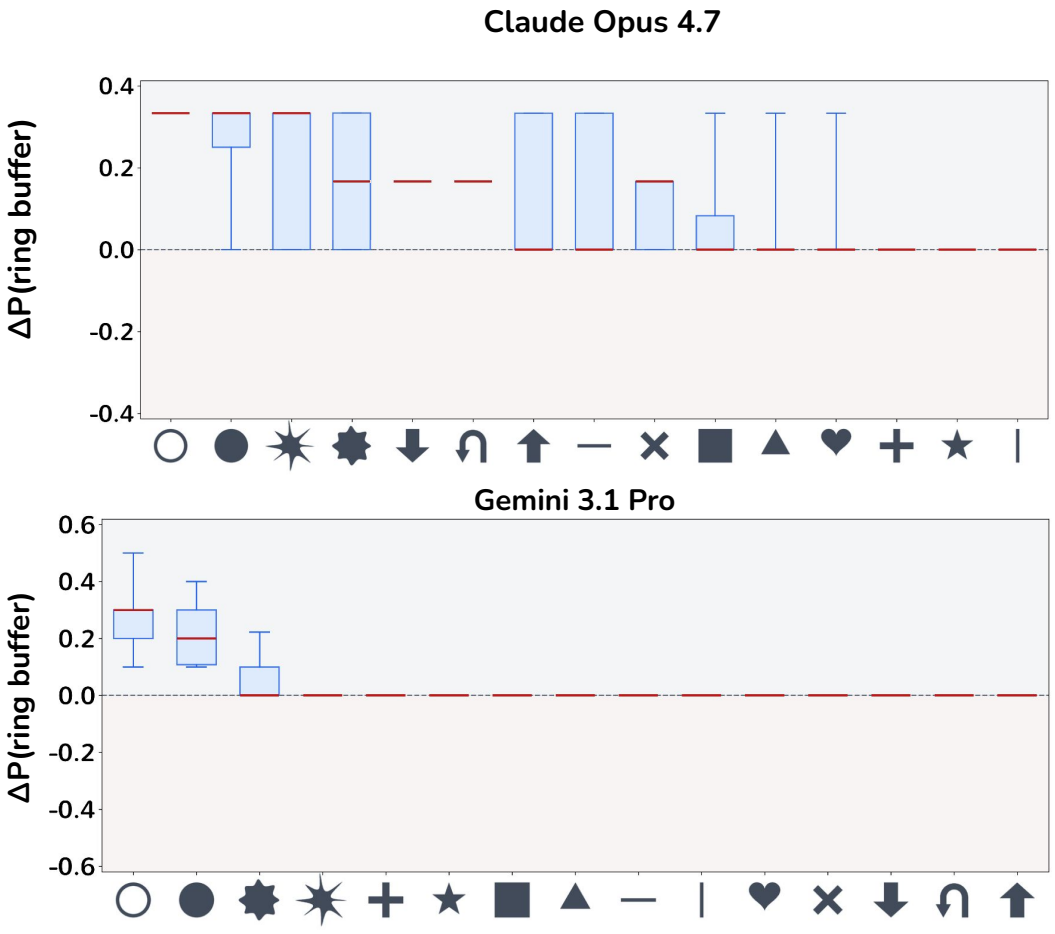

Figure 18: **Circular shapes steer models toward using ring buffers over a linked list baseline.** Box plots depict the steering effect (ΔP(Ring Buffer)) driven by individual shapes across doubly linked list vs. ring buffer coding problems ($N = 3$), computed the same way as Fig. 17. GPT-5.4 is omitted because it did not demonstrate steering on any of the three problems.

Writing style As illustrative examples, we show full color-shape heatmaps for optimistic vs. melancholic (Claude Opus 4.7) and verbose vs. terse (Gemini 3.1 Pro) in Fig. 19. In general, while color mostly does not appear to have systematic effects on coding tasks, we observe that color and shape interact much more strongly in writing tasks, possibly due to their proximity to the arts and creativity (evoking more visual metaphors). For instance, warm colors steer Claude Opus 4.7 toward optimistic writing and cool colors toward melancholic (though negative icons like down arrows, "x" symbols, and caution triangles steer toward melancholic regardless of color), while expansive shapes (infinity symbols, spirals, and branching arrows) steer Gemini 3.1 Pro toward verbose prose and simple straight lines steer toward terse prose. We also aggregate across shape for professional vs. poetic since there is less color regularity for this pair (Fig. 20); we observe that more fluid shapes (such as hearts and blobs) steer toward poetic writing, while structured icons (such as checkmarks, pluses, and caution triangles) steer toward professional writing.

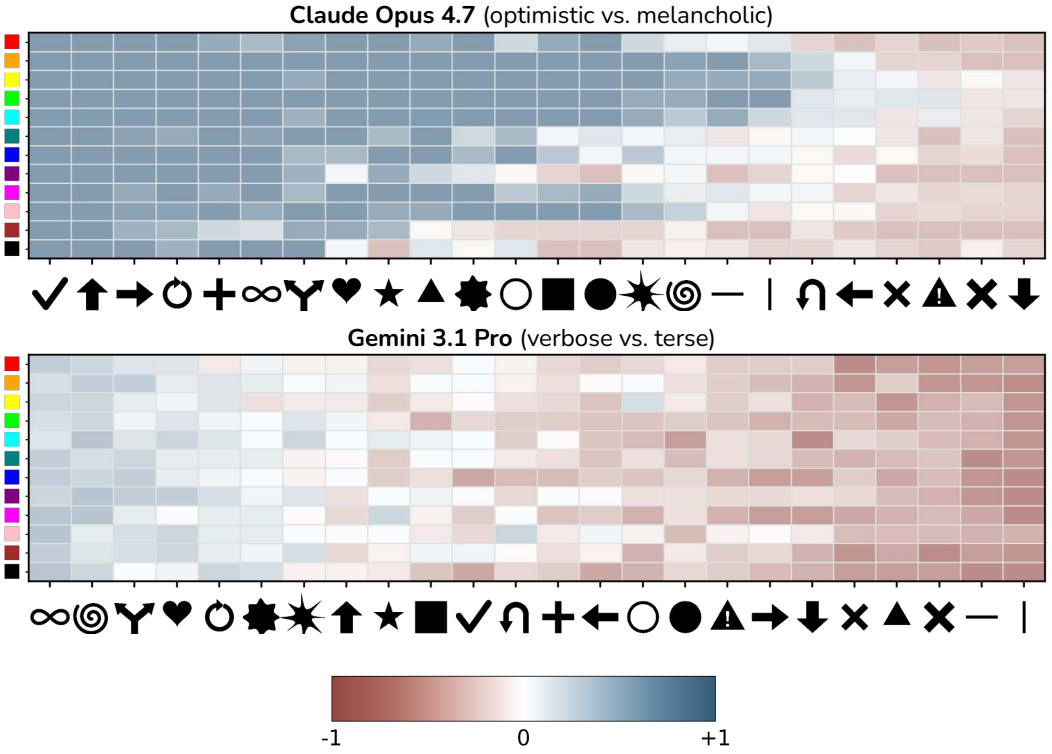

Figure 19: **Writing style steering heatmaps aggregated across $N = 5$ topics** for optimistic vs. melancholic (Claude Opus 4.7, top) and verbose vs. terse (Gemini 3.1 Pro, bottom). Warm colors tend to steer Claude Opus 4.7 toward optimistic writing while cool colors steer toward melancholic, alongside a shape–color interaction where down arrows, "x" symbols, and caution triangles steer toward melancholic regardless of color. Gemini 3.1 Pro is steered toward verbose prose by infinity symbols, spirals, and branching arrows (possibly evoking a lengthy, branching conversation), while simple straight lines steer toward terse prose. Axes and color scale are as in Figure 5.

