Expressive Text-to-Image Generation with Rich Text Appendix

In this appendix, we provide additional experimental results and details. In section A, we show the images generated by our model, Attend-and-Excite [2], Prompt-to-Prompt [3], and InstructPix2Pix [1] with various RGB colors, local styles, and detailed descriptions via footnotes. In section B, we provide additional details on the implementation and evaluation.

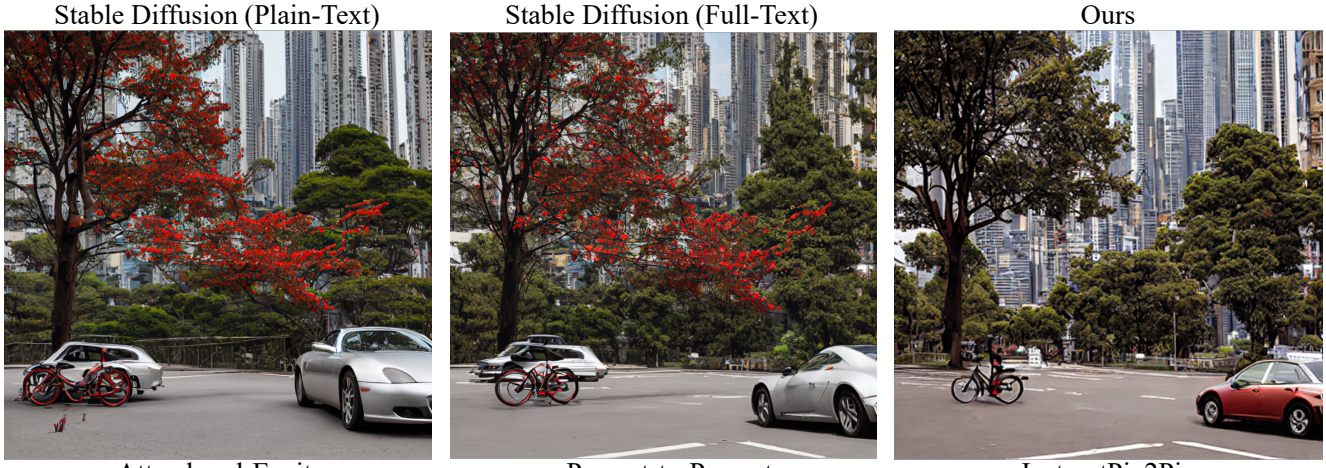
A. Additional Results

In this section, we first show additional results of rich-text-to-image generation on complex scene synthesis (Figures 15, 16, and 17), precise color rendering (Figures 18, 19, and 20), local style control (Figures 21 and 22), and explicit token re-weighting (Figure 23, 24, and 25). We also show an ablation study of the averaging and maximizing operations across tokens to obtain token maps in Figure 26. We present additional results compared with a composition-based baseline in Figure 27. Last, we show an ablation of the hyperparameters of our baseline method InstructPix2Pix [1] on the local style generation application in Figure 28.

A car[1] driving on the road. A bicycle[2] nearby a tree[3]. A cityscape[4] in the background.

[1]A sleek sports car gleams on the road in the sunlight, with its aerodynamic curves and polished finish catching the light. [2]A bicycle with rusted frame and worn tires.
[3]A dead tree with a few red apples on it. [4]A bustling Hongkong cityscape with towering skyscrapers.

Stable Diffusion (Plain-Text) | Stable Diffusion (Full-Text) | Ours



Attend-and-Excite | Prompt-to-Prompt | InstructPix2Pix

Figure 1. **Additional results of the footnote.** We show the generation from a complex description of a garden. Note that all the methods except for ours fail to generate accurate details of the mansion and fountain as described.

A lush garden[1] with a fountain[2]. A grand mansion[3] in the background.

[1]A garden is full of vibrant colors with a variety of flowers.

[2]A fountain made of white marble with multiple tiers. The tiers are intricately carved with various designs.

[3]An impressive two-story mansion with a royal exterior, white columns, and tile-made roof. The mansion has numerous windows, each adorned with white curtains.

Stable Diffusion (Plain-Text) | Stable Diffusion (Full-Text) | Ours

Attend-and-Excite | Prompt-to-Prompt | InstructPix2Pix

Figure 2. **Additional results of the footnote.** We show the generation from a complex description of a garden. Note that all the methods except for ours fail to generate accurate details of the mansion and fountain as described.

Figure 3. **Additional results of the footnote.** We show the generation from a complex description of an office. Note that all the methods except ours fail to generate accurate window overlooks and colorful binders as described.

Figure 4. **Additional results of the font color.** We show the generation of different objects with colors from the *Common category*. Prompt-to-Prompt has a large failure rate of respecting the given color name, while InstructPix2Pix tends to color the background and irrelevant objects.

Figure 5. **Additional results of the font color.** We show the generation of different objects with colors from the *HTML category*. Both methods fail to generate the precise color, and InstructPix2Pix tends to color the background and irrelevant objects.

Figure 6. **Additional results of the font color.** We show the generation of different objects with colors from the *RGB category*. Both baseline methods cannot interpret the RGB values correctly.

