Detail++: Training-Free Detail Enhancer for Text-to-Image Diffusion Models

Supplementary Material

1. Limitation and Discussion

The proposed Progressive Detail Injection (PDI) framework addresses detail binding issues in text-to-image generation through a training-free approach, yet it exhibits certain limitations. The method heavily relies on the quality and accuracy of initial Self-Attention maps—if the layout generation in the early phase is suboptimal, subsequent attribute injection processes may not fully rectify these issues, potentially constraining the final generation quality.

2. Implementation Details

Method details. Our method is implemented on SDXL. We extract the cross attention maps from the all layers with resolution of 32 × 32 of all blocks in U-Net, as we found this is comparatively fine-grained and accurate. Also, our all experiments is conduct in a single Nvidia H-20 GPU.

Baseline methods implementation. For quantitative comparison in Table. 1, we use official implementation of Stable diffusion 1.5 [48], Stable diffusion XL [42], Structured Diffusion Guidance [14], Composable diffusion [35], PixelArt-α [8], ELLA [22], ToME [21], Attention Regulation [64]. Since the SDXL checkpoint of Ranni [15] is not open-sourced, thus we directly refer to the score in their paper.

3. Sub-prompts Management

In this section, we compare the effects of different prompt management granularities on generation quality. For example, given the original prompt p_{ori} =*"a red dog with sunglasses and a blue cat with a necklace"*, we can manage the sub-prompts in two ways: 1) most complex first, or 2) simplest first. Regarding granularity, we can manage them in two ways: a) one subject per branch, or b) one attribute per branch. This results in a total of four possible configuration combinations. These are:

Config. A (1 + a):

$$\mathcal{P} = \left\{ \begin{array}{l} p_0 : \textit{"a red dog with sunglasses and a blue cat with a necklace."}, \\ p_1 : \textit{"a dog and a cat."}, \\ p_2 : \textit{"a red dog with sunglasses and a cat."}, \\ p_3 : \textit{"a dog and a blue cat with a necklace."} \end{array} \right\},$$

$$\mathcal{Q} = \left\{ \begin{array}{l} q_1 : \textit{"dog"}, \\ q_2 : \textit{"cat"} \end{array} \right\}.$$

Config. B (1+b):

$$\mathcal{P} = \left\{ \begin{array}{l} p_0 : \textit{"a red dog with sunglasses and a blue cat with a necklace."}, \\ p_1 : \textit{"a dog and a cat."}, \\ p_2 : \textit{"a red dog and a cat."}, \\ p_3 : \textit{"a dog with sunglasses and a cat."}, \\ p_4 : \textit{"a dog and a blue cat."}, \\ p_5 : \textit{"a dog and a cat with a necklace."} \end{array} \right\},$$

$$\mathcal{Q} = \left\{ \begin{array}{l} q_1 : \textit{"dog"}, \\ q_2 : \textit{"dog"}, \\ q_3 : \textit{"cat"}, \\ q_4 : \textit{"cat"} \end{array} \right\}.$$

Config. C (1 + a):

$$\mathcal{P} = \left\{ \begin{array}{l} p_1 : \textit{"a dog and a cat."}, \\ p_2 : \textit{"a red dog with sunglasses and a cat."}, \\ p_3 : \textit{"a dog and a blue cat with a necklace."} \end{array} \right\},$$

$$\mathcal{Q} = \left\{ \begin{array}{l} q_1 : \textit{"dog"}, \\ q_2 : \textit{"cat"} \end{array} \right\}.$$

Config. D (2+b):

$$\mathcal{P} = \left\{ \begin{array}{l} p_1 : \textit{"a dog and a cat."}, \\ p_2 : \textit{"a red dog and a cat."}, \\ p_3 : \textit{"a dog with sunglasses and a cat."}, \\ p_4 : \textit{"a dog and a blue cat."}, \\ p_5 : \textit{"a dog and a cat with a necklace."} \end{array} \right\},$$

$$\mathcal{Q} = \left\{ \begin{array}{l} q_1 : \textit{"dog"}, \\ q_2 : \textit{"dog"}, \\ q_3 : \textit{"cat"}, \\ q_4 : \textit{"cat"} \end{array} \right\}.$$

