

Seeing is Improving: Visual Feedback for Iterative Text Layout Refinement

Junrong Guo¹, Shancheng Fang^{2*}, Yadong Qu¹, Hongtao Xie¹

¹University of Science and Technology of China ²Shenzhen University

godrong@mail.ustc.edu.cn, fangsc@szu.edu.cn, qqyqyd@mail.ustc.edu.cn, htjie@ustc.edu.cn

Abstract

Recent advances in Multimodal Large Language Models (MLLMs) have enabled automated generation of structured layouts from natural language descriptions. Existing methods typically follow a code-only paradigm that generates code to represent layouts, which are then rendered by graphic engines to produce final images. However, they are blind to the rendered visual outcome, making it difficult to guarantee readability and aesthetics. In this paper, we identify visual feedback as a critical factor in layout generation and propose Visual Feedback Layout Model (VFLM), a self-improving framework that leverages visual feedback iterative refinement. VFLM is capable of performing adaptive reflective generation, which leverages visual information to reflect on previous issues and iteratively generates outputs until satisfactory quality is achieved. It is achieved through reinforcement learning with a visually grounded reward model that incorporates OCR accuracy. By rewarding only the final generated outcome, we can effectively stimulate the model’s iterative and reflective generative capabilities. Experiments across multiple benchmarks show that VFLM consistently outperforms advanced MLLMs, existing layout models, and code-only baselines, establishing visual feedback as critical for design-oriented MLLMs. Our code and data are available at <https://github.com/FoIspark/VFLM>.

1. Introduction

The emergence of Large Language Models (LLMs) [1, 9, 51] and Multimodal Large Language Models (MLLMs) [4, 18, 32, 42] has unlocked new possibilities for automated content generation, particularly for structured visual layouts. By generating structured representations (e.g., SVG code, custom JSON) [8, 11, 35, 52] that define the position, size, and style of each element [20, 22], LLMs can translate natural language descriptions directly into complex designs such as typographic posters, social media graphics,

*Corresponding author.

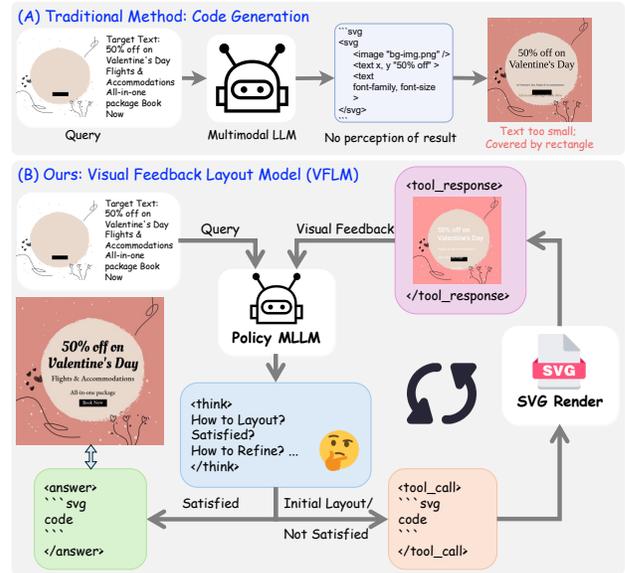


Figure 1. Method comparison. Traditional methods only generate through text code and have no perception of the generated results; our Visual Feedback method can iteratively reflect on the generated result and optimize the layout effect.

and documents. MLLMs further enhance this by leveraging cross-modal understanding, allowing generation to be conditioned on both text and visual inputs.

However, existing methods are limited to the code-only paradigm and can only generate codes for layout without considering visual feedback. For instance, COLE [22] and OpenCOLE [20] leverage LLMs to generate typography JSON files, while LGGPT [57] produces customized layout output formats, which are then composed into the final images by a graphic renderer. As shown in Fig. 1(A), while these models can produce layout structures that meet formal specifications, they remain unable to perceive the visual outcomes of their designs. Effective text layout design, however, depends on visual properties such as aesthetic balance, readability, and image–text coherence, which cannot be fully encoded by programmatic rules. Consequently, a model may generate syntactically valid SVG layouts that still suffer from overlapping elements, poor contrast, or mis-

aligned components.

Recent advances in LLMs [9, 12, 21, 32] research demonstrate that reflection, backtracking, and self-validation mechanisms can substantially improve performance on complex reasoning tasks. Moreover, Reinforcement Learning (RL) [33, 38, 40] techniques have proven effective in activating the reflective reasoning capabilities of LLMs. This motivates our core research question: Can such reflection capabilities be transferred to text layout generation to overcome the visual perception gap in existing approaches? We argue that the solution lies in incorporating **visual feedback** into the text layout generation process, leveraging MLLMs’ inherent cross-modal understanding capabilities. As illustrated in Fig. 1, models should not only generate layout code but also perceive the rendered results to evaluate quality, diagnose visual issues, and devise optimization strategies through iterative refinement.

In this paper, we propose **Visual Feedback Layout Model (VFLM)**, a novel self-improving framework for text layout generation that establishes a closed-loop process guided by RL. As shown in Fig. 1, VFLM first generates initial SVG layout code, which is rendered into a visual image. This rendered image is fed back to the same model for visual inspection and reflection. If issues are identified or the layout is unsatisfactory, VFLM generates revised code and repeats the process, creating a continuous loop of “*generation, rendering, reflection and refinement*” until a satisfactory layout is achieved.

VFLM utilizes a two-stage training pipeline. Firstly, we construct a multi-stage generation-reflection-refinement trajectory dataset by distilling advanced MLLMs, followed by Supervised Fine-Tuning (SFT) to endow model the ability of iterative generation. Secondly, we employ RL to enhance the model’s reflective capabilities. We utilize a reward model trained to evaluate layout quality holistically and incorporate text accuracy via Optical Character Recognition (OCR). While recent research on Agentic RL research [10, 25, 41] often relies on meticulously designed, complex, process-oriented reward mechanisms to guide each intermediate step, our analysis reveals this approach can lead to “reward hacking”. We demonstrate, counter-intuitively, that a simple outcome-based reward, which only assesses the final generated layout, is significantly more effective. This simpler signal compels the VFLM to leverage its inherent visual understanding to holistically balance initial generation quality and iterative refinement, enabling robust and stable performance improvements through visual feedback.

We validate this framework through extensive experiments based on Qwen2.5-VL-7B model [4] for the task of laying out target text on background images. Both quantitative and qualitative evaluations show VFLM significantly outperforms state-of-the-art layout generation approaches,

advanced MLLMs, and image editing models. Furthermore, our ablation studies, which provide fair comparisons against multiple code-only generation methods, confirm the significant and critical advantage of our VFLM. This superiority effectively validates visual feedback as a crucial component in generative text layout, establishing a practical framework for developing self-improving, MLLM-based design agents.

This work’s contributions can be summarized as follows:

- We propose VFLM, a self-improving framework that, to our knowledge, is the first to apply visual feedback to layout generation. It equips MLLMs with a “generation, rendering, reflection, refinement” cycle, using visual feedback to overcome the limitations of code-only methods.
- We design a two-stage SFT+RL training method, including a novel reward model for text layout, that successfully activates the model’s iterative refinement capabilities. Extensive experiments validate that our approach consistently outperforms strong existing methods.
- We demonstrate a key finding that simple, outcome-based rewards are more effective and robust for activating the model’s self-improvement capabilities than complex, process-oriented reward functions.

2. Related Work

2.1. Multimodal Large Language Model

Recent progress in Multimodal Large Language Models (MLLMs) is driven by integrating pretrained vision encoders [36, 54] with LLMs. The two modalities are typically aligned via lightweight projectors or Q-Former [24] structures, a paradigm that has spurred a suite of powerful models. This includes open-source series like LLaVA [29–31], Qwen-VL [3, 4, 45], and Intern-VL [7, 13, 46, 61], as well as large-scale proprietary systems such as GPT-4o [18], Gemini [43], and Claude [2], which continue to advance the state of the art through massive scaling and enhanced reasoning techniques [47, 58].

2.2. Graphic Layout Generation

Graphic layout generation has evolved from early Transformer-based architectures [27, 60] to methods centered on LLMs and MLLMs. Many approaches prompts LLMs to output structured formats like SVG [44], HTML [39], or JSON [6, 15, 20, 22, 26]. Others employ MLLMs to decompose the design process into ordered layers or sub-tasks [8, 22, 28, 35], often coordinating with multi-modal inputs [19, 52]. However, these methods operate in an open-loop, lacking visual perception of the rendered output. While some recent pipelines incorporate visual feedback, they typically rely on non-trainable, inference-only strategies with external advisors [53, 55], or decouple generation and reflection into separately trained

models [22]. In contrast, our work introduces visual feedback via RL, unifying these capabilities into a single, self-evolving policy with learnable autonomous judgment.

2.3. Reinforcement Learning

Reinforcement learning is a cornerstone for aligning LLMs with human preferences, standardized by the RLHF [33] pipeline which typically uses Proximal Policy Optimization (PPO) [38]. To mitigate the instability and high cost associated with PPO, recent alternatives like Direct Preference Optimization (DPO) [37] and Group Relative Policy Optimization (GRPO) [40] offer more direct and efficient optimization strategies. This alignment paradigm extends naturally to multimodal settings to improve visual grounding and reasoning. Works such as Vision-R1 [17], R1VL [56], and DeepEyes [59] adapt RL to MLLMs by incorporating multimodal rewards, chain-of-thought signals, and specialized replay mechanisms, demonstrating the power of RL in enhancing multimodal alignment and capability.

3. Method

This work aims to develop a self-improving VFLM for text layout generation that optimizes outputs through visual feedback. We achieve this with a two-stage training framework: (1) Cold-Start SFT to instill basic iterative generation and reflection capabilities, and (2) Reinforcement Learning to enhance performance using vision-based reward signals.

3.1. Visual Feedback Layout Model

As illustrated in Fig. 1, VFLM establishes a multi-round interaction mechanism between VFLM and a rendering environment. As shown in Algorithm 1, given a background image and target text, VFLM executes an iterative *generation, rendering, reflection, refinement* cycle:

1. **Initial Generation:** The VFLM first analyzes the input through reasoning, then generates initial layout code through a structured tool call.
2. **Rendering:** The rendering tool converts the SVG code into a visual image and feeds it back to the same VFLM.
3. **Visual Reflection:** The VFLM examines the rendered layout visual image through reasoning to evaluate whether the quality is satisfactory to it.
4. **Iterative Refinement:** If unsatisfied, the VFLM reasons about necessary modifications and a revised layout code is generated. The process repeats until the model determines satisfaction with the layout quality.

3.2. Cold-Start Supervised Fine-Tuning

The cold-start SFT stage enables the model to acquire iterative reflection capabilities, and tool usage specifications through distillation from an advanced teacher MLLM.

