Towards partial fulfillment for Undergraduate Degree Level Programme Bachelor of Technology in Computer Science and Engineering

An End Sem Project Evaluation Report on:

High-Level Text to Image : Transforming Descriptions into Visual Art

Prepared by:

Admission No. Student Name

_____

_____

_____

_____

Class: B.Tech. IV (Computer Science and Engineering) 7th Semester

Year: 2024-2025

Guided by: Dr. Mukesh A. Zaveri

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY,
SURAT - 395007 (GUJARAT, INDIA)

Student Declaration

This is to certify that the work described in this seminar report has been actually carried out and implemented by me i.e.

Sr.	Admission No.	Student Name
1	U20CS057	Sidhdharaj Dafda
2	U20CS082	Y. Venkata Ramana
3	U20CS058	Sarwade Swapnil
4	U21CS114	Dinesh Penigalapati

Neither the source code there in, nor the content of the seminar report have been copied or downloaded from any other source. I understand that my result grades would be revoked if later it is found to be so.

Signature of the Students:

Sr.	Student Name.	Signature of Student
1	Sidhdharaj Dafda	
2	Y.Venkata Ramana	
3	Sarwade Swapnil	
4	Dinesh Penigalapati	

Certificate

This is to certify that the seminar report entitled **High-Level Text to Image : Transforming Descriptions into Visual Art** is prepared and presented by

Sr.	Admission No.	Student Name
1	U20CS057	Sidhdharaj Dafda
2	U20CS082	Y.Venkata Ramana
3	U20CS058	Sarwade Swapnil
4	U21CS114	Dinesh Penigalapati

Final Year of Computer Science and Engineering and their work is satisfactory.

Signature:

Supervisor/s JURY HEAD OF DEPARTMENT

Acknowledgment

We wish to express our sincere gratitude and appreciation to several individuals and institutions whose support and guidance have been instrumental in the completion of this final year project on High-Level Text to Image : Transforming Descriptions into Visual Art.

First and foremost, we extend our deepest thanks to Dr. Mukesh A. Zaveri for his invaluable insights, unwavering encouragement, and the wealth of knowledge he shared throughout this endeavor. His guidance has been a beacon, illuminating our path in exploring the intricate field of text-to-image synthesis and generative models.

We would also like to acknowledge the support of Head of Department (HOD) Dr. Mukesh Zaveri, professors and mentors at Sardar Vallabhbhai National Institute of Technology (SVNIT), Surat. Their teachings and expertise have been instrumental in shaping my understanding of the subject matter. Their dedication to fostering a spirit of inquiry and research has been a constant source of inspiration.

Our gratitude is also extended to the academic institutions and libraries, including SVNIT, that provided access to extensive resources, enabling us to delve deeply into the study of text-to-image synthesis and generative models.

Furthermore, We appreciate the encouragement and patience of our peers and colleagues who provided their insights and feedback during the course of out research.

Last but not least, our heartfelt thanks go to our friends and families who stood by us with unwavering support and understanding, making this academic pursuit at SVNIT a truly enriching experience.

Sidhdharaj Dafda
Y.Venkata Ramana
Sarwade Swapnil
Dinesh Penigalapati

Abstract

Creating images from text descriptions has become an exciting field in AI, but using plain text often limits how much control users have over the final image. For example, it's difficult to specify exact details like precise colors, the importance of certain words, or complex scene layouts. To make this process easier and more flexible, we introduce a new approach that uses a rich-text editor as the input. This editor lets users customize text with attributes like font size, style, color, and even embedded images, allowing for more detailed and intuitive control.

Our method extracts these rich-text details and uses them to guide image generation. By breaking the text into regions based on attention maps, we apply specific styles and constraints to different parts of the image, ensuring they match the user's input. This helps achieve precise results, whether it's rendering accurate colors, emphasizing specific parts of the text, or creating detailed regions within an image.

We've tested this approach with various examples and found that it not only makes generating and editing images easier but also produces better results compared to existing methods. This new way of working with rich-text inputs opens up exciting possibilities for more creative and accurate image generation.

Keywords: Text-to-image synthesis, diffusion models, rich-text control, generative AI, image editing, attention maps, customizable prompts.