Therefore, we conduct plenty of experiments to test which one can best reflect our method's efficiency, as shown in Table. A2. Another tokenization approach is accumulative prompts, where the previously added attribute continues to appear in the prompt of the next branch. This is referred to as the **accUmu. prompt**:

$$\mathcal{P} = \left\{ \begin{array}{l} p_1 : \textit{"a dog and a cat."}, \\ p_2 : \textit{"a red dog and a cat."}, \\ p_3 : \textit{"a red dog with sunglasses and a cat."}, \\ p_4 : \textit{"a red dog with sunglasses and a blue cat."}, \\ p_5 : \textit{"a red dog with sunglasses and a blue cat a with necklace."} \end{array} \right\},$$

Table A1. Time Complexity of various methods. The highest scores are highlighted in blue, and second-highest in green.

Method	Inference Steps	Time Cost	Color	Texture	Shape
SDXL	50	30s	0.6275	0.5637	0.5408
ToMe	50	87s	0.6583	0.6371	0.5517
Attention Regulation	50	83s	0.5860	0.5173	0.4672
Detail++(Ours)	50	31s	0.7389	0.7241	0.5582

Table A2. Different configurations' effect.

Config	1/2	a/b	BLIP-VQA Color	BLIP-VQA Texture	BLIP-VQA Shape	Time Consumption	Memory Consumption
A	1	a	0.7286	0.7205	0.5573	30s	16.21G
B	1	b	**0.7389**	**0.7241**	**0.5582**	31s	20.10G
C	2	a	0.7238	0.7156	0.5567	30s	14.26G
D	2	b	0.7325	0.7192	0.5580	30s	18.14G
accumu. prompt			0.7314	0.7163	0.5541	30s	18.14G

$$\mathcal{Q} = \left\{ \begin{array}{l} q_1 : \textit{“dog”}, \\ q_2 : \textit{“dog”}, \\ q_3 : \textit{“cat”}, \\ q_4 : \textit{“cat”} \end{array} \right\}.$$

However, as shown in the second-to-last row of Table A2, the quantification results are not ideal. We hypothesize that this is due to modifier overflow in the longer prompts [21], which degrades the efficiency of our method.

4. Time complexity

Time complexity. As shown in Table. A2, we compare the time cost with the other two most recent training-free methods [21, 54]. Thanks to parallel inference, our method achieves a similar runtime to the baseline while delivering better results. In contrast, other methods typically require LLMs [15] for local inference or rely on test-time optimization during the denoising process, often resulting in suboptimal time consumption.

5. SCB details

Style Composition Benchmark(SCB) construction. Aiming to comprehensively evaluate text-to-image generation under multi-style composition scenarios, We adopt a unified prompt format—“A [type of style] style [subject] in a [type of style] style [background].” As shown in Table A3, the “[type of style]” column lists the common style categories used for prompt construction. while [subject] and [background] are randomly selected from the corresponding word pool in the table. Notably, to better evaluate the ability of different models to maintain style consistency, we ensure that the [subject] and [background] within the same prompt do not share the same style.

SCB evaluation metrics. We propose a novel style alignment evaluation methodology that employs object detection to segment composite images into subject and background components. Each segment is independently analyzed against its intended style description using CLIP score metrics. The final averaged score reflects the overall fidelity of the composite image to its intended style descriptors across all elements. As illustrated in Fig. A1 A2, our method effectively evaluates style alignment across composite images. In Fig. A1, the robot incorporates oil painting stylistic elements while the forest background exhibits traces of sketch characteristics, resulting in a lower CLIP score (0.2344) that indicates reduced alignment due to style contamination. In contrast, Fig. A2 demonstrates clearer stylistic delineation—a sketch-style robot (0.2818) against an oil painting forest background (0.2493)—yielding a higher average alignment score (0.2656). This granular approach enables precise measurement of style fidelity across different image elements. Noticed that, why we use a cropping way instead of a segmentation way, is the Clip cannot accurately recognize the images with partial vacancy.

