Empowering LLMs to Understand and Generate Complex Vector Graphics

Supplementary Material

Overview

In this supplementary material, we provide additional details and discussions related to our work on **LLM4SVG**. Specifically, this document covers the following aspects:

- **Comparison with Existing LLM-Based Methods** (Sec. A): We demonstrate the advantages of our approach over existing LLM-based methods for SVG generation, particularly in terms of visual appeal and the ability to handle numerical coordinates.
- **Dataset and Preprocessing Pipeline** (Sec. B): We introduce our newly collected dataset and describe a lossless preprocessing pipeline that enhances data quality for training and evaluation.
- **SVG Semantic Tokens** (Sec. C): We provide details on the SVG semantic tokens used in our model, along with an analysis of how different word embedding initialization methods affect model performance.
- **User Study Details** (Sec. D): We present additional details about the user study, including the number of participants, their backgrounds, and the methodology used to obtain evaluation metrics.
- **Primitive Ordering in SVG Generation** (Sec. E): We illustrate how our LLM4SVG model generates SVGs by following a structured primitive ordering strategy. We compare two different generation approaches—a step-by-step composition process and a top-down refinement method—and discuss their implications for SVG interpretability and usability.
- **Additional SVG Generation Results** (Sec. F): We showcase a diverse set of SVG illustrations generated by our model, demonstrating its ability to produce high-quality, semantically accurate, and stylistically consistent vector graphics across various categories, including objects, animals, food, and abstract symbols.

A. Comparison with LLM-based Methods

Apart from Figure 4 in the manuscript, we present additional qualitative comparisons of our method with existing LLM-based methods in this section, including ChatGPT-3.5 [36], GPT-4o [2], GPT-o1-preview [2], Claude 3.5-sonnet [4], Gemini 1.5-pro [9], Grok 2 [63], LLama 3.1 [13, 54], Qwen 2.5 [51, 57], and Yi [68].

As illustrated in Fig. S3, it is evident that our proposed LLM4SVG outperforms other methods in terms of overall visual effect and detail expression. Specifically, most LLMs struggle to generate complete SVG images, for example, *butterfly*, or produce outputs that accurately align with the provided textual descriptions, such as *two-hump camel* illustrated in the last third example. Some recent LLMs perform relatively better, such as GPT-4o, GPT-o1-preview [2], and Claude 3.5-sonnet [4]. However, their results are still less satisfactory as the shapes are overly simplistic (e.g., *flute*) and the colors lack harmony (e.g., *a bed*). In contrast, the results of our LLM4SVG are more complete, with shapes that are more diverse, colors that are more harmonious, and semantics that are more closely aligned with the prompts.

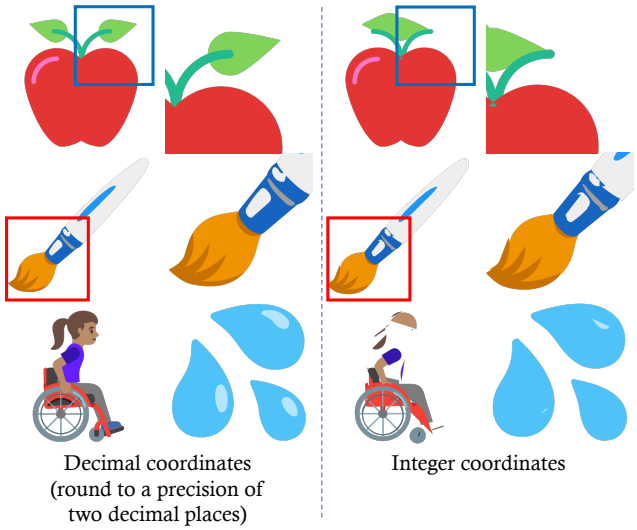

Figure S1. **Visual Comparison between Decimal Coordinates and Integer Coordinates in SVGs.** Only integer coordinates will lead to shape distortions and incompletion.