Contents

List of Figures

1 Introduction

The development of large-scale text-to-image generative models (Ramesh et al., 2021; Saharia et al., 2022; Rombach et al., 2022; Kang et al., 2023) has transformed the way we generate images, ushering in a new era of creative possibilities. These models provide incredible flexibility, enabling users to control the generation process using visual cues (Balaji et al., 2022; Gafni et al., 2022; Zhang and Agrawala, 2023) and text inputs (Brooks et al., 2023; Hertz et al., 2023). However, most existing approaches rely solely on plain text encoded by pretrained language models to guide image generation.

In everyday life, tasks like writing blogs or editing essays rarely rely on plain text alone. Instead, rich-text editors, which offer versatile formatting options, are the preferred tools (Colorado State University, 2012; Witten et al., 2009). In this work, we aim to bridge this gap by introducing accessible and precise control over image synthesis through the use of rich-text editors.

Figure 1. Plain text (left image) vs. Rich text (right image)

Fig. 1: Plain text (left image) vs. Rich text (right image) Our method allows a user to describe an image using a rich text editor that supports various text attributes like font family, size, color, and footnote. Given these text attributes extracted from rich-text prompts, our method enables precise control of text-to-image synthesis regarding colors, styles, and object details compared to plain text.

Rich-text editors offer unique advantages for specifying conditions beyond plain text. For instance, font colors can precisely define object colors, which is challenging to de-

scribe with plain text alone. Colors like "olive" or "orange" often have ambiguous meanings, and plain text struggles to convey RGB or Hex color values. By leveraging font colors, users can easily guide the generation process, such as specifying a particular shade of yellow for a marble statue (as demonstrated in Figure 1).

Moreover, rich-text editors allow intuitive word-level customizations. Font sizes can adjust the emphasis on specific tokens, while font styles can convey artistic attributes, such as rendering a region in the style of Ukiyo-e. Embedded images can also act as references, enabling highly personalized and context-aware image generation.

While one might convert rich-text attributes into plain text and feed them into existing methods (Rombach et al., 2022; Hertz et al., 2023; Brooks et al., 2023), this approach has significant limitations. Lengthy prompts describing multiple objects with distinct visual attributes often confuse text encoders, leading to uniform styles or mixed attributes within the generated image (Chefer et al., 2023). As a result, these methods struggle to maintain intricate details and distinct styles across different parts of an image.

To address these challenges, we propose a novel approach that decomposes rich-text prompts into two components: (1) a short plain-text prompt and (2) multiple region-specific prompts with associated text attributes. First, we use self- and cross-attention maps from a standard diffusion process with the plain-text prompt to identify specific regions for each word. Then, we generate region-specific prompts based on rich-text attributes. For example, the word "mountain" with a font style of "Ukiyo-e" translates to the prompt "mountain in the style of Ukiyo-e." For attributes like RGB colors, we iteratively refine regions using targeted guidance to achieve the desired appearance. This process ensures that regions with plain-text formatting remain consistent with the plain-text results, and only the specified attributes are modified.

Our method generates more precise colors, distinct styles, and intricate details compared to plain text-based approaches. To support this, we build a benchmark for rich-text-to-image generation, including a diverse set of prompts with varying colors, styles, and formats, providing new challenges for future research.

Additionally, we introduce the capability to use embedded images as rich-text attributes and demonstrate how this enhances control over image generation. We also explore the potential for editing real images using rich text, employing techniques like diffusion inversion and segmentation to achieve precise and intuitive editing results.

This work builds on preliminary research (Ge et al., 2023) and introduces several key contributions:

- A benchmark for evaluating rich-text-to-image generation, enabling quantitative comparisons with complex prompts.
- New methods for using embedded images as attributes to guide generation.
- Techniques for rich-text-based editing of real images, offering enhanced precision and control.