Data Construction Due to the absence of natural multi-round reflection data, we employ Doubao-Seed-1.6 [5] as a

Algorithm 1 Visual Feedback for Text Layout Self-Improvement

Require: Background Image I_b , Target Text T , max iterations N_{\max}

Ensure: Final layout code S

- 1: $S \leftarrow$ VFLM generates initial layout via tool call based on I_b and T
 - 2: $I_{\text{rendered}} \leftarrow$ Render(S) ▷ Tool Call
 - 3: **for** $i = 1 \rightarrow N_{\max}$ **do**
 - 4: Reflection \leftarrow VFLM(reason| I_{rendered})
 - 5: **if** Reflection indicates satisfaction **then**
 - 6: **return** S as the final result
 - 7: **end if**
 - 8: $S \leftarrow$ VFLM generates revised layout
 - 9: $I_{\text{rendered}} \leftarrow$ Render(S) ▷ Tool Call
 - 10: **end for**
 - 11: **return** S
-

teacher model for data synthesis. As illustrated in Fig. 2(A), the synthesis pipeline is divided into the following stages:

1. **Initial reasoning synthesis:** First, The teacher model is prompted with background images and ground-truth layouts to generate the reasoning processes for SVG code generation.
2. **Suboptimal layout generation:** This distilled reasoning data is used to fine-tune Qwen2.5-VL-7B. The inference outputs from this model are then collected to serve as the suboptimal initial attempts for the next stage.
3. **Multi-round reflection synthesis:** The suboptimal attempts (from Stage 2) are paired with their ground-truth layouts and fed back to the teacher model. The teacher is then instructed to perform iterative reflection and modification to reach the ground-truth solution. This process simulates realistic human design refinement and generates complete multi-round reflection trajectories.
4. **Data combination:** Finally, the synthesized data from the above stages are combined and organized using structured tags: intermediate rounds use `<think>` and `<tool_call>`, while the final round uses `<think>` and `<answer>` (See supplementary materials for specific data samples.).

Training Objective We fine-tune Qwen2.5-VL-7B using causal language modeling on the synthesized multi-round reflection trajectories data. To prevent the model from learning suboptimal outputs, the loss for initial responses in improvement sequences is masked, ensuring the model learns from the correction process rather than initial errors.

3.3. Iterative Reflection Reinforcement Learning

3.3.1. RL Algorithm

We adopt the Group Relative Policy Optimization (GRPO) [40] algorithm for RL and make certain improve-

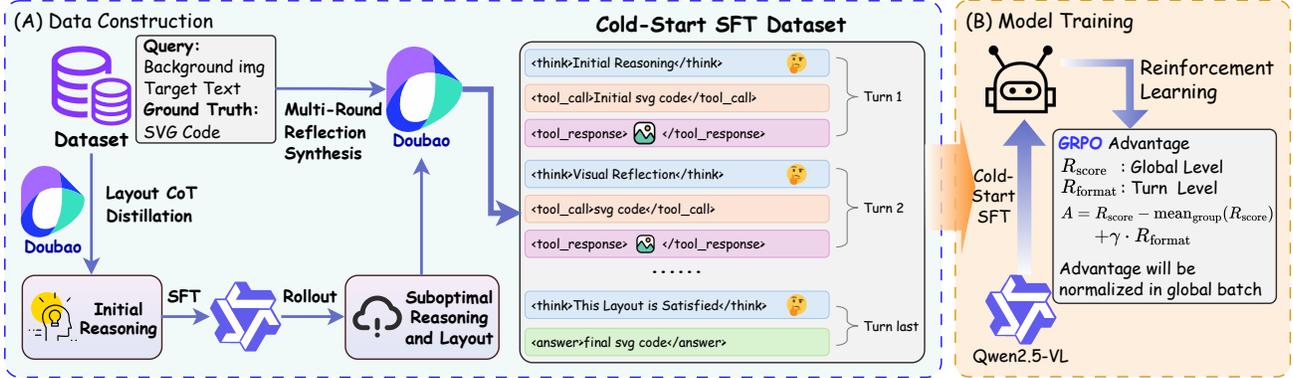


Figure 2. Pipeline of Visual Feedback Method, including Data Construction for Cold-Start SFT and RL with Modified Advantages.

ments to the advantage function. Compared with traditional policy optimization methods, GRPO performs policy gradient optimization within sample groups, enabling the model to learn in the direction of maximizing rewards. The optimization objectives of GRPO are as follows:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(O | q)] \\ \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_{\theta}(o_{i,t} | q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | q, o_{i,<t})} A_{i,t}, \right. \right. \\ \left. \left. \text{clip} \left(\frac{\pi_{\theta}(o_{i,t} | q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | q, o_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) A_{i,t} \right] - \beta \mathbb{D}_{\text{KL}}[\pi_{\theta} \| \pi_{\text{ref}}] \right\}, \quad (1)$$

where ϵ and β are the clipping hyperparameters and the KL divergence penalty coefficient, respectively.

For the reward function, we design a three-component reward function to score the layout effect: (1) R_{layout} (Sec. 3.3.2), a specialized reward model trained to evaluate overall layout quality; (2) R_{ocr} , which evaluates text accuracy based on OCR results from the rendered layout; and (3) R_{svg} , which measures text accuracy at the code level by comparing the extracted SVG strings with the target text. The total reward is the weighted sum of the three components:

$$R_{\text{score}} = R_{\text{layout}} + \alpha \cdot (R_{\text{ocr}} + R_{\text{svg}}), \quad (2)$$

where α balance aesthetic quality against functional accuracy. In addition, we incorporate a format reward to constrain the output format of the model:

$$R_{\text{format}} = \begin{cases} 1.0, & \text{if format is correct,} \\ -1.0, & \text{if format is incorrect.} \end{cases} \quad (3)$$

For the advantage function, since the multi-round nature of our approach, the format rewards are applied separately in each round, resulting in inconsistent rewards between rounds for a sequence. Referring to REINFORCE++ [16], we use the mean value of R_{score} within a group as baseline to reshape the reward. The format rewards are then incorporated with a scaling factor γ . Finally, the advantages in the global batch are normalized and used for GRPO training:

$$A_{\text{raw}} = R_{\text{score}} - \text{mean}_{\text{group}}(R_{\text{score}}) + \gamma \cdot R_{\text{format}}, \quad (4)$$

$$A = \frac{A_{\text{raw}} - \text{mean}_{\text{batch}}(A_{\text{raw}})}{\text{std}_{\text{batch}}(A_{\text{raw}})}. \quad (5)$$

3.3.2. Layout Reward Model

Training a robust reward model for text layout is uniquely challenging. Unlike tasks with binary correctness, layout quality is a holistic and fine-grained judgment of aesthetics, readability, and coherence. A high-quality preference dataset is crucial for training a model that can guide a stable RL without falling into reward hacking. However, no existing datasets or methodologies are specifically designed for layout generation reward modeling. To address this gap, we train a specialized reward model r_{θ} that takes triplets (B, T, I) as input and outputs a scalar score R_{layout} , where B denotes the background image, T denotes the target text, and I denotes the rendered layout image.

Dataset: To create the necessary preference data, we introduce a novel hierarchical data construction method that creates fine-grained quality distinctions across multiple layout quality levels.

To equip the reward model with comprehensive and robust evaluation capabilities, we construct four distinct quality levels to capture fine-grained layout performance:

- **Level-I:** High-quality ground-truth layouts serving as gold standards for design excellence.
- **Level-II:** Layouts generated by Qwen2.5-VL-7B after fine-tuning on 200K samples exhibit reasonable quality with minor imperfections.
- **Level-III:** Layouts from the Level-II model subjected to systematic spatial perturbations applied to layout elements, including random positional offsets that moderately compromise design coherence.
- **Level-IV:** Severely degraded layouts stem from the Level-II model under aggressive perturbations, including extensive positional displacement, random font size variations, selective text element deletion, image reference removal, and arbitrary SVG scaling transformations.

This hierarchical construction enables comprehensive preference learning through systematic pairwise comparisons. For each layout generation prompt, we create layouts at all four quality levels, then form all possible pairwise comparisons between different levels. This yields 6 preference pairs per problem, establishing clear quality orderings that capture nuanced distinctions essential for effective reward model training. This strategy forces the reward model to move beyond simple binary (good/bad) judgments and learn the subtle distinctions that separate excellent layouts from merely acceptable ones. The resulting dataset provides a comprehensive and reliable basis for training a highly discerning layout reward model r_θ .

Training: The method proposed in [34] is adopted for training the reward model. Specifically, the reward model is initialized with Qwen2.5-VL-3B. To adapt it for preference learning, the final layer is replaced with a linear layer yielding a scalar output. Subsequently, the reward model is trained via the negative log-likelihood loss function:

$$\mathcal{L}_{\text{RM}}(\theta) = -\mathbb{E}_{(B,T,I^+,I^-)\sim\mathcal{D}} [\log \sigma(r_\theta(B,T,I^+) - r_\theta(B,T,I^-))], \quad (6)$$

where I^+ and I^- denote the better-quality and lower-quality images, respectively, in a pairwise comparison.

To provide stable reward values for the RL process, the raw output of r_θ is normalized before being used as the final layout reward R_{layout} . Following the practice in [49], we standardize r_θ scores using the mean and standard deviation of the test set distribution. This procedure ensures that the reward signal maintains a consistent scale throughout training, which is crucial for effective policy optimization. The final reward is calculated as:

$$R_{\text{layout}} = \frac{r_\theta - \text{mean}_{\text{test}}(r_\theta)}{\text{std}_{\text{test}}(r_\theta)}. \quad (7)$$

4. Experiment

4.1. Experimental Setup

Evaluation Datasets For evaluation, we randomly sample 1K examples from our TextLayout dataset as the primary test set, ensuring no overlap with training data. We also conduct additional experiments on the Crello [50] and DE-SIGNERINTENTION [22] benchmarks, preprocessed into background-text pairs.

Evaluation metrics We adopt three groups of evaluation metrics. Text accuracy is measured using character-level F-measure based on OCR recognition. For graphic metrics, we adopt R_{ali} , R_{ove} , and R_{com} [35, 60], which capture text position alignment, text-text overlap, and pixel gradient smoothness within text regions. Additionally, we employ GPT-4o as a judge [14] to evaluate four dimensions: Text Accuracy, Text-Background Harmony, Text Presentation Quality, and Meaning Expression Adaptability. Refer to the supplementary materials for details.

Existing Methods We compare against three categories of existing methods: (1) Advanced MLLMs: GPT-4o [18], Claude 3.7 [2], Doubao-Seed-1.6[5], and Qwen2.5-VL-72B [4]; (2) Image editing models: GPT-4o(edit) [18], Qwen-Image-Edit [48], and FLUX-Kontext [23]; (3) Specialized layout generation: the open-source domain-specific model OpenCOLE [20] and IGD [35].

4.2. Comparison with Existing Methods

As shown in Tab. 1, VFLM demonstrates a clear quantitative advantage. On the TextLayout dataset, VFLM’s initial generation (VFLM-step1) already achieves a higher OCR F1-score (0.9071) than the strongest MLLM competitor, Claude 3.7 (0.8672), and the specialized layout model IGD (0.8481). The final optimized output (VFLM) widens this gap further to 0.9376. This superiority extends across key graphic metrics: VFLM excels at minimizing element overlap (R_{ove}) and optimizing composition (R_{com}), achieving the highest RM scores and overall GPT-4o evaluation. This strong performance generalizes robustly: VFLM consistently achieves state-of-the-art results on the Crello and DE-SIGNERINTENTION datasets, establishing a new SOTA on nearly all OCR and Graphic metrics.