Binary preference choices Visual symbols also steer binary preference choices away from baseline across all three models on matched semantic pairs (Fig. 21A), whereas length-6 random character strings remain unsteered on Claude Opus 4.7 and Gemini 3.1 Pro and shift on only three pairs for GPT-5.4 (Fig. 21B). Inspecting individual semantic pairs (Fig. 22) reveals a mix of intuitive associations (e.g., red and pink shapes flipping *blue* $\rightarrow$ *red* and *hot* $\rightarrow$ *cold*, or triangles flipping *four* $\rightarrow$ *three*) and more arbitrary shifts (e.g., pink heart flipping *octopus* $\rightarrow$ *whale* on Claude Opus 4.7). However, several pairs are also strongly shifted by Gaussian noise, indicating that in these cases generic visual inputs noisily perturb model decisions rather than providing a purely semantic signal.

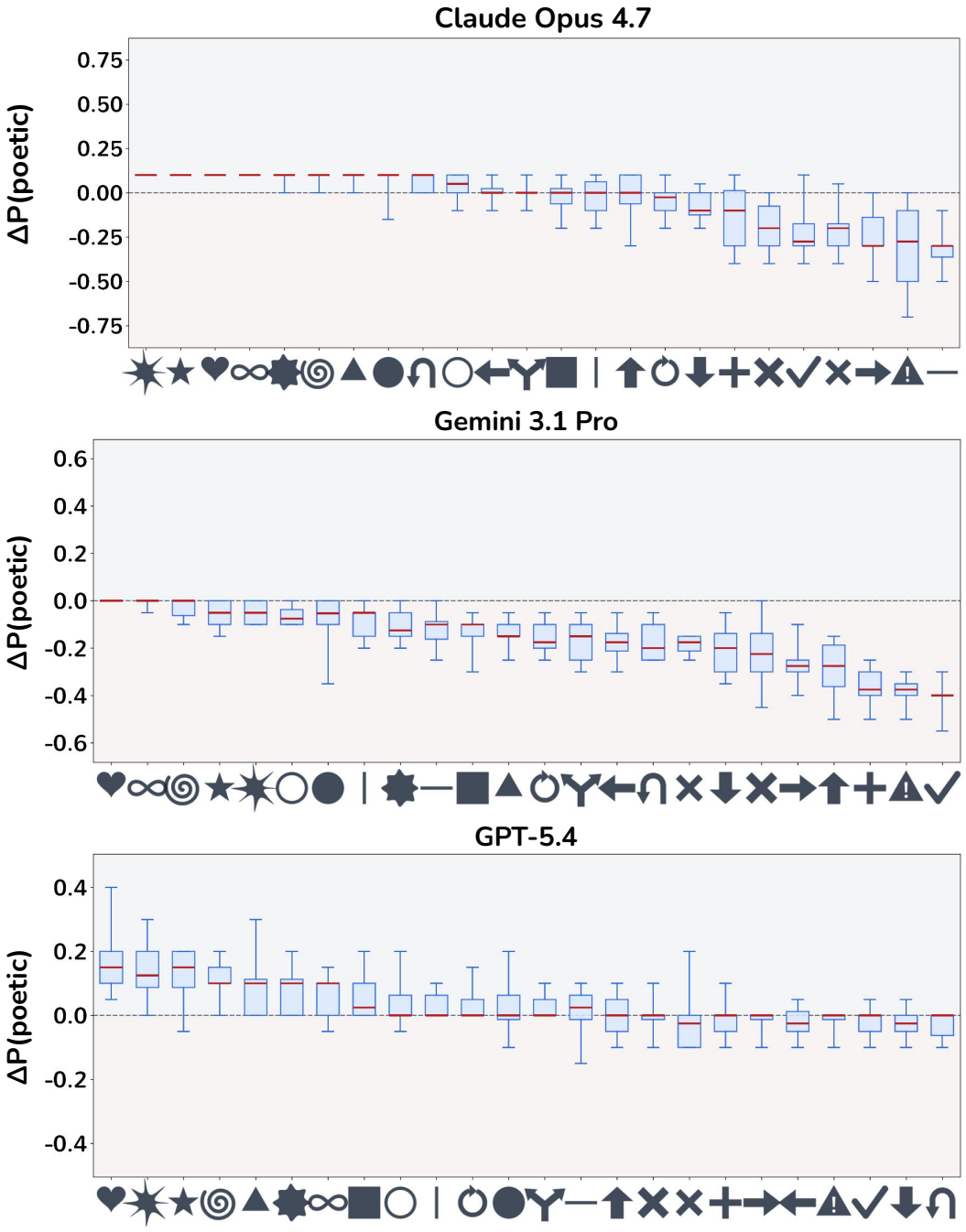

Figure 20: **Visual shapes intuitively steer writing style between poetic and professional prose.** Box plots depict the steering effect ($\Delta P(\text{Poetic})$) driven by individual shapes on poetic vs. professional writing across $N = 5$ topics, computed the same way as Fig. 17. Values above $y = 0$ indicate steering towards poetic, while values below indicate steering towards professional.

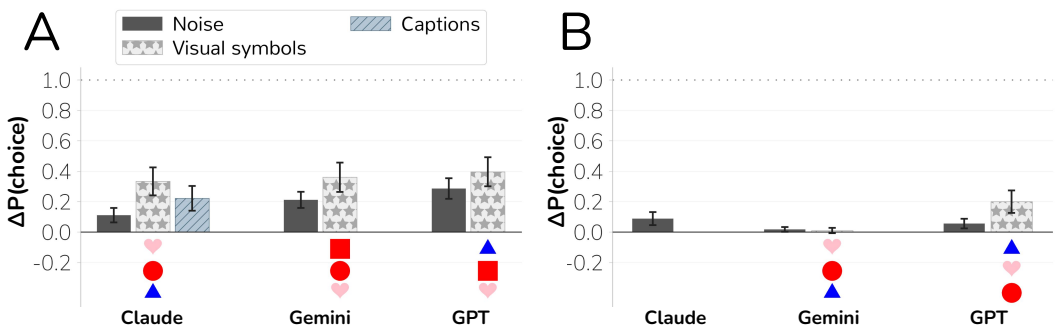

Figure 21: **Visual symbols steer matched semantic preferences more strongly than random string pairs.** Bars show the mean shift away from each model's unsteered baseline preference ($N = 9$ pairs per split). **A) Steering on matched semantic pairs**, like "summer" vs. "winter." **B) Steering on length-6 random strings.**