1.1 Applications

The rich-text-based text-to-image generation approach opens up exciting applications across various fields. In creative industries, it enables artists and designers to generate personalized artwork, illustrations, and concept art by specifying styles, colors, and embedded references directly within the text. It simplifies educational content creation by allowing teachers and students to produce interactive visuals for learning materials. Marketers and advertisers can benefit by creating brand-specific visuals and personalized campaigns with precise control over design elements. In gaming and entertainment, developers can quickly prototype scenes, characters, or assets with detailed styling instructions. This method also facilitates product mockups, fashion designs, and interior layouts, offering a fast and customizable way to visualize ideas. Additionally, it enhances image editing by allowing users to refine or alter specific parts of an image without affecting the overall content. Beyond these, it supports accessibility by converting descriptive inputs into visuals for those with visual impairments and aids researchers by providing benchmarks and data for model improvement. With its flexibility and user-friendly design, this approach bridges the gap between creativity and technology, making complex image generation more accessible and practical.

1.2 Motivation

The motivation for this project arises from the growing demand for more intuitive and precise ways to interact with AI-based text-to-image generation models. While existing methods have made remarkable progress, they often fall short when it comes to allowing users to fully articulate their creative visions. Plain text, as an input format, is inherently limited—it's challenging to specify precise details like exact colors, emphasize certain aspects of the prompt, or design intricate layouts. This limitation not only restricts the creative potential of artists and designers but also makes it difficult for everyday users to achieve the results they want. Rich-text editors, on the other hand, are tools we use in our daily lives for writing, editing, and formatting, offering intu-

itive options like changing font styles, sizes, colors, and adding images. These features naturally align with the need for more control and customization in image generation.

By introducing rich-text editors as an interface for text-to-image generation, this project aims to address these limitations and provide a user-friendly, precise, and highly customizable solution. The goal is to bridge the gap between creative intent and technological capability, empowering users to easily translate their ideas into visuals. This motivation is driven not only by the need to improve the user experience but also to expand the practical applications of AI in fields like art, education, marketing, and design. Whether it's generating personalized artwork, creating interactive educational content, or designing brand-specific visuals, this project seeks to make the process of image creation more accessible, expressive, and aligned with individual needs.

1.3 Objectives

1. **Enhance Customization in Text-to-Image Generation:** Develop a method that allows users to have precise control over image generation by incorporating rich-text attributes such as font style, size, color, and embedded images. This will enable users to specify detailed visual elements, ensuring the generated images align closely with their creative intentions.

2. **Bridge the Gap Between Plain Text and Rich-Text Inputs:** Create a system that seamlessly integrates rich-text inputs into the text-to-image generation process, overcoming the limitations of traditional plain text prompts. The aim is to provide a more intuitive and flexible interface for users to interact with AI models.

3. **Improve Image Accuracy and Detail:** Ensure that the generated images accurately reflect the user's input by applying region-based control. This approach will allow for better management of color, style, and layout, resulting in highly detailed and contextually relevant images.

4. **Facilitate Real-Time Image Editing:** Introduce the capability for users to not only generate new images but also edit existing ones using rich-text instructions. This will enable precise control over specific regions of the image, such as changing colors or styles without affecting the overall composition.

5. **Create a Benchmark for Rich-Text-Based Image Generation:** Build a comprehensive dataset and evaluation framework for testing and assessing the performance of text-to-image models using rich-text inputs. This benchmark will help guide future research and development in this area.

1.4 Organization of Project Report

Our project enhances text-to-image generation through rich text descriptions, addressing limitations of plain text such as specifying RGB color values and word importance. A rich-text editor supporting various formats was developed to allow for precise control of text-to-image synthesis.

Advanced techniques, like localized style control and explicit token reweighting, were integrated into a comprehensive framework, significantly improving the accuracy and detail of generated images. The system was deployed as a web application using Flask and TensorFlow, enabling users to input rich-text descriptions and receive detailed visual outputs.

Future improvements involve refining the model with larger datasets, optimizing parameters, implementing stacking techniques,and enhancing the user interface.

2 Literature Survey

In this chapter, we review the existing body of work in the field of text-to-image generation, which combines the power of natural language processing and computer vision to create images based on textual descriptions. Text-to-image synthesis has rapidly evolved, and various techniques have emerged to improve the quality, precision, and control of the generated images. This chapter explores these techniques, from early generative models to the latest diffusion models, examining their strengths, limitations, and the challenges that remain in the field. Furthermore, we will look at recent advancements that enhance the flexibility of text-to-image systems, such as integrating rich-text inputs that allow users to have more control over the generated outputs.

