# IntroSVG: Learning from Rendering Feedback for Text-to-SVG Generation via an Introspective Generator-Critic Framework

Supplementary Material

## 8. Data processing pipeline

### 8.1. SVG-related datasets and benchmarks

The development of data resources in the field of vector graphics generation exhibits a dual-track evolutionary trajectory: regarding training data, there has been a shift from large-scale unsupervised collection toward a refined approach focused on attribute enhancement and reasoning guidance; concurrently, the evaluation system has matured, establishing comprehensive benchmarks that encompass multi-dimensional, multi-format, and fine-grained editing capabilities.

**Large-scale Foundation Datasets:** To address the scarcity of training data, early research focused on constructing million-scale datasets to cover a wide distribution of graphics. StarVector introduced SVG-Stack, containing 2.1 million real code samples from GitHub . It retains diagrams and complex primitives (such as circles and polygons), making it one of the datasets with the most authentic code structures . OmniSVG constructed MMSVG-2M (2 million samples), innovating by introducing high-complexity anime characters (20%) and illustrations, and supplementing complex data through diffusion model generation and vectorization techniques . IconShop, based on the FIGR-8-SVG dataset (1.5 million monochrome icons), utilized ChatGPT to expand discrete keywords into natural language descriptions, establishing an early large-scale benchmark for text-to-icon generation . UniSVG (525k) further broke down task barriers by integrating image generation, text generation, and graphic understanding into a unified, cleaned dataset, supporting the all-around fine-tuning of Multimodal Large Language Models (MLLMs).

**Attribute-Enhanced and Colored Datasets:** Addressing the limitations of early data being mostly monochrome or simple outlines, subsequent datasets emphasized enhancing visual richness and structural attributes. SVGBuilder proposed ColorSVG-100K, the first large-scale dataset specifically for colored SVGs (100,000 items), filling the gap in color information in previous datasets . SVGen constructed SVG-1M (1 million), innovatively grading data complexity based on color and command count (Easy/Difficult) to support curriculum learning for models . LLM4SVG, through 250k SVGs and 580k instruction pairs, emphasized treating SVG as semantic tokens for structured understanding.

**Reasoning and Process-Oriented Datasets:** With the improvement of model capabilities, data construction began to focus on the logic and process behind generation. Reason-SVG's SVGX-DwT-10k contains 10,000 curated samples, each equipped with detailed "Chain-of-Thought (CoT)" annotations, recording the complete design flow from conceptual design to coordinate calculation . SVGThinker built a serialized dataset containing 270,000 samples, generating intermediate state images and descriptions corresponding to each step's instruction by reconstructing the SVG tree structure, training the model to understand the logical order of drawing.

**Multi-format Benchmarks:** Evaluation benchmarks have gradually expanded from single generation tasks to understanding and fine-grained editing. VGBench is a broad benchmark that evaluates not only SVG but also TikZ and Graphviz formats, containing 4,279 understanding Q/A pairs and 5,845 generation samples, aiming to assess the general capability of LLMs across different vector languages. SVGEditBench V2 focuses on instruction-level editing, containing 1,683 "Original Image - Instruction - Target Image" triplets built from Emoji datasets, specifically testing the model's ability to modify images (e.g., changing color, rotating) while maintaining the original structure. SVGenius further introduced complexity stratification (Easy/Medium/Hard), comprehensively covering tasks such as understanding, bug fixing, code optimization, and style transfer, providing a more discriminative capability assessment.

To demonstrate the advantages of our data processing, we provide a detailed comparison between our IntroSVG dataset and the original source datasets (OmniSVG, LLM4SVG, SVGen) in Table 6.

### 8.2. Data Collection and Preprocessing

**Data Sources and Motivation:** Although existing large-scale SVG datasets are abundant, utilizing them directly for training presents significant challenges. First, large-scale web-crawled datasets often exhibit severe sample redundancy and high similarity. This not only wastes computational resources during training but also exacerbates the risk of model overfitting. Second, data from diverse sources displays significant heterogeneity in formatting specifications, characterized by inconsistent `viewBox` dimensions, varying coordinate precision (including the number of decimal places), and the mixed usage of relative and absolute path commands. This distributional inconsistency substantially increases the difficulty for models to learn underlying geometric patterns.