Fig. 3 provides a qualitative comparison of VFLM against existing methods. Models like IGD and Claude 3.7, which generate layouts directly, inevitably suffer from visual artifacts such as text overlap and color conflicts. Meanwhile, image editing models like GPT4o-Image fundamentally conflict with our task, as they inevitably alter the background image. These editing models also struggle to guarantee quality in scenarios with dense text or Chinese characters. VFLM, in contrast, successfully integrates the element-wise consistency of code generation while further leveraging visual feedback to promptly discover and correct visual problems. This ensures a high-quality final output, overcoming the limitations of both competing approaches.

4.3. Layout Reward Model Evaluation

The trained reward model achieves a high pairwise prediction accuracy of 97.4% on the preference data test set. To further validate our methodology, we conducted two key analyses presented in Tab. 2. First, we verify the integrity of our four-level data hierarchy using objective metrics. As shown, the external Graphic and OCR metrics exhibit a clear monotonic degradation from Level-I to Level-IV. Among them, R_{ove} and R_{com} have slight improvement in Level-III and Level-IV because many texts in these two types of data have been lost in the layout, so text overlap and regional gradients will naturally decrease. This result confirms that our data construction process successfully creates a well-defined and reliable quality gradient.

Second, we evaluated whether our trained reward model internalizes this quality structure. The final column of the

Table 1. The Graphic quality metrics and OCR metrics on the TextLayout, Crello and DESIGNERINTENTION test set, where VFLM-step1 and VFLM respectively represent the metrics of our results in the first output and the final result output after iterative reflection.

Method	Model	OCR		Graphic			RM Score \uparrow	Text	Harmony	GPT-4o		Overall
		Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	Quality				Meaning		
TextLayout												
MLLM	GPT-4o	0.8258	0.0046	<u>0.0033</u>	18.8443	0.3561	8.5165	8.0341	7.2826	7.6553	7.8721	
	Claude3.7	0.8672	0.0053	0.0383	16.4401	0.5295	8.8058	8.4026	7.7691	8.2651	8.3106	
	Doubao-Seed-1.6	0.8484	0.0058	0.0216	18.5823	0.4063	8.5663	8.2304	7.4463	7.8917	8.0337	
	Qwen2.5-VL-72B	0.7178	0.0031	0.0358	19.9399	0.1989	7.8633	7.6094	6.5402	6.5582	7.1428	
	Qwen2.5-VL-7B	0.6195	0.0029	0.0229	25.7715	0.1166	7.5638	6.4431	5.4827	5.3861	6.5489	
Image Edit	GPT-4o	0.7198	-	-	-	0.3371	7.4809	8.9298	8.1672	8.1919	8.1924	
	Qwen-Image-Edit	0.7256	-	-	-	0.3062	5.4086	7.7783	6.1886	6.0040	6.3449	
	FLUX Kontext	0.1322	-	-	-	-0.5853	1.6473	6.8098	3.6345	2.2142	3.5765	
Layout	OpenCOLE	0.2147	1.2029	0.0316	25.1150	0.0397	2.6596	6.3873	3.7020	2.8896	3.9096	
	IGD	0.8481	0.0057	0.0114	14.8597	0.3646	8.4324	7.8125	7.3055	7.6894	7.8100	
Ours	VFLM-step1	<u>0.9071</u>	0.0035	0.0059	15.4583	<u>0.5415</u>	8.8880	8.3591	7.7255	8.2896	8.3155	
	VFLM	0.9376	0.0039	0.0009	11.8678	0.6018	9.0447	<u>8.7492</u>	<u>7.9679</u>	8.5969	8.5897	
	Δ (vs step1)	+0.0305	-0.0004	+0.005	+3.5905	+0.0603	+0.1567	+0.3901	+0.2424	+0.3073	+0.2742	
Crello												
MLLM	GPT-4o	0.9101	0.0017	0.0036	19.9216	0.4427	8.9612	7.9949	7.5237	7.9618	8.1104	
	Claude3.7	0.8610	0.0058	0.0205	21.0798	0.5880	8.9645	8.3805	<u>7.8999</u>	<u>8.4507</u>	<u>8.4239</u>	
	Doubao-Seed-1.6	0.8758	0.0051	0.0192	21.5918	0.4876	8.8927	8.2174	7.6658	8.2569	8.2582	
	Qwen2.5-VL-72B	0.8115	0.0044	0.0619	24.6138	0.2359	7.7332	7.4732	6.5589	6.5405	7.0764	
	Qwen2.5-VL-7B	0.7620	0.0028	0.0256	30.9031	0.2116	7.6289	6.2840	5.5726	5.5813	6.2667	
Image Edit	GPT-4o	0.8774	-	-	-	0.5849	8.8438	9.0127	8.8051	9.0069	8.9171	
	Qwen-Image-Edit	0.8326	-	-	-	0.3671	5.7509	7.2808	6.0635	5.9903	6.2714	
	FLUX Kontext	0.5558	-	-	-	-0.0834	3.2447	6.6992	4.5005	4.1499	4.6486	
Layout	OpenCOLE	0.6411	1.1902	0.0429	31.5831	0.3345	6.9777	7.3864	6.4068	6.5319	6.8257	
	IGD	0.8853	0.0128	0.0182	20.2092	0.3959	8.3001	7.5402	6.9126	7.4004	7.5383	
Ours	VFLM-step1	0.8774	0.0046	0.0061	19.5917	0.4392	8.2334	7.9055	7.2046	7.6543	7.7494	
	VFLM	0.9256	0.0025	0.0022	14.8063	0.5548	8.8260	<u>8.5957</u>	7.7267	8.2964	8.3562	
	Δ (vs step1)	+0.0482	+0.0021	+0.0039	+4.7854	+0.1156	+0.5926	+0.6902	+0.5221	+0.6421	+0.6068	
DESIGNERINTENTION												
MLLM	GPT-4o	0.9563	0.0015	0.0140	18.7249	0.5072	9.5180	8.2725	8.0200	8.4820	8.5731	
	Claude3.7	0.8753	0.0070	0.0601	15.0869	0.6465	9.5815	8.5473	8.2052	8.8229	8.7892	
	Doubao-Seed-1.6	0.8917	0.0046	0.0222	15.7470	0.5103	9.1440	8.0020	7.7400	8.2960	8.2955	
	Qwen2.5-VL-72B	0.5475	0.0069	0.1649	17.8489	0.3104	4.1891	7.7565	6.8813	4.4970	5.8310	
	Qwen2.5-VL-7B	0.6694	0.0011	0.0689	19.1119	0.2018	7.3688	6.1285	5.6576	5.4498	6.1512	
Image Edit	GPT-4o	0.8832	-	-	-	0.6278	<u>9.5231</u>	8.9287	9.0126	9.2558	9.1800	
	Qwen-Image-Edit	0.8518	-	-	-	0.5995	7.1626	7.7435	7.2625	7.1403	7.3272	
	FLUX Kontext	0.5412	-	-	-	0.4426	4.3908	7.2806	5.4208	5.1503	5.5606	
Layout	OpenCOLE	0.7308	0.6848	0.0408	20.2853	0.4684	8.4738	7.3488	6.9516	7.4435	7.5544	
	IGD	0.9283	0.0115	0.0065	14.5905	0.5349	8.8112	7.7505	7.4449	7.8677	7.9686	
Ours	VFLM-step1	0.9415	0.0033	0.0023	14.1230	0.5285	9.4020	8.1924	7.8640	8.3260	8.4461	
	VFLM	0.9663	0.0024	0.0008	12.1167	0.5688	9.5569	8.4511	7.9301	8.5130	8.6128	
	Δ (vs step1)	+0.0248	+0.0009	+0.0015	+2.0063	+0.0403	+0.1549	+0.2587	+0.0661	+0.1870	+0.1667	

Table 2. Metrics and RM score across four quality-level datasets.

level	OCR		Graphic		RM Score \uparrow
	Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	
Level-I	0.9474	0.0089	0.0038	6.6795	1.0594
Level-II	0.9049	0.0112	0.0134	12.2626	0.6343
Level-III	0.5657	0.0354	0.0444	21.2628	-0.2345
Level-IV	0.5324	0.0536	0.0375	17.5270	-1.3457

table reports the average RM Score, i.e., R_{layout} , which aligns well with the established hierarchy, decreasing consistently from a high for Level-I to a low for Level-IV. This strong discriminative performance across distinct quality levels confirms that our model has learned a nuanced understanding of layout quality, enabling it to provide a reliable and effective supervision signal for reinforcement learning.

4.4. Ablation Study

In the ablation study, we conducted comparative experiments using TextLayout test set. To comprehensively compare the effects of our visual feedback method, we compared it against several training approaches: (1) **Cold-Start Model**: The cold start model mainly ensures the format of the iterative output and does not significantly improve the layout quality; (2) **Single-Round RL**: We trained the cold-start model using RL but restricted generation to only one step, enabling fair comparison between single-round generation and iterative visual reflection; (3) **RL from Pre-trained Models**: Direct RL training from the pre-trained Qwen2.5-VL-7B model without SFT initialization; (4) **Direct Output**: SFT+RL training was performed using the



Figure 3. In comparison with existing methods, we selected one model from each category of methods as a representative. For a more comprehensive comparison, please refer to the supplementary materials.

same source data as our visual feedback method for direct SVG code generation; (5) **Direct Output SFT**: For fair comparison with our 40K-sample SFT+RL approach, we trained a direct SFT model on 40K samples. Since these models only require a single round of generation training, we use the standard SFT loss and GRPO Advantage calculation method. For more ablation experiments on reward functions and advantage algorithms, as well as training details, please refer to the supplementary materials.

Table 3. Ablation experiments on Graphic quality metrics and OCR metrics on TextLayout test set.

Model	OCR	Graphic			RM Score \uparrow
	Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	
VFLM-step1	0.9071	0.0035	0.0059	15.4583	0.5415
VFLM	0.9376	0.0039	0.0009	11.8678	0.6018
Cold-Start-step1	0.7913	0.0078	0.0297	19.0560	0.2572
Cold-Start	0.7980	0.0081	0.0298	19.0577	0.2608
Single-Round RL	0.8792	0.0024	0.0053	18.9428	0.4063
RL from Pretrained	0.8154	0.0003	6.7109	22.0273	0.2971
Direct Output	0.9237	0.0027	0.0021	17.0654	0.4964
Direct Output SFT	0.8551	0.0040	0.0153	12.9459	0.5332

The results of our ablation study, presented in Tab. 3,

demonstrate the clear superiority of our Visual Feedback method. Our VFLM achieves the best performance across the majority of metrics, substantially outperforming all RL baselines. More notably, the quality of our initial output (VFLM-step1) is already not inferior to any other competing methods. For instance, its RM score of 0.5415 even surpasses the second-best performing Direct Output SFT (0.5332). Subsequent iterative steps further widened this performance gap, increasing the RM score to 0.6018, and achieving top-notch results in both OCR and image quality. Our success highlights that our visual feedback framework is a more effective solution for layout generation, as it can establish a higher quality benchmark from the very first step and then optimize it to the state-of-the-art level.

4.5. Discussion: Effectiveness of Simple Rewards

Can simple outcome-based rewards effectively stimulate self-improvement capabilities in MLLMs? Our empirical investigation provides compelling evidence that they can, and even outperform more sophisticated alternatives.