Additionally, we compare the SVG source codes generated by our method and two of the most recent LLMs, GPT-o1-preview [2] and Claude 3.5-sonnet [4], to demonstrate the superiority of our method. As shown in Fig. S4, given the prompt "umbrella", both GPT-o1-preview [2] and Claude 3.5-sonnet [4] can only predict integer coordinates, whereas our LLM4SVG is capable of generating precise decimal coordinates accurate to two decimal places. In the context of SVG representation, retaining only integer values can lead to incomplete or distorted SVG shapes, as illustrated in Fig. S4. We present additional examples in Fig. S1 to further demonstrate the visual differences between integer and decimal coordinates.

B. SVGX-SFT Dataset Details

Figure S2 presents examples from our extensive and diverse SVGX-SFT dataset. This dataset includes primitives of varying complexity, ranging from minimal to highly

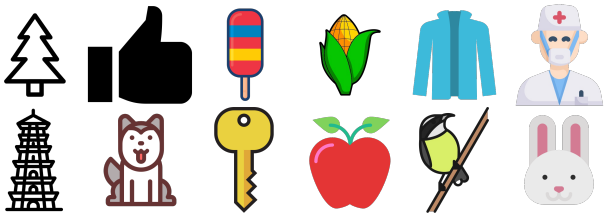

Figure S2. **Illustrative Samples from the SVGX-SFT Dataset**, showcasing its diversity in style, structure, and semantics. The dataset includes a wide range of objects, icons, and illustrations, making it well-suited for training and fine-tuning vector graphic generation models.

detailed, while maintaining a rich and harmonious color palette. Additionally, it covers a broad spectrum of subjects, including people, animals, objects, and symbols, making it a comprehensive resource for both SVG generation and understanding tasks.

Preprocessing Pipeline. As discussed in Section 3 of our manuscript, a significant portion of an SVG file consists of redundant metadata that does not contribute to its rendered appearance. To improve training efficiency while preserving visual fidelity, we introduce a lossless SVG preprocessing pipeline, as illustrated in Figure S5.

Our pipeline systematically removes unnecessary elements from SVG files, including XML declarations, comments, titles, descriptions, `<defs>` and `<class>` tags, as well as global `<g>` tags. Additionally, we optimize numerical representations by converting absolute coordinates to relative ones and rounding decimal values to a maximum of two decimal places. Furthermore, all SVGs are resized to a standardized 128×128 canvas, ensuring consistency across the dataset. These optimizations significantly reduce file size and computational overhead without altering the visual output.

Instruction-Based Dataset Construction. After preprocessing, we structured the dataset to support both SVG generation and understanding tasks. For understanding tasks, each SVG was rasterized into an image, and GPT-4 [2] was used to generate detailed descriptions, which serve as learning targets. For generation tasks, textual prompts were generated using BLIP [29], with the corresponding SVG source code serving as the learning content.

In total, our dataset comprises 580k high-quality SVG-text instruction-compliant samples, providing a robust foundation for training models in structured vector graphic generation and interpretation.

C. Our Proposed SVG Semantic Tokens

For an input SVG $\mathbf{X}_v$, we convert it from raw code into a structured representation. As shown in Tab. S1, we present a detailed taxonomy of the SVG semantic tokens employed in LLM4SVG, including 15 tag tokens, 30 attribute tokens and 10 path command tokens. These SVG tokens are used to replace all tags and attributes in the SVG source code, thus preventing the textual encoding of SVG tags and attributes as regular text. This ensures the uniqueness of SVG tags and attributes, and allows for their efficient integration into LLMs in a manner that is consistent with SVG definitions. The "Description" field is utilized to initialize the SVG Tokens based on Equation 1.

