SketchAgent: Language-Driven Sequential Sketch Generation

Yael Vinker[1] Tamar Rott Shaham[1] Kristine Zheng[2] Alex Zhao[1] Judith E Fan[2] Antonio Torralba[1]

[1]MIT
{yaelvink,tamarott,alexzhao,torralba}@mit.edu

[2]Stanford University
{jefan,kxzheng}@stanford.edu

https://sketch-agent.csail.mit.edu/

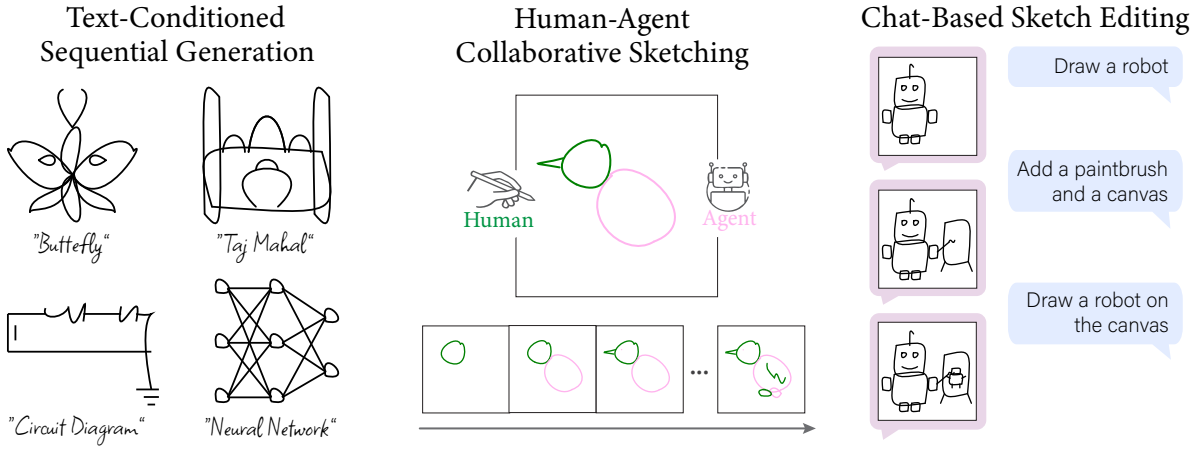


Figure 1. SketchAgent leverages an off-the-shelf multimodal LLM to facilitate language-driven, sequential sketch generation through an intuitive sketching language. It can sketch diverse concepts, engage in interactive sketching with humans, and edit content via chat.

Abstract

Sketching serves as a versatile tool for externalizing ideas, enabling rapid exploration and visual communication that spans various disciplines. While artificial systems have driven substantial advances in content creation and human-computer interaction, capturing the dynamic and abstract nature of human sketching remains challenging. In this work, we introduce SketchAgent, a language-driven, sequential sketch generation method that enables users to create, modify, and refine sketches through dynamic, conversational interactions. Our approach requires no training or fine-tuning. Instead, we leverage the sequential nature and rich prior knowledge of off-the-shelf multimodal large language models (LLMs). We present an intuitive sketching language, introduced to the model through in-context examples, enabling it to "draw" using string-based actions. These are processed into vector graphics and then rendered to create a sketch on a pixel canvas, which can be accessed again for further tasks. By drawing stroke by stroke, our agent captures the evolving, dynamic qualities intrinsic to sketching. We demonstrate that SketchAgent can generate sketches from diverse prompts, engage in dialogue-driven drawing, and collaborate meaningfully with human users.

1. Introduction

Sketching is a powerful tool for distilling ideas into their simplest form. Its fluid and spontaneous nature makes sketching a uniquely versatile tool for visualization, rapid ideation, and communication across cultures, generations, and disciplines [31, 113]. For example, designers use sketches to explore new ideas [44, 114], scientists employ them to formulate problems [55, 82], and children engage in sketching to learn and express themselves [32, 33] (see Fig. 2). Artificial systems, in principle, have the potential to support and enhance human creativity, problem-solving, and visual expression through sketching, adapting flexibly to their exploratory nature [26, 109, 133].

Traditionally, sketch generation methods rely on human-drawn datasets to train generative models [6, 8, 20, 41, 48, 67]. However, fully capturing the diversity of sketches within datasets remains challenging [31], limiting these methods in both scale and diversity. Recent advancements in vision-language models, such as CLIP [88] and text-to-image diffusion [92], have enabled sketch generation methods that reduce reliance on human-drawn datasets [34, 53, 116]. These methods leverage pretrained model guidance and differentiable rendering [66] to optimize parametric curves, creating sketches that go beyond predefined styles and categories.

While representing a significant step toward a general-purpose sketching system, these methods lack a crucial aspect of human drawing: the *process* itself. Current methods, though versatile, optimize all strokes simultaneously, making the intermediate sketching steps meaningless. As a result, the sketch cannot be decomposed into a coherent sequence of strokes that reflects the drawing process. In contrast, humans draw iteratively, stroke by stroke, incorporating visual feedback and continuously adapting—a dynamic, evolving process that fosters creativity, ideation, and communication [60, 98, 112].

In this work, we introduce SketchAgent, a sketch generation agent that leverages the prior knowledge and sequential nature of multimodal large language models (LLMs) to enable versatile, progressive, language-driven sketching. Our agent can generate sketches across a wide range of textual concepts—from animals to engineering principles (Fig. 1, left). Its sequential nature facilitates interactive human-agent sketching and supports iterative refinement and editing through a chat-based dialogue (Fig. 1, right).

Unlike vision-language models that directly generate images from text [85, 90, 92], multimodal LLMs [1, 3, 19, 64, 73, 84, 108] accept text and images as input but only output text. To produce visuals, they either utilize external "tools" (such as calling a text-to-image model) or are prompted to generate executable code (e.g., Python [49], SVG [12]) to create charts, diagrams, or graphics. However, prompting for such representations to directly produce sketches often results in a mechanical appearance with uniform, precise shapes that lack the subtle irregularities and spontaneous qualities characteristic of human sketches (see Fig. 3B). Additionally, despite their robustness in textual tasks, these models often struggle with fine-grained spatial reasoning [47, 129] as they are primarily optimized for text, making sketch editing more challenging.

To address these limitations, we introduce an intuitive sketching language that enables an off-the-shelf multimodal LLM agent to "draw" sketches on a canvas by providing string-based actions, without additional training or fine-tuning. We define the canvas as a numbered grid, allowing the agent to reference specific coordinates (e.g., `x2y8`) to enhance its spatial reasoning capabilities. We represent a sketch as a sequence of semantically meaningful strokes, each defined by a series of such coordinates. We leverage In-Context Learning (ICL) [9] to introduce the agent to the new representation, and Chain of Thought (CoT) [119] to enhance its planning capabilities. Given a sketching task, the agent produces a textual response following our representation, which we process by fitting a smooth Bézier curve to each coordinate sequence. The curves are then rendered onto the canvas to form the final sketch. We find this approach useful in emulating a more natural sketch appearance. For collaborative sketching, the canvas remains accessible to both the user and the agent throughout the session. The agent generates strokes sequentially and pauses according to an adjustable stopping token, allowing the user to add their own strokes directly to the canvas. These strokes are then integrated into the agent's sequence, enabling it to continue drawing, with real-time canvas updates.

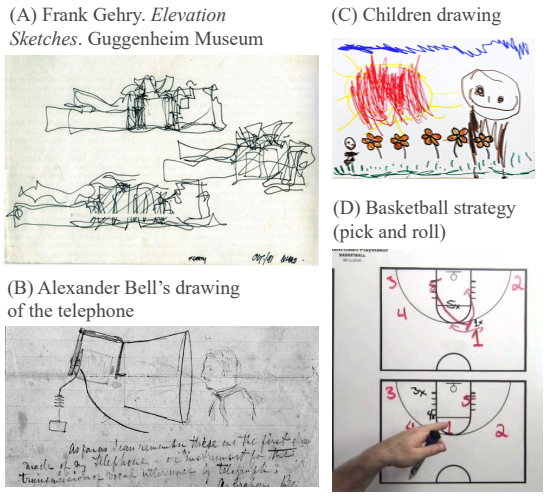


Figure 2. Examples of sketches used across disciplines and goals. (A) Ideation and design: *Process Elevation Sketches* by the architect Frank Gehry, Guggenheim Museum. (B) Engineering: Alexander Bell's telephone drawing. (C) Expressing emotions: Children's sketches. (D) Visual communication: Planning and communicating game strategy in basketball.

We demonstrate SketchAgent's capabil to generate sketches of diverse concepts while capturing the inherently sequential and dynamic nature of sketching. We showcase our agent's ability to collaborate effectively with humans in real time to create novel and meaningful sketches. Our method is the first to leverage pretrained multimodal LLMs for sequential sketching without additional training, paving the way for a general-purpose artificial sketching system that supports iterative, evolving interactivity.

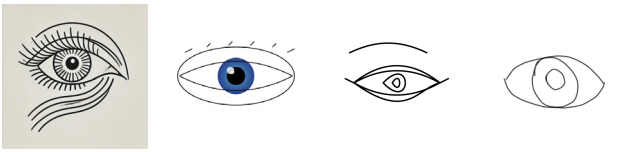
2. Related Work

Sketch Generation Early methods approached sketch generation by designing image filters to simulate sketch-like effects [13, 120]. With the advent of deep learning, data-driven approaches emerged to address a range of sketch-related tasks [128], including category-conditioned sketching [48, 86, 103], object sketching [67, 70], scene-sketching [16, 65, 68, 125], sketch completion [7, 71, 72, 105], portrait drawing [4, 131, 132], part-based generation [7, 42, 48, 140], and more. While sketch data collection has been broadly explored [25, 39, 46, 81, 95, 124], the wide variation in sketch styles and their adaptation to specific tasks [28] makes collecting datasets that encompass this diversity challenging. For example, Quick-Draw [54], the largest available sketch dataset with 50 mil-

lion sketches, covers only 345 object categories and primarily focuses on simple, iconic representations. This limits its data-driven methods to the style, abstraction level, and concepts seen during training. Recently, large pretrained vision-language models [85, 88, 90, 92, 94] have shown remarkable text-to-image generation capabilities by leveraging extensive visual knowledge from billions of training images [99]. While these models can be prompted to generate sketch-like images (see Fig. 3A), they do so in a single step and in pixel space, lacking the sequential, stroke-based process of human sketching. Subsequent approaches [18, 34, 36, 53, 116, 117, 126, 127, 135] leverage the priors of these models to guide an iterative optimization of parametric curves, with a differentiable rasterizer [66] linking pixel and vector representations. This approach reduces reliance on human-drawn datasets, enabling robust sketch generation beyond pre-defined categories. However, optimizing all strokes simultaneously results in sketches that lack temporal and semantic structure, and the process is time-consuming, taking 5 minutes to over an hour per sketch, making it suboptimal for collaborative sketching.

Sequential and Collaborative Sketching Collaborative human-machine sketching holds promise in enhancing creativity, ideation, communication, and learning, as explored in various fields, including human-computer interaction (HCI) [21, 52, 56, 57, 61, 62], computer graphics [63, 107], robotics [96, 97], cognitive science [29, 30, 40, 77], learning sciences [2, 43, 115], and more. Central to collaborative sketching is its sequential, adaptive, and dynamic process, with each action carrying intent. Existing methods employ diverse training strategies to account for the discrete nature of sequential sketches, including reinforcement and adversarial learning [37, 78, 140], multi-agent referential games [79, 87], transformers [6, 7, 14, 38, 69, 91, 123], and more. SketchRNN [48] is a pioneering work in this area, introducing the QuickDraw dataset [54], a crowd-sourced collection of real-time sketch sequences made by users. They utilize this dataset to train a recurrent neural network for sequential sketch generation, which was later shown [29, 83] to have potential for human-machine collaboration. However, this approach remains constrained by the predefined categories encountered during training.

Multimodel LLMs for Content Creation LLMs [10, 22, 89, 111] and multimodal LLMs [1, 3, 19, 64, 73, 84, 108] receive text as input (or text and images for multimodal) and output text. To enable visual content generation, these models are often paired with external "tools" that extend their functionality [51, 100, 122, 130]. For example, ChatGPT [84] generates images by internally calling a separate model, DALLE-3 [5]. Another approach involves prompting models to produce code in languages like Python [49], Processing [102], SVG [12], or TikZ [11] that can be rendered into visuals such as graphs, charts, and vector graphics. However, such code-generated content often looks rigid, with uniform and overly precise shapes that lack the subtle irregularities and spontaneous qualities characteristic of freehand sketches (see Fig. 3B). In contrast, we propose a sketching language grounded in spatial information that encourages the model to produce a more natural sketch appearance, which we then process into vector graphics. Common strategies for enhancing LLMs capabilities include Chain-of-Thought prompting [17, 80, 93, 101, 138], which breaks down tasks into smaller, logical steps to mimic human reasoning, and In-Context Learning (ICL) [23, 106, 134, 136], where examples of input-output pairs are provided to help the model infer task patterns.



(A) DALLE3 [5] (B) LLMs [3] (SVG) (C) SketchAgent (D) Human [54]

Figure 3. Sketch appearance. (A) Text-to-image diffusion models operate in pixel space, lacking thesequential nature of sketches. (B) Prompting LLMs to produce visuals with SVG results in a uniform, mechanical appearance. (C) Sketches produced by our agent appear less mechanical, more closely resembling the nature of (D) Human sketches, which are often spontaneous and irregular.

3. Preliminaries

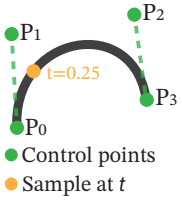


Figure 4. Cubic Bézier curve.

Vector Graphics and Bézier Curves Vector graphics allow us to create visual images directly from geometric shapes such as points, lines, curves, and polygons. Unlike raster images (represented with pixels), vector graphics are resolution-free, more compact, and editable. SVG [121] is an XML-based format for storing vector graphics, popular for its scalability and compatibility with modern web browsers. The process of transferring vector graphics into pixel images is called rasterization or rendering. Cubic Bézier curves are commonly used to represent sketches in vector graphics. A cubic Bézier curve (Fig. 4) is a smooth parametric curve defined by four points: a start point P_0, an end point P_3, and two control points P_1 and P_2 that shape the curvature . The set $P = \{P_0, P_1, P_2, P_3\}$ is often referred to as the curve's control points. The curve is described by the following polynomial equation:

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, \quad (1)$$

where $t \in [0, 1]$ is a parameter that moves the point along the curve from P_0 at $t = 0$ to P_3 at $t = 1$.

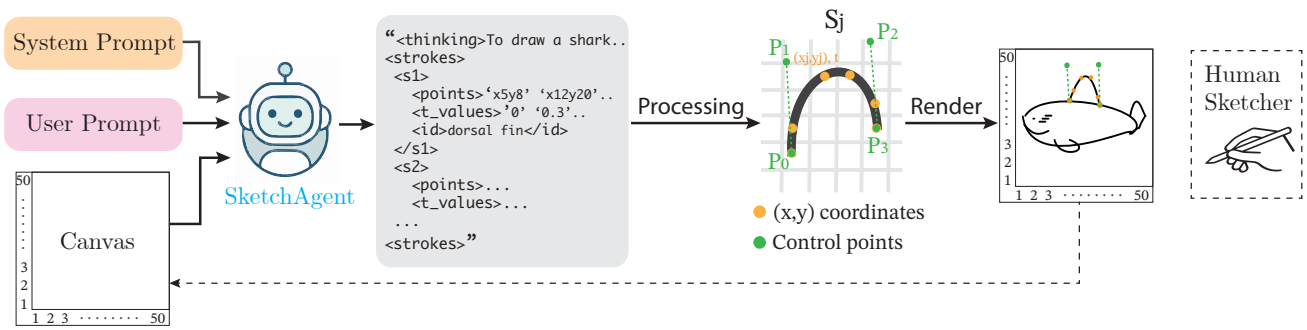


Figure 5. Method Overview. SketchAgent (blue) receives drawing instructions and generates a string representing the intended sketch. Inputs include: (1) a system prompt (orange) introducing the sketching language and canvas, (2) a user prompt (pink) specifying the task (e.g., "draw a shark"), and (3) a numbered canvas. The agent's response outlines a sketching strategy (in thinking tags) and a sequence of strokes defined by coordinates, which are processed into Bézier curves and rendered onto the canvas.

4. Method

Our goal is to enable an off-the-shelf pretrained multimodal LLM to draw sketches based on natural language instructions. An overview of our pipeline is illustrated in Fig. 5. We utilize a frozen multimodal LLM ("SketchAgent" shown in blue), which receives three inputs: (1) a system prompt containing guidelines for using our new sketching language, (2) a user prompt with additional task-specific instructions (e.g., "*Draw a shark*"), and (3) a blank canvas on which the agent can draw. Based on the given task, the agent generates a textual response, representing the sequence of strokes to be drawn, which we then process into vector graphics and render onto the canvas. The canvas can then be reused in two ways: it can be fed back into the model with an updated user prompt for additional tasks and editing, or it can be accessed by a human user who can draw directly on it to facilitate collaborative sketching. Next, we describe each component of the pipeline.

The Canvas Although multimodal LLMs demonstrate remarkable reasoning abilities, they often struggle with spatial reasoning tasks [35, 75, 110]. We present a simple example (see Fig. 6) to illustrate how this limitation affects the naive use of these models for sketch generation and interactive sketching. We provide GPT-4o [84] with an image depicting a simple line drawing of a partial house featuring five numbered points (from 1 to 5), and ask it to identify which points should be connected to complete the house. While the model correctly identifies the pair of points, it fails to select the correct pixel coordinates when given a basic `draw_line` tool that connects two points, even after multiple attempts. To enhance the model's spatial reasoning ability, we utilize a numbered canvas that forms a grid. This grid features numbers (1 to 50) along the x-axis and the y-axis (Fig. 5, left). Each cell is uniquely identified by a combination of the corresponding x-axis and y-axis numbers (e.g., the bottom-left cell is `x1y1`). The agent interacts with the canvas by specifying desired `(x,y)` coordinates.

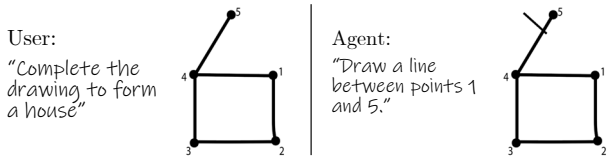


Figure 6. Although excelling in visual reasoning, multimodal LLMs often struggle to translate these abilities into spatial actions. In this example, GPT-4o [84] intends to draw a line between points 1 and 5 but fails to execute this with a `draw_line` function that accepts pixel coordinates.

Sketch Representation We define a sketch as a sequence of n ordered strokes $S = \{S_1, S_2, ...S_n\}$. Each stroke S_i is defined by a sequence of m cell coordinates on the grid: $S_i = \{(x_j, y_j)\}_{j=1}^{m}$, represented in string format as: `<points>x1y1, x15y20, ...</points>`.

A naive approach to processing the textual sequence of coordinates would be to use a polyline, connecting consecutive points with line segments. However, our grid-based representation sparsifies the canvas, resulting in a non-smooth and unnatural appearance when using polylines (see Fig. 7, left). To achieve a smoother appearance, an alternative approach is to treat the coordinates as a sequence of control points defining smooth curves. However, as illustrated in Fig. 4, the control points often do not lie directly on the curve. Consequently, if the agent aims for a stroke that passes through specific coordinates, it must derive the control points that define this stroke, which is challenging.

We propose an alternative approach: we treat the specified `(x,y)` coordinates as a set of desired points sampled **along** the curve, and fit a smooth Bézier curve to them (Fig. 7, right). To accommodate curves with complex curvature, we also task the model with determining **when** each point on the curve should be passed through, corresponding to the t value described in Eq. (1). Thus, for each stroke S_i, the agent provides a set of m sampled points $S_i = \{(x_j, y_j)\}_{j=1}^{m}$, along with a corresponding set of t values: $T_i = \{t_j\}_{j=1}^{m}$. Based on these, we fit a cubic Bézier

(A) Polyline (B) Control Points (C) Ours

Figure 7. Methods for processing the agent's coordinate sequence (in red): (A) Polyline results in an unnatural appearance. (B) Directly using coordinates as Bézier control points is challenging as they do not lie on the curve. (C) Fitting a Bézier curve to sampled coordinates provides smoother results.

curve to the sampled points by solving a system of linear equations using least squares, where the unknowns are the control points $P = \{P_0, P_1, P_2, P_3\}$:

$$P = \text{argmin}_P ||AP - B||, \tag{2}$$

where $A \in \mathbb{R}^{m\times 4}$ contains the cubic Bézier basis functions evaluated at specific t_j values (as described in Eq. (1)), and $B \in \mathbb{R}^{m\times 2}$ contains the m sampled points $\{(x_j, y_j)\}_{j=1}^{m}$. The least squares solution minimizes the error between the fitted Bézier curve and the sampled points. For long sequences resulting in a large fitting error, we recursively split the curve. Additionally, we account for Bézier curves of lower degrees, including quadratic curves, linear lines, and points. Upon completing this process, we render the parametric curves onto the canvas.