Table A3. Style Composition Benchmark

Type of style	Lego; Oil-painting; Cyberpunk; Sketch; Pixel-Art; Watercolor; Graffiti
Subject	Dog; Cat; Robot; Car; Unicorn
Background	forest; Space; Desert; City; Ruins

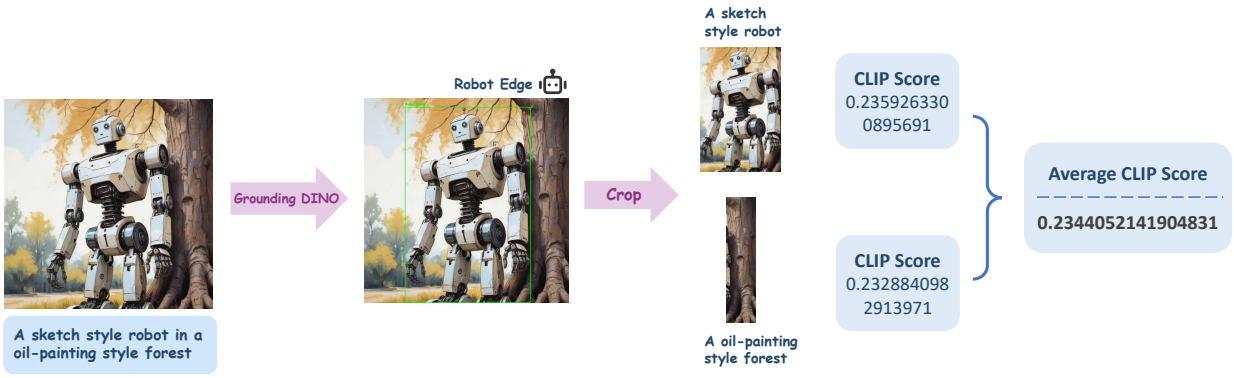

Figure A1. Pipeline of measure the style accuracy in circumstances of blending.

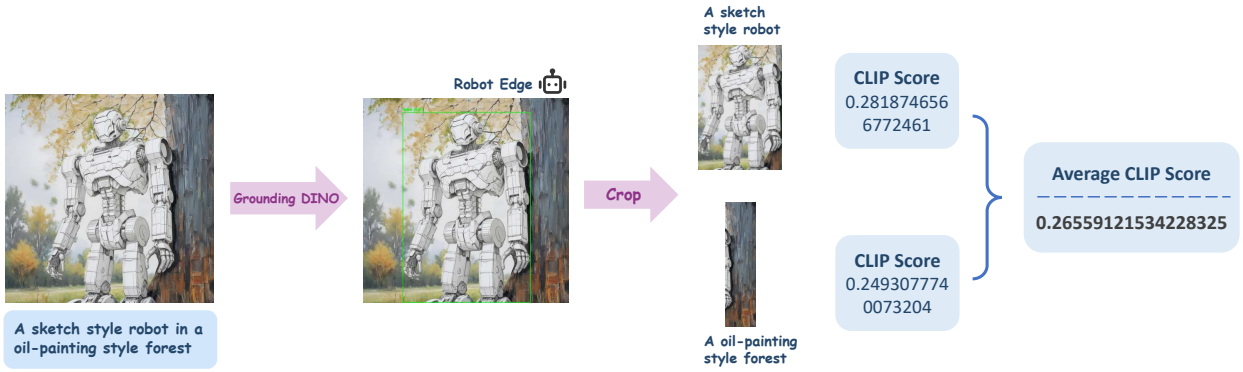

Figure A2. Pipeline of measure the style accuracy in circumstances for our methods.