Analysis of Word Embedding Initialization Method. As illustrated in Fig. S6, we used T-SNE [55] to map the newly added tokens into a two-dimensional visualization for better demonstration. In this visualization, the green dots represent tokens that were initialized using the text description of each token. Specifically, we averaged the values obtained from tokenizing these descriptions to initialize the tokens, a process detailed in Sec. 4.1. This strategy groups the token embeddings into a relatively compact region within the feature space, which helps reduce the difficulty of model training. The orange crosses, on the other hand, indicate tokens that were initialized without using the text description. These tokens exhibit a more scattered distribution across the feature space, making it challenging for the model to learn the accurate meaning of these tokens. The blue squares represent the positions of the tokens within the feature space post-training. It is evident that after the training phase, the token embeddings show more meaningful groupings, where tokens with similar semantics are clustered together while maintaining relative proximity to all SVG tokens. Additionally, these tokens remain closely within the overall feature space. This visualization demonstrates the rationale behind our token addition method and the effectiveness of our training approach.

D. More Details about User Study

We conducted a user survey to evaluate the effectiveness and practicality of the SVGs generated by our method LLM4SVG and two other popular LLMs, GPT-4o [2] and Claude-3.5 [4]. Specifically, the user study was structured as follows:

1. **Data Preparation**: We randomly sampled 17 text descriptions from the evaluation dataset and generated corresponding SVGs using the three models. Along with the original 17 SVGs from the dataset, this provided a total of 68 SVGs for the user study.
2. **Questionnaire Design**: Each questionnaire displayed 5 SVGs, randomly sampled from the pool of 68 SVGs. These SVGs were not necessarily generated from the same prompt.
   As illustrated in Fig. S7, participants were asked to evaluate each SVG on three aspects:
   - **Prompt Alignment**: How well does this SVG align with the given prompt? (Rated on a scale from 1 to 5,