Drawing Instructions We provide the model with a system prompt and a user prompt (marked in orange and pink in Fig. 5). In the system prompt, we supply the agent with context about its expertise ("*You are an expert artist specializing in drawing sketches*") and introduce it to the grid canvas along with examples of how to use our sketching language for drawing single-stroke primitives (full prompts are provided in the Appendix). The system prompt is fixed and can be applied to a variety of sketching tasks. The user prompt includes a description of the desired task and an example of a simple sketch of a house drawn with our sketching language. We find this to be crucial in assisting the agent with preserving the correct format that could be parsed directly [9]. The agent is tasked with responding in the format shown in the gray text box in Fig. 5. In the `<thinking>` tags, the agent is tasked to outline the overall sketching strategy [119]. This typically includes describing the different components of the sketch, the intended sketching order, and the overall placement of each part. The agent is also tasked with providing an ID tag following each stroke, which is useful for further analysis and for producing annotated sketches in scale.

4.1. In-Chat Editing and Collaborative Sketching

The above process can be repeated iteratively to support multiple sketching tasks and interactions. Text-based sketch editing in a chat dialogue is enabled by feeding the rendered canvas back to the agent (see dashed arrow in Fig. 5) and updating the user prompt with the desired edits. To support collaborative human-agent sketching, the canvas remains accessible to both the human user and the agent throughout the entire sketching session. We define an adjustable stopping token, `</s{j}>`, which instructs the agent to pause generating the sequence at stroke number j. We then process and render the generated strokes onto the canvas up to that point, then the user can add strokes directly to the canvas to continue the sketch. The user-drawn strokes are processed and converted into the agent's format by reversing our fitting process, i.e., sampling each stroke at multiple t values (as shown in Eq. (1)), and selecting the points closest to each cell's center on the grid. The converted user strokes are then chained with the agent's sequence, after which the agent resumes sketching until the next stopping token.

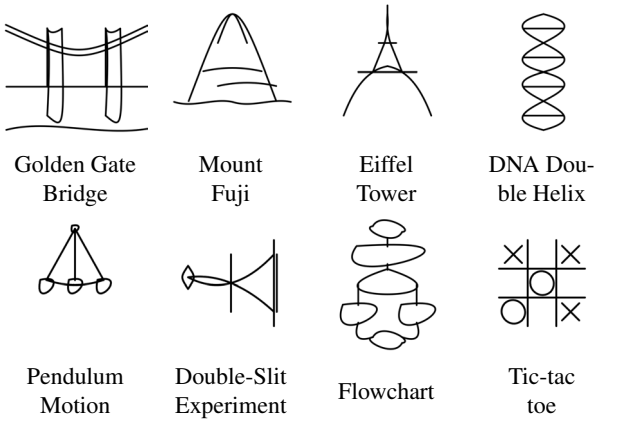


Figure 8. Sketches produced by SketchAgent for concepts beyond pre-defined categories. The textual input describing the desired concept shown below each image.

5. Results

We evaluate the performance of our method qualitatively and quantitatively across a selected set of sketching tasks. Additional tasks, evaluations, and examples are provided in the Appendix. All results presented in the paper were generated using Claude3.5-Sonnet [3] as our backbone model, unless stated otherwise.

5.1. Text-Conditioned Sketch Generation

Figures 1 and 8 demonstrate SketchAgent's capability to generate sketches of various concepts that extend beyond standard categories, which includes scientific concepts (e.g., "the double-slit experiment", "pendulum motion"), diagrams (e.g.,"circuit diagram", "a flowchart"), and notable landmarks (e.g., "Taj Mahal", "Eiffel Tower"). More examples are provided in the Appendix. To quantitatively

	GPT-4o	GPT-4o -mini	Claude3 Opus	Claude3.5 -Sonnet*	Claude3.5 -Sonnet (SVG)	Human (QD [54])
Top1	0.15 ±0.04	0.04 ±0.03	0.13 ±0.04	0.23 ±0.05	0.23 ±0.04	0.27 ±0.07
Top5	0.30 ±0.06	0.10 ±0.04	0.27 ±0.05	0.44 ±0.03	0.43 ±0.06	0.49 ±0.06
Vis.						

Table 1. Sketch recognition evaluation. Average Top-1 and Top-5 sketch recognition accuracy computed with CLIP zero-shot classifier on 500 sketches from 50 categories. The last row visualizes one sample from each experiment. *Indicates our default settings, which receives the highest accuracy among all models.

evaluate text-conditioned generation we utilize the QuickDraw dataset [54]. We randomly sample 50 categories (out of 345), and apply our method to generate 10 sketch instances per category, resulting in 500 sketches in total. Following common practice [116, 117, 126, 127], we utilize a CLIP zero-shot classifier [88] to evaluate how well the generated sketches depict the intended category. We compare the performance of different multimodal LLMs by repeating the same process with GPT-4o-mini [84], GPT-4o [84], and Claude3-Opus [3] as our backbone model (in addition to Claude3.5-Sonnet [3], our default backbone). As a baseline, we include human-drawn sketches sampled from the QuickDraw dataset [54]. The average Top-1 and Top-5 sketch classification accuracy are presented in Table 1. As can be seen, human sketches achieve the highest recognition accuracy, with Claude3.5-Sonnet performing best among all models, approaching human-level rates under the CLIP-score metric. More evaluation of confusion patterns and visualization of the data are provided in the Appendix.

We additionally compare to prompting Claude3.5-Sonnet to directly generate SVGs using the following prompt: *"Write SVG string that draws a sketch of a <concept>. Use only black and white colors"*. The corresponding scores are shown in the fifth column of Tab. 1. While this approach achieves recognition scores comparable to those of SketchAgent, the outputs are often characterized by uniform and precise shapes, failing to replicate the fluidity and natural irregularity of free-hand human sketches (e.g., Fig. 3). To evaluate how "human-like" our agent's sketches appear, we conduct a two alternative forced choice (2AFC) user study with 150 participants. Each participant was presented with pairs of sketches depicting the same object class produced by different methods, and asked to choose the sketch they believed was human-drawn. 150 sketches across 50 object classes were tested, comparing three methods: direct prompting, SketchAgent, and human sketches from QuickDraw (see Appendix for details). Results indicate SketchAgent's drawings appeared more human-like, being chosen as human-drawn in $74.90 \pm 3.35\%$ of cases when compared with direct prompting. When compared to human drawings, users slightly preferred human sketches ($54.68 \pm 4.61\%$) over SketchAgent's, while direct prompting was chosen only $38.9 \pm 5.55\%$ of the time.

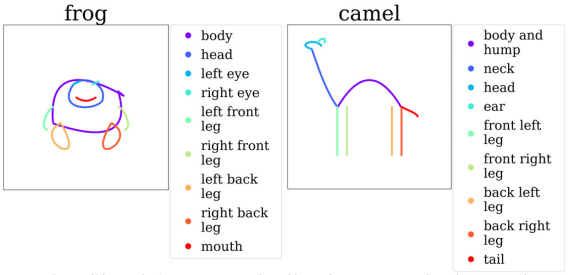


Figure 9. SketchAgent gradually draws stroke-by-stroke, each stroke is annotated by the agent with a semantic meaning.

5.2. Sequential Sketching

Figure 9 shows stroke-by-stroke sketch generation by SketchAgent, with the labels on the right indicating the sketching order and the meaning our agent associates with each generated stroke (see Appendix for more examples). Stroke annotation during generation is enabled by utilizing the prior of the backbone LLM, providing a valuable feature for analysis and data collection [42, 74, 118, 137, 139]. In Fig. 10, we illustrate why accounting for the sequential nature of sketching more closely emulates the process of human drawing. We present the sketch creation process of SketchAgent alongside SVGDreamer [127], SketchRNN [48], and a human sketch sampled from QuickDraw [54]. SVGDreamer (first row), is an optimization-based method, where a set of randomly initialized parametric curves (leftmost column) are iteratively refined to form a sketch, guided by a pretrained text-to-image diffusion model [92]. This process is time-consuming, taking 2000 iterations (1.6 hours), which makes it unsuitable for interactive sketching. While the final sketch (rightmost column) appears detailed and artistic due to the powerful vision backbone, the intermediate sketching and individual strokes lack clear semantic meaning. In contrast, SketchRNN (second row) is a sequential generative model trained on human-drawn dataset, producing sketches in real-time with strokes added progressively, emulating closer a human-like sketching process (as shown in the last row). Similarly, SketchAgent (third row) produces sketches gradually, with each stroke carrying a semantic meaning, by utilizing the sequential nature of its backbone model. While SketchRNN is restricted to generating sketches only within the 345 categories it was trained on, SketchAgent leverages the extensive prior knowledge of its backbone multimodal LLM, enabling it to create sketches of general visual concepts.

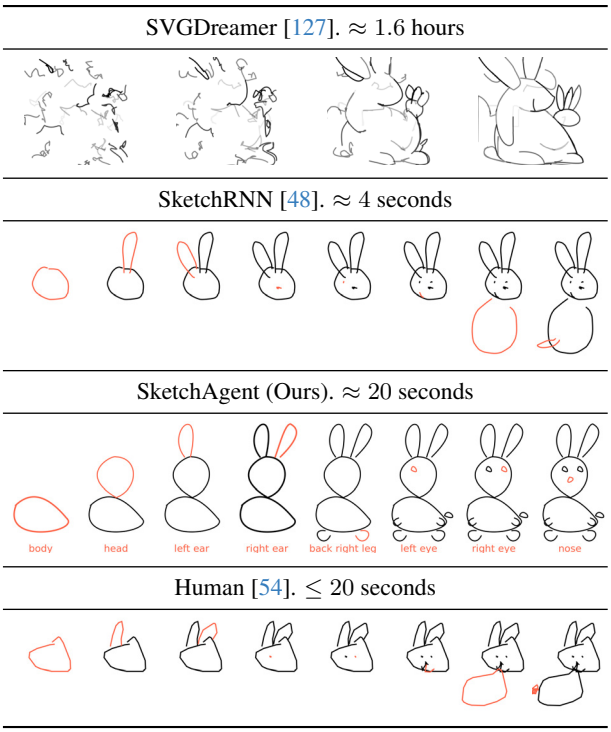


Figure 10. Sequential sketching process. SVGDreamer [127] requires 2000 iterations (1.6 hours) with intermediate steps lacking semantic meaning. SketchRNN [48] operates in real-time with coherent steps but is limited to QuickDraw categories. SketchAgent creates sketches gradually with meaningful strokes and no category restrictions. Human sketches also evolve through gradual, meaningful steps.

We use the set of 500 samples described in Sec. 5.1 to quantitatively analyze the sequential nature of our agent's sketches compared to human drawings. On the left of Fig. 11, we present histograms comparing the number of strokes in QuickDraw sketches (orange) and our sketches (blue). Most QuickDraw sketches contain 1 to 6 strokes, while our sketches show a broader distribution, peaking between 5 to 10 strokes. This suggests that, on average, QuickDraw sketches appear more abstract. To ensure a balanced comparison of sketches with similar levels of abstraction, we select sketches from both groups with a similar number of strokes (the largest intersection is found in sketches with 4-7 strokes, comprising 204 of our sketches and 120 from QuickDraw) and measure the change in CLIPScore as a function of the accumulated number of strokes (Fig. 11, right). Both QuickDraw and our sketches exhibit a generally similar pattern, with CLIPScore increasing as more strokes are added, suggesting that sketches become progressively more recognizable as they evolve.

5.3. Human-Agent Collaborative Sketching

We demonstrate the potential of our system for facilitating interactive human-agent collaboration, resulting in semantically meaningful and recognizable sketches. We design a web-based collaborative sketching environment (Fig. 12A) where users and SketchAgent take turns drawing on a shared canvas to create a recognizable sketch from a given textual concept. Following the evaluation protocol in collabdraw [29], we select 8 simple concepts, based on the agent's demonstrated ability to draw them independently, to focus evaluations on assessing the impact of *collaboration*. Participants sketched concepts in two modes: *solo*, where users drew independently, and *collab*, where users and SketchAgent collaborated, adding one stroke at a time until either was satisfied with the drawing. We collect sketches from 30 participants, resulting in 480 sketches in total. Average CLIP recognition rates are shown in Figure 12B. Collaboratively produced sketches (blue) achieve recognition levels close to those made solely by users and higher than those produced by the agent alone (dashed lines). To assess the contribution of each party in collaborative mode, we analyze partial sketches with only agent-made strokes (pink) or user-made strokes (green), resulting in a significant reduction in recognizability. This suggests that both user and agent contribute meaningfully to the recognizability of the complete sketch.

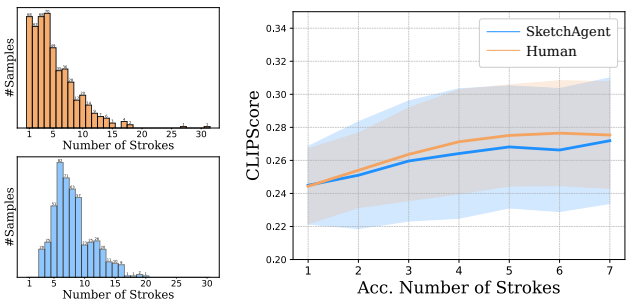
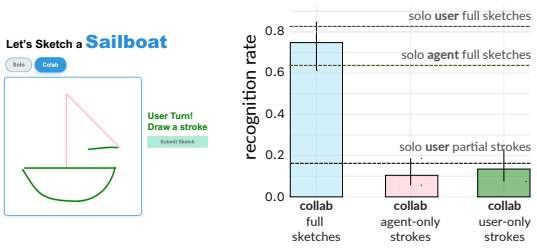


Figure 11. Sequential sketching analysis of SketchAgent (blue) and Humans [54] (orange). Left: Histograms of stroke distribution per sketch, showing QuickDraw sketches are more abstract on average. Right: CLIPScore as a function of the accumulated number of strokes for sketches containing 4-7 strokes, showing a similar recognition pattern over time.

5.4. Chat-Based Sketch Editing

We next demonstrate the effectiveness of our method in performing interactive text-based sketch editing within a chat dialogue, where the input to the agent combines both text and images. Inspired by [102], we explore edits that involve spatial reasoning and object relations. We focus on three object categories: outdoor, indoor, and animals, with three objects each, and design editing prompts to add objects to the input sketches. For outdoor and indoor objects, we specify relative locations of added concepts, e.g., "left to", "on top of" (see Fig. 47 left). For the animals category, we tasked the agent with adding accessories to each animal



(A) Sketching interface (B) Collaborative user study results

Figure 12. Collaborative sketching evaluation measured using CLIP classification. Sketches created collaboratively (blue) approaching those made solely by users (dashed lines). In collaborative sketches, keeping agent-only strokes (pink) or user-only strokes (green) significantly reduces recognizability.

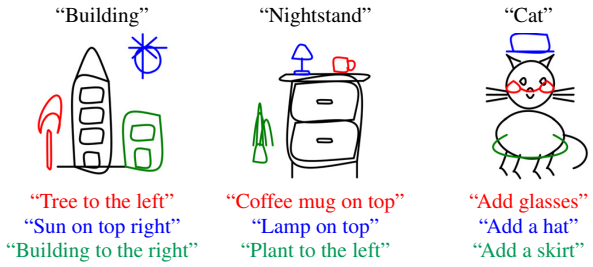


Figure 13. Chat-based sketch editing. We iteratively prompt SketchAgent to add objects to sketches through chat dialogues.

without guidance on their exact placement, testing its ability to infer placement based on semantics (e.g., placing a hat on a head (see Fig. 47 right). The full list of object and editing instructions is provided in the Appendix. We produced a total of 54 sketches. Evaluating the edited sketches reveals that SketchAgent correctly follows instructions 92% of the time, with 94% accuracy for specified relations and 88% accuracy for inferred semantic relations.

6. Ablation

We evaluate the impact of each component of our method by systematically removing them and measuring sketch recognition rates as detailed in 5.1. We assess the effects of removing the system prompt, omitting the CoT process (i.e., excluding thinking tags and 'think step-by-step' instructions), and modifying ICL (the complete sketch example provided in the user prompt). When modifying ICL, we use a correctly formatted single-stroke example instead of the complete sketch, as fully removing ICL results in outputs that do not follow the expected format making them unparsable. The results in Table 2 show that the full SketchAgent pipeline achieves the highest performance, highlighting the importance of each component. Interestingly, not providing a complete sketch example significantly reduces performance. Additional visualizations and analyses are provided in the Appendix.

	w/o System Prompt	w/o CoT	Modified ICL	SketchAgent (full)
Top1	0.20 ± 0.04	0.14 ± 0.02	0.07 ± 0.02	0.23 ± 0.04
Top5	0.42 ± 0.03	0.29 ± 0.04	0.16 ± 0.03	0.43 ± 0.06

Table 2. Ablation study. Average Top-1 and Top-5 CLIP recognition accuracy. We systematically remove each component in our pipeline, showcasing all components contribute to the agent's full performance.

7. Limitations and Future Work

SketchAgent has several limitations. First, it is constrained by the priors of the backbone model, primarily optimized for text rather than visual content. As a result, the agent often produces rich textual descriptions of object parts but struggles to convert these into effective sketching actions, resulting in overly abstract and unrecognizable outputs. For example, in Fig. 14A, the agent effectively describes key parts of a unicorn (e.g., the horn), but the sketch is unrecognizable. This constraint also impacts the depiction of human figures (Fig. 14B). While distinctive features (e.g., Frida Kahlo's eyebrows or Michael Jordan's dunk) may be captured well in language, the resulting sketches are overly simple, with an amateur style, lacking expressivity. We expect this issue to improve as future models advance in vision capabilities. Lastly, the agent may struggle with drawing letters and numbers. This could be improved in future work by providing relevant in-context examples.

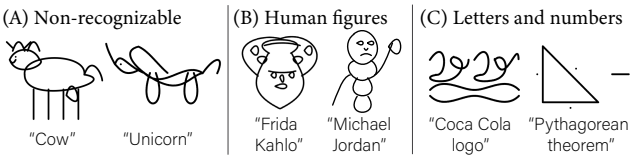


Figure 14. Limitations. Sketches of complex concepts (A) and human figures (B) appear too abstract and unrecognizable with non-professional style. (C) Fail to depict letters and numbers.

8. Conclusions

We presented a method for language-driven, sequential sketch generation, that can produce versatile sketches in real-time and meaningfully engage in collaborative sketching sessions with humans. We show that the prior knowledge embedded in pretrained multimodal LLMs can be effectively leveraged for sketch generation through an intuitive sketching language and a grid canvas, without requiring additional training or fine-tuning. We hope our work represents a meaningful step toward developing general-purpose sketching systems with the potential to enhance human-computer communication and computer-aided ideation.

9. Acknowledgements

We thank Yuval Alaluf, Hila Chefer, Assaf Ben Kish, Joanna Materzynska, Rinon Gal, Elad Richardson, Arnav Verma, and Ellie Arar for providing feedback on early versions of our manuscript. We are especially grateful to Yarden Frenkel for his insights, early explorations, and engaging discussions. This work was partially supported by NSF CAREER #2047191, NSF DRL #2400471, Stanford Human Centered AI Institute Hoffman-Yee Grant, Hyundai Motor Company, ARL grant W911NF-18-2-021, the Zuckerman STEM Leadership Program, and the Viterbi Fellowship. The funders had no role in the experimental design or analysis, the decision to publish, or manuscript preparation. The authors have no competing interests to report.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millicah, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. 2, 3

[2] Vanessa De Andrade, Sofia Freire, Mónica Baptista, and Yael Shwartz. Drawing as a space for social-cognitive interaction. *Education Sciences*, 12(1), 2022. 3

[3] Anthropic. Claude. https://www.anthropic.com/claude, 2023. 2, 3, 5, 6, 1

[4] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. Style and abstraction in portrait sketching. *ACM Trans. Graph.*, 32(4), 2013. 2

[5] James Betker, Gabriel Goh, Li Jing, † TimBrooks, Jianfeng Wang, Linjie Li, † LongOuyang, † JuntangZhuang, † JoyceLee, † YufeiGuo, † WesamManassra, † PrafullaDhariwal, † CaseyChu, † YunxinJiao, and Aditya Ramesh. Improving image generation with better captions. 3

[6] Ayan Kumar Bhunia, Ayan Das, Umar Riaz Muhammad, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. Pixelor: a competitive sketching ai agent. so you think you can sketch? *ACM Trans. Graph.*, 39:166:1–166:15, 2020. 1, 3

[7] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, Jorma Laaksonen, and Michael Felsberg. Doodleformer: Creative sketch drawing with transformers. *ECCV*, 2022. 2, 3

[8] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, Jorma Laaksonen, and Michael Felsberg. Doodleformer: Creative sketch drawing with transformers. In *European Conference on Computer Vision*, pages 338–355. Springer, 2022. 1

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901. Curran Associates, Inc., 2020. 2, 5

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901. Curran Associates, Inc., 2020. 3

[11] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. 3

[12] Mu Cai, Zeyi Huang, Yuheng Li, Haohan Wang, and Yong Jae Lee. Delving into LLMs' visual understanding ability using SVG to bridge image and text, 2024. 2, 3

[13] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 2

[14] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020. 3

[15] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 13

[16] Caroline Chan, Frédo Durand, and Phillip Isola. Learning to generate line drawings that convey geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7915–7925, 2022. 2

[17] Zhenfang Chen, Qinhong Zhou, Yikang Shen, Yining Hong, Zhiqing Sun, Dan Gutfreund, and Chuang Gan. Visual chain-of-thought prompting for knowledge-based visual reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1254–1262, 2024. 3

[18] Changwoon Choi, Jaeah Lee, Jaesik Park, and Young Min Kim. 3doodle: Compact abstraction of objects with 3d strokes. *ACM Trans. Graph.*, 43(4), 2024. 3

[19] Xiangxiang Chu, Jianlin Su, Bo Zhang, and Chunhua Shen.