We designed a sophisticated process-oriented reward function (Eq. (8)–Eq. (10)). This complex reward scheme

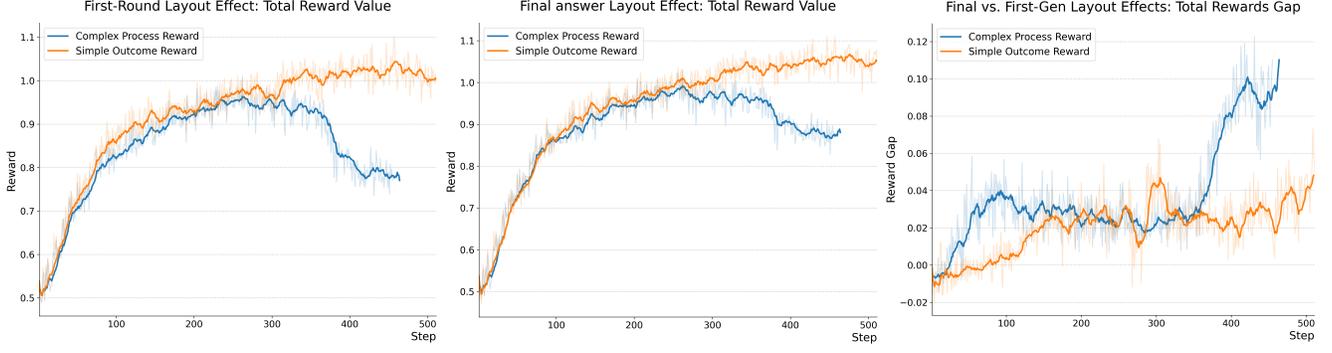


Figure 4. Comparison of training processes: simple outcome rewards vs. complex process rewards.

incorporates three distinct optimization objectives: (1) first-round quality maximization using group-wise mean baselines, (2) iterative improvement encouragement via maximum-quality baselines from prior rounds, and (3) strategic termination control via reward and length penalties to prevent reward hacking. Subsequent advantages will be normalized through Eq. (5).

$$A_{q,o_{t_i}} = \begin{cases} R_{q,o_{t_1}} - \text{mean}_{\text{group}}(R_{q,o_{t_1}}) & \text{if } i = 1, \\ 2 \cdot (R_{\text{answer}} + R_{\text{length}}) & \text{if } i = \text{last}, \\ 2 \cdot (R_{q,o_{t_i}} - \max(R_{q,o_{t_{\leq i}}})) & \text{else,} \end{cases} \quad (8)$$

where o_t represents a complete generation path, and o_{t_i} represents the response of the i -th round in this complete path. The terminal reward ($i = \text{last}$) components are defined as:

$$R_{\text{answer}} = \begin{cases} 0.7 & \text{if } R_{q,o_{t_{\text{last}}}} \geq \max(R_{q,o_{t_{\leq \text{last}}}}), \\ R_{q,o_{t_{\text{last}}}} - \max(R_{q,o_{t_{\leq \text{last}}}}) & \text{else,} \end{cases} \quad (9)$$

$$R_{\text{length}} = -2 \cdot \left[\left(R_{q,o_{t_{\text{last}}}} - \max_{\text{group}}(R_{q,o_{t_{\text{last}}}}) \right) \cdot \max(0, 4 - \text{tool_call_count}) \right]. \quad (10)$$

Fig. 4 illustrates the training dynamics of the two reward schemes. Before 250 steps, both algorithms improved stably: their final answer scores (middle subfigure) showed nearly identical trends, and first-round generation performance was comparable, the outcome-only reward algorithm even slightly outperform the complex process RL one. After 250 steps, the process RL converged and even suffered performance degradation, whereas the outcome-only reward continued to improve steadily until training concluded. The reward gap (right subfigure) reveals that process RL quickly widened this gap but plateaued later, which is likely due to restricted first-round learning in early training. In contrast, the outcome-only reward gradually mastered progressive iterative refinement. Tab. 4 quantifies these observations through comprehensive evaluation metrics. The simple outcome-based reward demonstrates superior effective-

ness in stimulating self-improvement capabilities across all evaluation dimensions.

Table 4. Comparison of simple outcome rewards and complex process rewards on our test set.

level	OCR	Graphic			RM Score \uparrow
	Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	
Outcome RL-step1	0.9071	0.0035	0.0059	15.4583	0.5415
Outcome RL	0.9376	0.0039	0.0009	11.8678	0.6018
Δ (vs step1)	+0.0305	-0.0004	+0.005	+3.5905	+0.0603
Process RL-step1	0.9032	0.0053	0.0030	16.6528	0.4936
Process RL	<u>0.9239</u>	0.0052	<u>0.0027</u>	16.4220	0.5241
Δ (vs step1)	+0.0207	+0.0001	+0.0003	+0.2308	+0.0305

These findings reveal a fundamental insight: under effective visual feedback mechanisms, simple outcome-based rewards can successfully harness the inherent visual understanding capabilities of MLLMs to elicit robust self-improvement behaviors, while complex process-oriented rewards may actually inhibit optimal performance, leading to the brittle local optima and “reward hacking” (policy collapse) observed in Fig. 4. This counterintuitive result suggests that the powerful internal representations and reasoning capabilities of modern MLLMs, when properly guided by clear outcome objectives and visual feedback, can autonomously develop sophisticated iterative refinement strategies without explicit process supervision.

5. Conclusion

This paper introduces VFLM, a self-improving framework for text layout design, bridging the critical “visual perception gap” of existing code-only methods. We implement a “generation, rendering, reflection, refinement” loop, which is trained via RL that leverages a novel, specialized layout reward model. A key finding is that rewarding only the final outcome, rather than complex intermediate steps, can more effectively stimulate the model’s self-improvement capabilities. Extensive experiments validate that VFLM consistently outperforms code-only methods, establishing visual feedback as essential for automated design and MLLM-based design agents.

Acknowledgments

This work is supported by the the National Nature Science Foundation of China (62425114, 62121002, U23B2028), and the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM103).

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1
- [2] Anthropic. Claude 3.7 sonnet system card. <https://www.anthropic.com/news/claude-3-7-sonnet>, 2025. 2, 5
- [3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023. 2
- [4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 1, 2, 5
- [5] ByteDance. doubao-seed-1.6. https://research.doubao.com/zh/seed1_6, 2025. 3, 5
- [6] Jian Chen, Ruiyi Zhang, Yufan Zhou, Jennifer Healey, Jiuxiang Gu, Zhiqiang Xu, and Changyou Chen. TextLap: Customizing language models for text-to-layout planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14275–14289, Miami, Florida, USA, 2024. Association for Computational Linguistics. 2
- [7] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024. 2
- [8] Yutao Cheng, Zhao Zhang, Maoke Yang, Nie Hui, Chunyuan Li, Xinglong Wu, and Jie Shao. Graphic design with large multimodal model. *arXiv preprint arXiv:2404.14368*, 2024. 1, 2
- [9] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. 1, 2
- [10] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025. 2
- [11] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36, 2024. 1
- [12] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025. 2
- [13] Zhangwei Gao, Zhe Chen, Erfei Cui, Yiming Ren, Weiyun Wang, Jinguo Zhu, Hao Tian, Shenglong Ye, Junjun He, Xizhou Zhu, et al. Mini-internvl: a flexible-transfer pocket multi-modal model with 5% parameters and 90% performance. *Visual Intelligence*, 2(1):1–17, 2024. 2
- [14] Daichi Haraguchi, Naoto Inoue, Wataru Shimoda, Hayato Mitani, Seiichi Uchida, and Kota Yamaguchi. Can gpts evaluate graphic design based on design principles? In *SIGGRAPH Asia 2024 Technical Communications*, pages 1–4, 2024. 5
- [15] HsiaoYuan Hsu and Yuxin Peng. Postero: Structuring layout trees to enable language models in generalized content-aware layout generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8117–8127, 2025. 2
- [16] Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. *arXiv preprint arXiv:2501.03262*, 2025. 4
- [17] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025. 3
- [18] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 1, 2, 5
- [19] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Towards Flexible Multi-modal Document Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14287–14296, 2023. 2
- [20] Naoto Inoue, Kento Masui, Wataru Shimoda, and Kota Yamaguchi. OpenCOLE: Towards Reproducible Automatic Graphic Design Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024. 1, 2, 5
- [21] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 2
- [22] Peidong Jia, Chenxuan Li, Yuhui Yuan, Zeyu Liu, Yichao Shen, Bohan Chen, Xingru Chen, Yinglin Zheng, Dong Chen, Ji Li, et al. Cole: A hierarchical generation framework for multi-layered and editable graphic design. *arXiv preprint arXiv:2311.16974*, 2023. 1, 2, 3, 5
- [23] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux.

- l kontekst: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025. 5
- [24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 2
- [25] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, et al. Deepagent: A general reasoning agent with scalable toolsets. *arXiv preprint arXiv:2510.21618*, 2025. 2
- [26] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang James Yang, Jian-Guang Lou, and Dongmei Zhang. Layoutprompter: Awaken the design ability of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2
- [27] Jinpeng Lin, Min Zhou, Ye Ma, Yifan Gao, Chenxi Fei, Yangjian Chen, Zhang Yu, and Tiezheng Ge. Autoposter: A highly automatic and content-aware design system for advertising poster generation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1250–1260, 2023. 2
- [28] Jiawei Lin, Shizhao Sun, Danqing Huang, Ting Liu, Ji Li, and Jiang Bian. From elements to design: A layered approach for automatic graphic design composition. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8128–8137, 2025. 2
- [29] 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
- [30] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26286–26296, 2024.
- [31] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024. 2
- [32] OpenAI. Thinking with images. <https://openai.com/index/thinking-with-images/>, 2025. 1, 2
- [33] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, pages 27730–27744. Curran Associates, Inc., 2022. 2, 3
- [34] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. 5
- [35] Yadong Qu, Shancheng Fang, Yuxin Wang, Xiaorui Wang, Zhineng Chen, Hongtao Xie, and Yongdong Zhang. Igd: Instructional graphic design with multimodal layer generation. *arXiv preprint arXiv:2507.09910*, 2025. 1, 2, 5
- [36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmlR, 2021. 2
- [37] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023. 3
- [38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 3
- [39] Jaeyung Seol, Seojun Kim, and Jaeyun Yoo. Posterllama: Bridging design ability of language model to contents-aware layout generation. *arXiv preprint arXiv:2404.00995*, 2024. 2
- [40] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 2, 3
- [41] Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025. 2
- [42] ByteDance Seed Team. Seed1.5-v1 technical report. *arXiv preprint arXiv:2505.07062*, 2025. 1
- [43] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 2
- [44] Feiyu Wang, Zhiyuan Zhao, Yuandong Liu, Da Zhang, Junyu Gao, Hao Sun, and Xuelong Li. Svgen: Interpretable vector graphics generation with large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 9608–9617, 2025. 2
- [45] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 2
- [46] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025. 2
- [47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 2