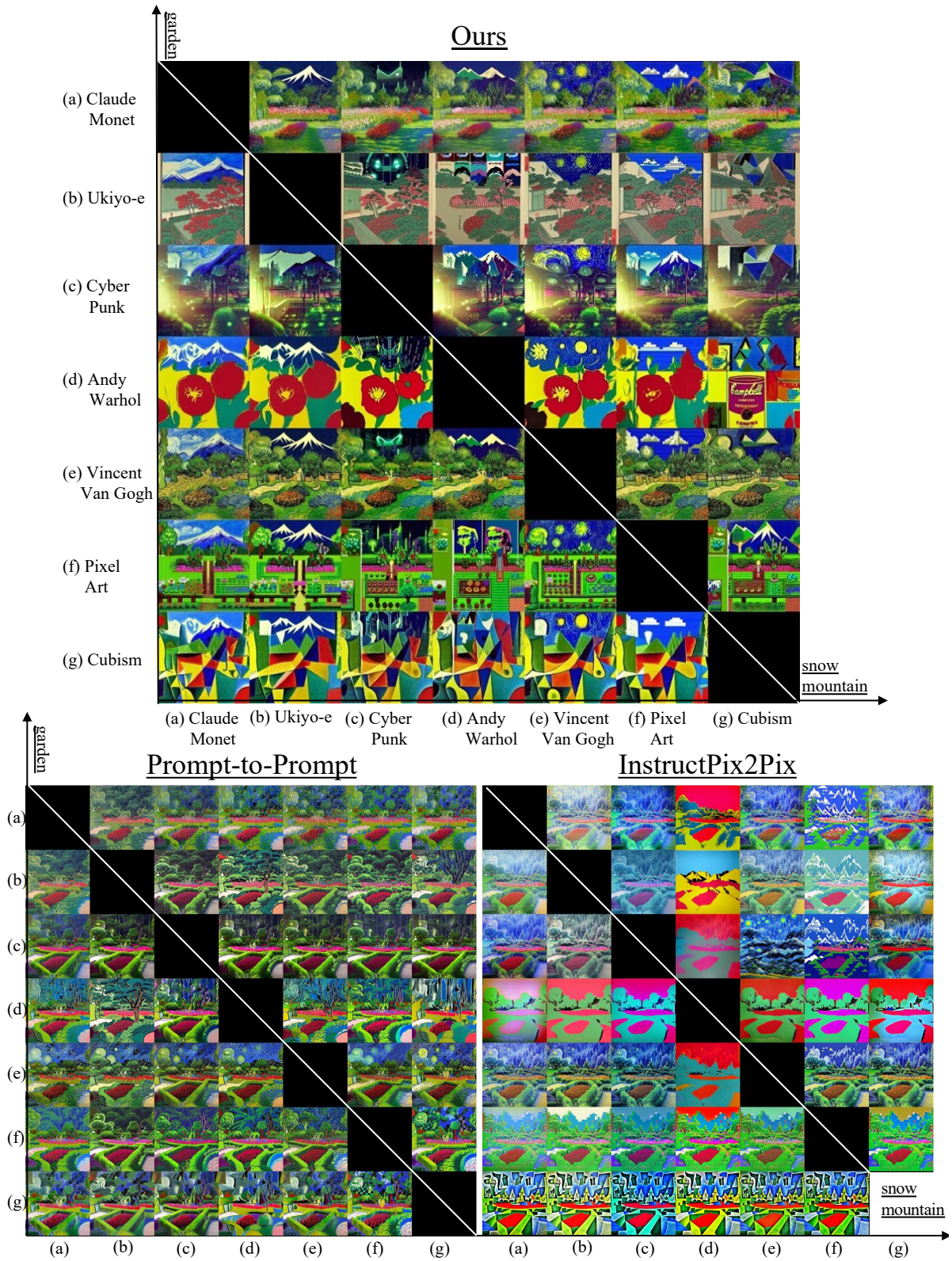


Figure 7. **Additional results of the font style.** We show images generated with different style combinations and prompt "a beautiful garden in front of a snow mountain". Each row contains "snow mountain" in 7 styles, and each column contains "garden" in 7 styles. Only our method can generate distinct styles for both objects.