Visionllama: A unified llama backbone for vision tasks, 2024. 2, 3

[20] Ayan Das, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Yi-Zhe Song. Béziersketch: A generative model for scalable vector sketches. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*, pages 632–647. Springer, 2020. 1

[21] Nicholas Davis, Chih-PIn Hsiao, Kunwar Yashraj Singh, Lisa Li, Sanat Moningi, and Brian Magerko. Drawing apprentice: An enactive co-creative agent for artistic collaboration. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, page 185–186, New York, NY, USA, 2015. Association for Computing Machinery. 3

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. 3

[23] Sivan Doveh, Shaked Perek, M Jehanzeb Mirza, Wei Lin, Amit Alfassy, Assaf Arbelle, Shimon Ullman, and Leonid Karlinsky. Towards multimodal in-context learning for vision & language models. *arXiv preprint arXiv:2403.12736*, 2024. 3

[24] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 4

[25] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph.*, 31(4), 2012. 2

[26] Ziv Epstein, Aaron Hertzmann, Laura Mariah Herman, Robert Mahari, Morgan R. Frank, Matthew Groh, Hope Schroeder, Amy Smith, Memo Akten, Jessica Fjeld, Hany Farid, Neil Leach, Alex Pentland, and Olga Russakovsky. Art and the science of generative ai. *Science*, 380:1110 – 1111, 2023. 1

[27] Hugging Face. clip-vit-large-patch14. https://huggingface.co/openai/clip-vit-large-patch14. 1

[28] Judy Fan, Wilma A. Bainbridge, Rebecca Chamberlain, and Jeffrey D. Wammes. Drawing as a versatile cognitive tool. *Nature Reviews Psychology*, 2:556 – 568, 2023. 2

[29] Judith E. Fan, Monica Dinculescu, and David Ha. collabdraw: An environment for collaborative sketching with an artificial agent. In *Proceedings of the 2019 Conference on Creativity and Cognition*, page 556–561, New York, NY, USA, 2019. Association for Computing Machinery. 3, 7

[30] 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. 3

[31] Judith E Fan, Wilma A Bainbridge, Rebecca Chamberlain, and Jeffrey D Wammes. Drawing as a versatile cognitive tool. *Nature Reviews Psychology*, 2(9):556–568, 2023. 1

[32] Logan Fiorella and Shelbi Kuhlmann. Creating drawings enhances learning by teaching. *Journal of Educational Psychology*, 112(4):811, 2020. 1

[33] Kenneth Forbus, Jeffrey Usher, Andrew Lovett, Kate Lockwood, and Jon Wetzel. Cogsketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science*, 3(4):648–666, 2011. 1

[34] Kevin Frans, L. B. Soros, and Olaf Witkowski. Clipdraw: exploring text-to-drawing synthesis through language-image encoders. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. 1, 3

[35] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. *arXiv preprint arXiv:2404.12390*, 2024. 4

[36] Rinon Gal, Yael Vinker, Yuval Alaluf, Amit Bermano, Daniel Cohen-Or, Ariel Shamir, and Gal Chechik. Breathing life into sketches using text-to-video priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4325–4336, 2024. 3

[37] Yaroslav Ganin, Tejas D. Kulkarni, Igor Babuschkin, S. M. Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *ArXiv*, abs/1804.01118, 2018. 3

[38] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. In *Neural Information Processing Systems*, 2021. 3

[39] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. Sketchycoco: Image generation from freehand scene sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5174–5183, 2020. 2

[40] 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. 3

[41] Songwei Ge, Vedanuj Goswami, C Lawrence Zitnick, and Devi Parikh. Creative sketch generation. *arXiv preprint arXiv:2011.10039*, 2020. 1

[42] Songwei Ge, Vedanuj Goswami, Larry Zitnick, and Devi Parikh. Creative sketch generation. In *International Conference on Learning Representations*, 2021. 2, 6

[43] Hannie Gijlers, Armin Weinberger, Alieke Mattia van Dijk, Lars Bollen, and Wouter van Joolingen. Collaborative drawing on a shared digital canvas in elementary science education: The effects of script and task awareness support. *International Journal of Computer-Supported Collaborative Learning*, 8(4):427–453, 2013. 3

[44] Gabriela Goldschmidt. Serial sketching: visual problem solving in designing. *Cybernetics and System*, 23(2):191–219, 1992. 1

[45] Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996. 13

[46] Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia Pont, Frédo Durand, and Adrien Bousseau.

Opensketch: a richly-annotated dataset of product design sketches. *ACM Trans. Graph.*, 38(6), 2019. 2

[47] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385, 2024. 2

[48] David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. 1, 2, 3, 6, 7

[49] Yucheng Han, China. Xiaoyan Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. Chartllama: A multimodal llm for chart understanding and generation. *ArXiv*, abs/2311.16483, 2023. 2, 3

[50] Robert D. Hawkins, Megumi Sano, Noah D. Goodman, and Judith E. Fan. Visual resemblance and communicative context constrain the emergence of graphical conventions, 2021. 13

[51] 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. *arXiv preprint arXiv:2406.09403*, 2024. 3

[52] Francisco Javier Ibarrola, Tomas Lawton, and Kazjon Grace. A collaborative, interactive and context-aware drawing agent for co-creative design. *IEEE Transactions on Visualization and Computer Graphics*, 30:5525–5537, 2022. 3

[53] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1911–1920, 2023. 1, 3

[54] Jongejan Jonas, Rowley Henry, Kawashima Takashi, Kim Jongmin, and Fox-Gieg Nick. The Quick, Draw! - A.I. Experiment, 2016. 2, 3, 6, 7, 8

[55] David Kaiser. *Drawing theories apart: The dispersion of Feynman diagrams in postwar physics*. University of Chicago Press, 2019. 1

[56] Pegah Karimi, Jeba Rezwana, Safat Siddiqui, Mary Lou Maher, and Nasrin Dehbozorgi. Creative sketching partner: an analysis of human-ai co-creativity. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, page 221–230, New York, NY, USA, 2020. Association for Computing Machinery. 3

[57] Pegah Karimi, Jeba Rezwana, Safat Siddiqui, Mary Lou Maher, and Nasrin Dehbozorgi. Creative sketching partner: an analysis of human-ai co-creativity. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, page 221–230, New York, NY, USA, 2020. Association for Computing Machinery. 3

[58] Günther Knoblich, Stephen Butterfill, and Natalie Sebanz. Chapter three - psychological research on joint action: Theory and data. In *Advances in Research and Theory*, pages 59–101. Academic Press, 2011. 13

[59] Kozea. Cairosvg. https://cairosvg.org/, 2023. 1

[60] Paul Laseau. *Graphic thinking for architects and designers*. John Wiley & Sons, 2000. 2

[61] Tomas Lawton, Kazjon Grace, and Francisco J Ibarrola. When is a tool a tool? user perceptions of system agency in human–ai co-creative drawing. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, page 1978–1996, New York, NY, USA, 2023. Association for Computing Machinery. 3

[62] Tomas Lawton, Francisco J Ibarrola, Dan Ventura, and Kazjon Grace. Drawing with reframer: Emergence and control in co-creative ai. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, page 264–277, New York, NY, USA, 2023. Association for Computing Machinery. 3

[63] Yong Jae Lee, C. Lawrence Zitnick, and Michael F. Cohen. Shadowdraw: real-time user guidance for freehand drawing. In *ACM SIGGRAPH 2011 Papers*, New York, NY, USA, 2011. Association for Computing Machinery. 3

[64] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *Proceedings of the 39th International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022. 2, 3

[65] Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1403–1412. IEEE, 2019. 2

[66] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020. 1, 3

[67] Yi Li, Yi-Zhe Song, Timothy M. Hospedales, and Shaogang Gong. Free-hand sketch synthesis with deformable stroke models. *CoRR*, abs/1510.02644, 2015. 1, 2

[68] Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. Im2pencil: Controllable pencil illustration from photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1525–1534, 2019. 2

[69] Hangyu Lin, Yanwei Fu, Yu-Gang Jiang, and X. Xue. Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6757–6766, 2020. 3

[70] Difan Liu, Matthew Fisher, Aaron Hertzmann, and Evangelos Kalogerakis. Neural strokes: Stylized line drawing of 3d shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14204–14213, 2021. 2

[71] Fang Liu, Xiaoming Deng, Yu-Kun Lai, Yong-Jin Liu, Cuixia Ma, and Hongan Wang. Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[72] Fang Liu, Xiaoming Deng, Yu-Kun Lai, Yong-Jin Liu, Cuixia Ma, and Hongan Wang. Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5839, 2019. 2

[73] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, pages 34892–34916. Curran Associates, Inc., 2023. 2, 3

[74] Bria Long, Judith Fan, Holly Huey, Zixian Chai, and Michael Frank. Parallel developmental changes in children's production and recognition of line drawings of visual concepts. *Nature Communications*, 15, 2024. 6

[75] Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Sasha Sax, and Aravind Rajeswaran. Openeqa: Embodied question answering in the era of foundation models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 4

[76] William P. McCarthy, Robert D. Hawkins, Haoliang Wang, Cameron Holdaway, and Judith E. Fan. Learning to communicate about shared procedural abstractions, 2021. 13

[77] William P. McCarthy, Justin Matejka, Karl D.D. Willis, Judith E. Fan, and Yewen Pu. Communicating design intent using drawing and text. In *Proceedings of the 16th Conference on Creativity & Cognition*, page 512–519, New York, NY, USA, 2024. Association for Computing Machinery. 3

[78] John FJ Mellor, Eunbyung Park, Yaroslav Ganin, Igor Babuschkin, Tejas Kulkarni, Dan Rosenbaum, Andy Ballard, Theophane Weber, Oriol Vinyals, and SM Eslami. Unsupervised doodling and painting with improved spiral. *arXiv preprint arXiv:1910.01007*, 2019. 3

[79] Daniela Mihai and Jonathon Hare. Learning to draw: Emergent communication through sketching. *Advances in Neural Information Processing Systems*, 34:7153–7166, 2021. 3

[80] Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain-of-thought prompting for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14431, 2024. 3

[81] Kushin Mukherjee, Holly Huey, Xuanchen Lu, Yael Vinker, Rio Aguina-Kang, Ariel Shamir, and Judith Fan. Seva: Leveraging sketches to evaluate alignment between human and machine visual abstraction. In *Advances in Neural Information Processing Systems*, 2023. 2

[82] Omar W Nasim. *Observing by hand: sketching the nebulae in the nineteenth century*. University of Chicago Press, 2019. 1

[83] Changhoon Oh, Jungwoo Song, Jinhan Choi, Seonghyeon Kim, Sungwoo Lee, and Bongwon Suh. I lead, you help but only with enough details: Understanding user experience of co-creation with artificial intelligence. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–13, New York, NY, USA, 2018. Association for Computing Machinery. 3

[84] OpenAI. Gpt-4 technical report, 2024. 2, 3, 4, 6

[85] Dustin Podell, Zion English, Kyle Lacey, A. Blattmann, Tim Dockhorn, Jonas Muller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *ArXiv*, abs/2307.01952, 2023. 2, 3

[86] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkang Li, and Yi-Zhe Song. Sketchlattice: Latticed representation for sketch manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 953–961, 2021. 2

[87] Shuwen Qiu, Sirui Xie, Lifeng Fan, Tao Gao, Song-Chun Zhu, and Yixin Zhu. Emergent graphical conventions in a visual communication game. *arXiv preprint arXiv:2111.14210*, 2021. 3

[88] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 1, 3, 6

[89] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), 2020. 3

[90] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2, 3

[91] Leo Sampaio Ferraz Ribeiro, Tu Bui, John P. Collomosse, and Moacir Antonelli Ponti. Sketchformer: Transformer-based representation for sketched structure. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14141–14150, 2020. 3

[92] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 1, 2, 3, 6

[93] Daniel Rose, Vaishnavi Himakunthala, Andy Ouyang, Ryan He, Alex Mei, Yujie Lu, Michael Saxon, Chinmay Sonar, Diba Mirza, and William Yang Wang. Visual chain of thought: bridging logical gaps with multimodal infillings. *arXiv preprint arXiv:2305.02317*, 2023. 3

[94] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 3

[95] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Trans. Graph.*, 35(4), 2016. 2

[96] Peter Schaldenbrand, James McCann, and Jean Oh. Frida: A collaborative robot painter with a differentiable, real2sim2real planning environment. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11712–11718, 2023. 3

[97] Peter Schaldenbrand, Gaurav Parmar, Jun-Yan Zhu, James McCann, and Jean Oh. Cofrida: Self-supervised fine-tuning for human-robot co-painting. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024. 3

[98] Donald A Schon and Vincent DeSanctis. The reflective practitioner: How professionals think in action, 1986. 2

[99] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022. 3

[100] Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. A multimodal automated interpretability agent. In *Forty-first International Conference on Machine Learning*, 2024. 3

[101] Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *arXiv preprint arXiv:2403.16999*, 2024. 3

[102] Pratyusha Sharma, Tamar Rott Shaham, Manel Baradad, Stephanie Fu, Adrian Rodriguez-Munoz, Shivam Duggal, Phillip Isola, and Antonio Torralba. A vision check-up for language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14410–14419, 2024. 3, 7

[103] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Learning to sketch with shortcut cycle consistency, 2018. 2

[104] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1504–1509, 2010. 13

[105] Guoyao Su, Yonggang Qi, Kaiyue Pang, Jie Yang, Yi-Zhe Song, and CVSSP SketchX. Sketchhealer: A graph-to-sequence network for recreating partial human sketches. In *BMVC*, page 5, 2020. 2

[106] Yanpeng Sun, Qiang Chen, Jian Wang, Jingdong Wang, and Zechao Li. Exploring effective factors for improving visual in-context learning. *arXiv preprint arXiv:2304.04748*, 2023. 3

[107] Ivan E. Sutherland. *Sketchpad—a man-machine graphical communication system*, page 391–408. Association for Computing Machinery, New York, NY, USA, 1998. 3

[108] Gemini Team. Gemini: A family of highly capable multimodal models, 2024. 2, 3

[109] Jakob Tholander and Martin Jonsson. Design ideation with ai - sketching, thinking and talking with generative machine learning models. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, page 1930–1940, New York, NY, USA, 2023. Association for Computing Machinery. 1

[110] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms, 2024. 4

[111] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023. 3

[112] Barbara Tversky. What do sketches say about thinking? *AAAI Spring Symp. Sketch Understanding Worksh.*, 2002. 2

[113] Barbara Tversky. Visualizing thought. In *Handbook of human centric visualization*, pages 3–40. Springer, 2013. 1

[114] Barbara Tversky, Masaki Suwa, Maneesh Agrawala, Julie Heiser, Chris Stolte, Pat Hanrahan, Doantam Phan, Jeff Klingner, Marie-Paule Daniel, Paul Lee, et al. Sketches for design and design of sketches. *Human Behaviour in Design: Individuals, Teams, Tools*, pages 79–86, 2003. 1

[115] Russell Tytler, Vaughan Prain, George Aranda, Joseph Ferguson, and Radhika Gorur. Drawing to reason and learn in science. *Journal of Research in Science Teaching*, 57(2): 209–231, 2020. 3

[116] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Trans. Graph.*, 41(4), 2022. 1, 3, 6

[117] Yael Vinker, Yuval Alaluf, Daniel Cohen-Or, and Ariel Shamir. Clipascene: Scene sketching with different types and levels of abstraction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4146–4156, 2023. 3, 6

[118] Jiawei Wang and Changjian Li. Contextseg: Sketch semantic segmentation by querying the context with attention. *arXiv preprint arXiv:2311.16682*, 2023. 6

[119] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. Red Hook, NY, USA, 2024. Curran Associates Inc. 2, 5

[120] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C. Olsen. Xdog: An extended difference-of-gaussians compendium including advanced image stylization. *Comput. Graph.*, 36:740–753, 2012. 2

[121] World Wide Web Consortium (W3C). *Scalable Vector Graphics (SVG)*, 1999. 3

[122] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023. 3

[123] Rong Wu, Wanchao Su, Kede Ma, and Jing Liao. Iconshop: Text-guided vector icon synthesis with autoregressive trans-

formers. *ACM Transactions on Graphics (TOG)*, 42:1 – 14, 2023. 3

[124] Chufeng Xiao, Wanchao Su, Jing Liao, Zhouhui Lian, Yi-Zhe Song, and Hongbo Fu. Differsketching: How differently do people sketch 3d objects? *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2022)*, 41 (4):1–16, 2022. 2

[125] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015. 2

[126] XiMing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. In *Advances in Neural Information Processing Systems*, pages 15869–15889. Curran Associates, Inc., 2023. 3, 6

[127] Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4546–4555, 2024. 3, 6, 7

[128] Peng Xu, Timothy M. Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey and a toolbox, 2020. 2

[129] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v(ision). *ArXiv*, abs/2309.17421, 2023. 2

[130] Zhengyuan Yang, Jianfeng Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. Idea2img: Iterative self-refinement with gpt-4v (ision) for automatic image design and generation. *arXiv preprint arXiv:2310.08541*, 2023. 3

[131] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. Apdrawinggan: Generating artistic portrait drawings from face photos with hierarchical gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10743–10752, 2019. 2

[132] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. Unpaired portrait drawing generation via asymmetric cycle mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8217–8225, 2020. 2

[133] C. Zhang, Weijie Wang, Paul Pangaro, Nikolas Martelaro, and Daragh Byrne. Generative image ai using design sketches as input: Opportunities and challenges. *Proceedings of the 15th Conference on Creativity and Cognition*, 2023. 1

[134] Jiahao Zhang, Bowen Wang, Liangzhi Li, Yuta Nakashima, and Hajime Nagahara. Instruct me more! random prompting for visual in-context learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2597–2606, 2024. 3

[135] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Text-to-vector generation with neural path representation. *ACM Trans. Graph.*, 43(4), 2024. 3

[136] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. What makes good examples for visual in-context learning? *Advances in Neural Information Processing Systems*, 36: 17773–17794, 2023. 3

[137] Zhengming Zhang, Xiaoming Deng, Jinyao Li, Yukun Lai, Cuixia Ma, Yongjin Liu, and Hongan Wang. Stroke-based semantic segmentation for scene-level free-hand sketches. *Vis. Comput.*, 39(12):6309–6321, 2022. 6

[138] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023. 3

[139] Yixiao Zheng, Kaiyue Pang, Ayan Das, Dongliang Chang, Yi-Zhe Song, and Zhanyu Ma. Creativeseg: Semantic segmentation of creative sketches. *IEEE Transactions on Image Processing*, 33:2266–2278, 2024. 6

[140] Tao Zhou, Chen Fang, Zhaowen Wang, Jimei Yang, Byungmoon Kim, Zhili Chen, Jonathan Brandt, and Demetri Terzopoulos. Learning to sketch with deep q networks and demonstrated strokes. *ArXiv*, abs/1810.05977, 2018. 2, 3

SketchAgent: Language-Driven Sequential Sketch Generation

Supplementary Material

Table of Contents

IRB Disclosure We received IRB approvals for all user studies, from all of the institutions involved. Accordingly, we took measures to ensure participant anonymity and refrained from showing them potentially offensive content.

A. Technical Details

We will publicly release the full source code, including our interactive platform. Our default backbone model is Claude3.5-Sonnet (version 20240620) [3]. We use the official API of Anthropic, with an average cost of $0.05 per sketch. We employ CairoSVG [59] for rendering the SVG onto the canvas. Our output sketches are also provided in SVG format to facilitate further editing if needed. SketchAgent generates a complete sketch in approximately 20 seconds, with individual strokes in collaborative mode taking about 8 seconds each. For the CLIP zero-shot classification, we use the clip-vit-large-patch14 model from Hugging Face [27]. Our canvas is defined as a 50×50 grid with numbers labeled on the bottom and left edges. Each cell corresponds to a patch of size 12×12 pixels, chosen to ensure a clear display of the grid numbers along the edges. This configuration results in a 612×612 pixels grid, with the drawing area confined to a 600×600 pixel range. All prompts used in our method are provided in Figs. 51, 52 and 55. The examples provided to the agent in the system and user prompts are visualized in Fig. 15 and Fig. 16 respectively.

We use Claude3.5-Sonnet in its default settings, which results in significant variability in results, given the highly diverse nature of LLMs. For example, in Fig. 17, we present 12 sketches produced by our method for the concept "rabbit", demonstrating high diversity in pose, structure, and quality. To generate variations in the experiments described in Section 5.1 of the main paper (where we applied our method 10 times per category), we use the default settings of Claude3.5-Sonnet. However, during controlled experimental conditions, we reduce variability by setting the temperature to 0 and top_k to 1, ensuring deterministic outputs. For general use, we recommend the stochastic version to encourage more varied and creative outputs.