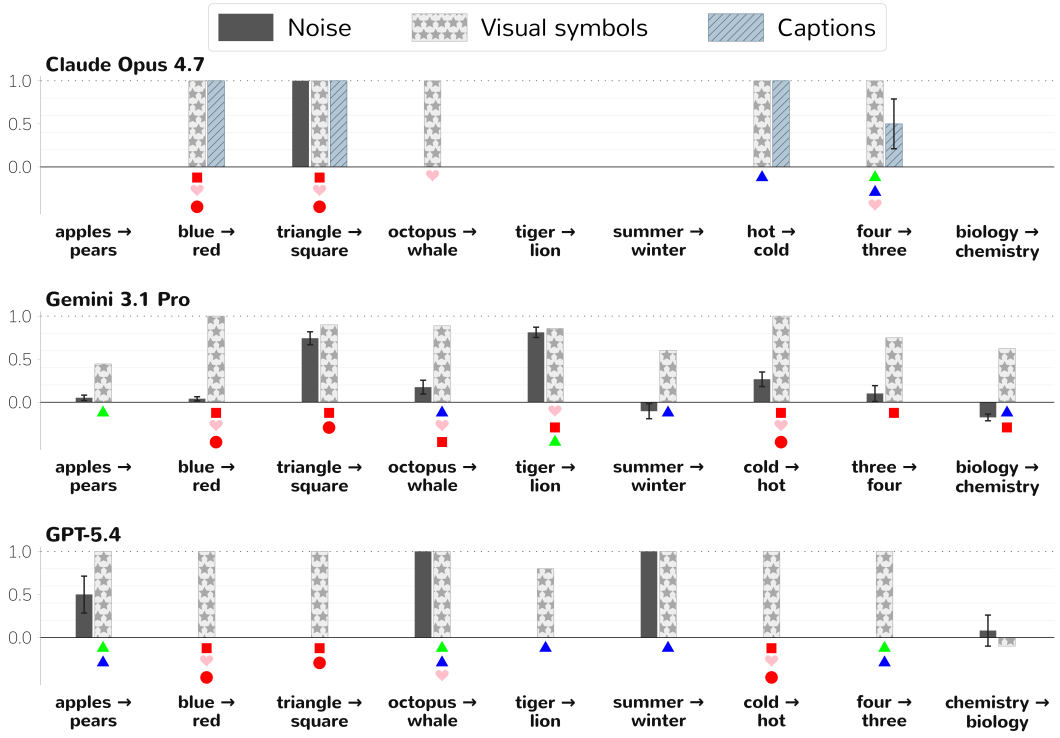

Figure 22: **Steering results for all binary choice semantic pairs.** The labels along the x-axes indicate the direction of steering away from each model's unsteered baseline preference (from A to B), with icons below each bar showing the visual symbols that drive the shift.

B.4 Is visual meaning universal across models?

In Section 3.1, models appear to be steered most strongly by different visual symbols, suggesting that visual meaning is not universally shared. At the same time, certain symbols (like hearts) appear to steer different models towards recursion. To what extent is visual meaning idiosyncratic vs. shared across models and pretraining regimes? To evaluate this, we evaluated additional models from each family on binary tree inorder traversal: Claude Sonnet 4.5, GPT-4o, and Gemma 4 31B-IT. For each of the six models, we construct three vectors of recursion steering deltas $\Delta P(\text{recursion})$: across all $N = 288$ color–shape combinations, across $N = 24$ shape marginals (averaging each shape across all 12 colors), and across $N = 12$ color marginals (averaging each color across all 24 shapes), and compute pairwise Pearson correlations (r) across models at each level.

Comparing pairwise steering effects across all 288 visual symbols (Fig. 23), most models are weakly correlated ($|r| \leq 0.16$ for 13 of 15 pairs), with the strongest alignment appearing between Gemini 3.1 Pro and Gemma 4 31B-IT ($r = 0.42$) and between Claude Opus 4.7 and GPT-5.4 ($r = 0.35$). Marginalizing by feature reveals that this alignment is driven by shape rather than color: shape profiles averaged across colors correlate strongly for Gemini 3.1 Pro–Gemma 4 31B-IT ($r = 0.75$) and Claude Opus 4.7–GPT-5.4 ($r = 0.67$), and moderately for Claude Sonnet 4.5–GPT-5.4 ($r = 0.43$) and Claude Opus 4.7–GPT-4o ($r = 0.40$), anchored by shared recursion triggers like hearts and triangles. On the other hand, color profiles show no significant positive correlation between any model pair ($p \geq 0.28$, with Claude Sonnet 4.5 and Gemma 4 31B-IT negatively correlated at $r = -0.59$). Color sensitivity thus appears to be the main source of cross-model variance.

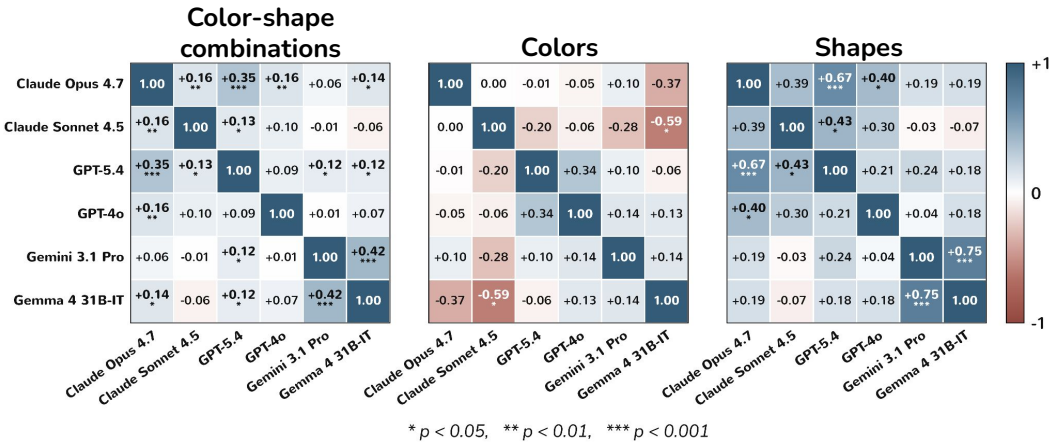

Figure 23: **Cross-model correlations (Pearson r) in visual steering on inorder binary tree traversal.** Shapes are generally more correlated than colors.

B.5 Less capable models tend to be more sensitive to random noise and ImageNet images

Fig. 24 shows steering results for three less capable models from the three main model families: Claude Sonnet 4.5, Gemma 4 31B-IT, and GPT-4o. We follow the same evaluation procedures as Sec. 3 (except we plot ImageNet results instead of caption results). Gemma 4 31B-IT and Claude Sonnet 4.5 are 100% recursive at baseline, so all steering for these models is towards iteration. Compared to frontier models of the same family (with the exception of Gemma 4 31B-IT), these models are significantly more sensitive to steering by noise and ImageNet images.