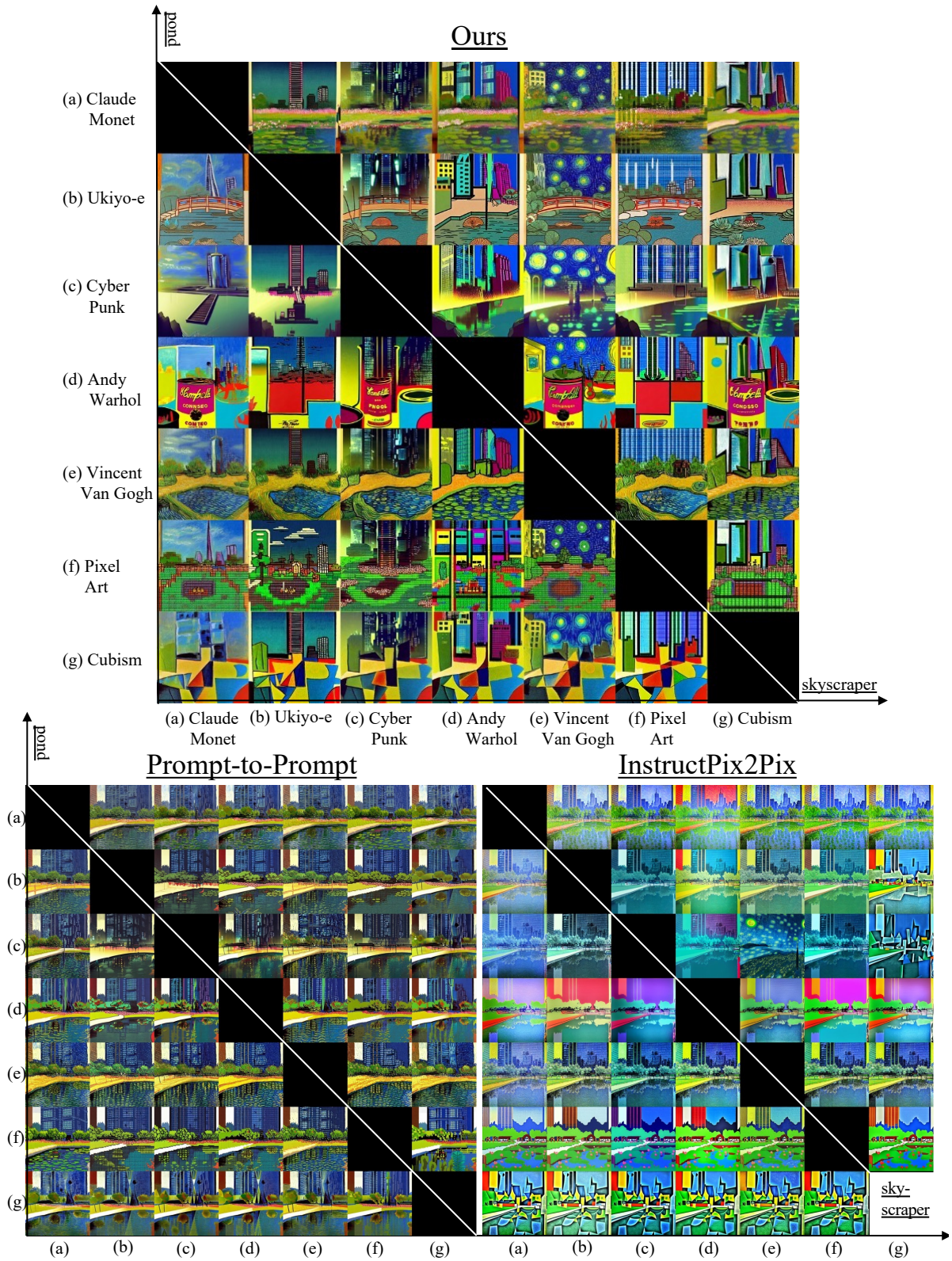


Figure 8. **Additional results of the font style.** We show images generated with different style combinations and prompt "a small pond surrounded by skyscraper". Each row contains "skyscraper" in 7 styles, and each column contains "pond" in 7 styles. Only our method can generate distinct styles for both objects.

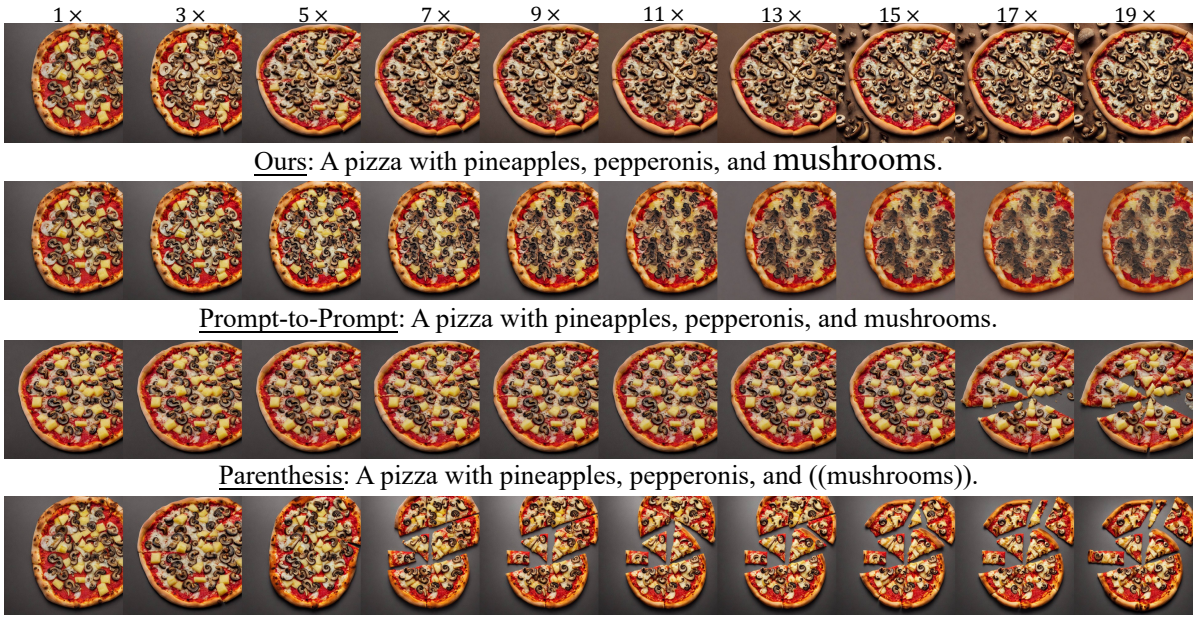


Figure 9. **Additional results of font sizes.** We use a token weight evenly sampled from 1 to 20 for the word 'mushrooms' with our method and Prompt-to-Prompt. For parenthesis and repeating, we show results by repeating the word 'mushrooms' and adding parentheses to the word 'mushrooms' for 1 to 10 times. Prompt-to-Prompt suffers from generating artifacts. Heuristic methods are not effective.