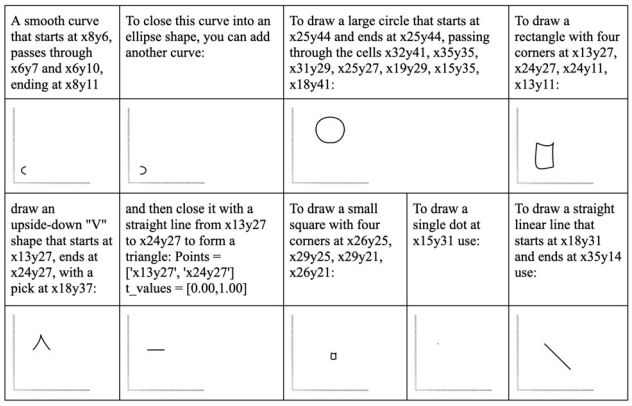


Figure 15. Visualization of single-stroke primitives used in the system prompt to introduce the grid and sketching language to the agent.

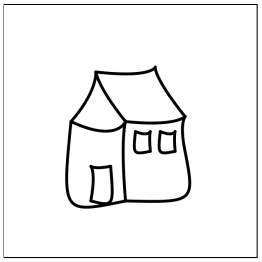


Figure 16. Visualization of the simple sketch of a house provided as an in-context example, represented with our sketching language through the user prompt.

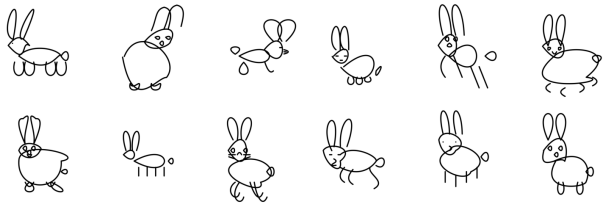


Figure 17. Sketch variability. Example of twelve different sketches produced for the concept "rabbit" by SketchAgent, with the same settings.

B. More Results and Analysis

As described in Section 5.1 of the main paper, SketchAgent is capable of generating sketches for a wide range of concepts that extend beyond standard categories. Here we provide additional results to support this claim. We define three unique categories that require general knowledge: Scientific Concepts, Diagrams, and Notable Landmarks, and utilize ChatGPT-4o to produce 10 ran-

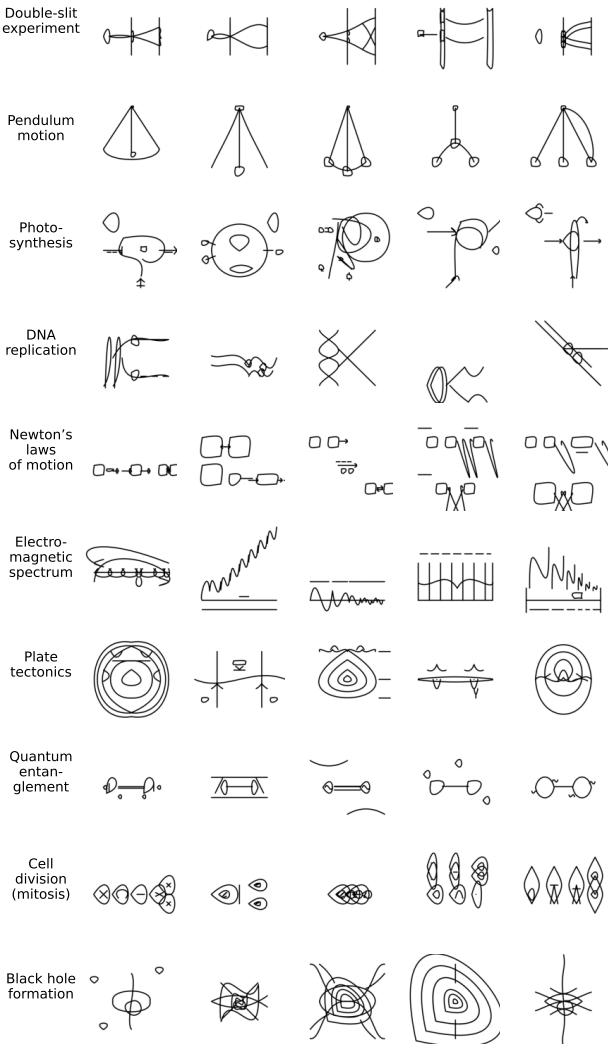


Figure 18. Randomly selected sketches of scientific concepts. Ten textual concepts were randomly selected using GPT-4o. Five sketches were generated per concept, showcasing the variability and diversity of the outputs.

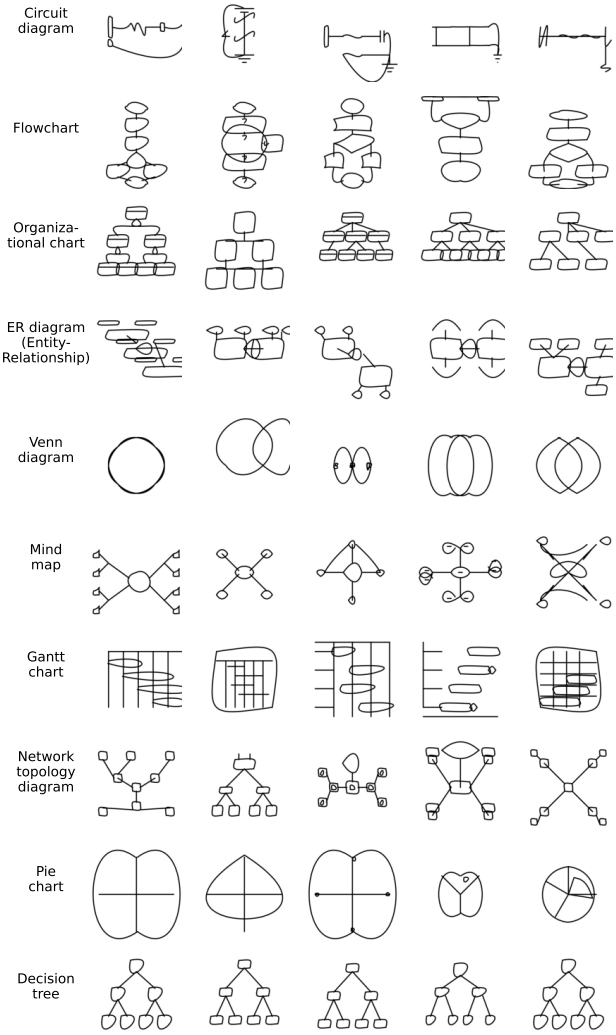


Figure 19. Randomly selected sketches of diagrams across fields. Ten textual concepts were randomly selected using GPT-4o. Five sketches were generated per concept, showcasing the variability and diversity of the outputs.

dom textual concepts for each category, resulting in the following random concepts:

- **Scientific Concepts:** *Double-slit experiment, Pendulum motion, Photosynthesis, DNA replication, Newton's laws of motion, Electromagnetic spectrum, Plate tectonics, Quantum entanglement, Cell division (mitosis), Black hole formation.*
- **Diagrams:** *Circuit diagram, Flowchart, Organizational chart, ER diagram (Entity-Relationship), Venn diagram, Mind map, Gantt chart, Network topology diagram, Pie chart, Decision tree.*
- **Notable Landmarks:** *Taj Mahal, Eiffel Tower, Great Wall of China, Pyramids of Giza, Statue of Liberty, Colosseum, Sydney Opera House, Big Ben, Mount Fuji, Machu Picchu.*

We generate five sketches for each concept (producing 50 sketches per category) by applying our method five times using its default settings. Figures 18 to 20 present the results for Scientific Concepts, Diagrams, and Notable Landmarks, respectively. The resulting sketches generally depict the concepts well, demonstrating diversity in the outputs. As can be seen, our method can generate a diverse set of different types and instances per concept (see *double-slit experiment, pendulum motion, Electromagnetic spectrum* in Fig. 18 and *Flowchart, Network typology diagram* in Fig. 19). Naturally, within each set, some concepts were depicted very successfully, while some outputs were less successful (e.g., Statue of Liberty, photosynthesis, pie chart).

B.1. Quantitative Text-Conditioned Analysis

In Section 5.1 of the main paper, we presented a quantitative analysis of text-conditioned sketch generation across 50 selected categories from the QuickDraw dataset [54]. Here, we provide addi-

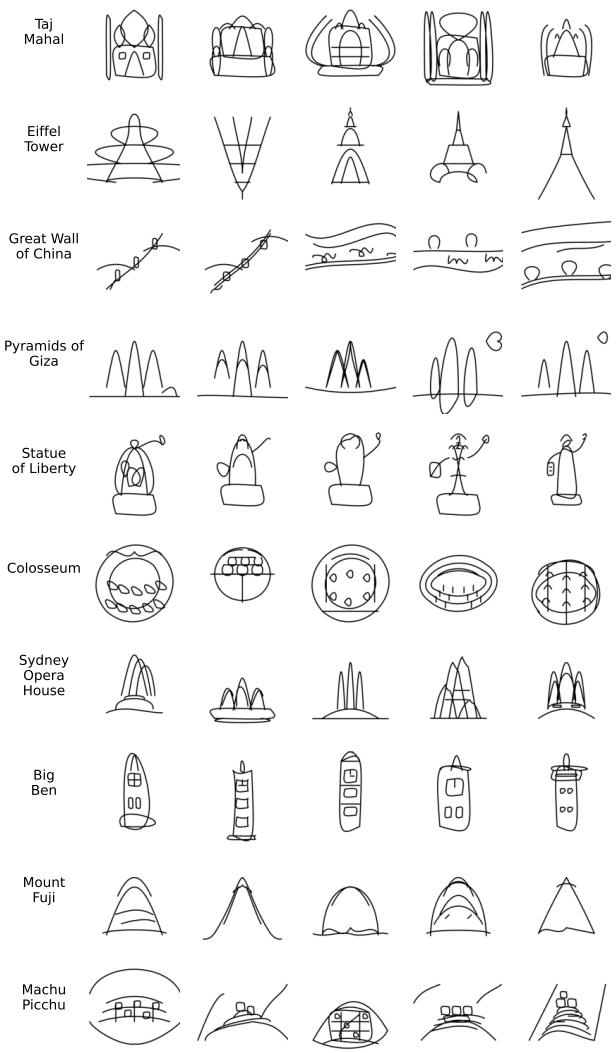


Figure 20. Randomly selected sketches of notable landmarks. Ten textual concepts were randomly selected using GPT-4o. Five sketches were generated per concept, showcasing the variability and diversity of the outputs.

tional details, visual examples, and further analysis of the experiment. We begin by providing further analysis of the CLIP classification rates of our default settings (Claude3.5-Sonnet) to explore recognition patterns. Figure 21 shows the confusion matrix (top 10 confused categories out of 50) for our set of 500 sketches. The most commonly confused classes are: "shark", which was often misclassified as a "fish", "octopus", which was frequently identified as a "spider"; and "snake", which was misclassified as a "squiggle". These confused classes often fall within highly related classes (such as a fish and a shark, or a school bus and a bus), suggesting that our method struggles with emphasizing distinctive features, likely due to its inherently abstract style. In Fig. 22, we visualize sketches from the six most confused classes with the correct class shown in green and the misclassified class shown in red. Figure 23 visualize the 10 top recognized classes.

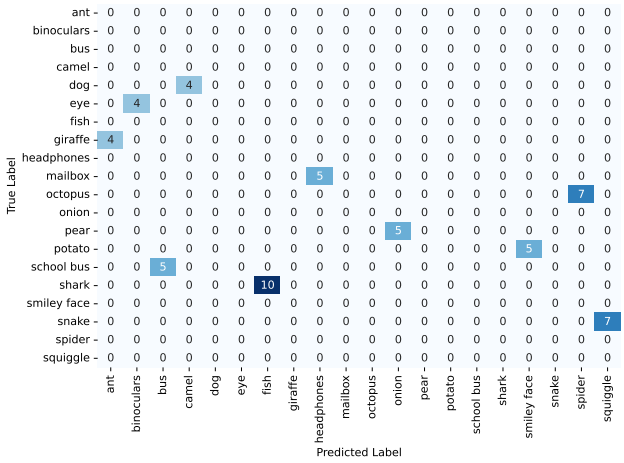


Figure 21. Confusion matrix (showing top 10 confused classes) for the set of 500 sketches generated with SketchAgent default settings (Claude3.5-Sonnet) across 50 categories

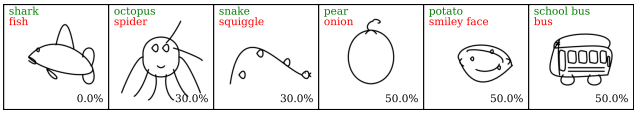


Figure 22. Visualization of sketches from the six most confused classes. The correct category is highlighted in green, while the misclassified category is highlighted in red.

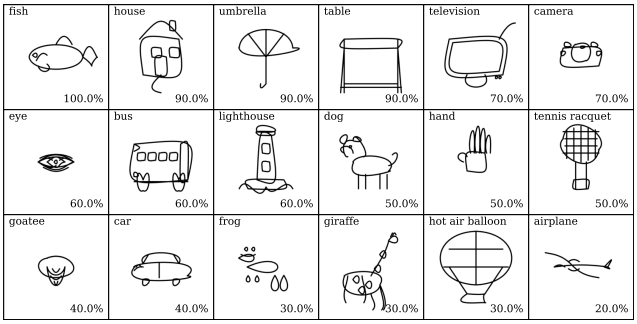


Figure 23. Visualization of the top recognized classes for the set of 500 sketches generated with our default settings (Claude3.5-Sonnet) across 50 categories.

The class "fish" was correctly identified across all seeds, followed by "house", "umbrella", and "table", which were correctly recognized in 90% of trials (9 out of 10). Recognition rates for other classes ranged from 70% to 20%. In Section 5.1 of the main paper, we compared the performance of different multimodal LLMs (GPT-4o-mini, GPT-4o, and Claude3-Opus) using our default prompts and settings. Figure 24 visualizes the eight most recognized classes across all backbone models. For this analysis, we select the top two recognized categories from each model and display the sketches with the highest classification probability for each. Note that some categories were at the top two of multiple models (such as house, fish, and eye), in that case, we

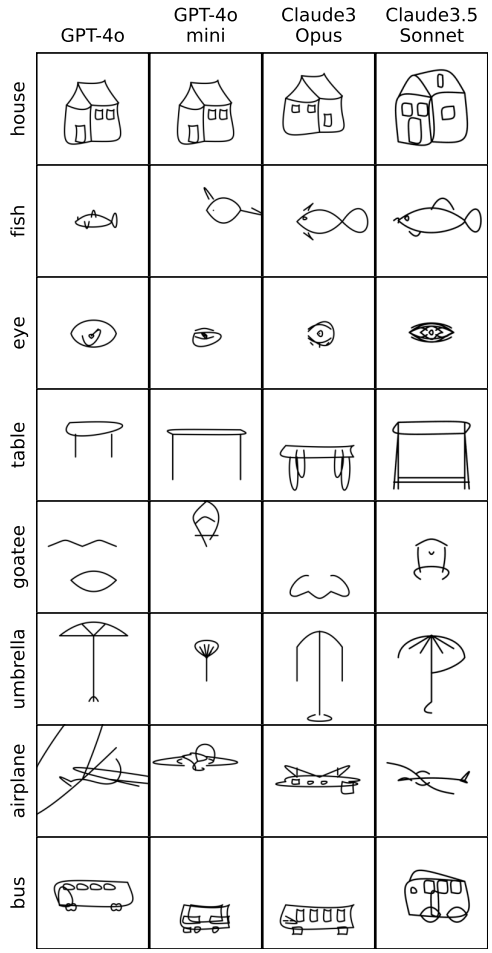


Figure 24. Visualization of sketches from the most recognized classes across all backbone models. The classes selected based on the two most recognizable classes in each model.

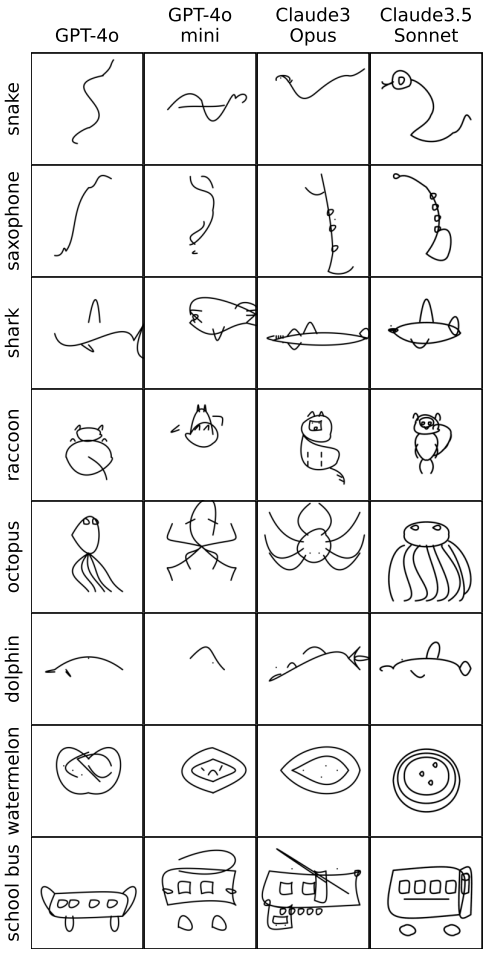


Figure 25. Visualization of sketches from the least recognized classes across all backbone models. The classes selected based on the two least recognizable classes in each model.

select the next top recognized category. The chosen top two categories for each model are: GPT-4o: house and eye, GPT-4o-mini: table and goatee, Claude3-Opus: fish and airplane, Claude3.5-Sonnet: umbrella and bus. Similarly, Figure 25 highlights the least recognized categories, chosen using the same selection criteria. The chosen worst two categories for each model are: GPT-4o: snake and school bus, GPT-4o-mini: saxophone and raccoon, Claude3-Opus: octopus and dolphin, Claude3.5-Sonnet: shark and watermelon. Note that snake, octopus, and shark, were all confused under at least three of the four backbones. The visualizations align well with the quantitative results presented in Table 1 of the main paper. Among the Anthropic models, Claude3.5-Sonnet produces better sketches than Claude3-Opus, and among the GPT models, GPT-4o outperforms GPT-4o-mini. Overall, the two best-performing backbone models are Claude3.5-Sonnet and GPT-4o. Interestingly, the sketching style differs between GPT-4o and Claude3.5-Sonnet. Although Claude3.5-Sonnet (our default backbone model) seems to yield the best results, this may be due to the fact that our method was primarily developed using this model. Consequently, the prompts we use were optimized for Claude3.5-Sonnet, and improved results for other models might be achievable with additional prompt engineering. We leave this exploration for future work.

SketchAgent using an open-source model While open-source models currently lag behind commercial closed-source models, they are rapidly advancing in size and capability, showing significant potential for facilitating sketch generation.

We begin by experimenting with Llama-3.2-11B-Vision [24], a multimodal large language model developed by Meta AI, as SketchAgent's backbone model. When used with our default prompts and framework, the model fails to generate meaningful sketches, frequently replicating the in-context example of a house provided in the user prompt (examples are shown in Fig. 26).

We therefore turn into exploring a larger available open-source model, Llama-3.1-405B-Instruct [24]. This model resulted in better sketches that manage to generalize well beyond the in-context example. We generated 500 random sketches and computed their classification rates using CLIP, as described in Section 5.1 of the

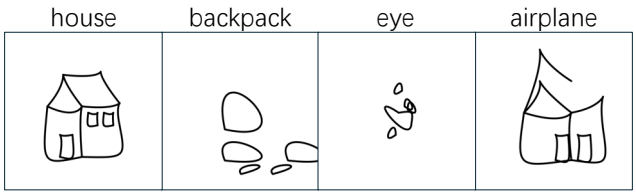


Figure 26. Sketches generated using Llama-3.2-11B-Vision as our backbone models. The model frequently replicates the in-context example of a house provided in the user prompt.

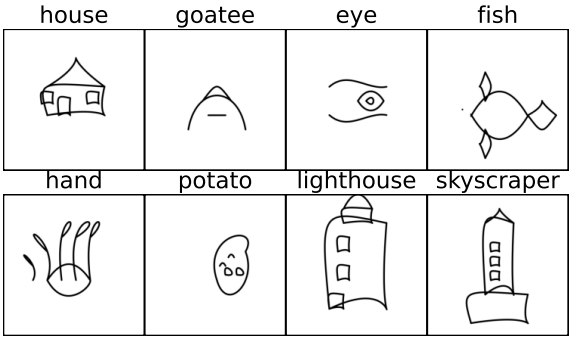


Figure 27. Visualization of the eight top recognized classes for the set of 500 sketches generated with Llama-3.1-405B-Instruct as our backbone model.

main paper. The results yielded lower scores compared to commercial models, with an average Top-1 recognition accuracy of 0.052 ± 0.03 and a Top-5 recognition accuracy of 0.1 ± 0.03. Visualizations of the top eight correctly classified classes are shown in Fig. 27, and the top eight most confused classes are presented in Fig. 28. Despite the lower recognition rates, the generated sketches are reasonable and visually coherent, showing promise as open-source models continue to improve. This experiment demonstrates the potential for SketchAgent to be implemented using publicly available models. While its performance does not match that of our default backbone, SketchAgent can still function effectively with open-source models, albeit with a slight compromise in performance.