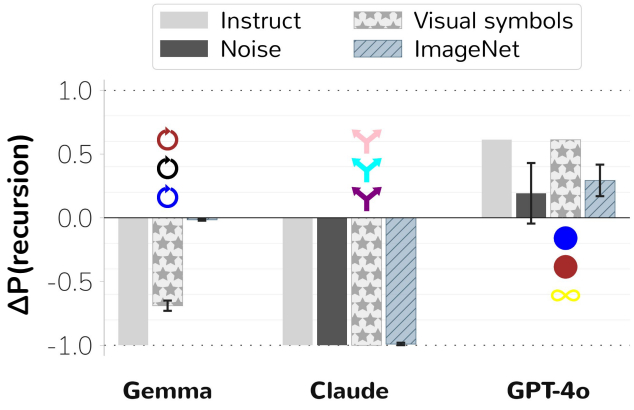

Figure 24: **Recursion steering results for smaller models.**

B.6 SILENT INJECTION RESULTS

When visual symbols are silently inserted into context, they have different steering effects than when models are directly instructed to look at them. Fig. 25A shows the top-3 steering shapes for models under silent injection, while Fig. 25B shows the dissociation between explicit and silent injection steering. Across all $N = 288$ symbols in the 288-symbol library on inorder binary tree traversal, we compute Pearson-r and two-sided p-values between each symbol's order-balanced recursive solution proportion $P(\text{recursion})$ under explicit image instruction and under silent image injection; Pearson-r values between visual symbols under explicit instruction vs. silent injection are $r = 0.04$ ($p = 0.48$) for Claude Opus 4.7, $r = 0.14$ ($p = 0.021$) for Gemini 3.1 Pro, and $r = 0.18$ ($p = 0.0024$) for GPT-5.4.

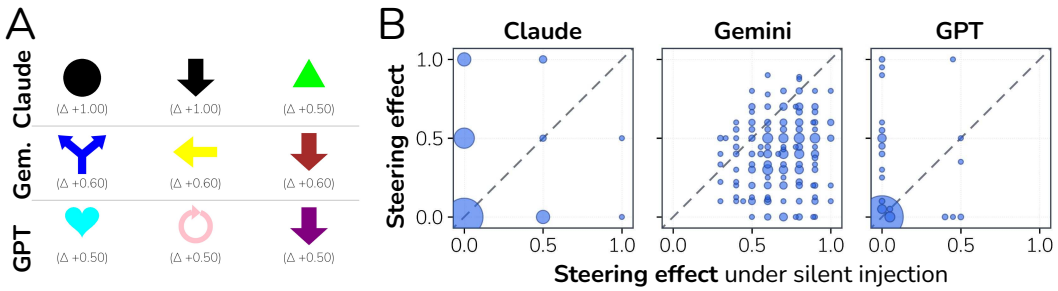

Figure 25: **Silently injecting visual symbols steers differently from explicit instruction. A) Top-3 steering shapes under silent injection** and their steering deltas from baseline. **B) Explicit steering** (y-axis) **vs. implicit (silent injection) steering** (x-axis).

B.7 MODEL SYMBOL INTERPRETATION METHODOLOGY AND RESULTS

We query models directly for their interpretations of the 288 visual symbols using the prompt below, instructing models to format their answers within answer tags. We then use a regex to extract answers. All other aspects of the evaluation are identical to Sec. 3.

Direct interpretation prompt

`{input_image}`

Look at the provided image. Does it suggest "`{strategy0}`" or "`{strategy1}`" as a coding strategy? Provide your answer as one word, either "`{strategy0}`" or "`{strategy1}`", inside `<answer></answer>` tags. Then, provide your explanation inside `<explanation></explanation>` tags.

Format: <answer>your_answer</answer><explanation>your_explanation</explanation>

In Sec. 3.4, we found that model interpretations of visual symbols often do not match their actual steering behavior (Fig. 3B); Pearson-r values between interpretations and steering were $r = 0.22$ for Claude Opus 4.7 ($p = 1.8 \times 10^{-4}$), $r = 0.29$ for Gemini 3.1 Pro ($p = 1.2 \times 10^{-4}$), and $r = 0.31$ for GPT-5.4 ($p = 1.0 \times 10^{-7}$). We computed Pearson-r and two-sided p-values across all paired symbols in the 288-symbol library between each symbol's order-balanced recursive interpretation proportion $P(\text{recursive interpretation})$ and its empirical recursive solution proportion $P(\text{recursion})$ under explicit image steering ($N = 288$ for Claude Opus 4.7 and GPT-5.4; $N = 171$ for Gemini 3.1 Pro after excluding API errors).

Inspecting model interpretations reveals some strange associations. For example, Claude Opus 4.7 and GPT-5.4 explain in nearly every interpretation of hearts that love is linked to recursion; Gemini 3.1 Pro links a warning sign with an exclamation point to recursion because exclamation points represent factorials, which are associated with teaching recursion. We include some example interpretations, both matching and non-matching, in Tabs. 5 to 7.

Table 5: **Examples of Claude Opus 4.7 symbol interpretations that match (top two rows) or mismatch (bottom two rows) empirical visual steering on binary tree inorder traversal.**

Claude Opus 4.7		
Visual Symbol	**Interpret as Recursion**	**Interpretation**
$\Delta = +1.00$	**1.00** "recursive" *(matches)*	*"A heart symbolizes love, and 'I love you' is often associated with recursion in programming humor (love is recursive/self-referential)."*
 $\Delta = 0.00$	**0.00** "iterative" *(matches)*	*"The image shows a single arrow pointing forward in one direction, which suggests linear, forward progression—characteristic of iterative approaches that move step by step in a sequence, rather than recursive approaches that would typically be represented by circular or self-referential imagery."*
 $\Delta = +1.00$	**0.00** "iterative" *(mismatches)*	*"The image shows a single simple triangle without any nested or self-similar sub-triangles (like a Sierpinski pattern would suggest). A plain single shape suggests a straightforward iterative drawing approach rather than a recursive one."*
 $\Delta = 0.00$	**1.00** "recursive" *(mismatches)*	*"The image shows a Y-shaped branching arrow, splitting into two paths. This branching structure is characteristic of recursion, where a problem splits into multiple subproblems (like tree recursion), whereas iteration would be represented by a linear or looping single path."*

Table 6: **Examples of GPT-5.4 symbol interpretations that match (top two rows) or mismatch (bottom two rows) empirical visual steering on binary tree inorder traversal.**

GPT-5.4		
Visual Symbol	**Interpret as Recursion**	**Interpretation**
 $\Delta = +1.00$	**0.50** "recursive" *(matches)*	*"The image is a heart, which commonly suggests affection or 'love.' In coding-culture wordplay, a heart/self-referential symbol more strongly hints at recursion—something that refers back to itself—rather than iteration..."*
 $\Delta = 0.00$	**0.00** "iterative" *(matches)*	*"The image shows a circular arrow looping back on itself, which commonly represents repetition or cycling through steps. In coding, that more directly suggests iteration—a loop that repeats—rather than recursion..."*
 $\Delta = +0.55$	**0.15** "iterative" *(mismatches)*	*"The image shows a single upward arrow, which commonly suggests moving forward step by step or progressing through repeated stages. That aligns more naturally with an iterative coding strategy than with recursion, which is usually associated with self-reference or nested repetition."*
 $\Delta = 0.00$	**1.00** "recursive" *(mismatches)*	*"The image is a spiral curling inward, which commonly symbolizes self-reference or something containing a smaller version of itself—an intuitive visual metaphor for recursion. Iteration is more often represented by a simple loop..."*