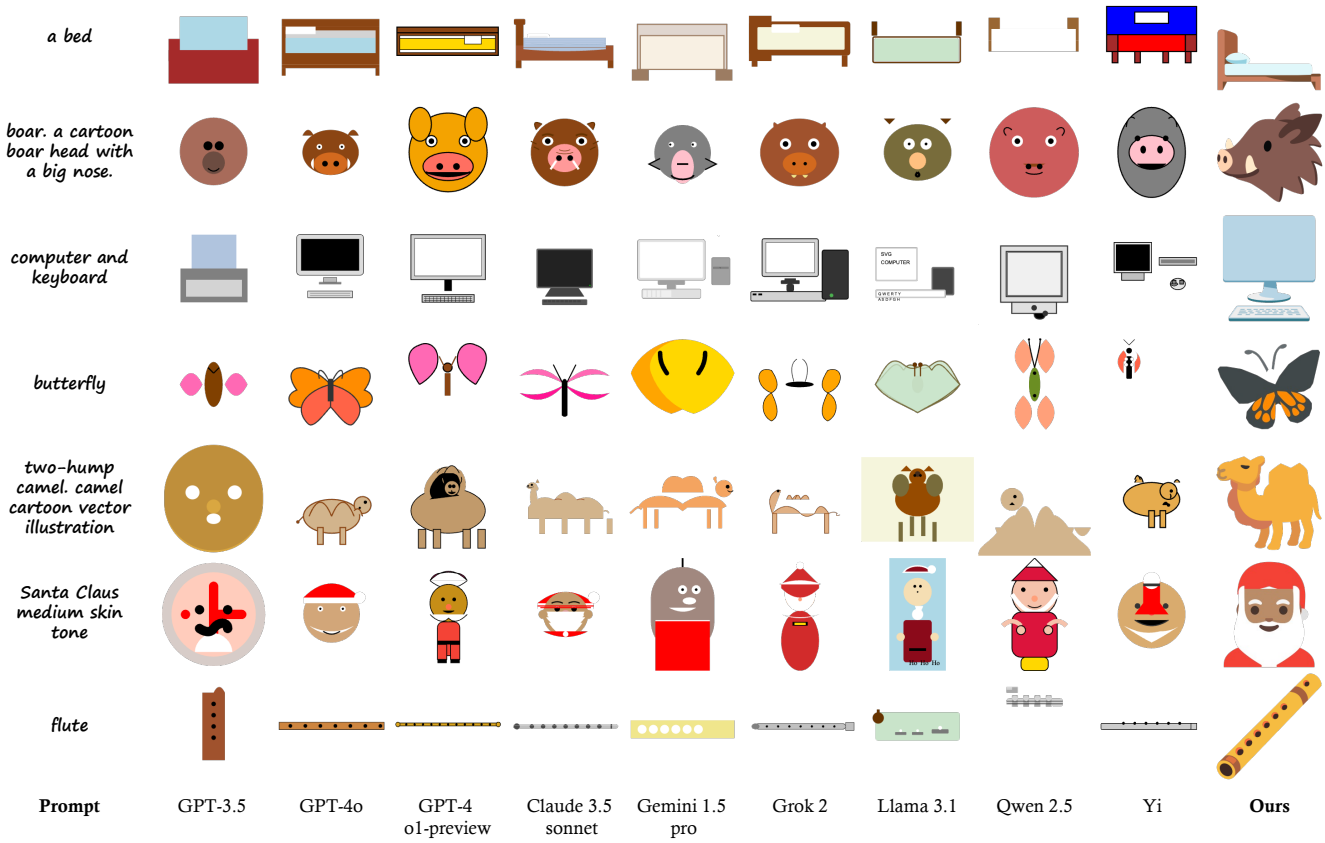

Figure S3. **Qualitative Comparison of LLM4SVG with Existing LLM-based Methods.** Given textual prompts (left), various LLMs generate corresponding vector graphics. The comparison highlights differences in abstraction, structural consistency, and fidelity to the input descriptions. Our LLM4SVG demonstrates improved coherence, accuracy, and stylistic refinement in SVG generation.

where 1 indicates the lowest score, while 5 represents the highest score.)

- **Visual Quality**: How appealing is the visual design of this SVG? (Rated on a scale from 1 to 5, where 1 indicates the lowest score, while 5 represents the highest score.)
- **Pick Score**: Would you consider using this SVG in real-world scenarios? (Yes/No)

3. **Result Calculation**: This user study involves 37 volunteers from backgrounds in computer science and the arts. Each volunteer was required to complete between 1 and 3 questionnaires. Scores presented in Table 3 of our manuscript were calculated by averaging the ratings for SVGs within the same category. For "Prompt Alignment" and "Visual Quality", the ratings were adjusted by a coefficient $\alpha = 0.2$, such that a score of 1 translates to 0.2, and a score of 5 counts to 1. For "Pick Score", a "Yes" was scored as 1, while a "No" was scored as 0.

E. Primitive Ordering in SVG Generation

Figure S8 illustrates how our LLM4SVG model generates SVGs by following a primitive ordering strategy that aligns with human design principles. This structured approach ensures that vector graphics are constructed in an intuitive and interpretable manner.

The upper sequence in Figure S8 demonstrates a step-by-step composition process. The design begins with a few basic primitives, such as lines and simple shapes, and progressively adds more details. This method closely resembles how human designers create illustrations—starting with foundational elements before refining them into a complete object. For example, the umbrella starts with a simple handle, then adds canopy sections until the final structured design emerges. Similarly, the dolphin illustration begins with a basic shape, followed by gradual refinements to enhance realism. This process ensures that each step maintains logical continuity, making the SVG more comprehensible and easier to manipulate.