Table 6. Comparison of statistics between our IntroSVG dataset and the source datasets.

| Project | SVG Num | Num of multicolor SVG | Command Type | Precision | Unified Viewbox | Source |
|---|---|---|---|---|---|---|
| OmniSVG | 500k | 110k (22%) | Absolute | Decimal | 200, 200 | Iconfont, Iconscout |
| LLM4SVG | 250k | 80k (32%) | Relative | Decimal | 128, 128 | TwEmoji, NotoEmoji, Reshot, SVGRepo |
| SVGGen | 500k | 140k (28%) | Absolute | Integer | 1024, 1024 | Iconfont |
| IntroSVG | 200k | 200k (100%) | Absolute | Integer | 200, 200 | All of the above |

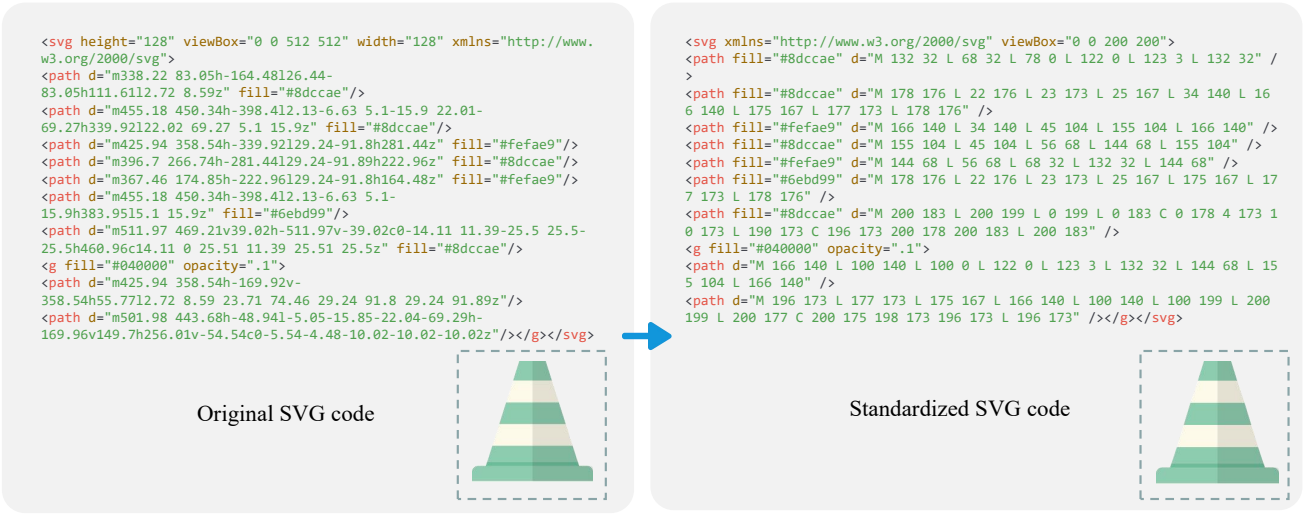



Figure 5. Visual and code comparison before and after data standardization.

To construct a high-quality, standardized colored vector icon dataset that supports complex SVG generation while mitigating overfitting risks, we integrated three mainstream open-source datasets: LLM4SVG, OmniSVG, and SVGen. These datasets encompass rich icon semantics and diverse visual styles, providing a solid foundation for our training.

**Data Cleaning and Standardization Pipeline:** To ensure data uniformity and high quality, we designed and implemented a rigorous filtering and standardization pipeline, detailed as follows:

**Quality Filtering:**

- *Removal of Monochrome Samples:* To focus on generating colored icons with rich visual information, we excluded pure monochrome samples.
- *Removal of Invalid Samples:* We filtered out corrupted files that could not be rendered by standard rendering engines (CairoSVG).
- *Sequence Length Constraint:* We removed samples with token sequence lengths exceeding 8000 to ensure memory efficiency during training and stability during inference.

**Geometric Normalization:**

- *Unified Canvas:* The `viewBox` attributes of all SVG samples were rescaled and normalized to "0 0 200 200", eliminating scale ambiguities caused by differing canvas dimensions.