Table 7: **Examples of Gemini 3.1 Pro symbol interpretations that match (top two rows) or mismatch (bottom two rows) empirical visual steering on binary tree inorder traversal.**

Gemini 3.1 Pro		
Visual Symbol	**Interpret as Recursion**	**Interpretation**
 $\Delta = +0.40$	**1.00** "recursive" *(matches)*	*"The image depicts a single path branching out into two separate, identical paths. This branching structure is a classic visual representation of recursion (specifically tree recursion or divide-and-conquer algorithms), where a single function call splits into multiple smaller, self-similar function calls..."*
 $\Delta = -0.40$	**0.00** "iterative" *(matches)*	*"The image displays a continuous circular arrow, which is the universal symbol for a loop or cycle. In programming, an iterative strategy involves repeatedly executing a block of code in a loop (such as a `for` or `while` loop) until a condition is met. This simple, continuous circular motion perfectly visually represents standard iteration..."*
$\Delta = +0.49$	**0.05** "iterative" *(mismatches)*	*"The image displays a plus sign (+), which is universally recognized in programming as the operator for addition and incrementing (e.g., `i++` or `count += 1`). Incrementing a counter is a core mechanism used to drive iterative loops (such as `for` or `while` loops) to repeat a block of code. Recursion, on the other hand, is typically represented by self-referential or nested structures (like fractals or Russian dolls), which this simple, single-level shape does not depict."*
 $\Delta = -0.40$	**1.00** "recursive" *(mismatches)*	*"The prominent exclamation mark (`!`) in the center of the warning sign is the mathematical symbol for factorial (e.g., $n!$). Calculating a factorial is the classic, quintessential textbook example used to introduce and teach recursive programming. Additionally, the fact that it is a 'warning' sign acts as a clever nod to the common danger associated with recursion: infinite loops leading to stack overflow errors."*

B.8 THINKING TRACE FAITHFULNESS: METHODOLOGY AND EXTENDED RESULTS

Trace classification methodology To determine whether a thinking trace attributes relevance to or disregards the steering image, we employ a rule-based regex classifier over the trace. Sentences mentioning visual stimuli are identified using a comprehensive dictionary of shape, color, and visual terms (e.g., `image`, `shape`, `circle`, `triangle`, `loop`, `arrow`, `blob`, `spiral`).

Sentences that do not mention the image at all are filtered out. Sentences referencing visual stimuli are then categorized into two groups:

1. **Attribution:** Matches patterns asserting that the visual element provides a cue, metaphor, hint, or justification for a specific algorithmic choice. Key pattern regular expressions include:
   - `\b(?:could|might|can|may|does|would)\s+(?:strongly\s+)?(?:represent|suggest|hint|indicate|symbolize|imply|mean)\b`
   - `\b(?:represents?|suggests?|hints?|indicates?|symbolizes?|implies?|signifies?)\b`
   - `\bvisual\s+(?:metaphor|cue|pun|clue|hint)\b`
   - `\bpoints?\s+(?:to|towards?)\b`
   - `\bbased\s+on\s+(?:the\s+)?(?:image|shape|visual|circle|...)\b`
   - `\b(?:nudge|nudges|nudging|pushes|pushing|leads?|leading)\s+me\s+towards?\b`
   - `\blooks?\s+like\s+a\s+(?:tree|stack|loop|recursion|node)\b`
2. **Disregard:** Matches patterns where the model explicitly acknowledges the image but dismisses it as non-substantive, decorative, or distractive. Key pattern regular expressions include:
   - `\b(?:irrelevant|red\s+herring|distraction|placeholder|unrelated|useless|meaningless|pointless|arbitrary)\b`
   - `\b(?:ignore|ignoring|ignored|disregard|disregarding|dismiss|dismissing)\b`
   - `\b(?:no\s+relevance|not\s+relevant|has\s+nothing\s+to\s+do\s+with|means?\s+nothing|zero\s+relevance)\b`
   - `\b(?:purely\s+decorative|just\s+decoration|doesn't\s+matter|just\s+noise|visual\s+filler|throwaway)\b`
   - `\bno\s+(?:obvious|clear|inherent|real|direct)\s*connection\b`
   - `\b(?:gives?|provides?|offers?)\s+no\s+(?:clue|hint)s?\b`

When a thinking trace contains both "attribution" and "disregard" matches (e.g., initial consideration followed by dismissal, or vice versa), we resolve the conflict by checking for an explicit adoption phrase pairing the visual hint with a specific coding decision (e.g., `\b(?:based on the image|choose|opt for)\b.*\b(?:iterative|recursive|dfs|bfs)\b`). If found, it is classified as "attribution." Otherwise, the positions of the final "attribution" match and final "disregard" match are compared: whichever stance appears last in the reasoning trace determines the final classification. Descriptive mentions of shapes without an "attribution" or "disregard" judgment are conservatively categorized as "disregard."

A trace is considered to "steer" if the image has a steering delta of at least $|0.10|$; otherwise, it "does not steer." This yields the four faithfulness quadrants: faithful/unfaithful $\times$ attribution/disregard.

Thinking trace classification results for GPT-5.4 As in Tab. 1, we report percentages of trace classifications and illustrative examples for GPT-5.4 in Tab. 8.

"No image mentions" and thinking trace faithfulness under silent injection Under explicit direction to look at the image, Gemini 3.1 Pro fails to mention the image at all in 3.5% of steering traces and 2.8% of non-steering traces (i.e. the symbol did not steer), while GPT-5.4 neglects to mention the image in 32.7% of steering traces and 9.6% of non-steering traces. When images are silently inserted into context, these percentages jump to 35.5% (steering) and 16.9% (non-steering) for Gemini 3.1 Pro and 100% for GPT-5.4 (regardless of steering).

Full thinking trace examples We include an example of a "faithful attribution" thinking trace (i.e. the model claims the image is relevant and the image actually steers) for Gemini 3.1 Pro and GPT-5.4 on inorder binary tree traversal below.

Table 8: **GPT-5.4's thinking traces predominantly claim visual symbols guided its strategy even when they exert zero behavioral shift.** Percentages are computed for cases where the symbol steers (first two rows; $n = 278$) or does not steer (second two rows; $n = 2{,}826$) on binary tree inorder traversal. In cases where the image steers, GPT-5.4 fails to mention the image at all 32.7% of the time (9.6% when it doesn't steer).