In contrast, the lower sequence showcases an alternative

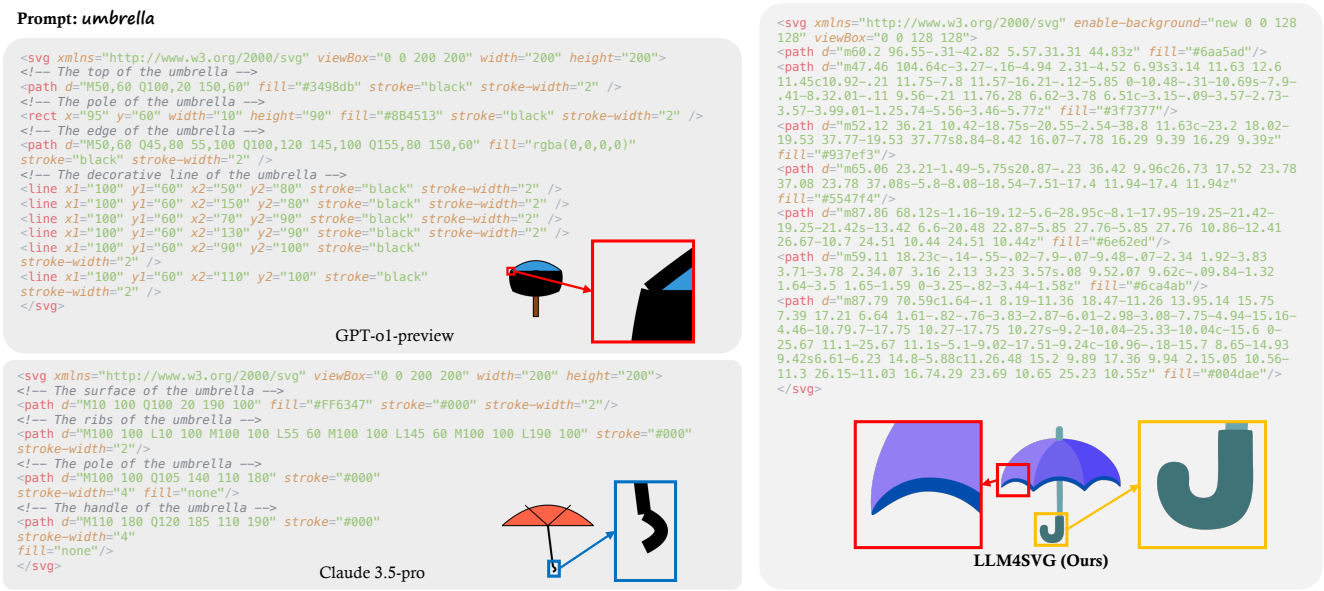

Figure S4. **Example SVG Code Comparison For the Prompt "umbrella"**. The figure contrasts SVG outputs generated by GPT-4o-preview, Claude 3.5-sonnet, and our LLM4SVG. While GPT-4o-preview and Claude 3.5-pro produce basic umbrella-like shapes, their structures contain artifacts or lack refinement. In contrast, LLM4SVG generates a more polished, visually coherent, and structured SVG representation, demonstrating improved detail, smoothness, and geometric accuracy.