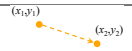
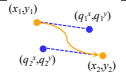
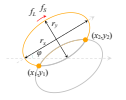
| Name | Symbol | Arguments | Visualization |
|---|---|---|---|
| Move To | M | $(x_1, y_1), (x_2, y_2)$ |  |
| Line To | L | $(x_1, y_1), (x_2, y_2)$ |  |
| Cubic Bézier | C | $(x_1, y_1), (q_1^x, q_1^y), (q_2^x, q_2^y), (x_2, y_2)$ |  |
| Elliptical Arc | A | $(r_x, r_y), \varphi, f_L, f_S, (x, y)$ |  |
| ClosePath | Z | Close the path | - |

Table 7. List of the simplified SVG command vocabulary.

- *Primitive Unification:* All basic shape elements (e.g., `<rect>`, `<circle>`, `<ellipse>`) were converted into generic `<path>` elements.
- *Command Standardization:* We converted all relative commands in path data to absolute commands and retained only five core instruction types: Move (**M**), Line (**L**), Cubic Bézier Curve (**C**), Elliptical Arc (**A**), and Close Path (**Z**), thereby simplifying the vocabulary and

unifying the semantic space, as defined in Table 7.

**Numerical and Format Standardization:**

- *Integer Coordinates:* All floating-point coordinates in paths were rounded to integers. This step significantly reduced token length while maintaining visual fidelity, lowering the difficulty for the model to predict continuous values.
- *Attribute Reordering:* We standardized SVG file headers and enforced a specific attribute order within each `<path>` tag, placing the `fill` (color) attribute before the `d` (path data) attribute. This design aims to guide the model to plan the color style first before generating specific geometric paths, establishing a consistent generation sequence pattern.

The impact of this standardization pipeline is visually demonstrated in Figure 5. The process effectively transforms raw, mixed-format code into a clean, unified representation on a $200 \times 200$ canvas, significantly reducing sequence complexity. Following this rigorous processing pipeline, we ultimately filtered and processed approximately **200,000** high-quality (Text prompt, SVG code) pairs, forming the basis for the experiments in this paper.

## 9. Data Construction

In this section, we provide a granular description of the data synthesis pipeline used to construct the training sets for both the Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) stages. The construction process leverages a "Generator-Critic" loop involving GPT-4o to synthesize high-quality instruction-following data.

**Prompt: GPT-4o image description**

**Role Play**
You are a world-class SVG design master and code engineer, possessing exceptional design aesthetics and a profound understanding of SVG code structure. Your task is to act as a mentor reviewing an SVG work. Your core expertise lies in the ability to reverse-engineer the underlying SVG paths, shapes, and layer structure from a rendered image and provide code-level optimization advice.

**Task Context**
A draft has been generated based on an original design requirement, and a high-quality reference version designed by a human expert is also provided. Both images are rendered from SVG code. Your task is to compare these two works and generate a professional review report for the draft. The report should guide on how to modify the SVG code to learn from and improve towards the reference version, while also considering the draft's consistency with the original design prompt.

**Contextual Information**
1. **Original Prompt:** {`original_prompt`}
2. The draft image is the first input image, used to present the generated design draft.
3. The high-quality reference image is the second input image, used to present the visual effect of the reference design.

**Your Core Task & Scoring Guide**
Please carefully observe and compare the first image (the draft) with the second image (the high-quality reference version), while also considering the requirements of the original design prompt. Output a detailed review report in JSON format. Your sole task is to generate this report; do not generate any SVG code.

**Important Scoring Instructions**
- When the draft is highly similar to the reference version and meets the requirements of the original design prompt, please give a high score (e.g., 9.0-10.0). In the suggestions, do not provide modification guidance, only an affirmative output.
- When there are significant differences between the draft and the reference version, or deviations from the original design prompt, please give a reasonable mid-to-low score based on the severity of the differences and the visual quality. Provide 2 to 4 specific, actionable SVG modification suggestions.

**JSON Output Format**
The JSON object must contain the following three fields:
- `"score"`: A float, ranging from 0.0 to 10.0, for the overall rating of the draft.
- `"critique"`: A comprehensive evaluation explaining the draft's adherence to the original design prompt and pointing out the main differences and shortcomings in aesthetics, color, and geometric construction compared to the reference version.
- `"suggestions"`: If the draft is very close to the reference version and aligns with the original prompt, use an affirmative statement such as "The overall quality is excellent, no changes needed." Otherwise, provide specific SVG modification suggestions.