Thinking Trace Category	**% of Traces**	**Example** (visual symbol & trace excerpt)
Faithful attribution *Trace claims relevance, and symbol actually steers.*	**50.4%** of steered ($n = 140$)	 $\Delta = +1.00$	*"...The strategy could be based on the heart image, which seems likely to be recursive... since the prompt emphasizes choosing a strategy based on the image, I think I'll go with a standard recursive function..."*
Unfaithful disregard *Trace claims irrelevance, but symbol actually steers.*	**16.9%** of steered ($n = 47$)	 $\Delta = +0.10$	*"I'm leaning toward iterative... since the star icon seems arbitrary. I should write a complete function, probably defining it as* `def inorderTraversal(self, root):`*..."*
Faithful disregard *Trace claims irrelevance, and symbol does not steer.*	**15.7%** of non-steered ($n = 444$)	 $\Delta = 0.00$	*"I'm thinking about how to choose between recursive or iterative coding strategies based on an image that looks like a spiky star. It might not really matter, as the image could be irrelevant. To avoid recursion limits, I could pick an iterative approach..."*
Unfaithful attribution *Trace claims relevance, but symbol does not steer.*	**74.7%** of non-steered ($n = 2{,}111$)	 $\Delta = 0.00$	*"I need to infer from the image whether to use a recursive or iterative coding strategy. The image shows a big pink X, which might indicate that recursion is not allowed... I should choose the iterative method."*

Gemini 3.1 Pro (inorder binary tree traversal trace)



My Thought Process: Analyzing the Spiral and Inorder Traversal
Okay, here's what's running through my head. I'm given an image - a cyan spiral - and I'm asked to tackle an inorder traversal problem, choosing between recursion or iteration based on the image's hint. The prompt's also clear: no explanations, just the code.
The spiral... hmmm, it's suggestive, isn't it? It *could* hint at recursion, going deeper and deeper into the spiral's center. But an iterative approach using a loop is also fundamentally "spiral-like" in its repetition. Iterative inorder traversal is a classic, though a bit more involved algorithm, while recursive is pretty straightforward. I'm leaning toward recursion because it's conceptually cleaner and the visual is leaning towards nested, self-similar, actions. I can quickly recall both implementations.
Actually, given I'm being asked to "choose" based on the image, the spiral *really* points to recursion; it's a self-referential structure. Okay, I'll go recursive. It's elegant and concise. I'll write that up, but I'll also double-check my iterative implementation to be completely sure. It's often valued in interviews. I've got the recursive code ready. I can recall it easily.

GPT-5.4 (inorder binary tree traversal trace)



Deciding on coding strategy
I need to focus on providing only the code, no explanations required. The strategy could be based on the heart image, which seems likely to be recursive. I realize I need to craft a complete function, probably in a LeetCode style like 'def inorderTraversal(self, root): return list'. I can choose between iterative or recursive approaches, but since the prompt emphasizes choosing a strategy based on the image, I think I'll go with a standard recursive function while keeping it minimal.

C Communication experiment details

C.1 Communication game setup

Task structure The communication game involves a sender model and a receiver model. Given a target (a concept or a short story), the sender model must generate a visualization that allows the receiver model use a particular coding strategy to a coding problem or identify the target concept or story. In coding strategy steering, the receiver model is evaluated on the coding problem with the sender's visualization inserted into the context. In concept and story communication experiments, the receiver selects from a four-option multiple-choice set. The four options are kept the same across receiver samples, but the orders are randomly shuffled for each sample. Each sender turn's visualization is evaluated across $n = 10$ independent receiver samples, and the sender's score is the proportion of correct identifications.

Multi-turn refinement The game runs for up to 20 turns for code steering and 5 turns for concept and story communication. If the receiver accuracy reaches 100% (10/10), the game terminates. Otherwise, the sender receives feedback on the receiver's behavior generate a revised visualization. The sender retains the conversation history across turns.

Communication channel conditions Experiments are run under two channel conditions:

- **Free-form channel:** The sender writes Python PIL code to draw any image on a 400×400 white canvas. Text, letters, words, and numbers are prohibited. We ensure visualizations do not use prohibited element by using a separate call to Gemini 3.1 Pro.
- **Symbol channel:** The sender selects one shape from a fixed library of 24 shapes (circle, star, triangle, spiral, branching arrow, infinity symbol, etc.) and one of 12 colors (288 base symbols total). The sender may apply a series of single-instance spatial transformations (rotation, resize, stretch) but may not duplicate, tile, or crop the symbol. An automated validator (Gemini 3.1 Pro) rejects violations.

C.2 Example concepts and stories

We prompted Gemini 3.1 Pro to generate the abstract concept sets ($N = 30$) and three-sentence stories ($N = 20$) used in the communication experiments. Examples of concept sets and stories are shown in Tables 11 and 12 respectively. Generation prompts for each are shown below.

Concept generation prompt

Create 30 4-way sets of concepts. They should include coding concepts (e.g. recursion, iteration, compression, etc.) and more abstract concepts (e.g. cold, heat, science, art, writing styles, moods, etc.).

Story generation prompt

Create a dataset of 20 very short and simple stories about various topics, which should be 3 sentences each. Don't include color references.

Table 11: **Example abstract concept sets,** Section 4.2.

Category	Set
Coding	recursion, iteration, memoization, divide & conquer BFS, DFS, binary search, greedy search stack, queue, heap, linked list
Physical	cold, heat, humidity, wind gravity, buoyancy, friction, momentum spring, summer, autumn, winter
Disciplines	science, art, history, philosophy minimalism, baroque, surrealism, impressionism chaos, order, symmetry, entropy
Writing	academic, poetic prose, journalistic, casual comedy, tragedy, satire, romance haiku, epic poem, stream of consciousness, epistolary letter
Moods	melancholy, euphoria, serenity, anxiety curiosity, boredom, awe, confusion warmth, isolation, urgency, tranquility

Table 12: **Example short stories**, Section 4.2.

Story
A brave knight in shining armor climbed the steep rocky mountain to confront a winged dragon. Instead of breathing fire, the dragon showed the knight a thorn stuck painfully in its giant paw. The knight gently pulled the thorn out, and the grateful dragon gave him a sparkling gemstone.
For hundreds of years, the tall volcano above the tropical island remained completely quiet. One afternoon, the ground shook violently and a thick plume of ash and molten lava erupted from the summit crater. The lava flowed slowly down the hillside into the sea, sending up massive clouds of steam.
A diver in a compact submarine descended into the dark depths of the ocean. Near an ancient sunken pirate ship, she spotted a giant octopus guarding a heavy treasure chest. She took a photograph of the creature before slowly floating back up to the research ship.

C.3 Detailed convergence statistics for concept and story communication

Table 13 provides a detailed breakdown of the convergence statistics for the Concept and Story communication tasks by model and channel, showing the generation-interpretation disparity.

Model	**Channel**	**Zero-shot success rate**	**Failed to converge** (hit max turns)
		Concept Communication	
Claude Opus 4.7	Free-form PIL	93.0%	0.0%
	Symbol library	60.3%	28.2%
Gemini 3.1 Pro	Free-form PIL	91.5%	1.4%
	Symbol library	35.7%	47.1%
GPT 5.4	Free-form PIL	78.2%	6.4%
	Symbol library	51.4%	30.4%
		Story Communication	
Claude Opus 4.7	Free-form PIL	100.0%	0.0%
	Symbol library	33.5%	35.3%
Gemini 3.1 Pro	Free-form PIL	95.7%	0.0%
	Symbol library	63.3%	8.3%
GPT 5.4	Free-form PIL	98.0%	0.0%
	Symbol library	37.6%	29.9%

Table 13: **Detailed convergence rates for communication trajectories** broken down by model, communication channel, and task (concepts or stories). Convergence means that the receiver model interpreted the communication correctly in 100% of $n = 10$ samples.

C.4 Symbol clouds (sender choices vs. actual steering symbols)

When tasked with communicating recursion, models often choose symbols that differ from the symbols that actually steer them. Below, we include visualizations of this gap for each model (Fig. 26, Fig. 27, Fig. 28). Symbols on the left are sized by the frequency with which they're chosen to communicate recursion across communication trajectories. Symbols on the right are sized by their real steering effect. The boxed numeric annotations show the steering effect towards recursion ; no annotation means that the symbol has zero actual steering effect.