Direct Prompting Analysis In Section 5.1 of the main paper, we compared our method to directly prompting Claude3.5-Sonnet for generating SVGs with a sketch-like appearance. In Fig. 29, we extend this analysis by visualizing the results of direct prompting with the other backbone models used in the quantitative experiment. This demonstrates how different models respond to direct SVG generation prompts. We present examples for the concepts "giraffe" and "lighthouse", using the following SVG generation prompt: *"Write an SVG string of a <concept>."*. For sketch-like SVGs, we used the same prompt as in the main paper (*"Write an SVG string that draws a sketch of a <concept>. Use only black and white colors"*). As shown, the outputs across all methods often feature uniform and precise geometric shapes (e.g., ellipses, triangles), which diverge from the natural variability and expressiveness characteristic of hand-drawn sketches. Interestingly, the SVGs generated by GPT-4o and Claude3.5-Sonnet appear more expressive and visually appealing compared to those produced by GPT-4o-mini and Claude3-Opus, aligning well with the performance differences observed in sketch generation.

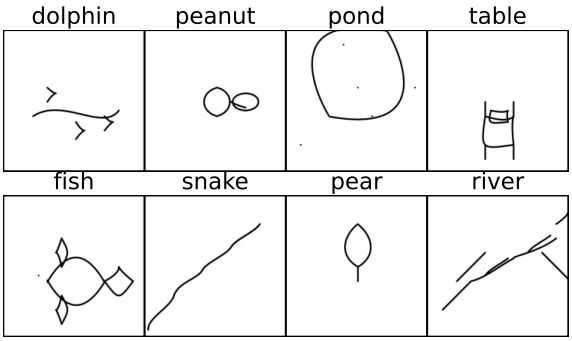


Figure 28. Visualization of sketches from the eight least recognized classes for the set of 500 sketches generated with Llama-3.1-405B-Instruct as our backbone model.

2AFC experiment In section 5.1 of the main paper, we also presented a 2AFC experiment to evaluate how "human-like" our agent's sketches appear compared to sketch-like SVGs generated with direct prompting and human sketches from the QuickDraw dataset. We utilize 50 sketches from 50 classes per method. We recruited a total of 150 workers through Amazon Mechanical Turk, each participating in 50 test sessions, as presented in Fig. 30. Before starting the test, workers were presented with instructions (Fig. 31). We filtered participants with a Mturk approval rate of 99.9% or higher and with a record of more than 1,000 surveys. Workers were paid $0.5 for completing the full test.

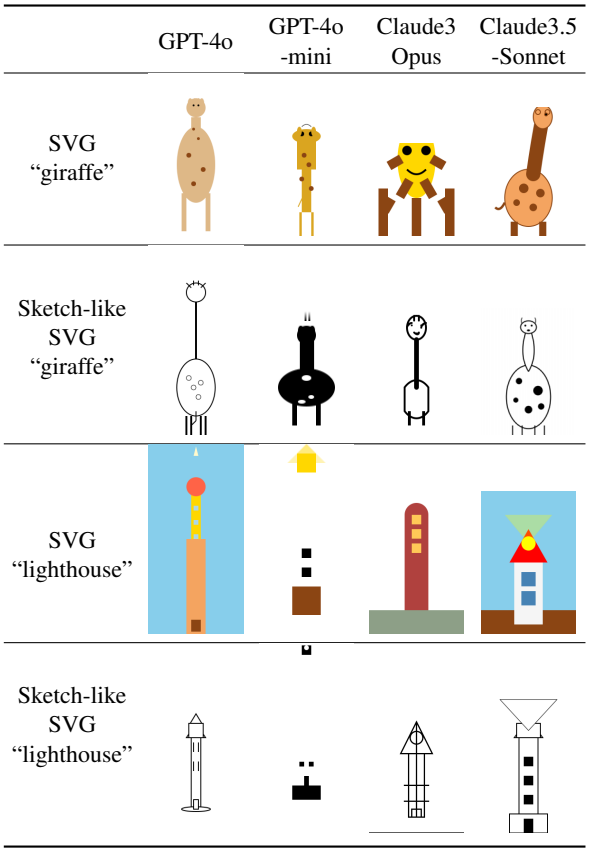


Figure 29. Direct prompting for SVG generation across different backbone models. The SVGs generated by GPT-4o and Claude3.5-Sonnet appear more expressive and visually appealing compared to those produced by GPT-4o-mini and Claude3-Opus, aligning well with the performance differences observed in sketch generation.

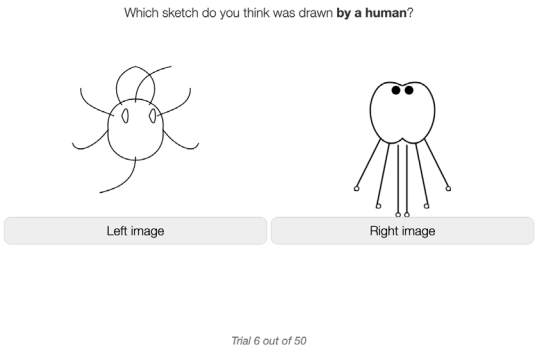


Figure 30. An example of our 2AFC session.

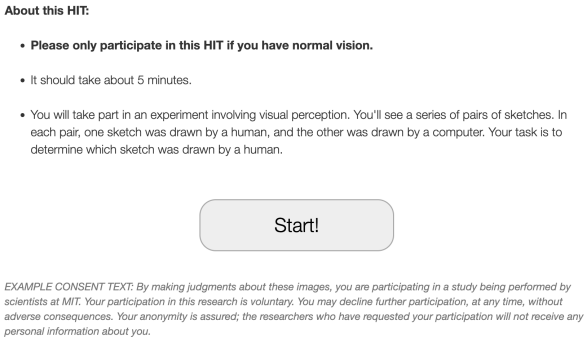


Figure 31. 2AFC instructions to users.

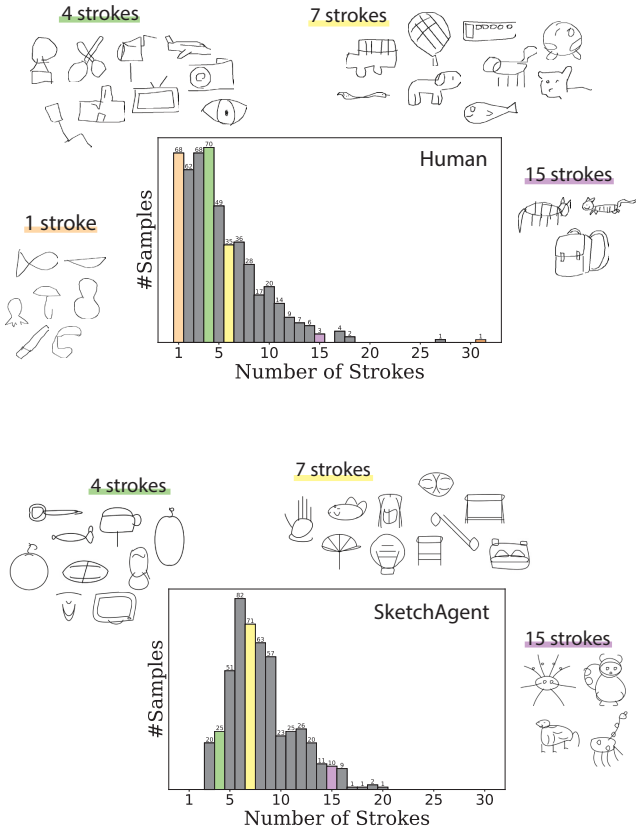


Figure 32. Distribution of human sketches [54] (top) and SketchAgent's sketches (bottom) based on the number of strokes per sketch. Representative examples are shown for sketches drawn with 1, 4, 7, and 15 strokes. Notably, in the QuickDraw dataset, single-stroke sketches often consist of a single long continuous line.

B.2. Sequential sketching

In Section 5.2 of the main paper, we analyze the sequential nature of our generated sketches. In Figs. 37 to 39, we present additional visualizations of annotated sequential sketches of 48 randomly selected animals, with the presented sketches also chosen randomly. As illustrated, due to the extensive prior knowledge of the backbone model, SketchAgent provides meaningful textual annotations for each stroke and sketches in a logical order. Typically, more significant body parts, such as the head and body, are drawn first. We next provide more details and visualizations of the quantitative analysis shown in Figure 11 of the main paper. Figure 32 displays histograms of the number of strokes in QuickDraw sketches (top) and our generated sketches (bottom), as shown in the main paper. Alongside these histograms, we include visualizations of sketches drawn with 1, 4, 7, and 15 strokes. Notably, in the QuickDraw dataset, single-stroke sketches often consist of a single long continuous line, making them recognizable after the first stroke. In contrast, sketches with a larger number of strokes rarely feature long continuous lines. For such cases, the sequential process of adding strokes gradually makes the sketches recognizable after several strokes. Figures 33 to 36 also demonstrates the sequential sketching process for both QuickDraw sketches and those generated by our method, providing a visual context for the trends observed in Figure 11.

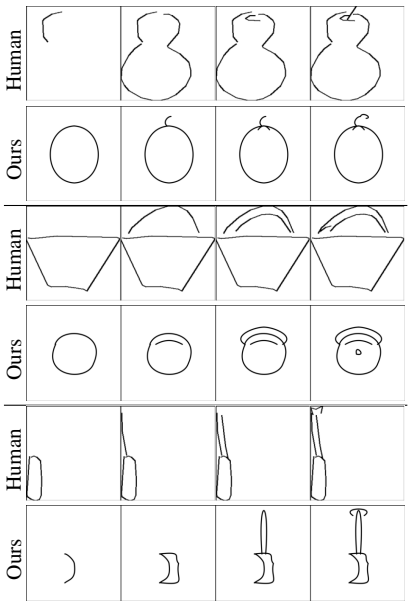


Figure 33. Sequential four-stroke sketches of a pear, purse, and screwdriver, created by humans [54] and by SketchAgent.

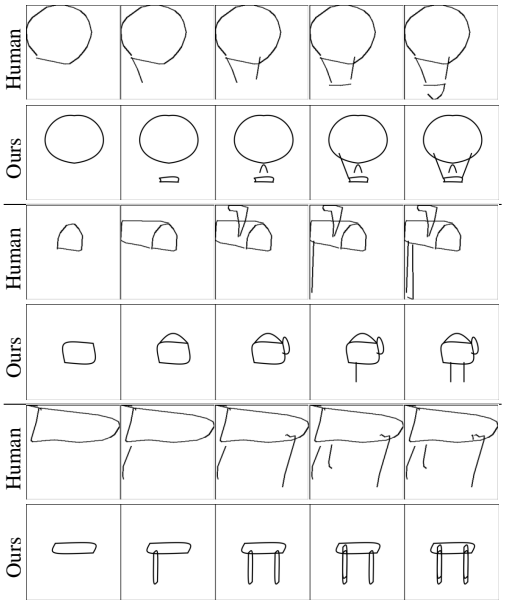


Figure 34. Sequential five-stroke sketches of a pear, purse, and screwdriver, created by humans [54] and by SketchAgent.

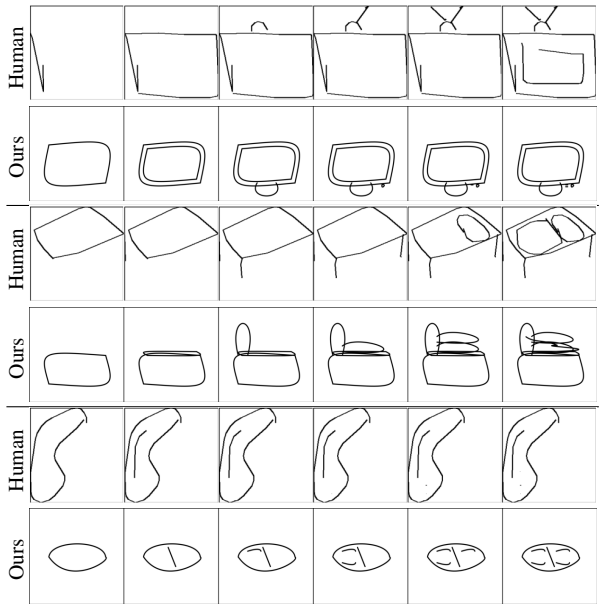


Figure 35. Sequential six-stroke sketches of a television, bed, and peanut, created by humans [54] and by SketchAgent.

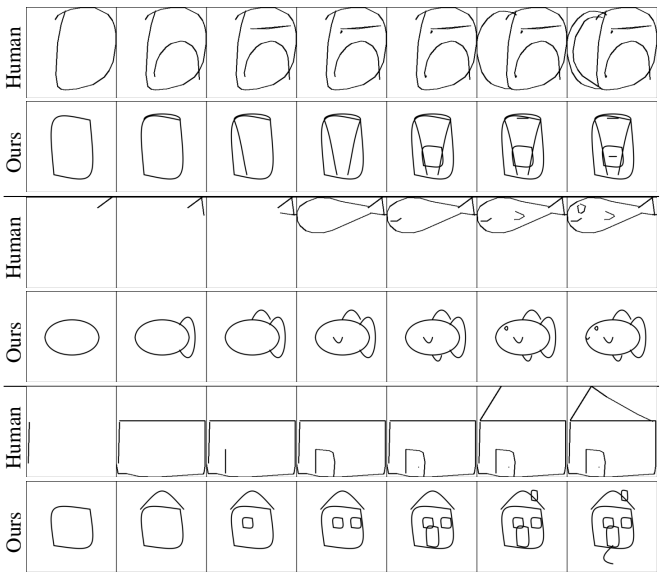


Figure 36. Sequential seven-stroke sketches of a backpack, fish, and house, created by humans [54] and by SketchAgent.

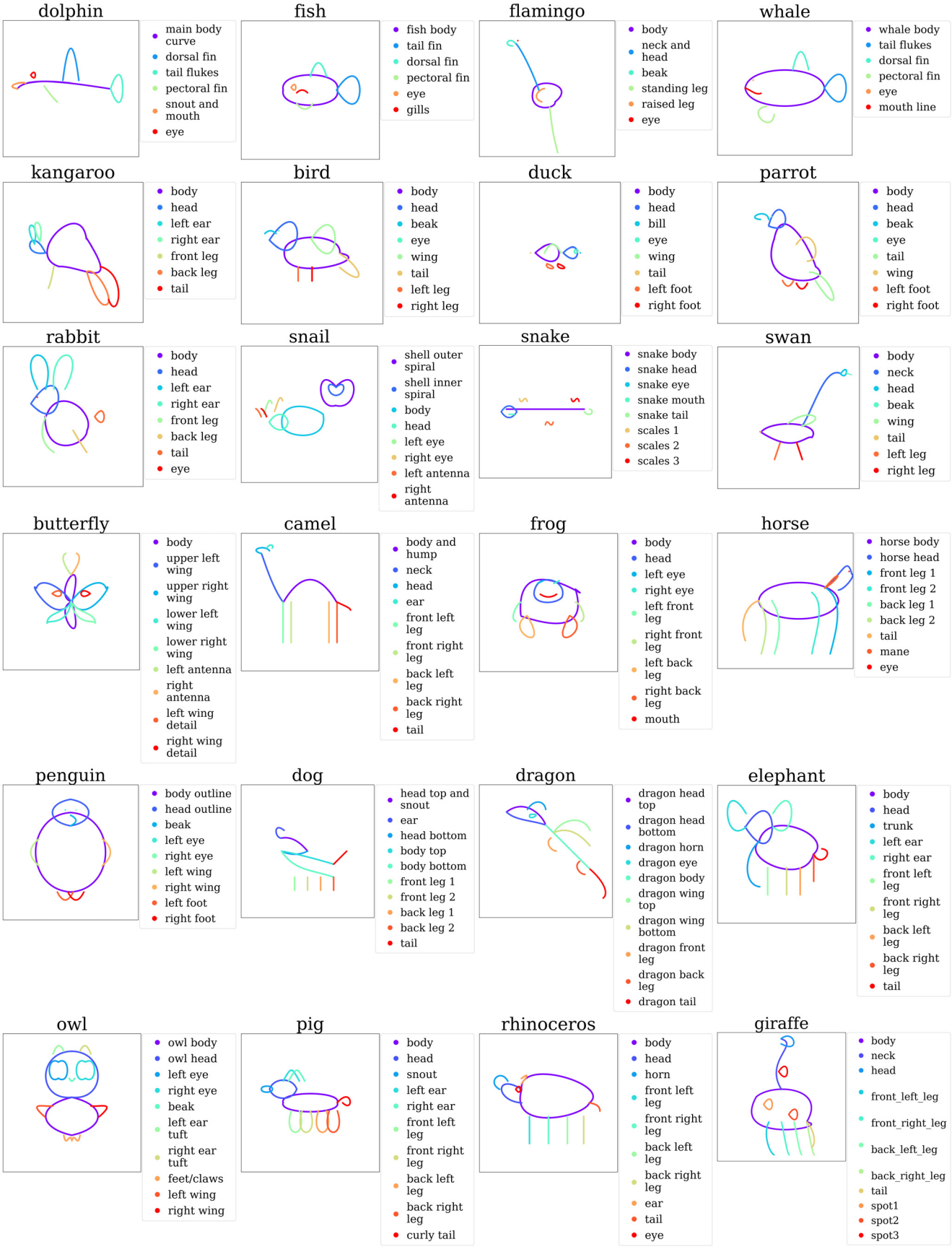

dolphin
main body curve
dorsal fin
tail flukes
pectoral fin
snout and mouth
eye
fish
fish body
tail fin
dorsal fin
pectoral fin
eye
gills
flamingo
body
neck and head
beak
standing leg
raised leg
eye
whale
whale body
tail flukes
dorsal fin
pectoral fin
eye
mouth line
kangaroo
body
head
left ear
right ear
front leg
back leg
tail
bird
body
head
beak
eye
wing
tail
left leg
right leg
duck
body
head
bill
eye
wing
tail
left foot
right foot
parrot
body
head
beak
eye
tail
wing
left foot
right foot
rabbit
body
head
left ear
right ear
front leg
back leg
tail
eye
snail
shell outer spiral
shell inner spiral
body
head
left eye
right eye
left antenna
right antenna
snake
snake body
snake head
snake eye
snake mouth
snake tail
scales 1
scales 2
scales 3
swan
body
neck
head
beak
wing
tail
left leg
right leg
butterfly
body
upper left wing
upper right wing
lower left wing
lower right wing
left antenna
right antenna
left wing detail
right wing detail
camel
body and hump
neck
head
ear
front left leg
front right leg
back left leg
back right leg
tail
frog
body
head
left eye
right eye
left front leg
right front leg
left back leg
right back leg
mouth
horse
horse body
horse head
front leg 1
front leg 2
back leg 1
back leg 2
tail
mane
eye
penguin
body outline
head outline
beak
left eye
right eye
left wing
right wing
left foot
right foot
dog
head top and snout
ear
head bottom
body top
body bottom
front leg 1
front leg 2
back leg 1
back leg 2
tail
dragon
dragon head top
dragon head bottom
dragon horn
dragon eye
dragon body
dragon wing top
dragon wing bottom
dragon front leg
dragon back leg
dragon tail
elephant
body
head
trunk
left ear
right ear
front left leg
front right leg
back left leg
back right leg
tail
owl
owl body
owl head
left eye
right eye
beak
left ear tuft
right ear tuft
feet/claws
left wing
right wing
pig
body
head
snout
left ear
right ear
front left leg
front right leg
back left leg
back right leg
curly tail
rhinoceros
body
head
horn
front left leg
front right leg
back left leg
back right leg
ear
tail
eye
giraffe
body
neck
head
front_left_leg
front_right_leg
back_left_leg
back_right_leg
tail
spot1
spot2
spot3