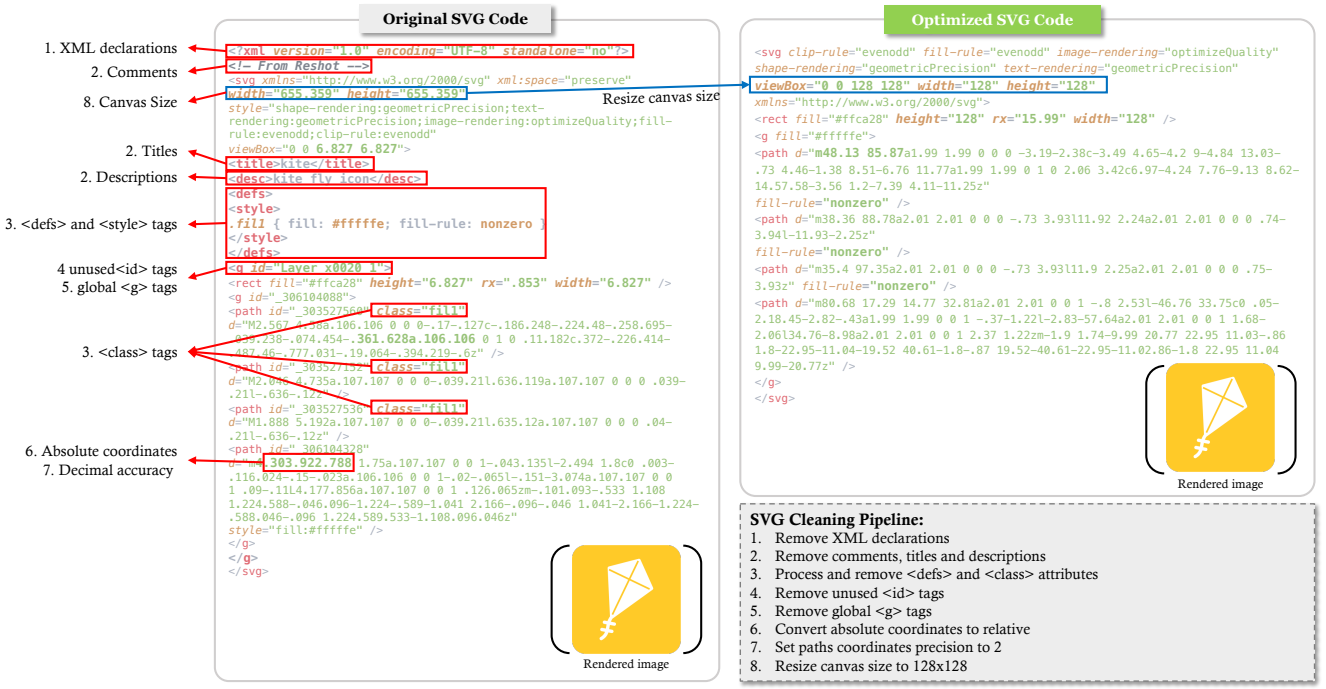

Figure S5. **Illustration of our SVG Processing Pipeline.** The left side shows the original SVG code, which contains redundant elements such as XML declarations, comments, metadata, unused tags, and absolute coordinates. The right side presents the optimized SVG code after applying our cleaning pipeline, which improves efficiency, readability, and scalability by removing unnecessary elements, converting absolute coordinates to relative, and standardizing canvas size. The rendered output remains visually consistent while significantly reducing file complexity.