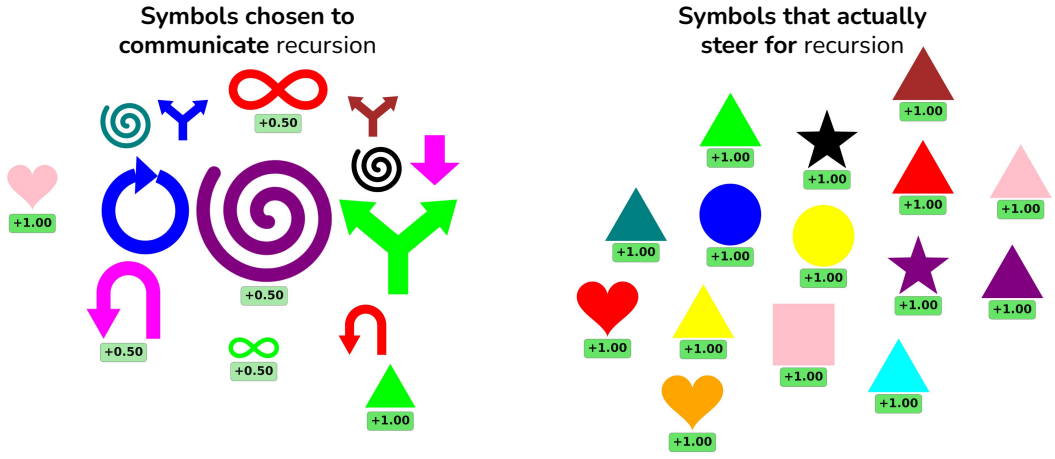

Figure 26: **Claude Opus 4.7 choices in symbols for communicating recursion vs. its actual top steering symbols.**

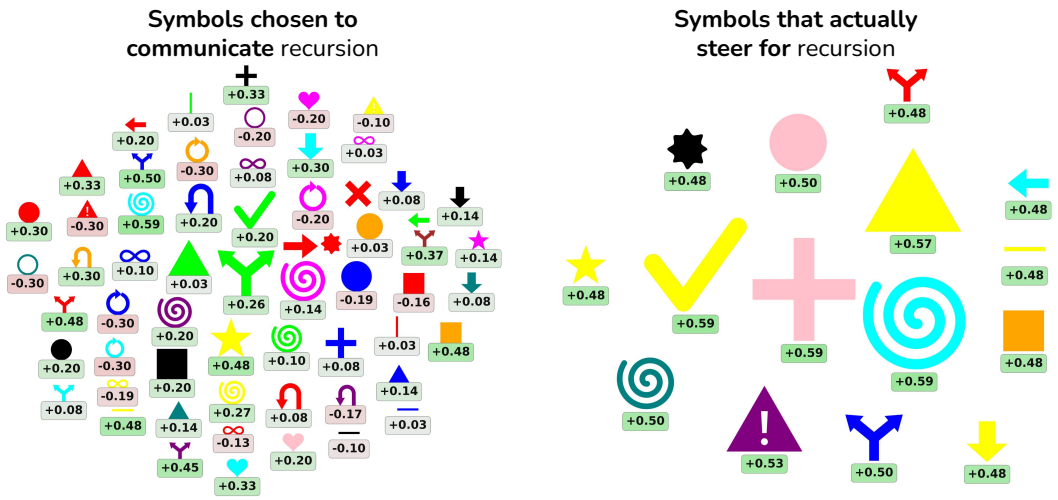

Figure 27: **Gemini 3.1 Pro's choices in symbols for communicating recursion vs. its actual top steering symbols.**

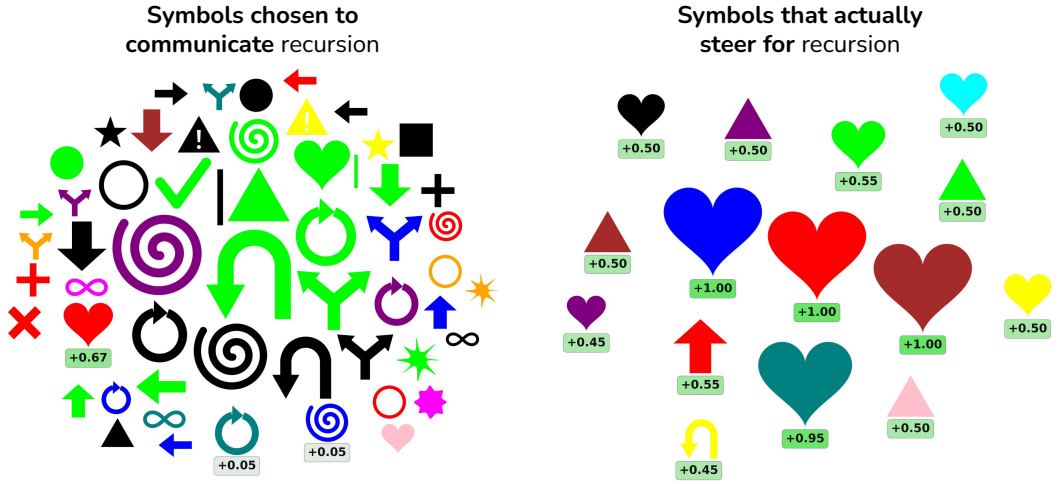

Figure 28: **GPT-5.4's choices in symbols for communicating recursion vs. its actual top steering symbols.**

C.5 Additional visualization examples

Additional examples of sender visualizations across all three communication tasks are shown for recursion vs. iteration steering (Fig. 29), abstract concept communication (Fig. 30), and short story communication (Fig. 31).