- [48] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. 5
- [49] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: learning and evaluating human preferences for text-to-image generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 15903–15935, 2023. 5
- [50] Kota Yamaguchi. Canvasvae: Learning to generate vector graphic documents. *ICCV*, 2021. 5
- [51] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 1
- [52] Tao Yang, Yingmin Luo, Zhongang Qi, Yang Wu, Ying Shan, and Chang Wen Chen. Posterllava: Constructing a unified multi-modal layout generator with llm, 2024. 1, 2
- [53] Zhengyuan Yang, Jianfeng Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. Idea2img: Iterative self-refinement with gpt-4v for automatic image design and generation. In *European Conference on Computer Vision*, pages 167–184. Springer, 2024. 2
- [54] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023. 2
- [55] Jiahao Zhang, Ryota Yoshihashi, Shunsuke Kitada, Atsuki Osanai, and Yuta Nakashima. Vascar: Content-aware layout generation via visual-aware self-correction. *arXiv preprint arXiv:2412.04237*, 2024. 2
- [56] Jingyi Zhang, Jiaying Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *arXiv preprint arXiv:2503.12937*, 2025. 3
- [57] Peirong Zhang, Jiabin Zhang, Jiahuan Cao, Hongliang Li, and Lianwen Jin. Smaller But Better: Unifying Layout Generation with Smaller Large Language Models. *International Journal of Computer Vision (IJCV)*, 133:3891–3917, 2025. 1
- [58] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025. 2
- [59] Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deep-eyes: Incentivizing “thinking with images” via reinforcement learning. *arXiv preprint*, 2025. 3
- [60] Min Zhou, Chenchen Xu, Ye Ma, Tiezheng Ge, Yuning Jiang, and Weiwei Xu. Composition-aware graphic layout gan for visual-textual presentation designs. *arXiv preprint arXiv:2205.00303*, 2022. 2, 5
- [61] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. 2

Seeing is Improving: Visual Feedback for Iterative Text Layout Refinement

Supplementary Material

1. Dataset Details

We collected approximately 200K samples, including free and paid data from the internet. Each sample contains a background image, target text, well-formatted SVG code, and the corresponding rendered image. Each sample has an average of 9.8 text boxes, with an average text length of 84.7. Fig. 5 is a data example. Based on these data, our training and testing data were constructed, including the dataset for the Cold-Start SFT phase, the queries for the reinforcement learning phase, the training and testing datasets for the reward model, and finally, the dataset used for evaluation.

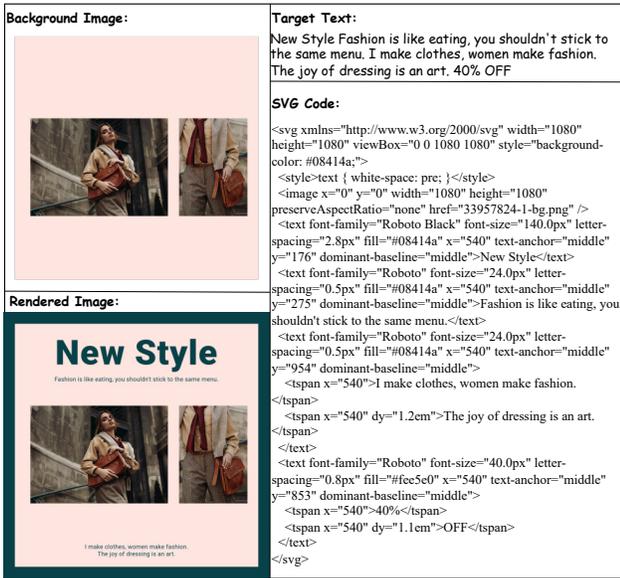


Figure 5. Data example, consisting of a background image, target text, SVG layout code, and the corresponding rendered image.

1.1. Cold-Start Dataset Details

We use the method described in the main text and set the maximum number of modifications to 3 during the data generation process. Since each data trajectory includes an initial generation and a final "satisfied" output, the total number of turns is the number of modifications plus two. This process yielded a total of 8K trajectories, comprising 2,359 two-turn samples (0 modifications), 1,266 three-turn samples (1 modification), 2,030 four-turn samples (2 modifications), and 2,537 five-turn samples (3 modifications). Total data volume is 8K.

During cold-start data construction, the prompts used to guide Doubao-Seed-1.6 are prompt 1 and prompt 2, where

prompt 1 is utilized for Initial Reasoning synthesis and prompt 2 for Multi-round reflection synthesis.

1.2. Layout Reward Model Dataset

Herein, a qualitative comparison of data across the four quality levels (Level-I, Level-II, Level-III, Level-IV) is presented. These four levels exhibit distinct differences in typesetting quality: Level-I demonstrates the best typesetting quality, while Level-IV, by contrast, shows the worst.



Figure 6. Qualitative comparison of four levels of data in the training data of the reward model

2. Training Details

2.1. VFLM

All experiments are conducted on a cluster of 16 NVIDIA H200 GPUs. For the Cold-Start SFT stage, we trained the model with 8k data for 2 epochs with a learning rate of 1×10^{-5} and a batch size of 64. During this stage, the image `max_pixels` was set to $1024 \times 28 \times 28$.

For the RL stage, we set the maximum number of tool calls to 4. The weights for R_{ocr} and R_{svg} (denoted as α) were set to 0.25, while the weight for R_{format} was set to 0.1. We prepare up to 32K samples for training, with early stopping based on reward metrics during RL training. We employed a strict on-policy training strategy with the following configuration: batch size of 64, 8 rollouts per sample, sampling temperature of 1.0, KL divergence coefficient of $1e-3$, and learning rate of $1e-6$.

2.2. Layout Reward Model

We trained the reward model on a preference dataset constructed from 200K layout samples. Four quality levels (Level-I, Level-II, Level-III, Level-IV) were generated for each query, yielding 1.2M preference pairs. We randomly

select 25K pairs as the test set, using the remainder for training. During training, we use a batch size of 512 and train for 2100 steps.

2.3. Ablation Study

In the ablation experiments, Single-Round RL, RL from Pretrained, and Direct Output adopt the same hyperparameter configuration as VFLM. For Direct Output SFT, 40k samples are used for SFT, with a batch size of 128, a learning rate of 1e-5, and training conducted for 2 epochs.

3. Evaluation Details

3.1. Evaluation metrics

We use an OCR engine¹ to recognize text in design images and evaluate the accuracy of rendered text using character-level f-measure. In the RL reward function, R_{ocr} and R_{svg} are evaluated using accuracy. Specifically, a character in the OCR recognition result is defined as a True Positive (TP) if it appears in the annotation; otherwise, it is classified as a False Positive (FP). A False Negative (FN) indicates that a character is only present in the annotation but absent from the OCR recognition result. Accordingly, character-level precision, recall, f-measure and accuracy can be formulated as follows:

$$\begin{aligned}
 Char_P &= \frac{TP}{TP + FP}, \\
 Char_R &= \frac{TP}{TP + FN}, \\
 Char_F &= \frac{2 \times Char_P \times Char_R}{Char_P + Char_R}, \\
 Char_Acc &= \frac{TP}{TP + FP + FN}.
 \end{aligned} \tag{11}$$

For the GPT4o evaluation, we assess the effect along four dimensions: Text Accuracy, Text-Background Harmony, Text Presentation Quality, and Meaning Expression Adaptability. The evaluation prompt is shown in prompt 5. Fig. 7 is a detailed evaluation sample.

3.2. Model Prompt

3.2.1. VFLM System prompt

The VFLM system prompt is shown in prompt 3, includes tool definitions and task descriptions.

3.2.2. MLLM System prompt

The system prompt of MLLM models, as shown in prompt 4, only omits the definition of tools and the statement of multi-round responses compared with that of VFLM.

¹<https://github.com/PaddlePaddle/PaddleOCR>

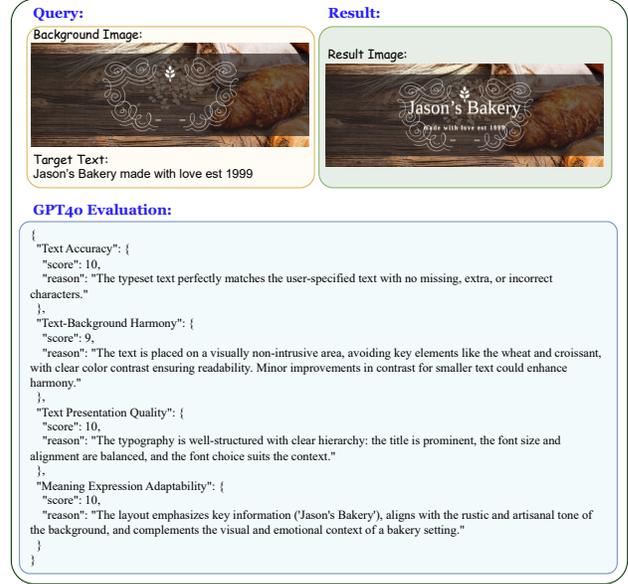


Figure 7. A detailed GPT4o evaluation output.

4. More Experiments

4.1. Number of rounds of reflection

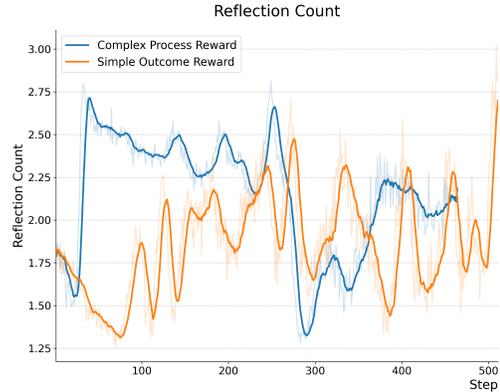


Figure 8. Training processes: reflection times of VFLM, constrained between 1 and 4.

Fig. 8 illustrates the dynamic evolution of reflection counts during training. While the complex process reward employs strategic termination control (Eq. (9)-Eq. (10)) to enforce stability, VFLM under the simple outcome reward exhibits a distinct, insightful trajectory. Specifically, in the initial phase (the first 100 steps), we observe a decline in reflection turns. Combining with Fig. 4, we can analyze this is attributed to the model's initial instability in output formatting, where iterative attempts often degraded quality compared to the initial generation, prompting the model to curtail its reasoning depth. However, beyond 100 steps, as the output format stabilizes, the model discovers that iter-

ative optimization yields superior rewards. Consequently, the reflection count begins to fluctuate and rise, reflecting the model’s autonomous realization that deeper reflection correlates with better layout quality, rather than relying on rigid, pre-defined constraints.



Figure 9. **Reflection Turns vs. Input Complexity.** The positive trend (red line) demonstrates that VFLM autonomously increases reflection depth for more complex tasks.

To further analyze the number of reflection rounds VFLM performs, we examined the correlation between input complexity (measured by the token length of the Target Text) and the number of reasoning rounds. As shown in Figure Fig. 9, which presents the model outputs of 1000 samples from VFLM on the TextLayout test set, the gray scatter points depict the raw distribution of inference steps, which are inherently discrete integers. To visualize the underlying trend amidst this variance, the red solid line tracks the average reflection turns across complexity intervals. A clear positive trend is observable: for concise inputs (< 100 tokens), the model efficiently converges with fewer refinement steps (averaging ~ 2.5 turns). In contrast, as input complexity increases to over 400 tokens, the model adaptively increases its reasoning depth, approaching the maximum of 4 turns. This confirms that VFLM actively perceives layout difficulty and allocates computational resources accordingly.

4.2. Human Study

To further enhance evaluation reliability, we supplement our analysis with a blind human study involving 16 participants, who evaluated outputs from Claude3.7, GPT4o-Image, OpenCOLE, and VFLM on three dimensions across 320 randomly selected queries from three datasets. A total of 960 votes were collected. As shown in the Figure 10, VFLM performed comparably to GPT4o-Image on text coordination and achieved the best results in text accuracy and overall aesthetics.