6. Comparison between SA and CA mechanism

In this section, we present key concepts and techniques that serve as the foundation for our proposed method. These preliminary components provide a clearer understanding of the motivations behind our approach and the technical mechanisms leveraged in its design.

7. Why first 80% timesteps self-extension?

Why SA map? While many recent studies [5, 34, 38, 51] indicate that self-attention maps carry structural information and thus have strong layout-preserving capabilities, the cross-attention map-based editing approach [18] still has a significant impact. So, why choose to replace the self-attention map rather than the cross-self attention map for progressive editing? Here, we visualize the difference between replacing self-attention maps (SA) and cross-attention maps (CA) at various steps. As shown in Fig. A5, editing based on the self-attention map tends to provide stronger editing capabilities, consistent with the findings in FPE [34].

We hypothesize that the relatively weaker effect of editing based on cross-attention arises because it only replaces or influences the attention map of the edited object. In contrast, editing based on the self-attention map is more like conditional generation under the prior assumption of a diffusion model's layout, which resembles the effect described in [61]. In this way, our framework, **Detail++**, ensures that each editing step is guided by an accurate detail binding relation prompt, enabling precise detail binding and preventing issues such as overflow, mismatching, and blending.

8. Cross-attention Layers Selection

In this section, we compare different layer selections for binary mask generation. We visualize cross-attention maps from various layers (see Fig. A3) and provide additional detailed visualizations in Fig. A4. It is evident that extracting the 32×32 resolution cross-attention map from the down, up, and mid blocks produces the cleanest and most accurate subject mask.