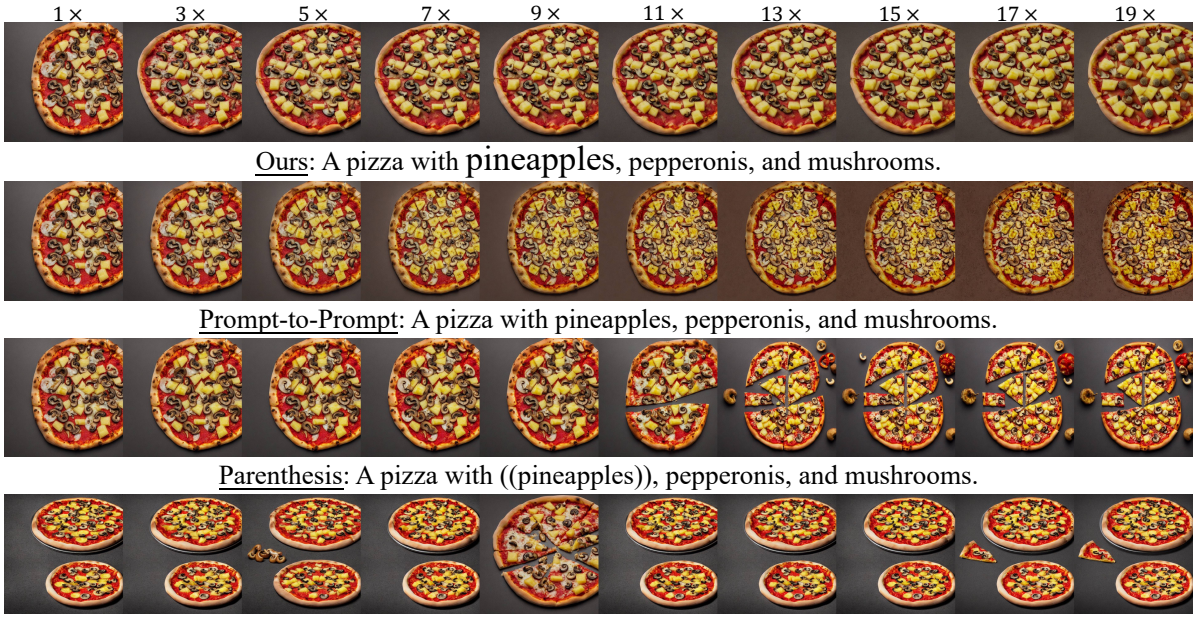


Figure 10. **Additional results of font sizes.** We use a token weight evenly sampled from 1 to 20 for the word 'pineapples' with our method and Prompt-to-Prompt. For parenthesis and repeating, we show results by repeating the word 'pineapples' and adding parentheses to the word 'pineapples' for 1 to 10 times. Prompt-to-Prompt suffers from generating artifacts. Heuristic methods are not effective.

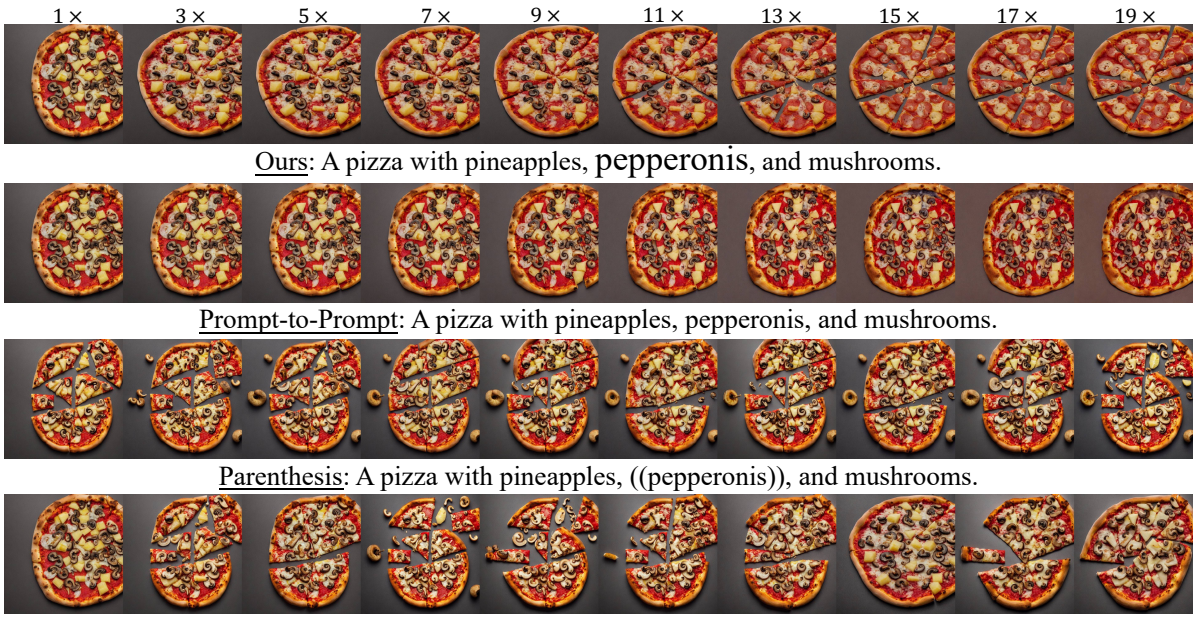


Figure 11. **Additional results of font sizes.** We use a token weight evenly sampled from 1 to 20 for the word 'pepperonis' with our method and Prompt-to-Prompt. For parenthesis and repeating, we show results by repeating the word 'pepperonis' and adding parentheses to the word 'pepperonis' for 1 to 10 times. Prompt-to-Prompt suffers from generating artifacts. Heuristic methods are not effective.