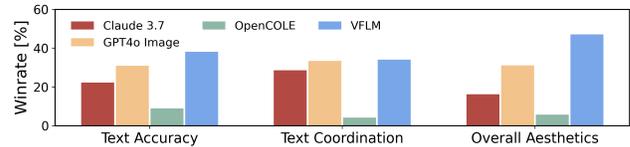


Figure 10. Results of human study.

4.3. Compute Cost and Latency

As shown in Figure 11, we report the average inference time of VFLM, OpenCOLE, and IGD on the TextLayout test set using 4x4090 GPUs. Although VFLM has the highest latency, it yields substantial performance gains.

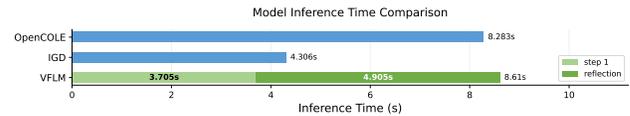


Figure 11. Comparison of inference time and latency with OpenCOLE and IGD.

4.4. More Baseline

Table 5. Metrics and RM score on OpenCOLE* and MLLM feedback baseline.

Model	Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	RM Score \uparrow
OpenCOLE*	0.7671	0.7114	0.0143	21.8279	0.3620
VFLM	0.9071	0.0035	0.0059	15.4583	0.5415
Qwen2.5-VL-7B-feedback	0.6830 \blacktriangle	0.0025 \blacktriangle	0.0188 \blacktriangle	28.8221 \blacktriangledown	0.0910 \blacktriangledown
GPT-4o-feedback	0.8494 \blacktriangle	0.0039 \blacktriangle	0.0028 \blacktriangle	20.8852 \blacktriangledown	0.3662 \blacktriangle

The table above shows the results of OpenCOLE trained on the same training set as VFLM. Although retraining brings substantial improvements over the original OpenCOLE, it still underperforms VFLM. We further integrate visual feedback into GPT-4o and Qwen2.5-VL-7B baselines. As shown in the table, without task-specific training, introducing visual feedback to MLLMs yields only marginal gains, with some metrics even showing slight declines.

4.5. Ablation on Other Datasets

Table 6. Ablation experiments on Graphic quality metrics and OCR metrics on Crello test set.

Model	OCR	Graphic			RM Score \uparrow
	Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	
VFLM-step1	0.8774	0.0046	0.0061	19.5917	0.4392
VFLM	0.9256	0.0025	0.0022	14.8063	0.5548
Cold-Start-step1	0.7904	0.0118	0.0410	24.3143	0.2721
Cold-Start	0.7928	0.0121	0.0402	24.0888	0.2774
Single-Round RL	0.8829	0.0010	0.0094	24.4845	0.4007
RL from Pretrained	0.8792	0.0004	0.0012	30.3397	0.3482
Direct Output	0.9092	0.0009	0.0022	19.3983	0.4596
Direct Output SFT	0.8960	0.0032	0.0192	18.3754	0.4680

Table 7. Ablation experiments on Graphic quality metrics and OCR metrics on DESIGNERINTENTION test set.

Model	OCR	Graphic			RM Score \uparrow
	Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	
VFLM-step1	0.9415	0.0033	0.0023	14.1230	0.5285
VFLM	0.9663	0.0024	0.0008	12.1167	0.5688
Cold-Start-step1	0.8860	0.0130	0.0264	17.5229	0.3784
Cold-Start	0.8881	0.0127	0.0257	16.9280	0.3850
Single-Round RL	0.9382	0.0006	0.0033	16.7466	0.4768
RL from Pretrained	0.9430	0.0005	0.0001	18.0272	0.4692
Direct Output	0.9590	0.0005	0.0010	14.6338	0.5167
Direct Output SFT	0.9434	0.0017	0.0082	13.1702	0.5398

Tab. 6 and Tab. 7 demonstrate the metrics of all models in the ablation experiments on the Crello and DESIGNER-INTENTION test sets. Consistent with the conclusions in the TextLayout dataset, VFLM also achieves significant advantages over other baselines on these two datasets, fully demonstrating the generalization ability of our visual feedback method.

4.6. Ablation on Reward Functions

In our RL training process, the score for layout performance (R_{score}) from the reward model and two rewards based on text accuracy:

$$R_{score} = R_{layout} + \alpha \cdot (R_{ocr} + R_{svg}). \quad (12)$$

We perform ablation to investigate whether using only R_{score} from the reward model is effective. To save validation time, tests are conducted on the models of the two ablation experiments: Single-Round RL and Direct Output.

Table 8. Ablation experiments on the reward function on the Text-Layout test set.

Model	OCR	Graphic			RM Score \uparrow
	Char-F \uparrow	R_{ali} \downarrow	R_{ove} \downarrow	R_{com} \downarrow	
Single-Round RL	0.8792	0.0024	0.0053	18.9428	0.4063
Single-Round RL-w/o OCR	0.8696	0.0024	0.0054	20.0621	0.4153
Direct Output	0.9237	0.0027	0.0021	17.0654	0.4964
Direct Output-w/vo OCR	0.9146	0.0026	0.0059	16.6961	0.5039

The results are presented in Tab. 8. It can be observed from the table that the two models exhibit little difference in RL training performance with and without the R_{ocr} and R_{svg} rewards, with many metrics being relatively close. Since R_{ocr} and R_{svg} have low computational costs and the task essentially involves text typesetting, these two reward functions are retained in the final training of VFLM.

5. Case Study

Fig. 12 presents a detailed case generated by VFLM. The SVG initially produced by VFLM exhibits several obvious issues in terms of rendering effect; for instance, the main title "Merry Christmas" is too large and overlaps with the silhouette of the castle. After receiving visual feedback from

the rendered image, it can be observed from the model's reasoning content (highlighted in the figure) that the model identified these problems and implemented appropriate improvements. Following the second round of visual feedback, the model deemed the current layout satisfactory and output it to user as the final result. Fig. 13 is another case.

Fig. 14 show the more output cases of VFLM. Fig. 15 and Fig. 16 demonstrate the comparison between VFLM and all other models.



Target Text:
 "Christ is Born",
 "Merry Christmas",
 "Glorify Him"

<think>First, determine the hierarchy of the text. The main title "Merry Christmas" should be the most prominent, so I choose a large font size (108px), use the "Source Han Serif SC SemiBold" font, set the color to #eec0d8, and center it at the position (602, 255).
 Next, for the text "Christ is Born Glorify Him", as it is a subtitle, I use the "Source Han Sans SC Heavy" font, size 21.7px, letter-spacing 3.3px, and the color #eec0d8. I center it at (602, 331).
 Throughout the process, I ensure that the text is centered by setting the x-coordinate to 602 (the horizontal center of the 1200px width background) and adjust the y-coordinate to position the text appropriately in the vertical space.
 Let me export the image first to see how the layout looks.</think>
 <tool_call>
 TOOL: svg_to_image_tool
 PARAMS:
 svg_code:
 ```svg  
 <svg xmlns="http://www.w3.org/2000/svg" width="1200" height="628" style="background-color: #8865aa;">  
 <style>text { white-space: pre; }</style>  
 <image x="0" y="0" width="1200" height="628" preserveAspectRatio="none" href="background-image.png" />  
 <text font-family="Source Han Serif SC SemiBold" font-size="108.0px" fill="#eec0d8" x="602" text-anchor="middle" y="255" dominant-baseline="middle">Merry Christmas</text>  
 <text font-family="Source Han Sans SC Heavy" font-size="21.7px" letter-spacing="3.3px" fill="#eec0d8" x="602" text-anchor="middle" y="331" dominant-baseline="middle">Christ is Born Glorify Him</text>  
 </svg>  
 </tool\_call>



<think>I will check the SVG rendering effect of my version. First, **the main title "Merry Christmas" is too large and overlaps with the silhouette of the castle, making the overall layout feel cluttered. The text color also blends too much with the background, reducing readability.** On the positive side, the text content is complete with no missing parts. To improve, **I need to restructure the text hierarchy and positioning.** I'll reduce the font size of "Merry Christmas" to make it more proportional to the background, adjust its position to the upper middle area to avoid overlapping with the castle silhouette, **change the color to a warmer tone** like #fcedd9 to enhance contrast with the background, and split the text into two lines for better visual balance. Next, I will improve my SVG code.</think>  
 <tool\_call>  
 TOOL: svg\_to\_image\_tool  
 PARAMS:  
 svg\_code:  
 ```svg  
 <svg xmlns="http://www.w3.org/2000/svg" width="1200" height="628" style="background-color: #926c98;">
 <style>text { white-space: pre; }</style>
 <image x="0" y="0" width="1200" height="628" preserveAspectRatio="none" href="background-image.png" />
 <text font-family="Source Han Sans SC Heavy" font-size="51.0px" fill="#fcedd9" x="600" text-anchor="middle" y="157" dominant-baseline="middle">Merry Christmas</text>
 <text font-family="Source Han Sans SC Heavy" font-size="21.0px" letter-spacing="3.1px" fill="#fcedd9" x="600" text-anchor="middle" y="219" dominant-baseline="middle">Christ is Born Glorify Him</text>
 </svg>
 </tool_call>



<think>I will check the SVG rendering effect of this version. The main title "Merry Christmas" is now properly positioned in the upper middle area, with an appropriate font size and color that contrasts well with the background, making it the focal point of the layout. The subtitle "Christ is Born Glorify Him" is centered below the main title, creating a clear visual hierarchy. There is no text overlapping with the silhouette elements of the background, and all text content is complete and clearly readable. The overall layout feels balanced and harmonious, effectively conveying the Christmas message. **I believe the current SVG layout effect is very good and can be output to the user.**</think>
 <answer>
 ```svg  
 <svg xmlns="http://www.w3.org/2000/svg" width="1200" height="628" style="background-color: #926c98;">  
 <style>text { white-space: pre; }</style>  
 <image x="0" y="0" width="1200" height="628" preserveAspectRatio="none" href="background-image.png" />  
 <text font-family="Source Han Sans SC Heavy" font-size="51.0px" fill="#fcedd9" x="600" text-anchor="middle" y="157" dominant-baseline="middle">Merry Christmas</text>  
 <text font-family="Source Han Sans SC Heavy" font-size="21.0px" letter-spacing="3.1px" fill="#fcedd9" x="600" text-anchor="middle" y="219" dominant-baseline="middle">Christ is Born Glorify Him</text>  
 </svg>  
 </answer>



Figure 12. A detailed VFLM output case.



User

Target Text:

超值组合  
招牌牛肉套餐

**think**—First, I need to determine the placement of two texts: “超值组合” and “招牌牛肉套餐”. Looking at the background, the red area on the right is a suitable space, so I center both texts there. For “招牌牛肉套餐”, as the key information, I choose a bold font (Source Han Sans SC Heavy) with a larger size (50px) to emphasize it. For “超值组合”, I use a regular font (Source Han Sans SC Regular) with a slightly smaller size (29px) to create a hierarchy.

To make the text stand out against the red background, I use a double - outline effect. I define each text in <defs> first. Then, I create a mask: the mask combines a white stroke and a black fill of the text. After that, I use <use> to apply the fill color (#ff3b7) for “超值组合” and #ee1c1e for “招牌牛肉套餐”) and the masked stroke (#ae1311). The stroke width is set to 4px, which is wide enough for a clear outline without being too thick.

Both texts are center - aligned using text - anchor=“middle”. “招牌牛肉套餐” is positioned higher (y = 61) as the main headline, and “超值组合” is below it (y = 116) as a supporting line. Using white - space: pre in the style ensures proper text formatting. Reusing <use> elements for fill, stroke, and mask keeps the code efficient while achieving the desired visual effects.

Let me export the image first to see how the layout looks.</think>