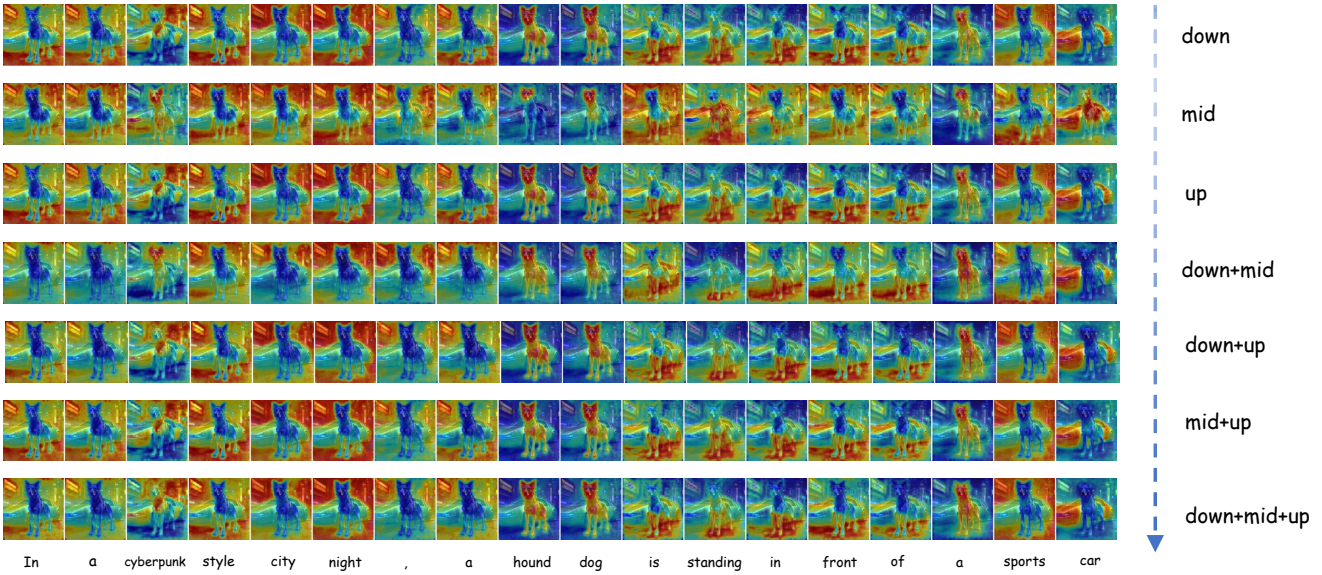

Figure A3. Visualization of cross-attention maps at different layers. "Down" denotes the cross-attention layers in the first downsampling block, while "Up" indicates those in the second upsampling block. Other layers with a 64×64 cross-attention map require significantly more memory and are thus omitted here to avoid excessive GPU usage.

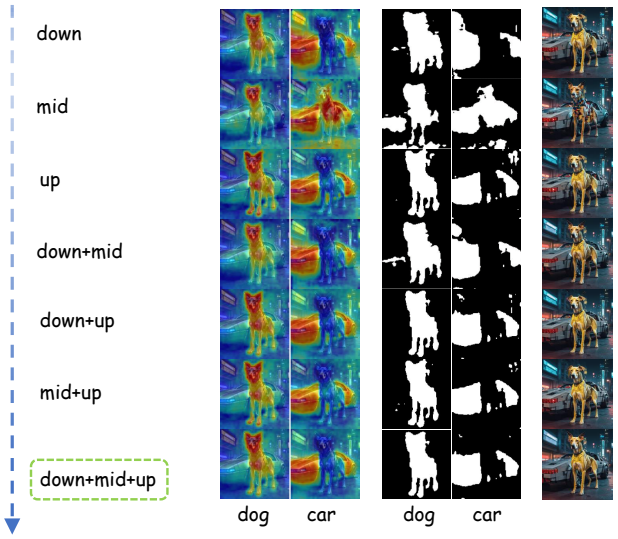

Figure A4. Comparison of the effects of different cross-attention layer selections in binary mask extraction. It can be observed that extracting the 32×32 resolution cross-attention map from the down, up, and mid blocks yields the cleanest and most accurate subject mask.

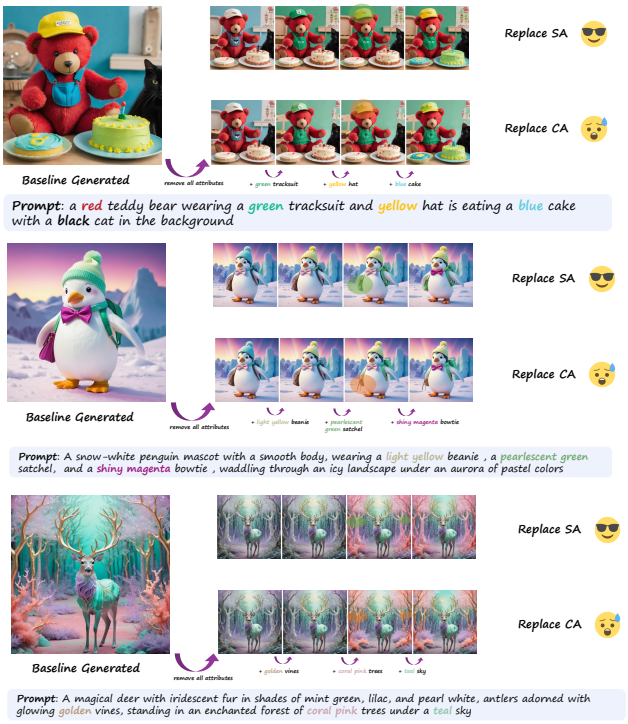

Figure A5. Comparative analysis of attention map types in image generation. As indicated by the marked areas, replacing the cross-attention map results in weaker editing effects, making it less effective at accurately assigning attributes to the subject. In contrast, replacing the self-attention map allows for more precise attribute assignment in the same scenario.