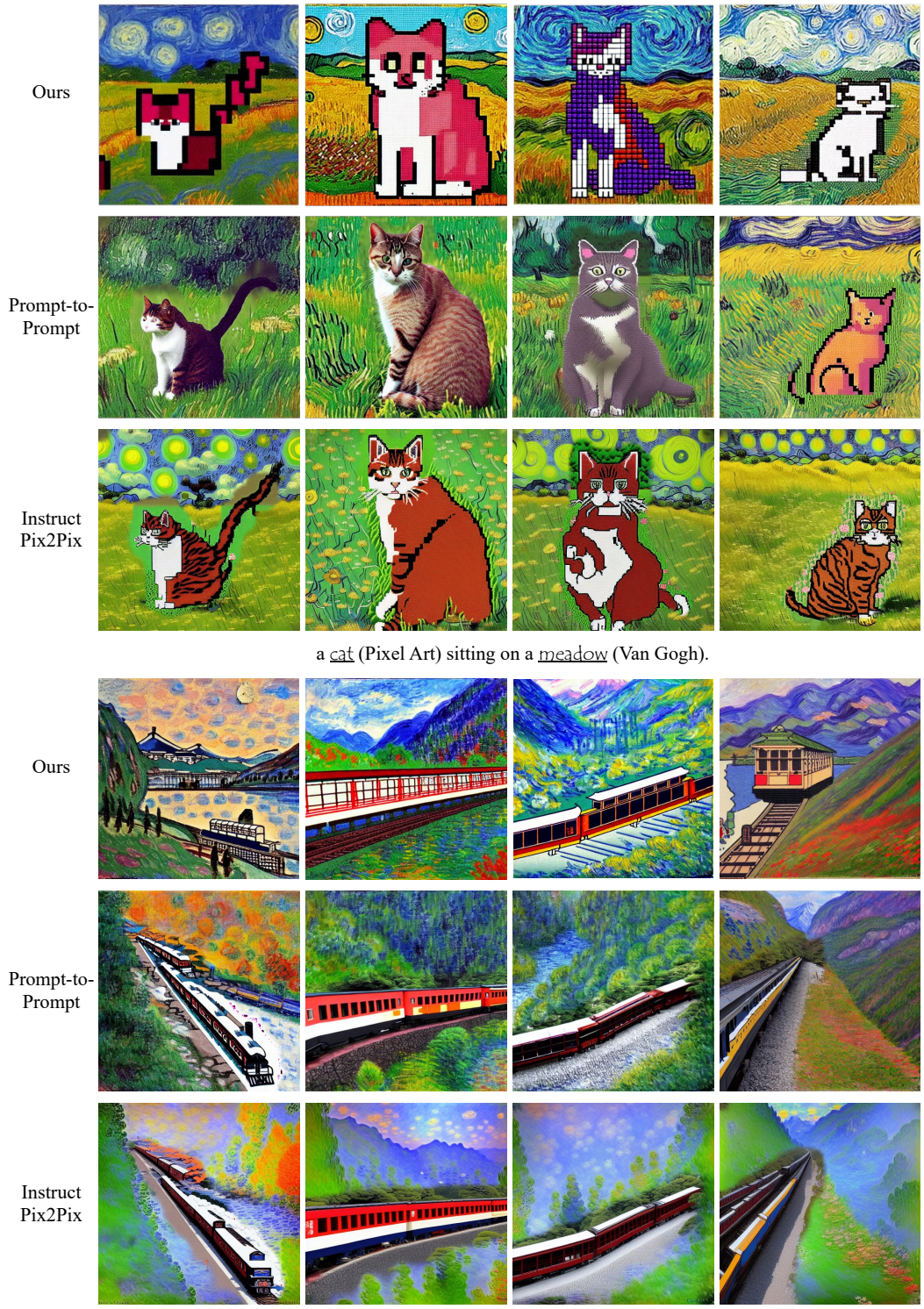


Figure 12. **Comparison with a simple composed-based method using different random seeds.** Since the methods like Prompt-to-Prompt [3] cannot generate multiple styles on a single image, one simple idea to fix this is to apply the methods on two regions separately and compose them using the token maps. However, we show that this leads to sharp changes and artifacts at the boundary areas.