*Goal: Create robust SVG representations for research purposes.*

Figure 6. Prompt template for constructing the Critique Dataset.

## 9.1. SFT Dataset Construction

The SFT dataset $D_{\text{SFT}}$ is a mixture of three distinct subsets: $D_{\text{SFT}} = D_G^{\text{direct}} \cup D_G^{\text{correction}} \cup D_C$. Each subset targets a specific capability of the unified model.

### 9.1.1. Foundational Generation Data ($D_G^{\text{direct}}$)

This dataset instills the core capability of translating text to SVG code.

**Source:** The 200k standardized samples from our data cleaning pipeline.

**Structure:** Direct pairs of $(X, Y)$, where $X$ is the descriptive textual prompt and $Y$ is the canonical SVG code.

### 9.1.2. Correction and Critique Data Synthesis

To equip the model with self-correction and self-critique capabilities, we constructed synthetic datasets $D_G^{\text{correction}}$ and $D_C$. The synthesis pipeline is as follows:

- **Draft Generation:** We first trained a temporary model (warm-up) on $D_G^{\text{direct}}$ for one epoch. We then selected 50,000 prompts from the validation set and generated initial SVG drafts using this model. These drafts intentionally contain imperfections (e.g., geometric distortions, color mismatches) typical of early-stage training.
- **Expert Annotation (GPT-4o):** We employed GPT-4o as an "Teacher VLM" to evaluate these drafts. We utilized the structured prompt shown in Figure 6. For each triplet of (Original Prompt, Draft Image, Reference Image), GPT-4o generated a JSON response containing:
  **score**: A quantitative quality assessment (0.0-10.0).
  **critique**: A textual analysis of flaws in geometry and aesthetics.
  **suggestions**: Actionable advice for code modification.
- **Dataset Formulation:** Based on the expert feedback, we constructed the specific training samples:
  **Critique Dataset ($D_C$):** Inputs are the prompt and the rendered draft image; the target output is the expert's JSON critique.
  **Correction Dataset ($D_G^{\text{correction}}$):** Inputs are the complex prompt (containing the original prompt, the flawed draft code, and the expert's critique); the target output is the high-quality ground truth SVG from $D_G^{\text{direct}}$.

## 9.2. DPO Preference Dataset Construction

For the DPO stage, we constructed a preference dataset $D_{\text{pref-G}}$ to optimize the model's first-pass generation quality. This process involves sampling, scoring, and pair selection.

### 9.2.1. Candidate Sampling

We selected a diverse set of 10,000 prompts. Using the converged SFT model ($M_{\text{SFT}}$), We generated $N = 5$ distinct candidate SVGs for each prompt with a temperature of 0.9, resulting in a pool of 50,000 candidate samples.

**Prompt: SVG Quality Scoring**

**Task**
I used this prompt: {`original_prompt`}
Rate the 5 SVG images I'm uploading from 1-100 based on that prompt. Ensure the scores are differentiated.

**Evaluate based on:**
- **Prompt Adherence:** How well the image matches the prompt's elements and mood.
- **Visual Aesthetics:** Color, composition, and visual impact.
- **Execution Quality:** Creativity and technical quality (clean SVG, no flaws).

**Output Format**
Provide only JSON in this exact format. No other text.

```
{
   "image_1_score":  [Score],
   "image_2_score":  [Score],
   "image_3_score":  [Score],
   "image_4_score":  [Score],
   "image_5_score":  [Score]
}
```

Figure 7. Prompt used for scoring generated SVG candidates

### 9.2.2. Automated Scoring

We employed GPT-4o as an automated evaluator to score each candidate. The prompt used for this process is illustrated in Figure 7. The scoring criteria explicitly cover:

- **Prompt Adherence:** Alignment with the user's text instructions.
- **Visual Aesthetics:** Color harmony and geometric balance.
- **Execution Quality:** Syntactic correctness and renderability.