```

<tool_call>
TOOL: svg_to_image_tool
PARAMS:
svg_code:
'''svg
<svg xmlns="http://www.w3.org/2000/svg" width="686" height="176" viewBox="0 0 686 176" style="background-color: #fff;">
<style>text { white-space: pre; }</style>
<image x="0" y="0" width="686" height="176" preserveAspectRatio="none" href="background-image.png" />
<defs>
<text id="text-1" font-family="Source Han Sans SC Regular" font-size="29.0px" x="454" text-anchor="middle" y="116" dominant-baseline="middle">超值组合</text>
<mask id="text-1-outer-1">
<use href="#text-1" stroke="white" stroke-width="4" />
<use href="#text-1" fill="black" />
</mask>
</defs>
<defs>
<text id="text-2" font-family="Source Han Sans SC Heavy" font-size="50.0px" x="454" text-anchor="middle" y="61" dominant-baseline="middle">招牌牛肉套餐</text>
<mask id="text-2-outer-1">
<use href="#text-2" stroke="white" stroke-width="4" />
<use href="#text-2" fill="black" />
</mask>
</defs>
</defs>
<use href="#text-1" fill="#ff3b7" />
<use href="#text-1" fill="none" />
<use href="#text-1" mask="url(#text-1-outer-1)" stroke="#ae1311" stroke-width="4" />
<use href="#text-2" fill="#ee1c1e" />
<use href="#text-2" fill="none" />
<use href="#text-2" mask="url(#text-2-outer-1)" stroke="#ae1311" stroke-width="4" />
</svg>
'''
</tool_call>

```



SVG



**think**—I will check the SVG rendering effect of my version. First, I notice that both key texts “超值组合” and “招牌牛肉套餐” are present and not overlapping, which is a good foundation. However, the visual hierarchy seems inverted. The prominent red background area on the right should highlight the main promotional message, but currently “招牌牛肉套餐” is placed there in large red text - **this causes poor readability against the red background and weakens the focus.** Meanwhile, “超值组合” is confined to the small white rectangular box below, **reducing its impact as a key selling point.**

To improve, **I need to swap the text content between the two positions.** “超值组合” should be the headline, placed prominently on the red background with a larger size and white color for contrast. “招牌牛肉套餐” should then occupy the white box, using a smaller size and dark red color to complement the box. I'll adjust the first text element's content to “超值组合”, set font-size to 72px, fill to #fff, and center it horizontally on the red area (x around 328, y around 61). The second text element will be updated to “招牌牛肉套餐”, with font-size 32px, fill #b50101, and centered in the white box (x around 350, y around 120). This will fix readability and establish correct visual hierarchy. Next, I will improve my SVG code.</think>

```

<tool_call>
TOOL: svg_to_image_tool
PARAMS:
svg_code:
'''svg
<svg xmlns="http://www.w3.org/2000/svg" width="686" height="176" viewBox="0 0 686 176" style="background-color: #fff;">
<style>text { white-space: pre; }</style>
<image x="0" y="0" width="686" height="176" preserveAspectRatio="none" href="background-image.png" />
<text font-family="PangZhenGui-PMZD" font-size="72.0px" letter-spacing="-5.6px" fill="#fff" x="328" y="61" dominant-baseline="middle">超值组合</text>
<text font-family="PangZhenGui-PMZD" font-size="32.0px" letter-spacing="-1.3px" fill="#b50101" x="350" y="120" dominant-baseline="middle">招牌牛肉套餐</text>
</svg>
'''
</tool_call>

```



SVG



**think**—I will check the SVG rendering effect of this version. The layout now presents a clear visual hierarchy: “超值组合” stands out as the main headline in large white text against the red background, immediately capturing attention, which effectively emphasizes the promotional focus. “招牌牛肉套餐” is neatly positioned within the white rectangular area with red text, creating strong contrast against the white background for excellent readability. Both texts are centered appropriately, with no overlapping, blocking, or exceeding the background's aesthetic, making the promotional message clear and visually appealing. **The current SVG layout effect is very good and can be output to the user.**</think>

```

<answer>'''svg
<svg xmlns="http://www.w3.org/2000/svg" width="686" height="176" viewBox="0 0 686 176" style="background-color: #fff;">
<style>text { white-space: pre; }</style>
<image x="0" y="0" width="686" height="176" preserveAspectRatio="none" href="background-image.png" />
<text font-family="PangZhenGui-PMZD" font-size="72.0px" letter-spacing="-5.8px" fill="#fff" x="328" y="61" dominant-baseline="middle">超值组合</text>
<text font-family="PangZhenGui-PMZD" font-size="32.0px" letter-spacing="-1.3px" fill="#b50101" x="350" y="120" dominant-baseline="middle">招牌牛肉套餐</text>
</svg>
'''
</answer>

```



SVG





VFLM


VFLM


VFLM

Figure 13. Another detailed VFLM output case.

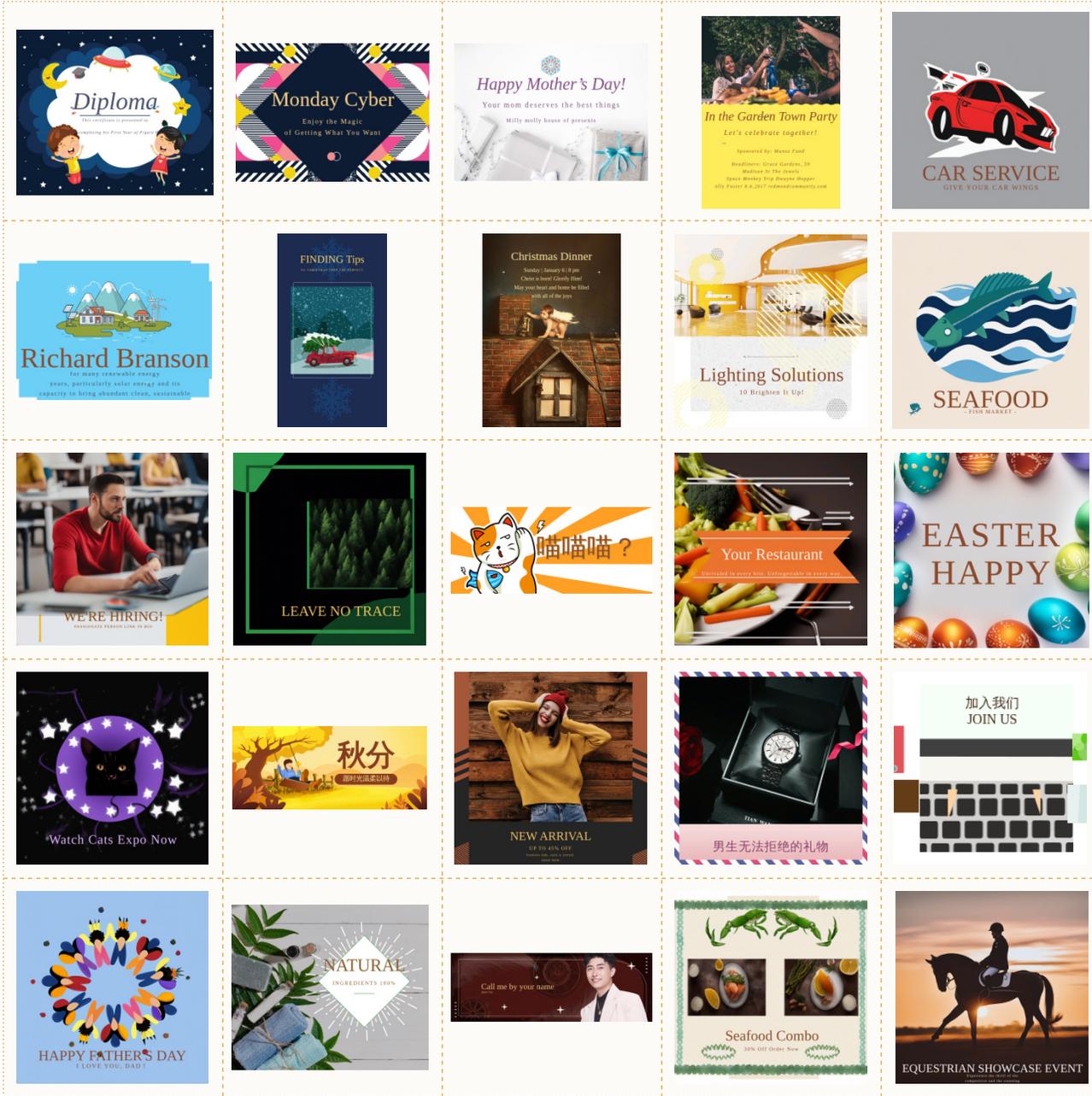


Figure 14. More VFLM output cases.

Target Text

www.fridayfantasy.com  
For more information visit  
our website  
FANTASY FILM FESTIVAL  
FRIDAY NIGHT MAGIC

BURGER  
Club  
OPEN  
24/7

FISH MARKET ORDER  
NOW SELLING FRESH  
SEAFOOD ON ICE GET  
THE BEST PRICE FROM  
US

认识兴趣是力求认识世界，兴趣是最好的老师，学生有心曲的学习，即可在学习活动中体验效率，又可体验成功过，提高效率，所以应培养并激发学生学习的兴趣，才能保证课堂的效率。关键词：兴趣、学习、数学、效率 导论 INTRODUCTION

·热爱互联网行业，对未来充满期待·好奇心强，喜欢探索研究新事物·自驱力强，时时刻刻鞭策自己，敢想敢做·有较强的目标导向意识及抗压能力，沟通能力强 我们希望

Background image

GPT4o

Claude3.7

Doubao-Seed-1.6

Qwen2.5-VL-72B

GPT4o-Image



Figure 15. In comparison with all existing methods(Part1).

Target Text

www.fridayfantasy.com  
For more information visit  
our website.  
FANTASY FILM FESTIVAL  
FRIDAY NIGHT MAGIC

BURGER  
Club  
OPEN  
24/7

FISH MARKET ORDER  
NOW SELLING FRESH  
SEAFOOD ON ICE GET  
THE BEST PRICE FROM  
US

认识兴趣是力求认识世界，兴趣是最好的老师，学生有心曲的学习，即可在学习活动中体验效率，又可体验成功，提高效率，所以应培养并激发学生学习的兴趣，才能保证课堂的效率。关键词：兴趣、学习、教学、效率  
导论 INTRODUCTION

热爱互联网行业，对未来充满期待，好奇心强，喜欢探索研究新事物，自驱力强，时时刻刻鞭策自己，敢想敢做，有较强的目标导向意识及抗压能力，沟通能力强 我们希望

Background image



Qwen-Image-Edit

FLUX Kontext

IGD

OpenCOLE

VFLM

Figure 16. In comparison with all existing methods(Part2).

## Initial Reasoning Process Prompt

### Role setting:

You are an experienced Layout and SVG engineer.

### Task:

Here is a result of using SVG code to typeset specific text on an input background image. I will provide you with the designed SVG code and the rendered image of this code, which has a very beautiful layout effect.

Now, assuming you are the designer of typesetting this SVG, what is your thought process when typesetting this SVG?

Could you please use the voice of a designer to briefly describe your thought process when designing this SVG based on the SVG code and rendering results? How did you design this SVG?

Ensure that your design ideas are consistent and closely related to the design results of this SVG. Do not fabricate content that is not included in the SVG, as the SVG only typesets the given text based on the given background image. Therefore, the typesetting ideas should focus on the implementation of the text, including text position, font, size, beautification effect, etc. Don't pay attention to the text already in the background image. Do not pay attention to unnecessary other graphic icons and other elements, and do not analyze how good the rendered image is from the perspective of typesetting, because your thinking is based on not seeing the rendered image.