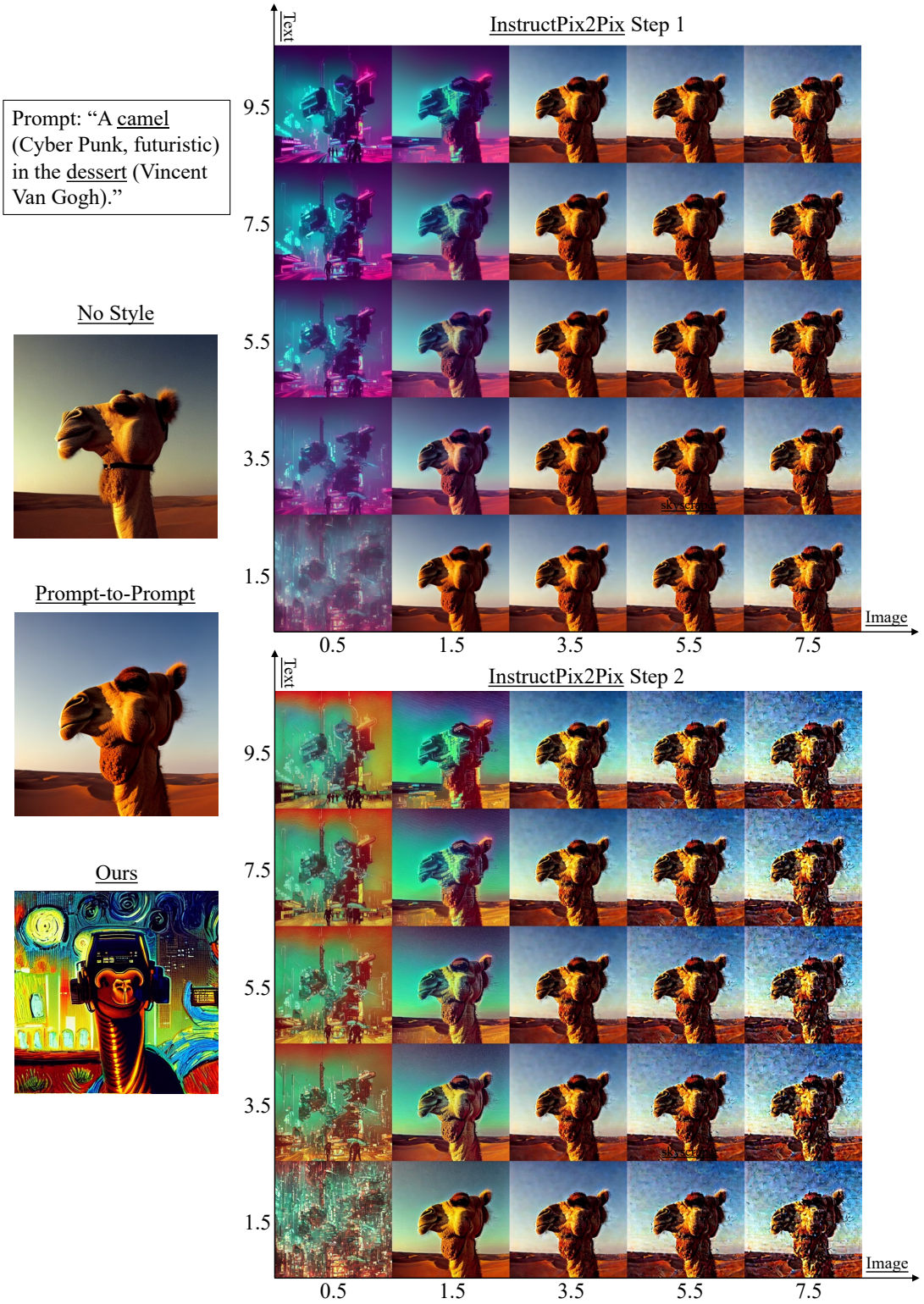


Figure 13. **Ablation of the classifier free guidance of InstructPix2Pix.** We show that InstruxtPix2Pix fails to generate both styles with different image and text classifier-free guidance (cfg) weights. When image-cfg is low, the desert is lost after the first editing. We use image-cfg= 1.5 and text-cfg= 7.5 in our experiment.

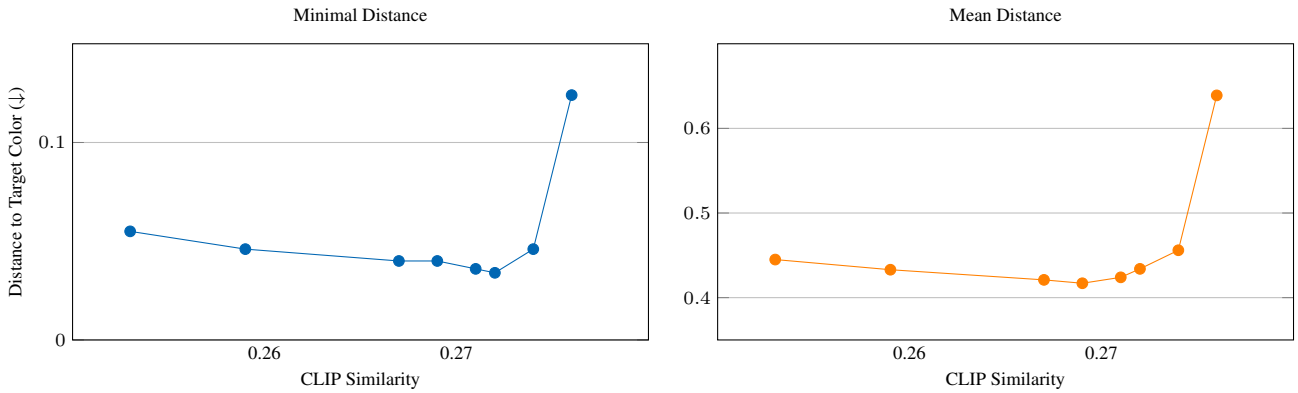


Figure 14. **Ablation on the hyperparameter λ in Equation (7).** We report the trade-off of CLIP similarity and color distance achieved by sweeping the strength of color optimization λ.

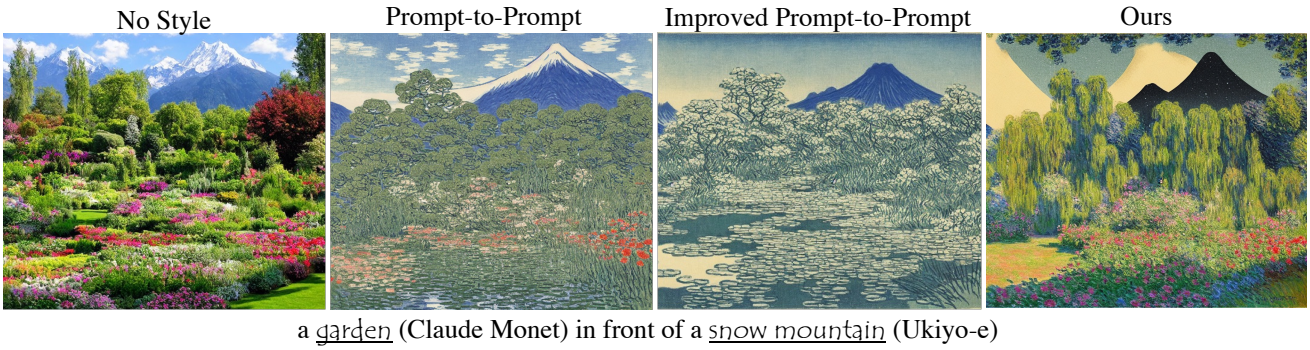


Figure 15. **Improved Prompt-to-Prompt.** Further constraining the attention maps for styles does not resolve the mixed style issue.

Ablation of the color guidance weight. Changing the guidance strength λ allows us to control the trade-off between *fidelity* and *color precision*. To evaluate the fidelity of the image, we compute the CLIP score between the generation and the plain text prompt. We plot the CLIP similarity vs. color distance in Figure 14 by sweeping λ from 0 to 20. Increasing the strength always reduces the CLIP similarity as details are removed to satisfy the color objective. We find that larger λ first reduces and then increases the distances due to the optimization divergence.