Category	Token	Description
SVG Container Tags	[<|START_OF_SVG|>]	start of svg
	[<|END_OF_SVG|>]	end of svg
	[<|start_of_g|>]	start of svg group
	[<|end_of_g|>]	end of svg group
SVG Geometry Tags	[<|svg_path|>]	svg path element
	[<|svg_circle|>]	svg circle element
	[<|svg_rect|>]	svg rectangle element
	[<|svg_ellipse|>]	svg ellipse element
	[<|svg_polygon|>]	svg polygon element
	[<|svg_line|>]	svg line element
	[<|svg_polyline|>]	svg polyline element
	[<|svg_text|>]	svg text element
SVG Gradient Tags	[<|svg_linearGradient|>]	svg linear gradient element
	[<|svg_radialGradient|>]	svg radial gradient element
	[<|svg_stop|>]	svg stop element
Path Commands	[<|moveto|>]	svg path command, move to
	[<|lineto|>]	svg path command, line to
	[<|horizontal_lineto|>]	svg path command, horizontal line to
	[<|vertical_lineto|>]	svg path command, vertical line to
	[<|curveto|>]	svg path command, curve to
	[<|smooth_curveto|>]	svg path command, smooth curve to
	[<|quadratic_bezier_curve|>]	svg path command, quadratic bezier curve
	[<|smooth_quadratic_bezier_curveto|>]	svg path command, smooth quadratic bezier curve
	[<|elliptical_Arc|>]	svg path command, elliptical arc
	[<|close_the_path|>]	svg path command, close the path, close-form
Attribute Tokens	[<|id|>]	svg element attribute id
	[<|d|>]	svg element attribute define the path
	[<|fill|>]	svg element attribute fill
	[<|stroke-width|>]	svg element attribute stroke-width
	[<|stroke-linecap|>]	svg element attribute stroke-linecap
	[<|stroke|>]	svg element attribute stroke
	[<|opacity|>]	svg element attribute opacity
	[<|transform|>]	svg element attribute transform
	[<|gradientTransform|>]	svg element attribute gradient transform
	[<|offset|>]	svg element attribute offset
	[<|width|>]	svg element attribute width
	[<|height|>]	svg element attribute height
	[<|cx|>]	svg element attribute x coordinate of circle center
	[<|cy|>]	svg element attribute y coordinate of circle center
	[<|rx|>]	svg element attribute x radius of ellipse
	[<|ry|>]	svg element attribute y radius of ellipse
	[<|r|>]	svg element attribute radius of circle
	[<|points|>]	svg element attribute points
	[<|x1|>]	svg element attribute x1 coordinate
	[<|y1|>]	svg element attribute y1 coordinate
	[<|x2|>]	svg element attribute x2 coordinate
	[<|y2|>]	svg element attribute y2 coordinate
	[<|x|>]	svg element attribute x coordinate
	[<|y|>]	svg element attribute y coordinate
	[<|fr|>]	svg element attribute fr
	[<|fx|>]	svg element attribute fx
	[<|fy|>]	svg element attribute fy
	[<|href|>]	svg element attribute href
	[<|rotate|>]	svg element attribute rotate
	[<|font-size|>]	svg element attribute font-size

Table S1. **SVG Semantic Tokens Defined by Our LLM4SVG.** We define 15 tag tokens (including 4 SVG container tags, 8 SVG geometry tags, and 3 SVG gradient tags), 30 attribute tokens, and 10 path command tokens in our LLM4SVG. The "Token" field corresponds to the Token defined in Table 1. The "Description" field is used to initialize the Token .

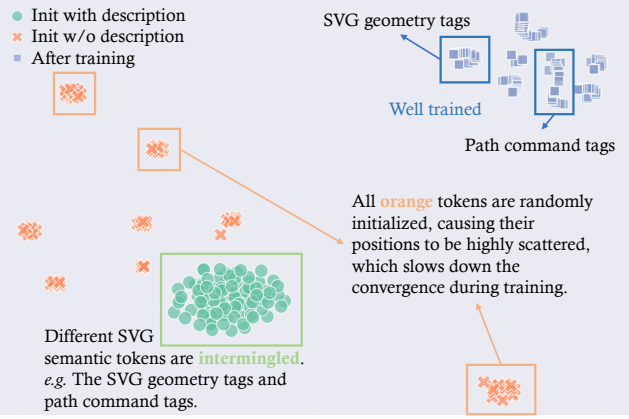

Figure S6. **t-SNE Visualization of Token Embeddings.** The green dots represent the SVG token embeddings initialized with descriptions, while the orange crosses indicate those initialized without descriptions. The blue squares represent SVG token embeddings after training.