Figure 37

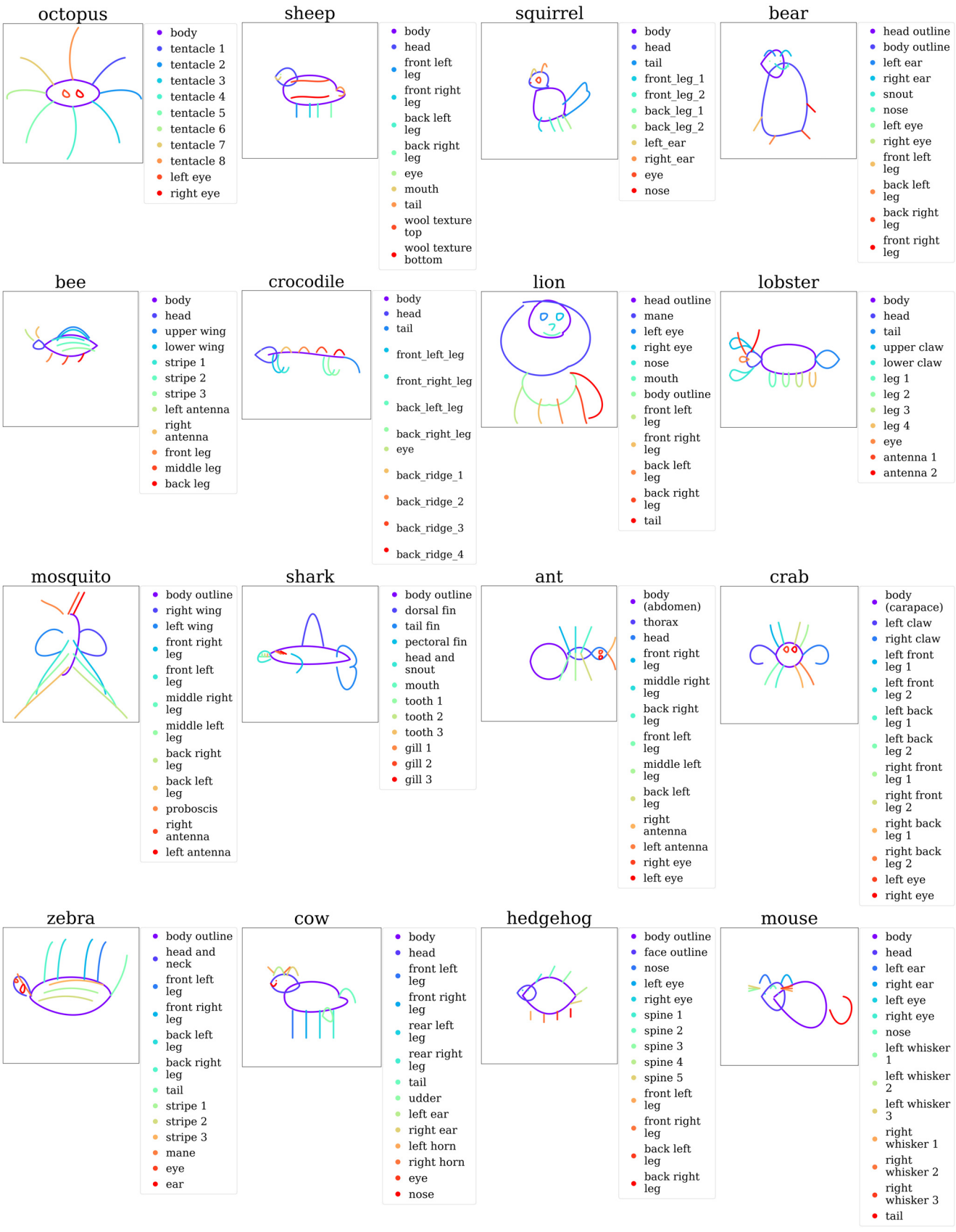

octopus
body
tentacle 1
tentacle 2
tentacle 3
tentacle 4
tentacle 5
tentacle 6
tentacle 7
tentacle 8
left eye
right eye
sheep
body
head
front left leg
front right leg
back left leg
back right leg
eye
mouth
tail
wool texture top
wool texture bottom
squirrel
body
head
tail
front_leg_1
front_leg_2
back_leg_1
back_leg_2
left_ear
right_ear
eye
nose
bear
head outline
body outline
left ear
right ear
snout
nose
left eye
right eye
front left leg
back left leg
back right leg
front right leg
bee
body
head
upper wing
lower wing
stripe 1
stripe 2
stripe 3
left antenna
right antenna
front leg
middle leg
back leg
crocodile
body
head
tail
front_left_leg
front_right_leg
back_left_leg
back_right_leg
eye
back_ridge_1
back_ridge_2
back_ridge_3
back_ridge_4
lion
head outline
mane
left eye
right eye
nose
mouth
body outline
front left leg
front right leg
back left leg
back right leg
tail
lobster
body
head
tail
upper claw
lower claw
leg 1
leg 2
leg 3
leg 4
eye
antenna 1
antenna 2
mosquito
body outline
right wing
left wing
front right leg
front left leg
middle right leg
middle left leg
back right leg
back left leg
proboscis
right antenna
left antenna
shark
body outline
dorsal fin
tail fin
pectoral fin
head and snout
mouth
tooth 1
tooth 2
tooth 3
gill 1
gill 2
gill 3
ant
body (abdomen)
thorax
head
front right leg
middle right leg
back right leg
front left leg
middle left leg
back left leg
right antenna
left antenna
right eye
left eye
crab
body (carapace)
left claw
right claw
left front leg 1
left front leg 2
left back leg 1
left back leg 2
right front leg 1
right front leg 2
right back leg 1
right back leg 2
left eye
right eye
zebra
body outline
head and neck
front left leg
front right leg
back left leg
back right leg
tail
stripe 1
stripe 2
stripe 3
mane
eye
ear
cow
body
head
front left leg
front right leg
rear left leg
rear right leg
tail
udder
left ear
right ear
left horn
right horn
eye
nose
hedgehog
body outline
face outline
nose
left eye
right eye
spine 1
spine 2
spine 3
spine 4
spine 5
front left leg
front right leg
back left leg
back right leg
mouse
body
head
left ear
right ear
left eye
right eye
nose
left whisker 1
left whisker 2
left whisker 3
right whisker 1
right whisker 2
right whisker 3
tail

Figure 38

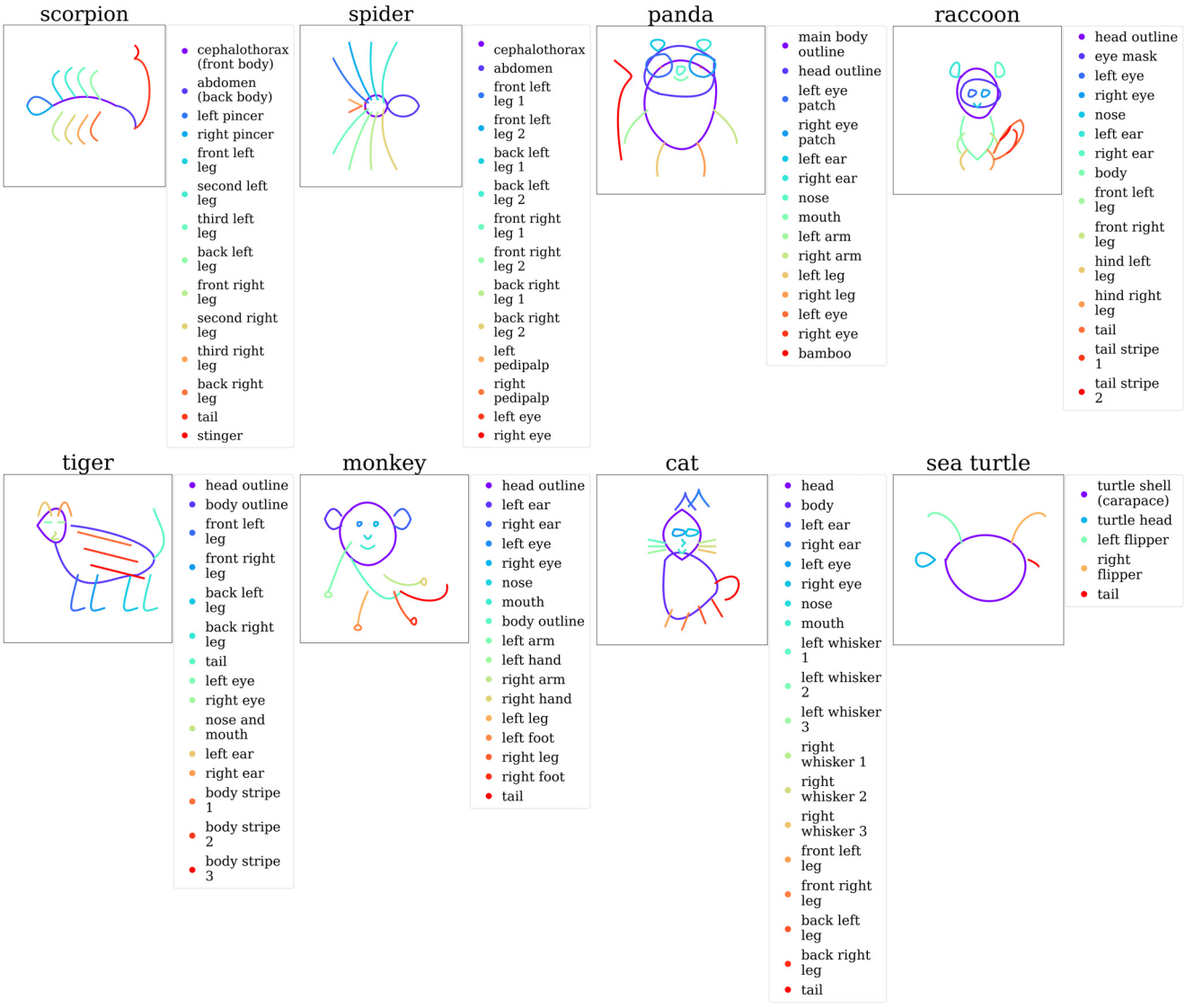

scorpion
cephalothorax (front body)
abdomen (back body)
left pincer
right pincer
front left leg
second left leg
third left leg
back left leg
front right leg
second right leg
third right leg
back right leg
tail
stinger
spider
cephalothorax
abdomen
front left leg 1
front left leg 2
back left leg 1
back left leg 2
front right leg 1
front right leg 2
back right leg 1
back right leg 2
left pedipalp
right pedipalp
left eye
right eye
panda
main body outline
head outline
left eye patch
right eye patch
left ear
right ear
nose
mouth
left arm
right arm
left leg
right leg
left eye
right eye
bamboo
raccoon
head outline
eye mask
left eye
right eye
nose
left ear
right ear
body
front left leg
front right leg
hind left leg
hind right leg
tail
tail stripe 1
tail stripe 2
tiger
head outline
body outline
front left leg
front right leg
back left leg
back right leg
tail
left eye
right eye
nose and mouth
left ear
right ear
body stripe 1
body stripe 2
body stripe 3
monkey
head outline
left ear
right ear
left eye
right eye
nose
mouth
body outline
left arm
left hand
right arm
right hand
left leg
left foot
right leg
right foot
tail
cat
head
body
left ear
right ear
left eye
right eye
nose
mouth
left whisker 1
left whisker 2
left whisker 3
right whisker 1
right whisker 2
right whisker 3
front left leg
front right leg
back left leg
back right leg
tail
sea turtle
turtle shell (carapace)
turtle head
left flipper
right flipper
tail

Figure 39

B.3. Human-Agent Collaborative Sketching

In Section 5.3 of the main paper, we demonstrate that humans and SketchAgent can effectively collaborate to produce meaningful sketches through genuine interaction. The sketching interface (Fig. 40) consists of a 400×400 plain canvas shared between the user and the agent. It highlights the current concept to be sketched and displays the active sketching mode, which can be either *solo* or *collab*. Additionally, the interface includes a submit button that allows users to finalize the sketch when they consider it complete. In *solo* mode, users independently sketch the given concept using green strokes. In *collab* mode, users and SketchAgent take turns adding strokes, with user strokes displayed in green and agent strokes in pink. At the beginning of each session, users are provided with general instructions about the experiment and the types of sketches they will be asked to draw (Fig. 41). Specifically, they are instructed to create recognizable sketches, stroke by stroke, while minimizing the number of strokes by planning ahead. Next, users begin by sketching two warm-up concepts in both "solo" and "collab" modes to familiarize themselves with the web environment. Each session includes all eight primary concepts in a randomized order, resulting in a total of 10 sketches per user (including the two warm-up sketches). The concepts are as follows:

- **Warm up concepts**: *jellyfish, house*
- **Text concepts**: *butterfly, fish, rabbit, duck, sailboat, coffee mug, eyeglasses, car*

For each concept, participants sketched in both solo and collaboration modes, with the order of these modes randomized to mitigate potential biases. The 30 users are counterbalanced: 15 users produced the first stroke in collaboration with the agent (and all odd-numbered strokes thereafter), while the other 15 users produced the second stroke in collaboration with the agent (and all even-numbered strokes). In total we collected responses from 32 users, however, two users were excluded from the analysis due to incomplete sketching sessions, leaving a total of 30 users. In Fig. 46 we present examples of sketches from each mode, focusing on those with high recognition rates across categories. Solo sketches are shown in green, agent-only sketches in pink, and collaborative sketches are depicted with a combination of both colors. To analyze "collab" and "solo" sketches, we rendered all complete and partial sketches (agent-only and user-only strokes) from SVG to pixel images. We then utilized a CLIP zero-shot classifier, as described in the main paper, to evaluate how effectively each sketch represented the intended concept. Tab. 3 summarizes the results (as shown in the graph in Fig. 12B of the main paper). These results highlight that both users and the agent contributed meaningfully to the final "collab" sketches. Variants of collaborative sketches containing only the user's strokes or only the agent's strokes were found to contain substantially less semantic information about the intended concept compared to the complete collaborative sketches. Additionally, the average number of strokes per completed sketch was consistent across modes, indicating similar levels of complexity. Specifically, the average stroke counts were as follows: collaborative full sketches: 7.333; solo agent sketches: 7.321; solo user sketches: 7.708. This suggests that collaboration produces sketches with a level of detail comparable to those created independently.

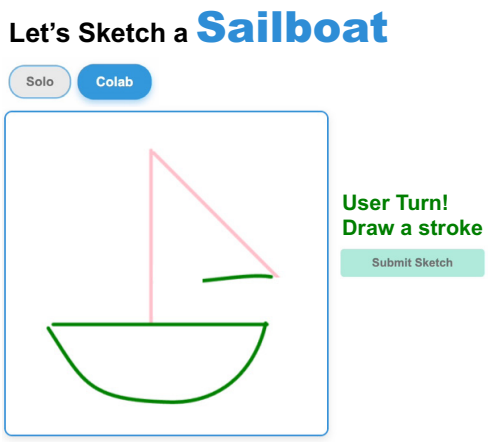


Figure 40. Screenshot of our web interface.

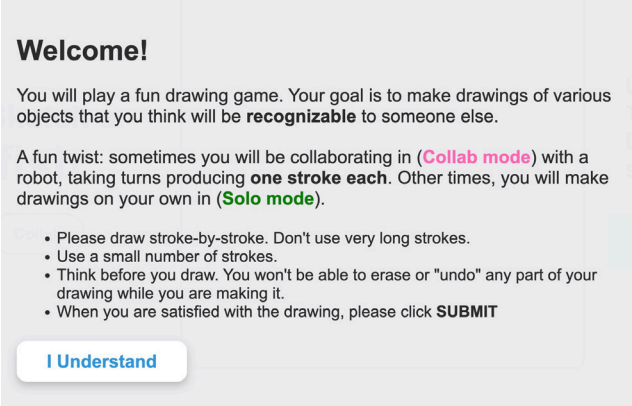


Figure 41. User instructions in the sketching interface.

We analyze the classification confusion patterns for collaborative and solo sketches (240 sketches each) in Figs. 43 and 44, revealing similar trends. For instance, a "coffee cup" was often misclassified as a "teapot", a "car" was frequently identified as a "turtle", and a "duck" was misclassified as a "bird". Additionally, "car" sketches were sometimes mistaken for a "pickup truck". In most cases, the misclassifications occur within closely related categories (e.g., "car" to "truck" or "pickup truck") or among categories sharing similar visual structures (e.g., the rounded dome and four base components of a "car" resembling a "turtle"). This highlights a challenge in emphasizing distinctive features within specific categories, likely stemming from the inherently abstract nature of our sketches.

Figure 42 presents the recognition rates with 95% confidence interval (CI) error bars for each concept across all three sketching conditions: "collab" (blue), "solo-user" (green), and "solo-agent" (pink). Overall, the recognition rates for collaborative sketches are comparable to those produced by users alone or the agent alone for each unique category. Notably, sketches of "car" exhibit the lowest recognition rate across all conditions. This is likely due to confusion with semantically similar categories, such as "truck", "pickup truck", "airplane", and "speedboat", as indicated by the

Variation	Recognition Rate	95% CI
Collab full sketch	0.75	[0.61, 0.85]
Collab agent-only strokes	0.10	[0.06, 0.19]
Collab user-only strokes	0.13	[0.07, 0.23]

Table 3. Recognition rate and 95% CI across collaborative full and partial sketches. In collaborative sketches, keeping agent-only strokes or user-only strokes significantly reduces recognizability.

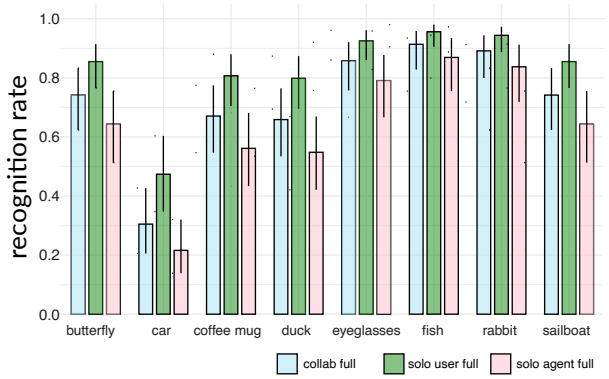


Figure 42. CLIP recognition rate by class for collaborative, solo user, and solo agent full sketches.

confusion matrices. Similarly, as discussed earlier, "coffee cup" and "duck" are frequently misclassified as related categories with overlapping visual features.

We observe that in some cases of collaborative sketching (14 out of 240 sketches), the agent-human pair faces challenges in interpreting each other's intentions and the meanings of strokes. Achieving effective collaboration and communication between different parties [45] is a challenge that often requires prior planning, social reasoning, and repeated interactions to establish shared intentions and representations. These complex processes continue to be studied across various contexts, including in interactions between humans [50, 58, 76], between humans and agents [15], and between agents [104]. Fig. 45 highlights the few collaborative sketches where the CLIP classification is correct, but the agent and user appear to lack a shared understanding of different stroke groups, resulting in the conflicting creation of duplicate concept components (i.e. two heads).

B.4. Chat-Based Editing

In section 5.4 of the main paper we demonstrate chat-based editing using SketchAgent. Below, we provide more details about the implementation of the experiment we performed. To enable chat editing, we use the following prompt: *"<editing instruction>. Describe the location of the added concepts first in <thinking> tags. Only provide the added strokes. Respond in the same format as before. Be concise."*, where <editing instruction> contains the desired edit such as *"Add glasses to the given cat"*. The chosen objects per category, as well as the editing prompts, are provided:

- **Animals:** *fish, bird, cat*. Editing instruction: *"Add glasses", "Add a hat", "Add a skirt"*.

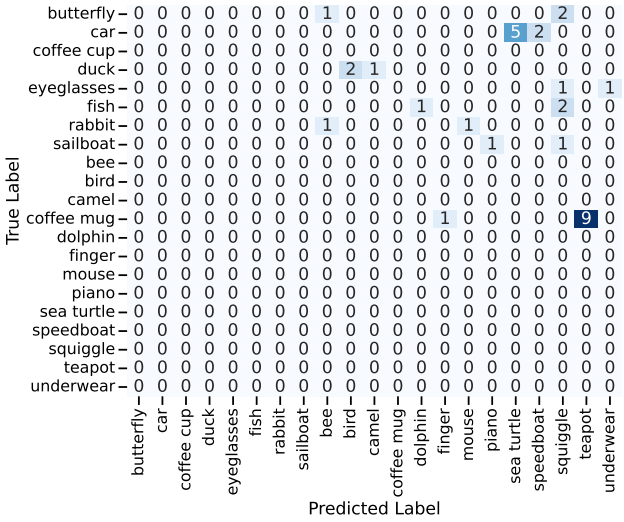


Figure 43. Confusion matrix from CLIP classification with categories from the QuickDraw dataset for 240 collaborative sketches across 8 categories.

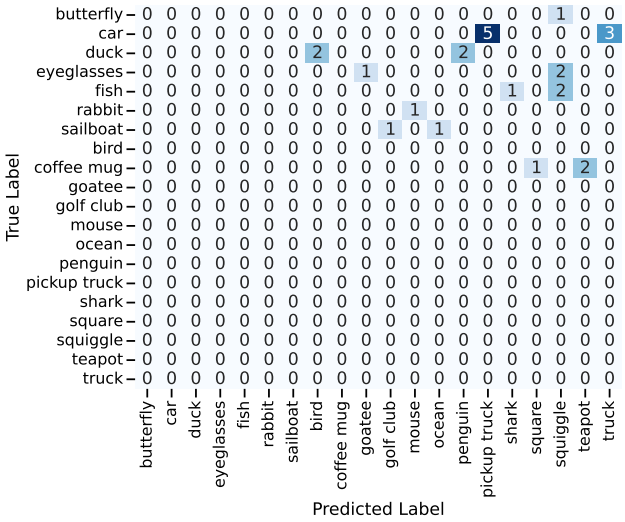


Figure 44. Confusion matrix from CLIP classification with categories from the QuickDraw dataset for 240 solo user sketches across 8 categories.

- **Outdoor:** *bus, building, boat*. Editing instruction: *"Add a tree to the left of the <concept>", "Add a sun on the top right, above the <concept>", "Add another smaller <concept> to the right of this <concept>"*.
- **Indoor:** *shelf, nightstand, table*. Editing instruction: *"Add a coffee mug on the top of the <concept>", "Add a lamp on the top of the <concept>", "Add an indoor plant to the left of the <concept>"*.

The resulting edited sketches are shown in Figure 47.