Constrained Prompt-to-Prompt. The original Attention Refinement proposed in Prompt-to-Prompt [3] does not apply any constraint to newly added tokens' attention maps, which may be the reason that it fails with generating distinct styles. Therefore, we attempt to improve Prompt-to-Prompt by injecting the cross-attention maps for the newly added style tokens. For example, in Figure 15, we use the cross attention map of "garden" for the style "Claude Monet". However, the method still produces a uniform style.

Human Evaluation We conduct a user study on crowdsourcing platforms. We show human annotators a pair of generated images and ask them which image more accurately expresses the reference color, artistic styles, or supplementary descriptions. To compare ours with each baseline, we show 135 font color pairs, 167 font style pairs, and 21 footnote pairs to three individuals and receive 1938 responses. As shown in the table below, our method is chosen more than 80% of the time over both baselines for producing more precise color and content given the long prompt and more than 65% of the time for rendering more accurate artistic styles. We will include a similar study at a larger scale in our revision.

Table 1. Human evaluation results.

	Color	Style	Footnote
Ours vs. Prompt-to-Prompt	88.2%	65.2%	84.1%
Ours vs. InstructPix2Pix	80.7%	69.8%	87.3%

B. Additional Details

This section details our quantitative evaluation of the font style and font color experiments.

Font style evaluation. To compute the local CLIP scores at each local region to evaluate the stylization quality, we need to create test prompts with multiple objects and styles. We use seven popular styles that people use to describe the artistic styles of the generation, as listed below. Note that for each style, to achieve the best quality, we also include complementary information like the name of famous artists in addition to the style.

```
styles = [
    'Claud Monet, impressionism, oil on canvas',
    'Ukiyoe',
    'Cyber Punk, futuristic',
    'Pop Art, masterpiece, Andy Warhol',
    'Vincent Van Gogh',
    'Pixel Art, 8 bits, 16 bits',
    'Abstract Cubism, Pablo Picasso'
]
```

We also manually create a set of prompts, where each contains a combination of two objects, for stylization, resulting in 420 prompts in total. We generally confirm that Stable Diffusion [4] can generate the correct combination, as our goal is not to evaluate the compositionality of the generation as in DrawBench [5]. The prompts and the object tokens used for our method are listed below.

```
candidate_prompts = [
    'A garden with a mountain in the distance.': ['garden', 'mountain'],
    'A fountain in front of an castle.': ['fountain', 'castle'],
    'A cat sitting on a meadow.': ['cat', 'meadow'],
    'A lighthouse among the turbulent waves in the night.': ['lighthouse', 'turbulent waves'],
    'A stream train on the mountain side.': ['stream train', 'mountain side'],
    'A cactus standing in the desert.': ['cactus', 'desert'],
    'A dog sitting on a beach.': ['dog', 'beach'],
    'A solitary rowboat tethered on a serene pond.': ['rowboat', 'pond'],
    'A house on a rocky mountain.': ['house', 'mountain'],
    'A rustic windmill on a grassy hill.': ['rustic', 'hill'],
]
```

Font color evaluation. To evaluate precise color generation capacity, we create a set of prompts with colored objects. We divide the potential colors into three levels according to the difficulty of text-to-image generation models to depend on. The *easy-level* color set contains 17 basic color names that these models generally understand. The complete set is as below.

```
COLORS_easy = {
    'brown': [165, 42, 42],
    'red': [255, 0, 0],
    'pink': [253, 108, 158],
    'orange': [255, 165, 0],
    'yellow': [255, 255, 0],
    'purple': [128, 0, 128],
    'green': [0, 128, 0],
    'blue': [0, 0, 255],
    'white': [255, 255, 255],
    'gray': [128, 128, 128],
    'black': [0, 0, 0],
    'crimson': [220, 20, 60],
    'maroon': [128, 0, 0],
    'cyan': [0, 255, 255],
```

```
    'azure': [240, 255, 255],
    'turquoise': [64, 224, 208],
    'magenta': [255, 0, 255],
}
```

The *medium-level* set contain color names that are selected from the HTML color names [1]. These colors are also standard to use for website design. However, their names are less often occurring in the image captions, making interpretation by a text-to-image model challenging. To address this issue, we also append the coarse color category when possible, e.g., "Chocolate" to "Chocolate brown". The complete list is below.

```
COLORS_medium = {
    'Fire Brick red': [178, 34, 34],
    'Salmon red': [250, 128, 114],
    'Coral orange': [255, 127, 80],
    'Tomato orange': [255, 99, 71],
    'Peach Puff orange': [255, 218, 185],
    'Moccasin orange': [255, 228, 181],
    'Goldenrod yellow': [218, 165, 32],
    'Olive yellow': [128, 128, 0],
    'Gold yellow': [255, 215, 0],
    'Lavender purple': [230, 230, 250],
    'Indigo purple': [75, 0, 130],
    'Thistle purple': [216, 191, 216],
    'Plum purple': [221, 160, 221],
    'Violet purple': [238, 130, 238],
    'Orchid purple': [218, 112, 214],
    'Chartreuse green': [127, 255, 0],
    'Lawn green': [124, 252, 0],
    'Lime green': [50, 205, 50],
    'Forest green': [34, 139, 34],
    'Spring green': [0, 255, 127],
    'Sea green': [46, 139, 87],
    'Sky blue': [135, 206, 235],
    'Dodger blue': [30, 144, 255],
    'Steel blue': [70, 130, 180],
    'Navy blue': [0, 0, 128],
    'Slate blue': [106, 90, 205],
    'Wheat brown': [245, 222, 179],
    'Tan brown': [210, 180, 140],
    'Peru brown': [205, 133, 63],
    'Chocolate brown': [210, 105, 30],
    'Sienna brown': [160, 82, 4],
    'Floral White': [255, 250, 240],
    'Honeydew White': [240, 255, 240],
}
```