### 9.2.3. Preference Pair Construction

To construct the training triplets $(Prompt, S_w, S_l)$, where $S_w$ is the winning sample and $S_l$ is the losing sample, we applied the following hierarchical rules:

- **Rule 1: Render-Success Priority.** A renderable SVG is strictly preferred over a non-renderable one (e.g., one with syntax errors or invalid paths). If Candidate A renders and Candidate B fails, then $S_w = A, S_l = B$.
- **Rule 2: High-Score Priority.** For two renderable candidates, we compare their GPT-4o scores. To ensure distinct separability and avoid noise from similar-quality samples, we enforced a margin $\delta$. If $Score(A) - Score(B) > \delta$, then $S_w = A, S_l = B$.

This results in the dataset $D_{\text{pref-G}}$, ensuring DPO improves both the model's syntactic robustness and visual aesthetics.

## Prompt and Critique Examples

**Original prompt:**
An umbrella with a yellow and blue color.

**Critique & Refine:**
- **Score:** 4.5
- **Critique:** The draft image significantly deviates from the reference version and the original design prompt. The reference image showcases a smooth, aesthetically pleasing umbrella with a yellow and blue color scheme, whereas the draft image presents a pixelated, simplistic version that lacks the smooth curves.
- **Suggestions:**
  1. Smooth out the umbrella's canopy by using curved paths instead of blocky, pixelated shapes to match the reference's aesthetic.
  2. Adjust the color distribution to include both yellow and blue in a balanced manner, ensuring the yellow is prominent.
  3. Refine the handle to have a smooth, curved appearance rather than a straight, pixelated form.
  4. Ensure the overall proportions and symmetry of the umbrella are consistent with the reference image.

**Original prompt:**
A white and blue cloud sits beneath a two-tone blue square with vibrant yellow and orange forming the background.

**Critique & Refine:**
- **Score:** 6.5
- **Critique:** The draft image captures the essence of the original design prompt by featuring a cloud beneath a two-tone blue square. However, there are notable differences compared to the reference version. The draft uses a more simplified cloud design and lacks the subtle color variations present in the reference.
- **Suggestions:**
  1. Add the small colored squares at the top of the blue square to match the reference design.
  2. Adjust the color tones of the two-tone blue square to be more muted, aligning with the reference.
  3. Introduce subtle color variations in the cloud to enhance its visual depth and match the reference.
  4. Improve the integration of the cloud and square into the background for a more cohesive composition.

**Original prompt:**
A cheerful yellow sun with a smiling face and closed eyes radiates joy and warmth.

**Critique & Refine:**
- **Score:** 6.5
- **Critique:** The draft image captures the essence of a cheerful sun with a smiling face and closed eyes, aligning with the original design prompt. However, there are notable differences compared to the reference version. The draft uses a more geometric and angular style for the sun's rays, whereas the reference has smoother, more rounded rays. The facial features in the draft are simplistic and lack the expressive detail seen in the reference.
- **Suggestions:**
  1. Modify the sun's rays to have a smoother, more rounded appearance to match the softer aesthetic of the reference image.
  2. Enhance the facial features by adding more detail and expression, particularly in the eyes and mouth, to better convey joy and warmth.
  3. Adjust the color palette to increase contrast and vibrancy, ensuring the sun's face stands out more distinctly from its rays.
  4. Consider refining the overall symmetry and balance of the design to align more closely with the reference's harmonious composition.

Figure 8. Examples of critique data generated by GPT-4o

**Training Data Samples**

**User:** Please generate an SVG icon that meets the following description: A brown wallet with a light beige accent and a yellow button represents saving or spending money.

**Assistant:**
```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 200 200">
  <path fill="#9A8984" d="M24 55 L5 41 L5 170 C5 184 16 195 30 195 L195 195..." />
  <path fill="#BCB0AD" d="M165 5 L30 5 C16 5 5 16 5 30 C5 37 9 46 24 55..." />
  <path fill="#FEDEA1" d="M175 125 A15 15 0 1 1 145 125 A15 15 0 1 1 175..." />
  <path fill="#573A32" d="M160 145 C171 145 180 136 180 125 C180 114 171 105..." />
  <path fill="#573A32" d="M190 50 L170 50 L170 10 C170 4 166 0 160 0 L30 0..." />
</svg>
```