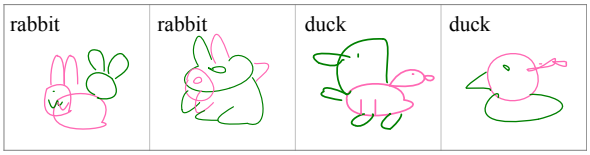


Figure 45. Examples of sketches created in "collab" mode that were correctly classified by CLIP but considered unsuccessful as collaborations due to conflicting agent-user interpretations of sub-components.

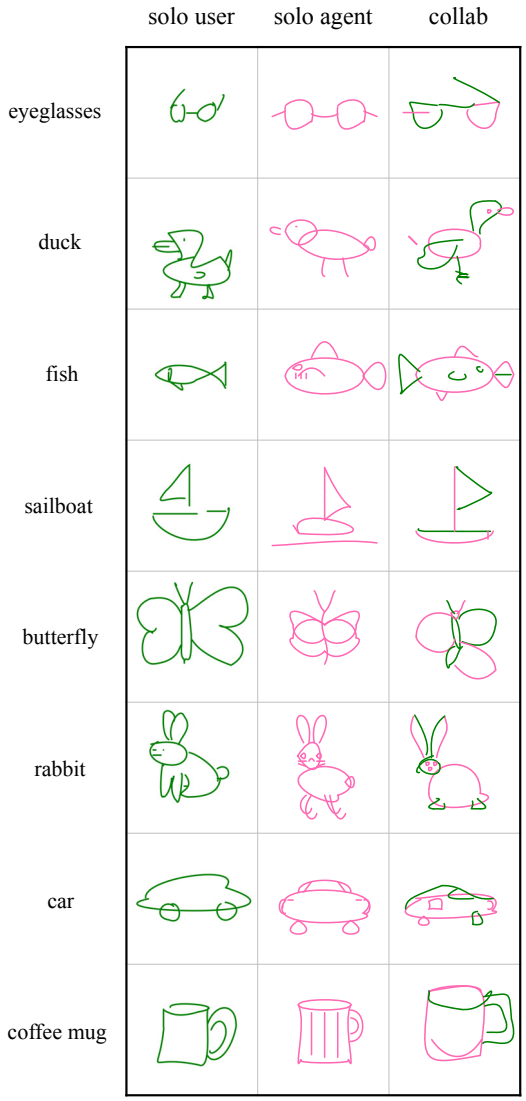


Figure 46. Examples of sketches from our collaborative human study that received high recognition rates. From left to right are sketches drawn in "solo" mode by users, "solo" mode by the agent, and collaboratively by both.

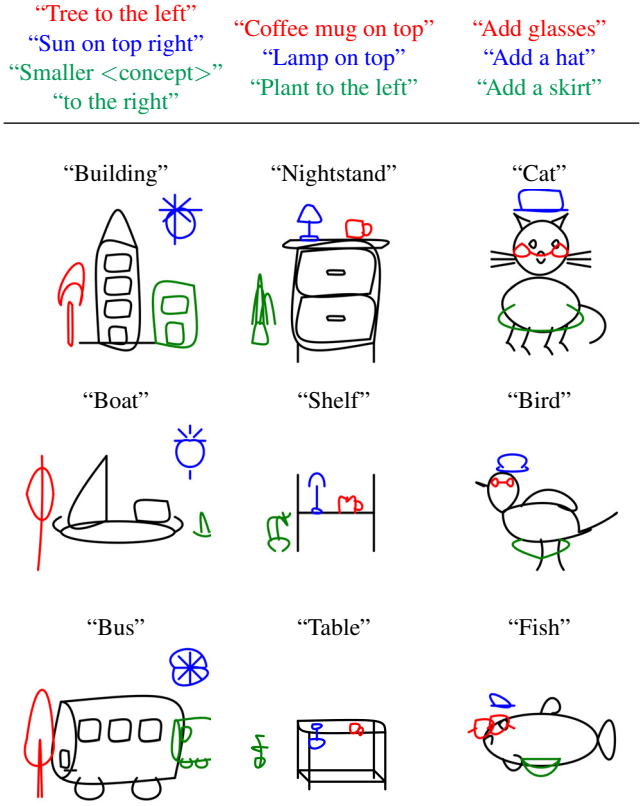


Figure 47. Chat-based sketch editing. We iteratively prompt SketchAgent to add objects to sketches through chat dialogues.

C. Ablation Study

In Section 6 of the main paper, we presented an ablation study by systematically removing key components of our method and computing the resulting classification rates. Here, we provide further analyses and discussions on the ablation study.

Table 2 in the main paper shows the CLIP classification rates for 500 sketches (across 50 categories) per experiment. In Fig. 48 we include a visualization of six sketches from six different concepts, covering both structures and animals. As shown, incorporating chain-of-thought reasoning and our in-context example of a house significantly enhances the quality of the results.

We find that the examples used in in-context learning (ICL) can influence both the quality and appearance of the generated sketches, suggesting an interesting direction for future research. Here, we analyze the impact of different types of in-context examples. To investigate whether the theme of the in-context example affects the output (e.g., whether using a house example aids in sketching related concepts like a hospital or if using a cat example helps with sketching other animals), we constructed an alternative sketch of a cat. This sketch used the same number of strokes as the house example to isolate the effect of the theme from complexity. We then applied our method using this alternative example in ICL. In Fig. 49 we illustrate the influence of different ICL examples on related concepts. The example used in each experiment is shown

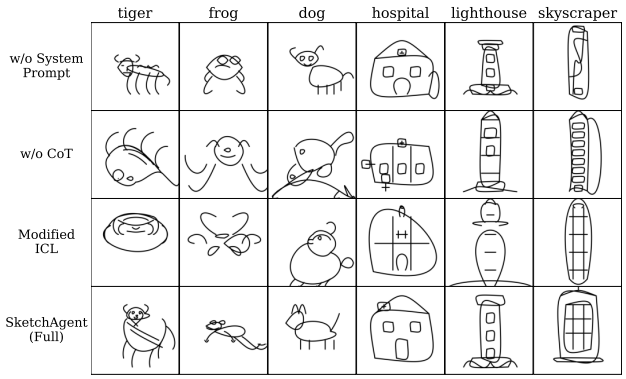


Figure 48. Visualization of sketches produced in different cases of our ablation study.

on the left, with the top figure presenting the effect on animal concepts and the bottom figure depicting the effect on structures. The results indicate that animal sketches are generally more influenced by an animal-based in-context example. For instance, the eyes in the generated sketches tend to resemble the eyes of the cat example more closely, while they vary more when a house example is used. However, there is no definitive conclusion regarding the overall quality or recognizability of these results. Conversely, for structures (bottom), the use of the cat example seems to result in smoother and more rounded shapes, while sketches generated using the house example generally appear more refined and cohesive.

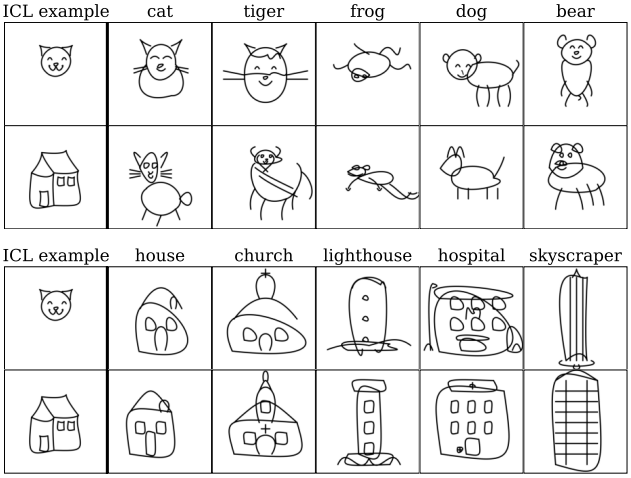


Figure 49. ICL example ablation study. We examine the impact of changing the concept in the ICL example (e.g., from a house to a cat) on the generation of related concepts. The example used in each experiment is shown on the left, with the top figure illustrating the effect on animal concepts and the bottom figure showing the effect on structural concepts.

We also examine the impact of example complexity, specifically how using a more detailed sketch with additional strokes affects the output. To test this, we enhanced the cat example by adding more details and then applied our method with the new, more complex example. The results are presented in Fig. 50. As shown, when a more detailed example is used, the generated sketches tend to overfit, closely replicating the original cat sketch. In contrast, using a simpler example leads to greater variation in the output.

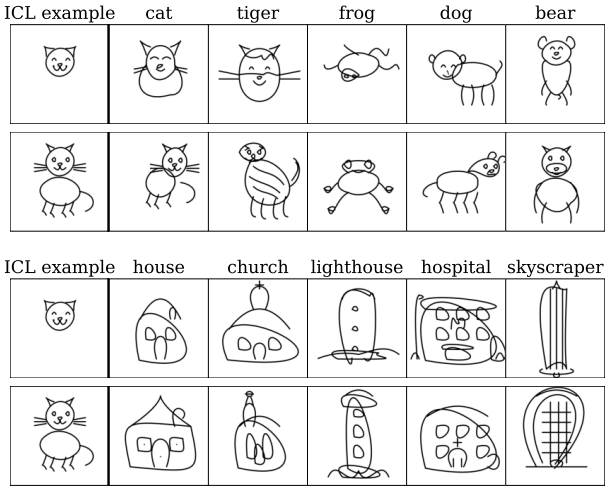


Figure 50. ICL example ablation study. We examine the impact of varying the complexity of the sketch presented in the ICL example while keeping the semantic concept (a cat) constant. The example used in each experiment is shown on the left, with the top figure illustrating the effect on animal concepts and the bottom figure showing the effect on structural concepts.

D. Prompts and More Results

We present the full prompts used in our system, as well as our randomly generated sketches used for the quantitative evaluation presented in Section 5.1 of the main paper, and the full set of sketches made by users and in collaborative mode from our human study.

You are an expert artist specializing in drawing sketches that are visually appealing, expressive, and professional.
You will be provided with a blank grid. Your task is to specify where to place strokes on the grid to create a visually appealing sketch of the given textual concept. The grid uses numbers (1 to res) along the bottom (x axis) and numbers (1 to res) along the left edge (y axis) to reference specific locations within the grid. Each cell is uniquely identified by a combination of the corresponding x axis numbers and y axis number (e.g., the bottom-left cell is 'x1y1', the cell to its right is 'x2y1'). You can draw on this grid by specifying where to draw strokes. You can draw multiple strokes to depict the whole object, where different strokes compose different parts of the object.
To draw a stroke on the grid, you need to specify the following:
Starting Point: Specify the starting point by giving the grid location (e.g., 'x1y1' for column 1, row 1).
Ending Point: Specify the ending point in the same way (e.g., 'xresyres' for column res, row res).
Intermediate Points: Specify at least two intermediate points that the stroke should pass through. List these in the order the stroke should follow, using the same grid location format (e.g., 'x6y5', 'x13y10' for points at column 6 row 5 and column 13 row 10).
Parameter Values (t): For each point (including the start and end points), specify a t value between 0 and 1 that defines the position along the stroke's path. t=0 for the starting point. t=1 for the ending point.
Intermediate points should have t values between 0 and 1 (e.g., "0.3 for x6y5, 0.7 for x13y10").
Examples:
To draw a smooth curve that starts at x8y6, passes through x6y7 and x6y10, ending at x8y11:
Points = ['x8y6', 'x6y7', 'x6y10', 'x8y11'] t_values = [0.00,0.30,0.80,1.00]
To close this curve into an ellipse shape, you can add another curve:
Points = ['x8y11', 'x11y10', 'x11y7', 'x8y6'] t_values = [0.00,0.30,0.70,1.00]
To draw a large circle that starts at x25y44 and ends at x25y44, passing through the cells x32y41, x35y35, x31y29, x25y27, x19y29, x15y35, x18y41: Points = ['x25y44', 'x32y41', 'x35y35', 'x31y29', 'x25y27', 'x19y29', 'x15y35', 'x18y41', 'x25y44'] t_values = [0.00, 0.125, 0.25, 0.375, 0.50, 0.625, 0.75, 0.875, 1.00]
To draw non-smooth shapes (with corners) like triangles or rectangles, you need to specify the corner points twice with adjacent corresponding t values. For example, to draw an upside-down "V" shape that starts at x13y27, ends at x24y27, with a pick (corner) at x18y37: Points = ['x13y27', 'x18y37','x18y37', 'x24y27'] t_values = [0.00,0.55,0.5,1.00]
To draw a triangle with corners at x10y29, x15y33, and x9y35, start with drawing a "V" shape that starts at x10y29, ends at x9y35, with a pick (corner) at x15y33:
Points = ['x10y29', 'x15y33', 'x15y33', 'x9y35'] t_values = [0.00,0.55,0.5,1.00]
and then close it with a straight line from x13y27 to x24y27 to form a triangle:
Points = ['x13y27', 'x24y27'] t_values = [0.00,1.00]
Note that for a triangle, the start and end points should be different from each other.
To draw a rectangle with four corners at x13y27, x24y27, x24y11, x13y11:
Points = ['x13y27', 'x24y27', 'x24y27', 'x24y11', 'x24y11', 'x13y11', 'x13y11', 'x13y27'] t_values = [0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00]
To draw a small square with four corners at x26y25, x29y25, x29y21, x26y21:
Points = ['x26y25', 'x29y25', 'x29y25', 'x29y21', 'x29y21', 'x26y21', 'x26y21', 'x26y25'] t_values = [0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00]
To draw a single dot at x15y31 use: Points = ['x15y31'] t_values = [0.00]
To draw a straight linear line that starts at x18y31 and ends at x35y14 use: Points = ['x18y31', 'x35y14'] t_values = [0.00, 1.00].
If you want to draw a big and long stroke, split it into multiple small curves that connect to each other. These instructions will define a smooth stroke that follows a Bezier curve from the starting point to the ending point, passing through the specified intermediate points. To draw a visually appealing sketch of the given object or concept, break down complex drawings into manageable steps. Begin with the most important part of the object, then observe your progress and add additional elements as needed. Continuously refine your sketch by starting with a basic structure and gradually adding complexity. Think step-by-step.

Figure 51. System prompt.

I provide you with a blank grid. Your goal is to produce a visually appealing sketch of a {concept}.
Here are a few examples:
<examples>
{gt-sketches}
</examples>

You need to provide x-y coordinates that construct a recognizable sketch of a concept.
You will receive feedback on your sketch and you will be able to adjust and fix it. Note that you will not have access to any additional resources. Do not copy previous sketches.

Think before you provide the x-y coordinates in <thinking> tags.
First, think through what parts of the concept you want to sketch and the sketching order.
Then, think about where the parts should be located on the grid.
Finally, provide your response in <answer> tags, using your analysis.

Provide the sketch in the following format with the following fields:
<formatting>
<concept>The concept depicted in the sketch.</concept>
<strokes>This element holds a collection of individual stroke elements that define the sketch.
Each stroke is uniquely identified by its own tag (e.g., <s1>, <s2>, etc.).
Within each stroke element, there are three key pieces of information:
<points>A list of x-y coordinates defining the curve. These points define the path the stroke follows.</points>
<t_values>A series of numerical timing values that correspond to the points. These values define the progression of the stroke over time, ranging from 0 to 1, indicating the order or speed at which the stroke is drawn.</t_values>
<id>A short descriptive identifier for the stroke, explaining which part of the sketch it corresponds to.</id>
</strokes>
</formatting>

Figure 52. User prompt. This prompt contains the specific sketching task as well as details about the expected format.

<example>
To draw a house, start by drawing the front of the house:
<concept>House</concept>
<strokes>
<s1>
<points>'x13y27', 'x24y27', 'x24y27', 'x24y11', 'x24y11', 'x13y11', 'x13y11', 'x13y27'</points>
<t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
<id>house base front rectangle</id>
</s1>
<s2>
<points>'x13y27', 'x18y37','x18y37', 'x24y27'</points>
<t_values>0.00,0.55,0.5,1.00</t_values>
<id>roof front triangle</id>
</s2>
</strokes>

Next we add the house's right section:
<concept>House</concept>
<strokes>
<s1>
<points>'x13y27', 'x24y27', 'x24y27', 'x24y11', 'x24y11', 'x13y11', 'x13y11', 'x13y27'</points>
<t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
<id>house base front rectangle</id>
</s1>
<s2>
<points>'x13y27', 'x18y37','x18y37', 'x24y27'</points>
<t_values>0.00,0.55,0.5,1.00</t_values>
<id>roof front triangle</id>
</s2>
<s3>
<points>'x24y27', 'x36y28', 'x36y28', 'x36y21', 'x36y21', 'x36y12', 'x36y12', 'x24y11'</points>
<t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
<id>house base right section</id>
</s3>
<s4>
<points>'x18y37', 'x30y38', 'x30y38', 'x36y28'</points>
<t_values>0.00,0.55,0.5,1.00</t_values>
<id>roof right section</id>
</s4>
</strokes>

Now that we have the general structure of the house, we can add details to it, like windows and a door:
<concept>House</concept>
<strokes>
<s1>
<points>'x13y27', 'x24y27', 'x24y27', 'x24y11', 'x24y11', 'x13y11', 'x13y11', 'x13y27'</points>
<t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
<id>house base front rectangle</id>
</s1>
<s2>
<points>'x13y27', 'x18y37','x18y37', 'x24y27'</points>
<t_values>0.00,0.55,0.5,1.00</t_values>
<id>roof front triangle</id>
</s2>

Figure 53. ICL example. This is the example of a sketch of a house we provide to the model.

```
    <s3>
        <points>'x24y27', 'x36y28', 'x36y28', 'x36y21', 'x36y21', 'x36y12', 'x36y12', 'x24y11'</points>
        <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
        <id>house base right section</id>
    </s3>
    <s4>
        <points>'x18y37', 'x30y38', 'x30y38', 'x36y28'</points>
        <t_values>0.00,0.55,0.5,1.00</t_values>
        <id>roof right section</id>
    </s4>
    <s5>
        <points>'x26y25', 'x29y25', 'x29y25', 'x29y21', 'x29y21', 'x26y21', 'x26y21', 'x26y25'</points>
        <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
        <id>left window square</id>
    </s5>
    <s6>
        <points>'x31y25', 'x34y25', 'x34y25', 'x34y21', 'x34y21', 'x31y21', 'x31y21','x31y25'</points>
        <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
        <id>right window square</id>
    </s6>
    <s7>
        <points>'x17y11', 'x17y18', 'x17y18', 'x21y18', 'x21y18', 'x21y11', 'x21y11', 'x17y11'</points>
        <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
        <id>front door</id>
    </s7>
</strokes>
and here is the complete example:
<concept>House</concept>
<strokes>
    <s1>
        <points>'x13y27', 'x24y27', 'x24y27', 'x24y11', 'x24y11', 'x13y11', 'x13y11', 'x13y27'</points>
        <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
        <id>house base front rectangle</id>
    </s1>
    <s2>
        <points>'x24y27', 'x36y28', 'x36y28', 'x36y21', 'x36y21', 'x36y12', 'x36y12', 'x24y11'</points>
        <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
        <id>house base right section</id>
    </s2>
    <s3>
        <points>'x13y27', 'x18y37','x18y37', 'x24y27'</points>
        <t_values>0.00,0.55,0.5,1.00</t_values>
        <id>roof front triangle</id>
    </s3>
    <s4>
        <points>'x18y37', 'x30y38', 'x30y38', 'x36y28'</points>
        <t_values>0.00,0.55,0.5,1.00</t_values>
        <id>roof right section</id>
    </s4>
```

Figure 54. ICL example. This is the example of a sketch of a house we provide to the model.

```
    <s5>
      <points>'x26y25', 'x29y25', 'x29y25', 'x29y21', 'x29y21', 'x26y21', 'x26y21', 'x26y25'</points>
      <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
      <id>left window square</id>
    </s5>
    <s6>
      <points>'x31y25', 'x34y25', 'x34y25', 'x34y21', 'x34y21', 'x31y21', 'x31y21','x31y25'</points>
      <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
      <id>right window square</id>
    </s6>
    <s7>
      <points>'x17y11', 'x17y18', 'x17y18', 'x21y18', 'x21y18', 'x21y11', 'x21y11', 'x17y11'</points>
      <t_values>0.00,0.3,0.25,0.5,0.5,0.75,0.75,1.00</t_values>
      <id>front door</id>
    </s7>
</strokes>
</example>
```

Figure 55. ICL example. This is the example of a sketch of a house we provide to the model.