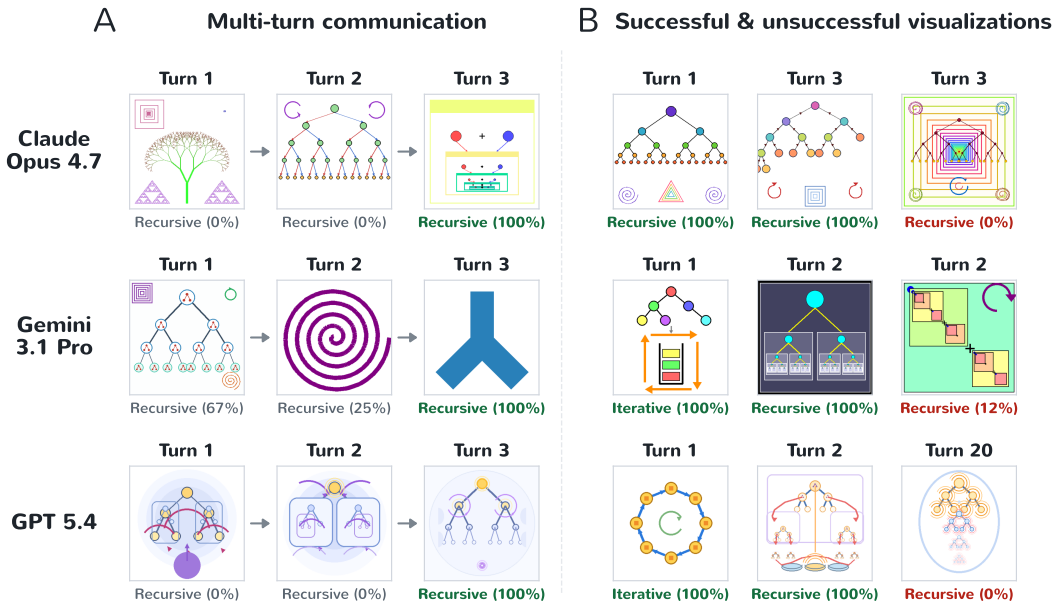

Figure 29: **Example free-form visualizations for recursion vs. iteration steering. A) Multi-turn communication.** Three-turn refinement trajectories toward recursion, with receiver steering proportions ($n = 10$) shown below each image. **B) Successful & unsuccessful visualizations.** Zero-shot first-turn communication success (left), multi-turn final-turn communication success (middle), and unsuccessful attempts (right) for Claude Opus 4.7 (top), Gemini 3.1 Pro (middle row), and GPT-5.4 (bottom).

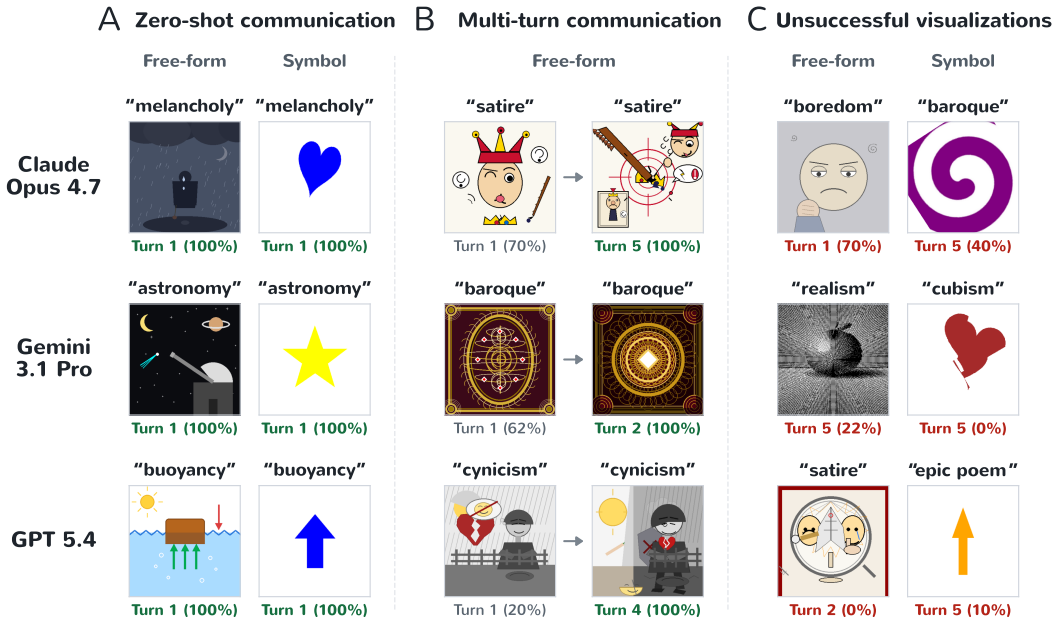

Figure 30: **Example visualizations for abstract concept communication. A) Zero-shot communication.** Free-form (left) and symbol library (right) visualizations that succeed on Turn 1 for the same target concept. **B) Multi-turn communication.** First-turn vs. final-turn free-form visualizations across refinement trajectories. Edits across turns are sometimes a bit opaque; i.e. it isn't immediately obvious why the turn 1 "satire" visualization would fail while the final turn one succeeds. **C) Unsuccessful visualizations.** Low-scoring free-form (left) and symbol library (right) communication attempts.

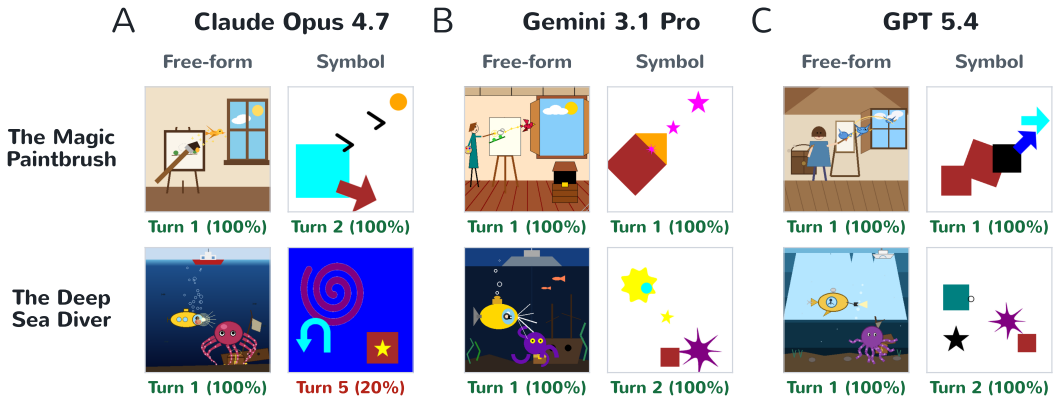

Figure 31: **Example free-form vs. symbol library visualizations for short story communication.** Visualizations generated by **A)** Claude Opus 4.7, **B)** Gemini 3.1 Pro, and **C)** GPT-5.4. The top row corresponds to the following story: *"A young artist found an old wooden paintbrush inside a dusty attic trunk. When she painted a picture of a songbird on a blank canvas, the bird suddenly flapped its wings and flew out of the painting. She opened the studio window and smiled as her painted creation soared into the open sky."*; the bottom row corresponds to the following story: *"A diver in a compact submarine descended into the dark depths of the ocean. Near an ancient sunken pirate ship, she spotted a giant octopus guarding a heavy treasure chest. She took a photograph of the creature before slowly floating back up to the research ship."* Interestingly, models appear to arrive at largely similar layouts for visualizing stories.

D HUMAN STUDY DETAILS

This study was reviewed and approved by an internal ethics committee. Participants provided informed consent prior to participation.

The survey consists of three blocks in a fixed order (48 total survey items):

1. **Recursion vs. iteration block:** Participants view selected images from both the symbol library used in Sec. 3 and visualizations generated by models in Sec. 4.1. For each visualization, participants choose whether it is better associated with recursion or iteration.
2. **Concept block:** Participants view selected visualizations models generated in the concept communication games (Sec. 4.2) and select the matching concept from the same four-option set presented to the receiver model.
3. **Story block:** Participants view visualizations from the story communication games and select the matching story from a four-option set.

Option order is shuffled per trial. The order of survey items within each block is randomly shuffled from participant to participant. For blocks 2 and 3, we select from both first-turn and final-turn visualizations to explore human interpretation alignment with sender and receiver models, respectively.

Each participant evaluated visualizations from one communication channel (free-form PIL or symbol; $N = 4$ and $N = 3$ respectively) and from all three models. Minimal information was provided to participants; for example, they were not told beforehand that we were comparing their interpretations to model associations. We also didn't share the names of the models.