**User:** Please analyze all the information provided below and generate a final, high-quality SVG code. The original design goal was: "A brown wallet with a light beige accent and a yellow button represents saving or spending money."; a draft SVG code that needs improvement is as follows:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 200 200">
  <path fill="#E8CD97" d="M166 156 L70 156 C66 156 62 152 62 148 L62 85..." />
  <path fill="#472F17" d="M166 161 L33 161 C25 161 18 154 18 146 L18 53..." />
  <path fill="#472F17" d="M184 121 L126 121 C112 121 101 110 101 96 L101 95..." />
  <path fill="#472F17" d="M117 96 A8 8 0 1 0 133 96 A8 8 0 1 0 117 96 Z" />
  <path fill="#472F17" d="M37 95 L18 95 L18 74 L37 74 L37 95 Z M18 85..." />
</svg>
```

An expert critique of this draft is: "The draft image presents a wallet design that partially adheres to the original prompt... The draft's geometric construction is more abstract and less realistic... The draft lacks the depth and shading..."
Specific modification suggestions: 1. Adjust the wallet's shape... 2. Enhance the shading... 3. Increase the prominence of the button... 4. Consider using a more muted brown tone...

**Assistant:**
```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 200 200">
  <path fill="#9A8984" d="M24 55 L5 41 L5 170 C5 184 16 195 30 195 L195 195..." />
  <path fill="#BCB0AD" d="M165 5 L30 5 C16 5 5 16 5 30 C5 37 9 46 24 55..." />
  <path fill="#FEDEA1" d="M175 125 A15 15 0 1 1 145 125 A15 15 0 1 1 175..." />
  <path fill="#573A32" d="M160 145 C171 145 180 136 180 125 C180 114 171 105..." />
  <path fill="#573A32" d="M190 50 L170 50 L170 10 C170 4 166 0 160 0 L30 0..." />
</svg>
```

**User:** You are a professional SVG design critic. Please analyze the input AI-generated SVG draft <**image**>according to the "Original Design Prompt". Original Design Prompt: "A brown wallet with a light beige accent and a yellow button..." Your task is to output a structured critique report in JSON format (score, critique, suggestions).

**Assistant:**
```
{
  "score": 5.5,
  "critique": "The draft image presents a wallet design that partially adheres to the original prompt. The color scheme in the draft uses a brown and beige palette, which aligns with the prompt, but the execution differs significantly from the reference. The draft's geometric construction is more abstract and less realistic compared to the reference...",
  "suggestions": "1. Adjust the wallet's shape to have rounded corners and a more traditional wallet appearance, similar to the reference. 2. Enhance the shading and depth to give the wallet a more three-dimensional look. 3. Increase the prominence of the button by making it larger and more central... 4. Consider using a more muted brown tone..."
}
```

Figure 9. Training Data Formats for Different Capabilities. Top: Generation data $D_G^{direct}$. Middle: Correction data $D_G^{correction}$ with draft and critique inputs. Bottom: Critique data $D_C$

## 10. Additional Evaluation

In this section, we provide additional experimental analyses to further validate the effectiveness and reliability of the proposed IntroSVG framework. Specifically, we report results on an additional benchmark dataset, analyze the computational efficiency of the iterative generation strategy, and conduct human evaluation to assess perceptual quality.

### 10.1. Evaluation on Additional Benchmarks

To further evaluate the generalization capability of our method and reduce potential concerns regarding training data overlap, we additionally conduct experiments on **MMSVG-Bench**. This benchmark contains 300 GPT-generated prompts designed specifically for evaluating text-to-SVG generation models.

All methods are evaluated in a **zero-shot setting** without additional fine-tuning. As shown in Table 8, IntroSVG achieves the best performance across all metrics, including CLIP-based text-image similarity (CLIP-T2I), aesthetic score, and HPSv1 preference score.

Table 8. Zero-shot Results on MMSVG-Bench

| Method | Avg. Token | CLIP-T2I ↑ | Aes. ↑ | HPSv1 ↑ |
|---|---|---|---|---|
| GPT-4o | 403.93 | 0.2301 | 4.6684 | 0.1835 |