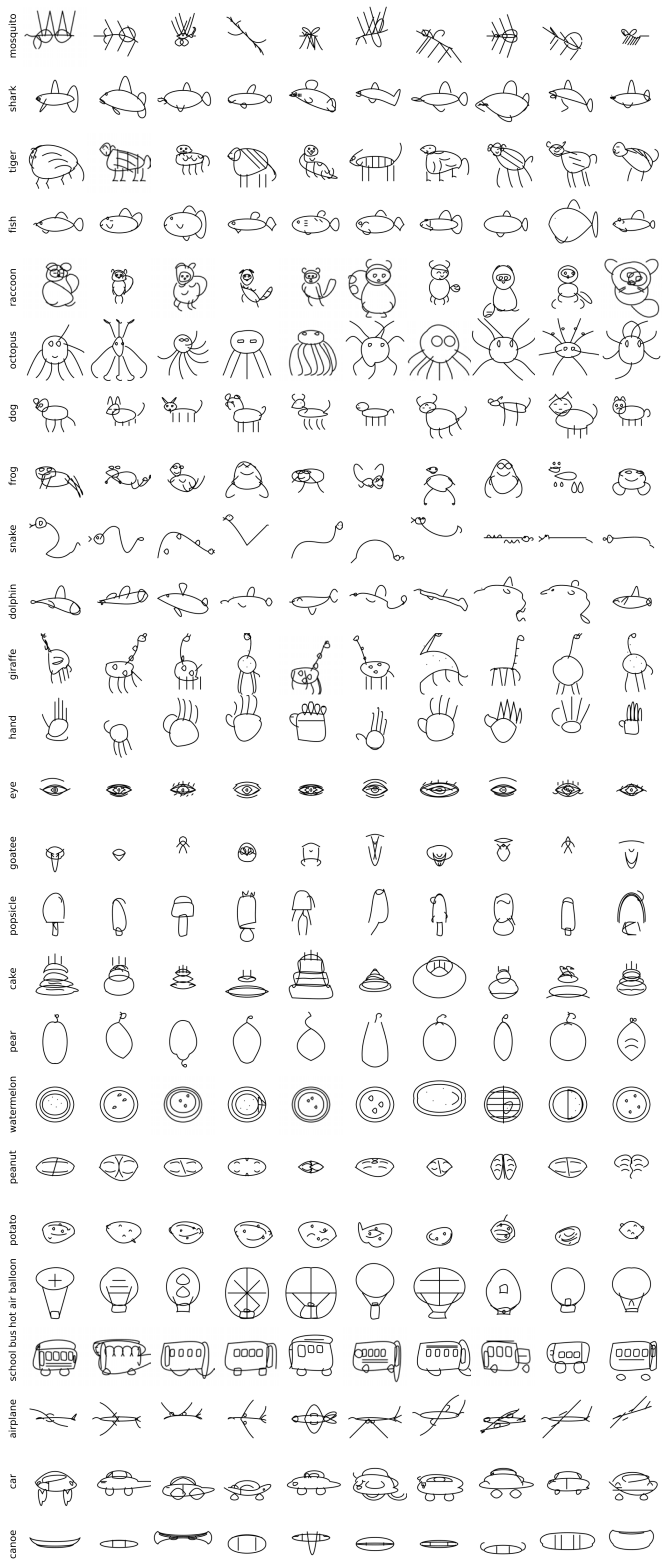


Figure 56. Randomly generated sketches used in the quantitative analysis (ten sketches per category).

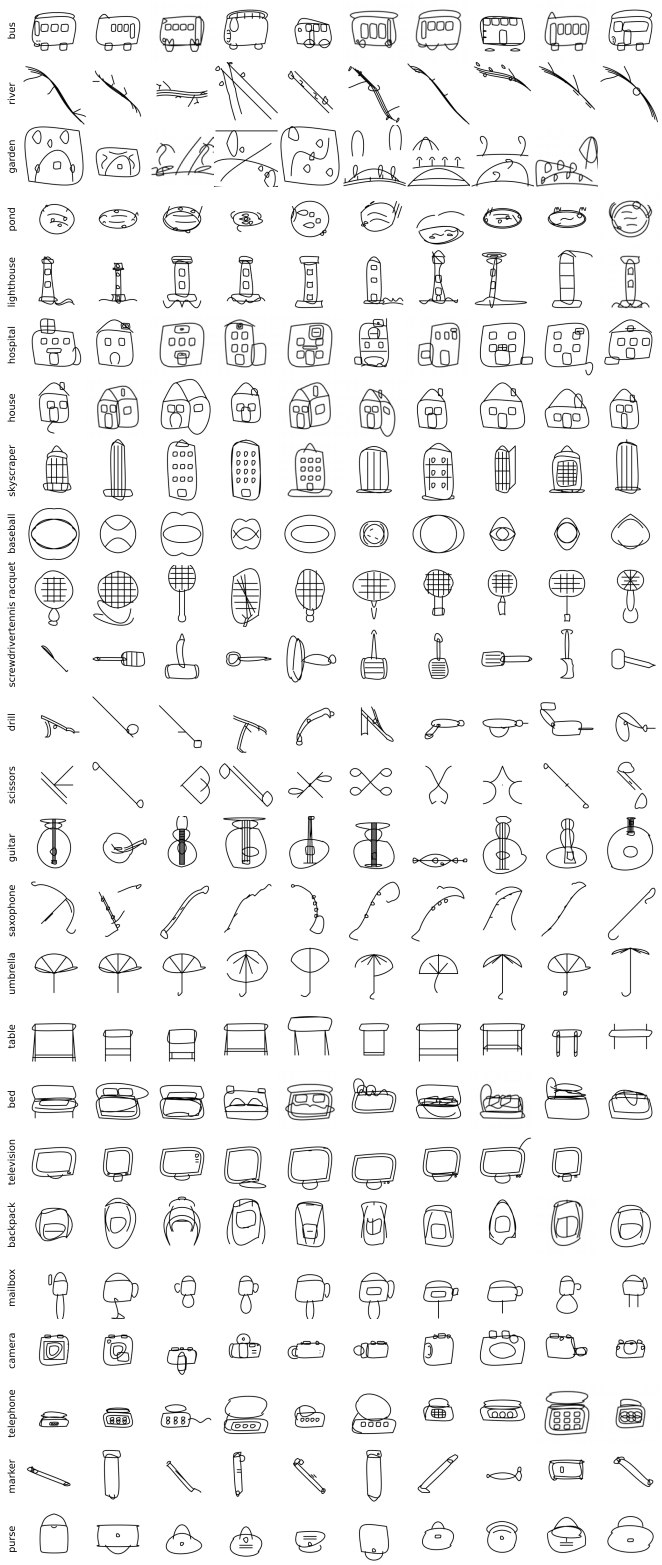


Figure 57. Randomly generated sketches used in the quantitative analysis (ten sketches per category).

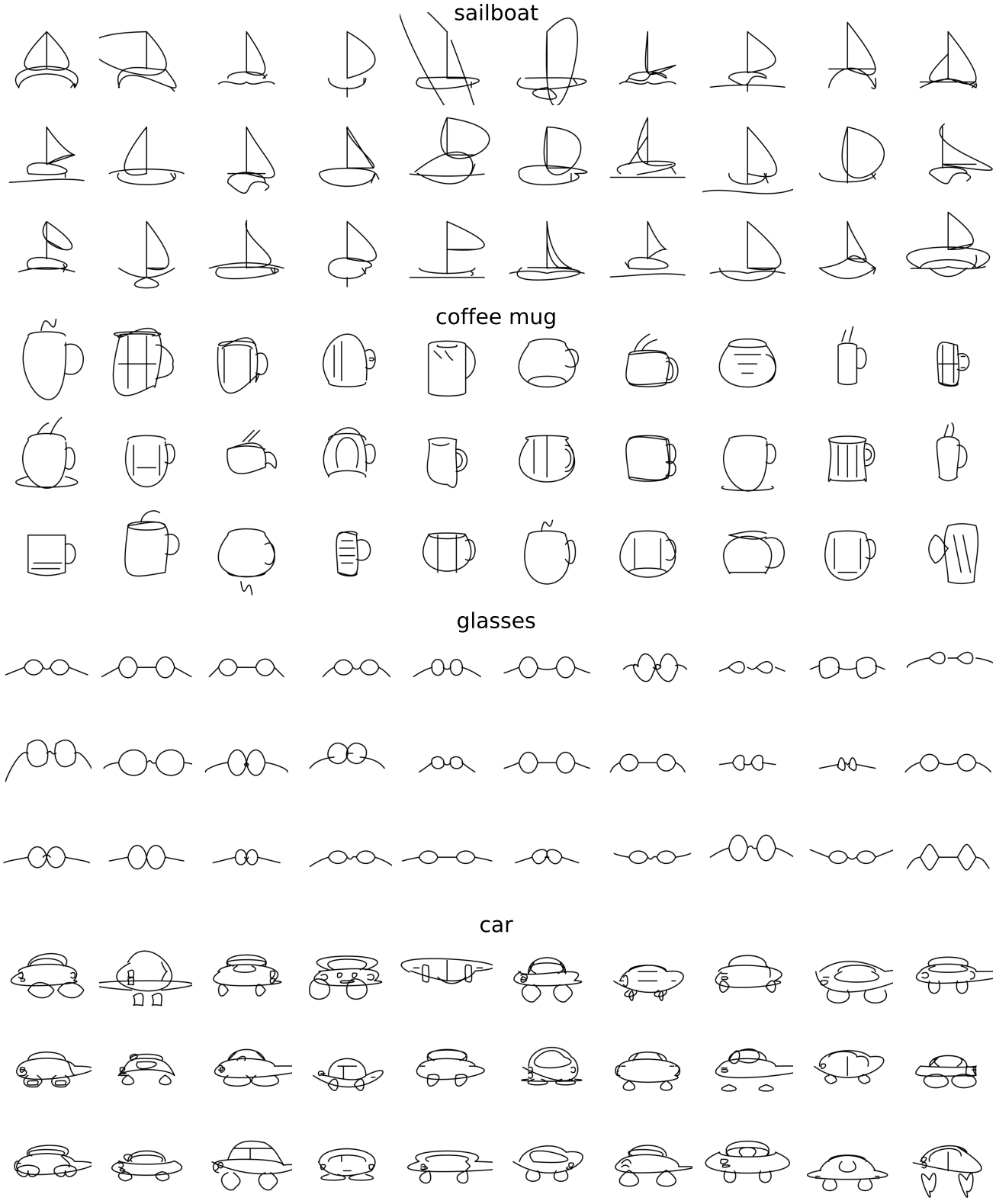


Figure 58. Sketches generated by SketchAgent for the eight categories of our human-agent collaborative study.

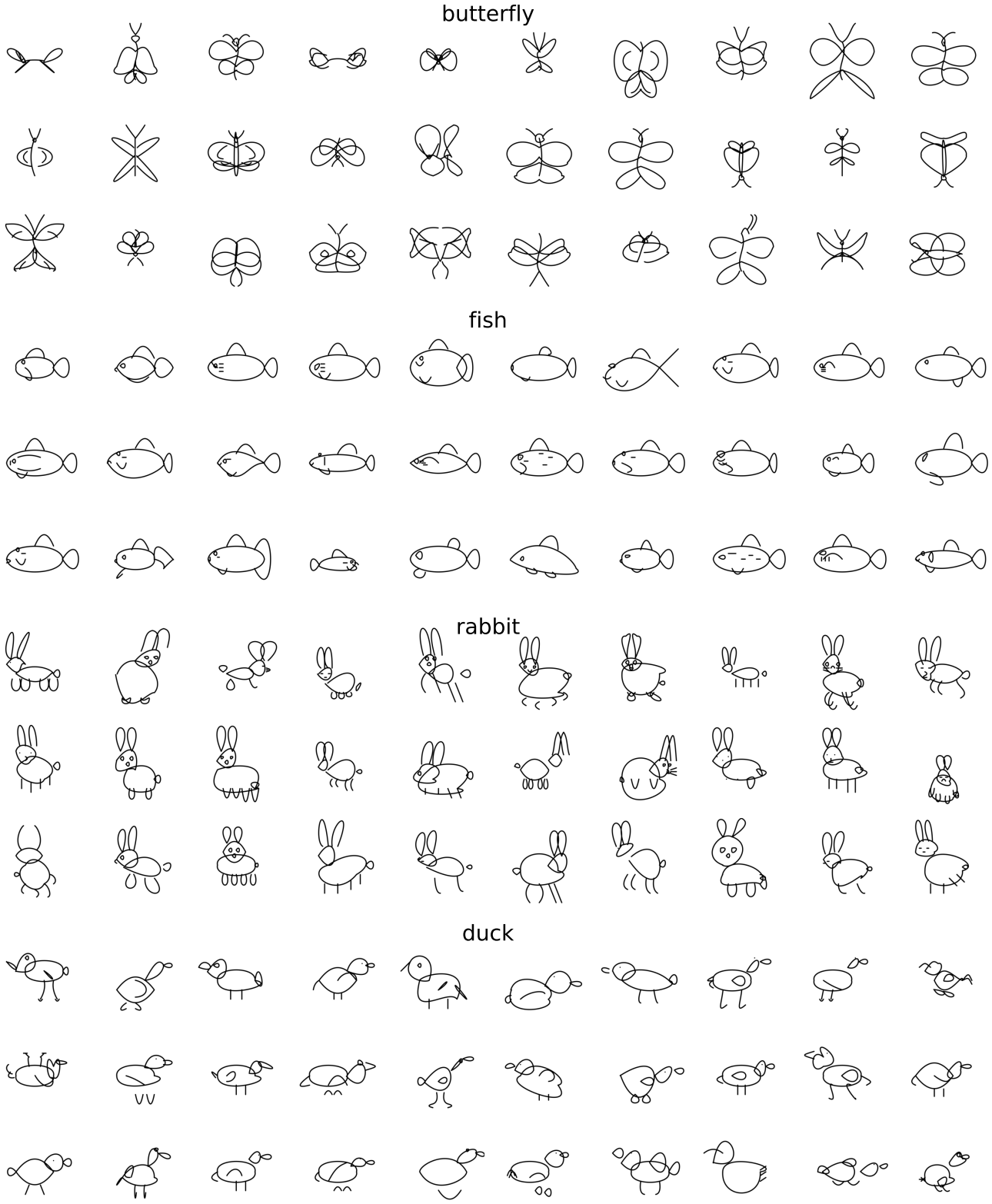


Figure 59. Sketches generated by SketchAgent for the eight categories of our human-agent collaborative study.

Figure 60. Sketches created collaboratively by users and SketchAgent as part of our collaborative human study.

Figure 61. Sketches created collaboratively by users and SketchAgent as part of our collaborative human study.

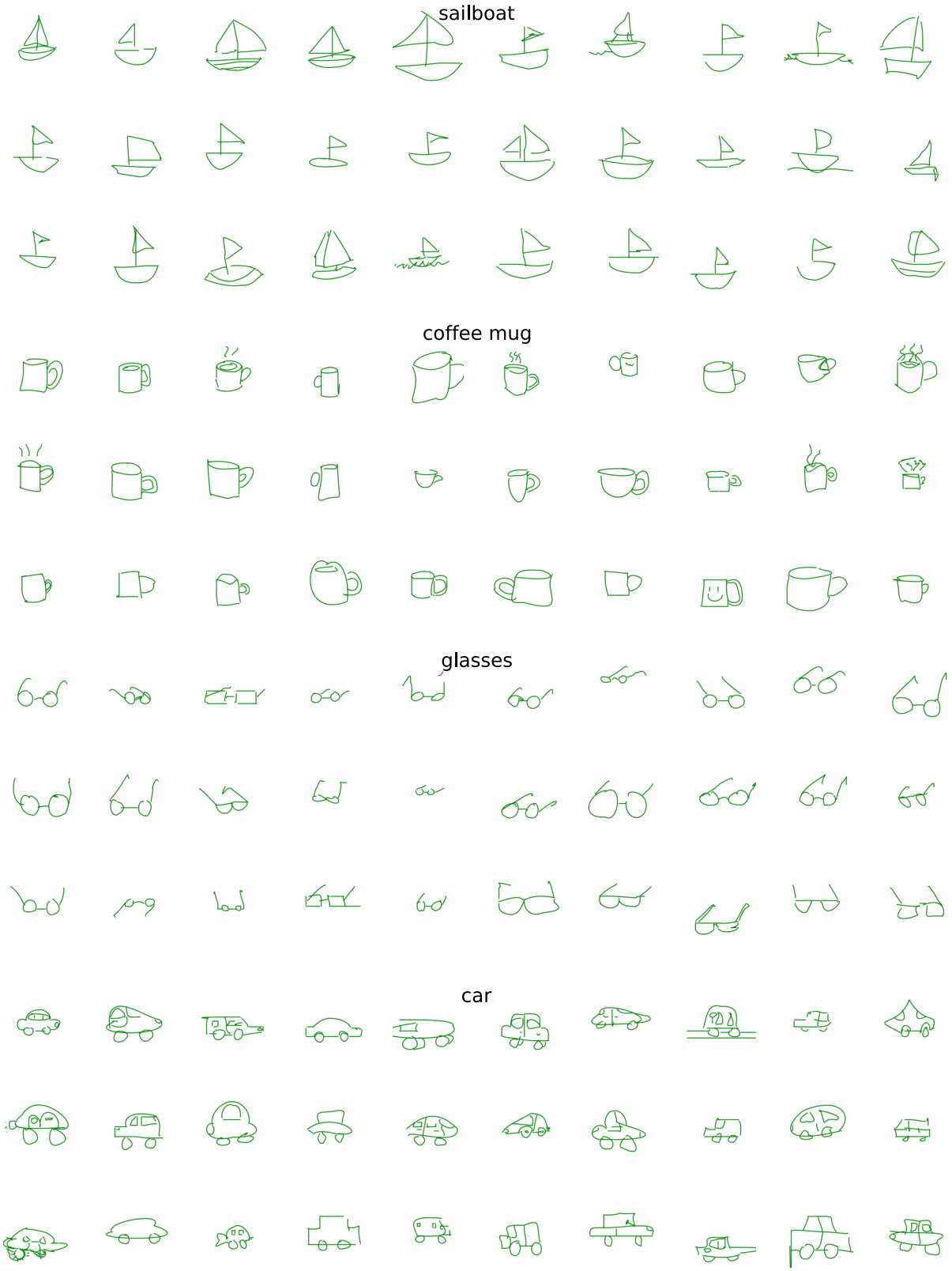


Figure 62. Sketches created by users in “solo” mode as part of our collaborative human study.

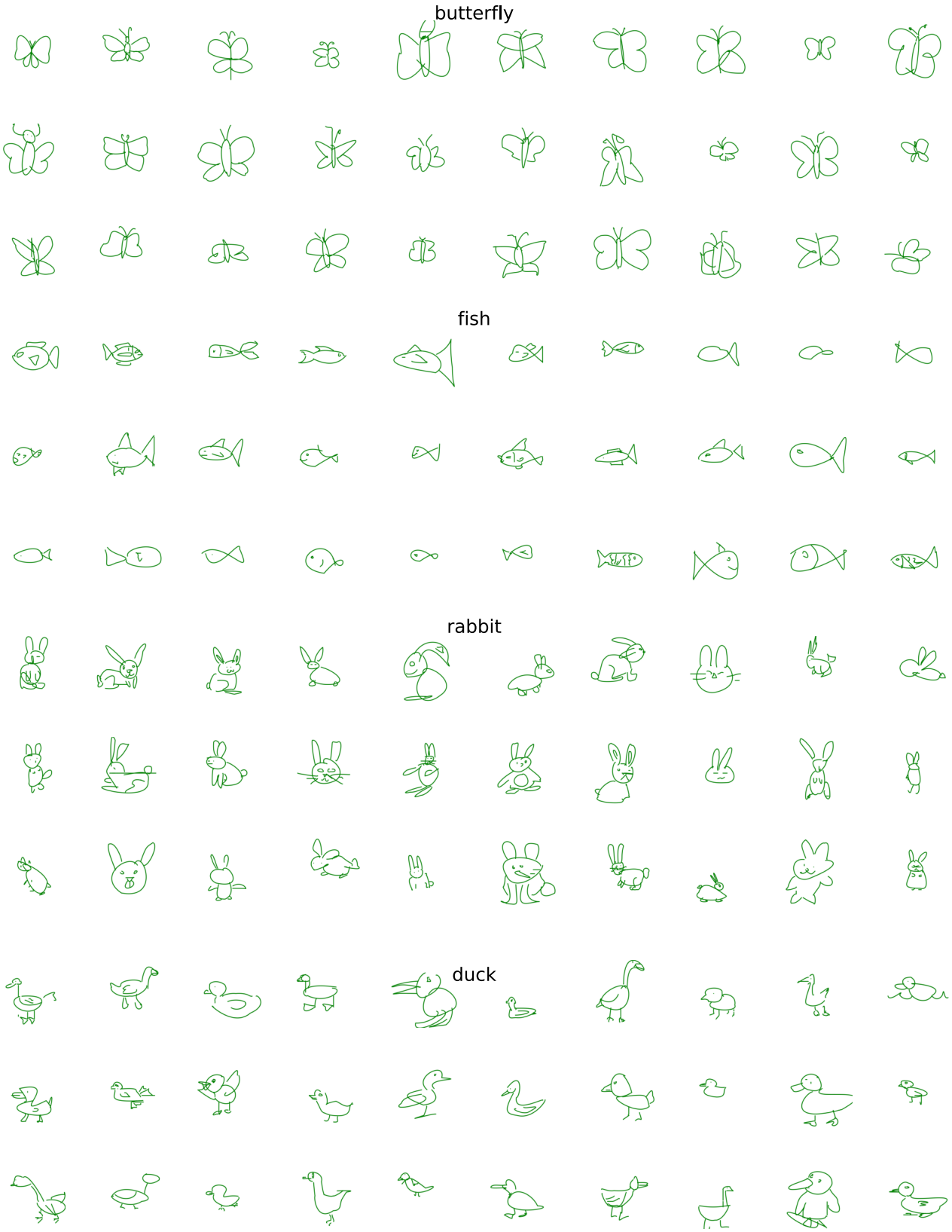


Figure 63. Sketches created by users in “solo” mode as part of our collaborative human study.