The *hard-level* set contains 50 randomly sampled RGB triplets as we aim to generate objects with arbitrary colors indicated in rich texts. For example, the color can be selected by an RGB slider.

```
COLORS_hard = {
    'color of RGB values [68, 17, 237]': [68, 17, 237],
    'color of RGB values [173, 99, 227]': [173, 99, 227],
    'color of RGB values [48, 131, 172]': [48, 131, 172],
```

[1] https://simple.wikipedia.org/wiki/Web_color

```
    'color of RGB values [198, 234, 45]': [198, 234, 45],
    'color of RGB values [182, 53, 74]': [182, 53, 74],
    'color of RGB values [29, 139, 118]': [29, 139, 118],
    'color of RGB values [105, 96, 172]': [105, 96, 172],
    'color of RGB values [216, 118, 105]': [216, 118, 105],
    'color of RGB values [88, 119, 37]': [88, 119, 37],
    'color of RGB values [189, 132, 98]': [189, 132, 98],
    'color of RGB values [78, 174, 11]': [78, 174, 11],
    'color of RGB values [39, 126, 109]': [39, 126, 109],
    'color of RGB values [236, 81, 34]': [236, 81, 34],
    'color of RGB values [157, 69, 64]': [157, 69, 64],
    'color of RGB values [67, 192, 60]': [67, 192, 60],
    'color of RGB values [181, 57, 181]': [181, 57, 181],
    'color of RGB values [71, 240, 139]': [71, 240, 139],
    'color of RGB values [34, 153, 226]': [34, 153, 226],
    'color of RGB values [47, 221, 120]': [47, 221, 120],
    'color of RGB values [219, 100, 27]': [219, 100, 27],
    'color of RGB values [228, 168, 120]': [228, 168, 120],
    'color of RGB values [195, 31, 8]': [195, 31, 8],
    'color of RGB values [84, 142, 64]': [84, 142, 64],
    'color of RGB values [104, 120, 31]': [104, 120, 31],
    'color of RGB values [240, 209, 78]': [240, 209, 78],
    'color of RGB values [38, 175, 96]': [38, 175, 96],
    'color of RGB values [116, 233, 180]': [116, 233, 180],
    'color of RGB values [205, 196, 126]': [205, 196, 126],
    'color of RGB values [56, 107, 26]': [56, 107, 26],
    'color of RGB values [200, 55, 100]': [200, 55, 100],
    'color of RGB values [35, 21, 185]': [35, 21, 185],
    'color of RGB values [77, 26, 73]': [77, 26, 73],
    'color of RGB values [216, 185, 14]': [216, 185, 14],
    'color of RGB values [53, 21, 50]': [53, 21, 50],
    'color of RGB values [222, 80, 195]': [222, 80, 195],
    'color of RGB values [103, 168, 84]': [103, 168, 84],
    'color of RGB values [57, 51, 218]': [57, 51, 218],
    'color of RGB values [143, 77, 162]': [143, 77, 162],
    'color of RGB values [25, 75, 226]': [25, 75, 226],
    'color of RGB values [99, 219, 32]': [99, 219, 32],
    'color of RGB values [211, 22, 52]': [211, 22, 52],
    'color of RGB values [162, 239, 198]': [162, 239, 198],
    'color of RGB values [40, 226, 144]': [40, 226, 144],
    'color of RGB values [208, 211, 9]': [208, 211, 9],
    'color of RGB values [231, 121, 82]': [231, 121, 82],
    'color of RGB values [108, 105, 52]': [108, 105, 52],
    'color of RGB values [105, 28, 226]': [105, 28, 226],
    'color of RGB values [31, 94, 190]': [31, 94, 190],
    'color of RGB values [116, 6, 93]': [116, 6, 93],
    'color of RGB values [61, 82, 239]': [61, 82, 239],
}
```

To write a complete prompt, we create a list of 12 objects and simple prompts containing them as below. The objects would naturally exhibit different colors in practice, such as “flower”, “gem”, and “house”.

```
candidate_prompts = [
    'a man wearing a shirt': 'shirt',
```

```
    'a woman wearing pants': 'pants',
    'a car in the street': 'car',
    'a basket of fruit': 'fruit',
    'a bowl of vegetable': 'vegetable',
    'a flower in a vase': 'flower',
    'a bottle of beverage on the table': 'bottle beverage',
    'a plant in the garden': 'plant',
    'a candy on the table': 'candy',
    'a toy on the floor': 'toy',
    'a gem on the ground': 'gem',
    'a church with beautiful landscape in the background': 'church',
]
```

Baseline. We compare our method quantitatively with two strong baselines, Prompt-to-Prompt [3] and InstructPix2Pix [1]. The prompt refinement application of Prompt-to-Prompt allows adding new tokens to the prompt. We use plain text as the base prompt and add color or style to create the modified prompt. InstructPix2Pix [1] allows using instructions to edit the image. We use the image generated by the plain text as the input image and create the instructions using templates "turn the *[object]* into the style of *[style]*," or "make the color of *[object]* to be *[color]*". For the stylization experiment, we apply two instructions in both parallel (InstructPix2Pix-para) and sequence (InstructPix2Pix-seq). We tune both methods on a separate set of manually created prompts to find the best hyperparameters. In contrast, it is worth noting that our method *does not* require hyperparameter tuning.

Running time. The inference time of our models depends on the number of attributes added to the rich text since we implement each attribute with an independent diffusion process. In practice, we always use a batch size of 1 to make the code compatible with low-resource devices. In our experiments on an NVIDIA RTX A6000 GPU, each sampling based on the plain text takes around 5.06 seconds, while sampling an image with two styles takes around 8.07 seconds, and sampling an image with our color optimization takes around 13.14 seconds.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[2] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *arXiv preprint arXiv:2301.13826*, 2023.

[3] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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