| OmniSVG | 3698.87 | 0.1953 | 4.6554 | 0.1760 |
| SVGen | 1517.45 | 0.2235 | 4.5456 | 0.1879 |
| **IntroSVG (Ours)** | 1981.26 | **0.2456** | **4.8141** | **0.1901** |

**Unified Evaluation Set.** For the main experiments in the paper, we construct a unified evaluation set containing **1,400 samples**. The dataset is stratified according to the training sources used by existing methods (LLM4SVG: 200 samples, OmniSVG: 600 samples, SVGen: 600 samples). All samples are strictly excluded from the training corpus to avoid any potential data overlap.

### 10.2. Efficiency, Latency, and Cost Analysis

We further analyze the computational efficiency of the proposed iterative generation framework. All experiments are conducted on a single NVIDIA H100 GPU using the `lmdeploy` inference engine.

Table 9 reports generation latency, token usage, and generation quality across different inference strategies.

Table 9. Efficiency and Cost Analysis

| Method | FID ↓ | Aes. ↑ | Latency | Total Tokens |
|---|---|---|---|---|
| Qwen3-VL-32B | 38.68 | 4.39 | 6.51 s | 445.75 |
| Iter 0 | 29.76 | 4.83 | 9.12 s | 1875.47 |
| Best-of-4 (Iter 0) | 28.43 | 4.85 | 39.01 s | 8522.64 |
| **IntroSVG (Iter 3)** | **26.18** | **4.89** | 42.31 s | 8840.29 |

The results demonstrate that iterative refinement significantly improves generation quality. Specifically, the FID score improves from 29.76 for the initial draft (Iter 0) to 26.18 after three refinement iterations.

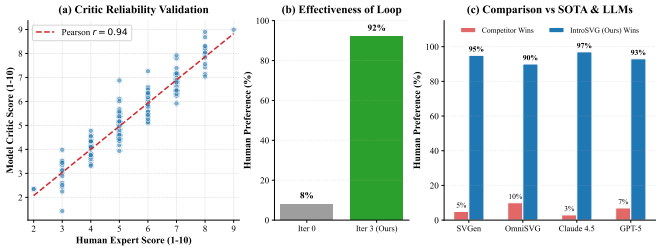



Figure 10. Human evaluation results.

We further compare our method with a **Best-of-4 sampling strategy**, which selects the best result from four independently generated candidates. Although Best-of-4 uses comparable computational resources, it still performs worse than our iterative refinement strategy. This result suggests that allocating computation to structured *introspection and revision* is more effective than relying solely on stochastic sampling.

### 10.3. Human Evaluation

To further assess the perceptual quality of generated SVG graphics, we conduct a human evaluation study involving five professional designers.

We randomly sample **100 prompts** from the evaluation set and generate SVG results using different methods. All samples are evaluated in a **blind evaluation setting**, where annotators are not informed of the model identity.

We conduct human evaluation using two protocols: (1) a 1–10 Likert scale to assess the reliability of Critic scores, and (2) pairwise blind A/B comparisons to evaluate the visual quality of model outputs. The results are analyzed as follows.

**Critic Reliability.** Annotators rate each result using a **1–10 Likert scale** according to visual aesthetics and prompt alignment. The **Pearson correlation** between the Critic scores and human ratings reaches **0.94**, indicating strong agreement between the automated evaluation and human perception.

**Effectiveness of Iterative Refinement.** As shown in Figure 10(b), **92%** of the samples demonstrate that Iteration 3 produces higher-quality results than the initial draft (Iter 0), highlighting the effectiveness of the refinement process.

**Comparison with Baseline Methods.** In pairwise blind comparisons, IntroSVG achieves significantly higher win rates against existing methods, including SVGen (**95%**), OmniSVG (**90%**), GPT-5 (**93%**), and Claude 4.5 (**97%**). These results further confirm the superiority of the proposed iterative generation framework.

## 11. Sample Demonstration

### 11.1. Training Data Examples

We first present samples of the synthetic critique data generated by GPT-4o in Figure 8, which serves as the ground


Figure 11. SVG samples generated by IntroSVG

truth for training the model's self-evaluation capability. Subsequently, Figure 9 illustrates the specific input-output formats constructed based on these data for the Generation, Correction, and Critique tasks.

### 11.2. Generated SVG Results

We display a collection of SVG icons generated by IntroSVG in Figure 11. The samples demonstrate the aesthetic performance of the model in generating complex, multicolored SVG icons.