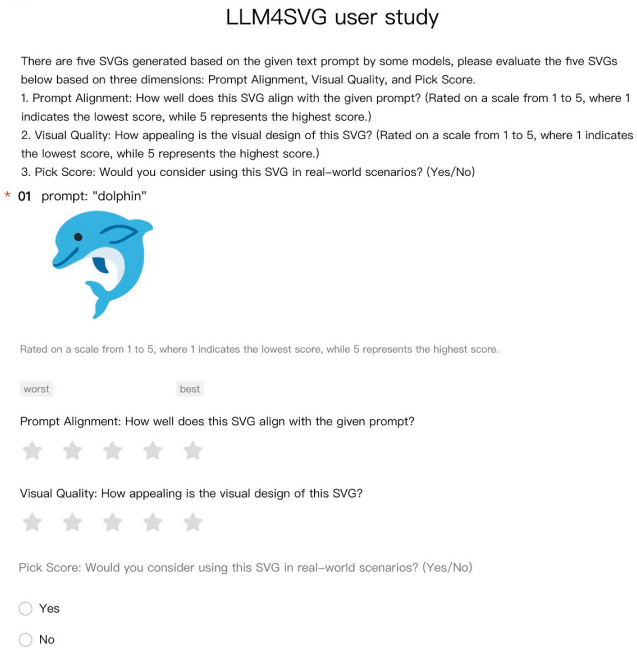

Figure S7. **Screenshot of Our Questionnaire used in the LLM4SVG User Study.** Participants evaluate five SVGs generated from a given text prompt based on three key criteria: (1) Prompt Alignment—how well the SVG matches the given prompt, (2) Visual Quality—the aesthetic appeal of the SVG, and (3) Pick Score—whether the participant would consider using the SVG in real-world applications. Ratings are provided on a 5-point scale, with an additional Yes/No selection for practical usability.

approach where the entire object is introduced first, followed by successive refinements. This method mimics a top-down design strategy, where an overall form is quickly established before adding intricate details. While this approach can be efficient for certain applications, it lacks the

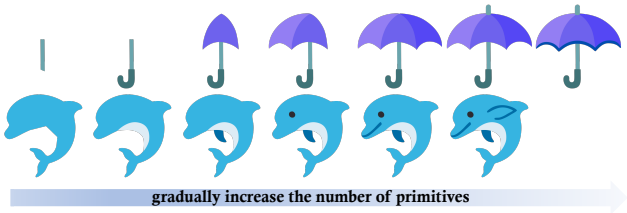

Figure S8. **Our LLM4SVG model generates SVGs with primitive ordering that aligns with human design principles.** The upper example demonstrates a gradual design process, progressing from individual components to the complete design. In contrast, the lower example begins with the overall design before detailing each individual component.

structured progression seen in human sketching workflows, which typically emphasize incremental construction.

By structuring SVG generation in a way that mirrors human cognitive processes, our LLM4SVG model enhances the interpretability and usability of vector graphics. This structured ordering makes the model particularly useful for applications such as educational tools that teach step-by-step drawing, design software that supports intuitive vector editing, and automated graphic generation systems that produce clear and logically constructed icons. The ability to generate illustrations progressively ensures that the output is both aesthetically pleasing and functionally adaptable.

F. Additional Results Generated by Our LLM4SVG

Figure S9 showcases a diverse collection of SVG illustrations generated by our LLM4SVG model. These results demonstrate the model's capability to produce high-quality, semantically accurate, and visually appealing vector graphics across a wide range of categories. The generated SVGs span various domains, including:

- **Objects:** Everyday items such as a pen, paperclip, key, and teapot.
- **Animals:** Illustrations of a dolphin, parrot, horse, fish, and giraffe.
- **Food:** Fruits, vegetables, and prepared dishes, including a tomato, orange, lime, and a rice bowl.
- **People and Expressions:** Human figures representing different professions, emotions, and activities.
- **Symbols and Abstract Concepts:** Medals, graphs, weather symbols, and a heart icon.

These results highlight the effectiveness of our model in generating stylistically consistent and visually coherent vector graphics. The illustrations maintain a balanced level of abstraction while preserving key details, making them suitable for real-world applications such as digital icons, UI elements, and educational materials.

Figure S9. **Additional results generated by our LLM4SVG model.** This collection showcases a diverse range of high-quality SVG illustrations covering various categories, including objects, animals, food, people, symbols, and abstract concepts. The results demonstrate the model's ability to produce visually appealing, semantically accurate, and stylistically consistent vector graphics.