```
background-image.png: <image>
image size : {image_width} * {image_height}
texts: {target_text}
```

```
svg code:
```svg
{svg_code}
```
```

The image result rendered by this svg: <image>

### Output requirements:

Please directly output the thinking process of the person in the designer's tone, without any other descriptive content. Answer in English.

## Multi-Round Reflection Data Synthesis Prompt

USER:

You are an experienced typesetter and SVG engineer, skilled at elegantly typesetting specified text on user provided background images. You know how to apply unique aesthetic principles to design professional and attractive layout, using SVG code to create beautiful layouts. Please design the final layout plan based on the background image and text content provided by the user. In SVG code, use the image tag to reference the background image: href="background-image.png", while other elements only need to be designed with content related to the text. Please design an SVG code layout scheme based on the background image and text content provided by the following users. You first think about it, and then output the final SVG code. The format is<think>...</think>\n<answer>...</answer>

```
background-image.png: <image>
image size: {image_width}*{image_height}
Please arrange the following text on the background image:
{target_text}
```

ASSISTANT:

{Reasoning Process and SVG generated by Qwen2.5-VL-7B}

USER:

Now, I'll show you the effect of this version of SVG layout, and you need to improve this SVG layout effect.

I'll also give you a standard SVG layout result, and you need to improve your SVG layout according to this standard SVG layout result.

- In your output, you need to speak in the tone of a designer, stating that you've reviewed the SVG result of your initial layout, then reflected on it and made corrections. Note that you've designed an initial version of the SVG, and now I've provided you with the rendered image. Your output should focus on examining the image, ensuring it's a reflection and correction of your initial SVG layout result. The direction of correction is the correct effect I gave you, but don't expose in the output that you're improving based on the standard effect. Pretend you've thought it out on your own.

- The output should include your thinking process for SVG layout, how to improve your SVG layout result step by step. You need to point out which parts of your initial layout were good and which were bad and needed modification. For each modification point, be specific about how to modify the SVG code. Don't just qualitatively say which aspects you'll modify. Pay attention to the tone, which should be like that of a designer, and the content of the output should conform to the designer's way of thinking.

- During the modification process, key considerations should be text position, whether there is any text missing, text overlapping, text being blocked, and whether the text exceeds the background image range, etc. These considerations need to be included in the output.

- Your output modification process may involve multiple steps. If your initial layout is not very different from the standard one, you can make only one modification; if there is a large gap, multiple steps of modification are required. You need to simulate the designer's thinking process and gradually improve the SVG layout. Each time you modify, choose the part with the worst effect to improve. Explain the specific SVG code improvements in the thinking process. After modifying one version, only make changes to the SVG part that needs to be modified in this step, and don't change the other parts for now. Output the complete SVG code; then proceed to the next modification until you think the SVG layout effect is very good. Don't make too many modifications. Ensure that each modification is better than the previous one, with a maximum of 3 modifications. The SVG code after the last modification needs to be output, and its effect should be the same as that of the standard code I gave you.

- You need to answer one modification each time, and then I'll show you the rendered effect of the SVG you modified, and you'll make the next modification.

- Based on the rendered image effect I give you after each of your modifications, decide whether the next modification is needed. Each modification should have a significant improvement, not just a minor one. For example, when the order of different text tags doesn't affect the SVG rendering effect, there's no need for additional modification. Since I require you to make as few steps of modification as possible, each modification should have a significant improvement.

Your output is the thinking process of a designer improving the SVG layout after reviewing the first version they designed. I've given you the standard SVG code, and you should modify the SVG code in this direction. However, note that your output is based on not having seen this standard SVG, as if the designer is reflecting after designing the initial draft and modifying it to the final standard SVG version through multiple steps.

After the last modification, you need to output the final inspection, indicating that after checking the image, you think the current SVG layout effect is very good and can be replied to the user.

Your initial SVG layout effect is shown in the figure below:

<image>

This is standard and beautiful SVG code. The code and its rendered effect diagram are as follows.

```
```svg
{svg_code}
```
```

<image>

Output requirements:

Please directly output the thinking process in the tone of a designer, without any other descriptive content. Be careful not to reveal that you have seen the standard SVG effect. Transform it into your own thinking. The output should conform to the designer's thinking process, that is, how you think about improving the layout by yourself, not by comparing with the standard effect. Do not output the word "standard".

If improvement is needed, the first sentence in each step of the thinking process should be: "I will check the SVG rendering effect of my version...", and the last sentence should be: "Next, I will improve my SVG code."

These two beginning and ending sentences are necessary and cannot be omitted, but you can modify the language to maintain the same meaning and make the output diverse.

Your output needs to specifically point out which effects in your first version are good and do not need improvement, which effects are poor and need improvement, and how to specifically modify the SVG code. If you think the SVG layout effect of your first version is very close to or even better than the standard SVG rendering effect I provided, you can describe your satisfaction with this SVG layout and that you think it has achieved a very good effect and does not need further improvement.

Answer in English.

Output requirements:

- You need to output in the form of multi - round conversations. According to the number of modifications you decide, the output format for each modification is as follows:

# Step {current modification number} of modification:

## Thinking process: Here, think about how to make the modification.

## SVG code: Modify the complete SVG code.

- After the final modification, the rendering result of your SVG code should be exactly the same as that of the reference SVG code I provided.

- After the last modification is output, I will provide you with the rendered image again. Then you need to output a final reflection, indicating that you will check the SVG rendering effect of this version and think that the current SVG layout effect is very good and does not need to be improved further, and it can be output to the user. The output format of the final reflection is:

# Final rethink: ...

USER:

Your current SVG layout effect is shown in the figure below:

<image>

ASSISTANT:

...

## VFLM System Prompt

System:

You are a helpful assistant.

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:

```
<tools>
{
 "type": "function ",
 "function": {
 "name": "svg_to_image_tool",
 "description": "Convert SVG code to an image.",
 "parameters": {
 "type": "object",
 "properties": {
 "svg_code": {
 "type": "string",
 "description": "The SVG code to convert to an image."
 }
 },
 "required": [
 "svg_code"
]
 }
 }
}
</tools>
```

# How to call this tool

Wrap the SVG code with specific markers (`` and ``) within <tool\_call></tool\_call> XML tags.

**\*\*Example\*\*:**

```
<tool_call>
TOOL: svg_to_image_tool
PARAMS:
svg_code:
``svg
...
</tool_call>
```

You are an experienced visual layout designer and SVG engineer, skilled at elegantly typesetting specified text on background images provided by users.

You know how to apply unique aesthetic principles to design professional and appealing layouts, using SVG code to create beautiful layouts. Please design a final layout plan based on the background image and text content provided by the user.

In the SVG code, use the image tag to reference the background image:

href="background-image.png", and other elements only need to design content related to the text.

Please design an SVG code layout plan based on the following background image and text content provided by the user.

You should first view the background image, think about how to typeset the text on the background image, design a version of SVG code, correctly reference the background image in the SVG code, then call the `svg_to_image` tool, and you will get the picture of your SVG. Then, based on the picture, judge whether the typesetting of your picture meets the expectations, whether the background image is correctly referenced, and whether the text is beautiful. If the typesetting effect is not good enough, modify the SVG code, and repeatedly reflect after tool calls until the typesetting effect is better. Finally, output the final SVG code.

Format: `<think>...</think>\n<tool_call>...</tool_call>`(if tools needed)  
`<answer>...</answer>`

User:

`background-image.png: <image>`  
`image size: {image_width}*{image_height}`  
`texts: {text}`

### MLLM System Prompt

System:

You are an experienced layout designer and SVG engineer, proficient in elegantly laying out specified text on a background image provided by the user. You have a deep understanding of how to use unique aesthetic principles to design a professional and attractive layout. Use SVG code to create a beautiful layout. Please design the final layout plan according to the background image and text content provided by the user. In the SVG code, use the image tag to reference the background image: `href="background-image.png"`, and only design the elements related to the text. Please design the SVG code layout plan according to the background image and text content provided below.

User:

`background-image.png: <image>`  
`image size: {image_width}*{image_height}`  
`texts: {text}`

## GPT4o Evaluation Prompt

You are an autonomous AI Assistant specializing in evaluating the typesetting effects of a typesetting model. This model's core task is to typeset user-specified text on a background image; your goal is to provide objective, targeted, and constructive scoring and feedback based on text-typesetting-specific principles and practical application needs. Your evaluation covers four independent dimensions: text content accuracy, text-background visual harmony, text presentation quality, and meaning expression adaptability. You will be provided with the background image, the user's original specified text, and the typeset result (background image + typeset text). Your task is to score the typesetting effect objectively based on the following 4 criteria and provide concise reasoning for each score.

### Scoring rules:

- For each of the 4 criteria, score objectively and rigorously on an independent scale of 1-10. For a single criterion, a score of 10 means flawless performance (no issues, fully meeting expectations); a score of 7 indicates minor flaws (no impact on core performance); a score of 4 reflects significant shortcomings (affecting core performance); a score of 1-2 signifies severe issues (rendering the function of this criterion ineffective).
- Keep reasoning concise (1-2 sentences per criterion), focusing on specific performance. If the output is too long, it will be truncated.
- Only respond in JSON format with 4 top-level keys corresponding to the 4 Grading criteria. Each key's value is an object containing "score" (integer 1-10) and "reason" (string). No other information.

### Grading criteria:

1. Text Accuracy (1-10): Evaluate consistency with the user's original text (no missing/extra/wrong characters, no spelling/grammatical errors in Chinese/English). Score 10: 100% accurate; Score 1: massive errors or unrecognizable characters.

2. Text-Background Harmony (1-10): Evaluate visual coordination: (1) text avoids blocking the background's main subject (key figures, core graphics); (2) text color/transparency ensures clear contrast with the background (no blurring). Score 10: no blocking, perfect contrast; Score 1: complete blocking or unreadable due to poor contrast.

3. Text Presentation Quality (1-10): Evaluate text's own properties: (1) structural rationality (clear title/body hierarchy, compliance with reading habits, balanced spacing); (2) physical readability (appropriate font selection, suitable size, neat alignment). Score 10: clear structure, highly readable; Score 1: chaotic structure and physically unreadable.

4. Meaning Expression Adaptability (1-10): Evaluate meaning transmission: (1) key information is highlighted (via weight/color/size); (2) layout matches text's emotional tone (e.g., serious text uses rigorous typography); (3) text position aligns with the background's semantic context (e.g., "ocean protection" text near ocean elements). Score 10: amplifies meaning, matches tone, aligns with background semantics; Score 1: contradicts meaning/tone or conflicts with background semantics.

### The background image:

<image>

User's original specified text: {text\_content}

### The typeset result:

<image>