The chapter is structured into several key sections to provide a comprehensive overview. It begins with the theoretical background, covering foundational concepts that are crucial for understanding the methodologies discussed later. Then, we categorize and analyze the different approaches to text-to-image generation, focusing on their architectural differences and key contributions. We also discuss the challenges these methods face and highlight recent innovations in the field. Finally, we wrap up with a summary that synthesizes the insights from the literature, helping to contextualize the contributions of this project in advancing the field of text-to-image synthesis.

2.1 Theoretical Background

To better understand the methods used in text-to-image generation, it is essential to first introduce some key concepts and models that form the foundation of this research. The following topics are central to the techniques we explore:

2.1.1 Generative Models:

At the heart of text-to-image generation lies generative modeling, which is the task of learning how to generate data from a given distribution. Early models such as GANs (Generative Adversarial Networks) and VAEs (Variational Autoencoders) laid the groundwork for the field. More recently, diffusion models have gained significant attention due to their ability to produce highly detailed images with fewer artifacts and better generalization.

2.1.2 Attention Mechanisms:

Attention mechanisms play a crucial role in guiding the model to focus on important parts of the text and image. Self-attention and cross-attention mechanisms allow models to selectively focus on relevant features in the input, such as words in the description or regions in the image, improving the alignment between text and the generated image.

2.1.3 Diffusion Models:

Diffusion models have revolutionized text-to-image synthesis by introducing a novel process for generating images. Unlike GANs, which rely on adversarial training, diffusion models progressively refine random noise into a high-quality image through a series of steps. These models, including Stable Diffusion, Region Diffusion SDXL, and DALL-E 2, have shown impressive results in generating visually appealing and diverse images based on textual input.

2.1.4 Text Encoding:

Textual input is an essential component of text-to-image generation. Models like CLIP (Contrastive Language-Image Pretraining) are used to encode text into vector representations, allowing the system to relate the textual description to the visual content. This is key for generating images that accurately reflect the semantics of the input text.

2.2 Survey of Text-to-Image Generation Methods

In this section, we review the various approaches to text-to-image generation, focusing on the evolution of models and their performance in generating images from textual descriptions.

2.2.1 Early Approaches :

Early models like AttnGAN and StackGAN made significant contributions by introducing the use of attention mechanisms in GANs for text-to-image synthesis. These models could generate images from text descriptions by focusing on key words, but they still faced challenges such as lack of detail and inconsistencies in image quality.

2.2.2 Generative Adversarial Networks (GANs):

The development of more advanced GANs, such as BigGAN and StyleGAN, allowed for higher-quality image generation. These models used progressively deeper architectures and refined training techniques to improve the fidelity of generated images. However, they still faced limitations in aligning textual descriptions with generated images, especially for more complex scenes.

2.2.3 Text-to-Image Synthesis with Diffusion Models:

The introduction of diffusion models marked a breakthrough in the field of text-to-image synthesis. Models like DALL-E 2, Stable Diffusion, and Imagen have demonstrated a remarkable ability to generate images that are not only high in quality but also exhibit a greater understanding of complex and nuanced textual prompts. These models excel in tasks that require a high degree of detail and creativity.

2.2.4 Rich-Text Inputs for Enhanced Control:

As the demand for more control over the image generation process grew, researchers explored methods that allow users to provide richer forms of input. Rich-text inputs, such as those using different fonts, colors, and embedded images, provide users with greater flexibility in specifying the details of the generated image. This approach is especially useful for creative applications where precise customization is important.

2.3 Text-to-Image Generation: Key Challenges

Despite the advancements in text-to-image generation, several challenges persist:

2.3.1 Precision in Color and Detail:

One of the most significant hurdles is ensuring that generated images reflect the correct colors, styles, and fine details described in the text. Even with powerful models like diffusion models, accurately translating vague or abstract textual descriptions into specific image details remains a challenge.

2.3.2 Scene Composition and Layout:

Another difficulty lies in correctly arranging multiple objects and ensuring that their spatial relationships are represented accurately in the generated images. Complex scenes with multiple entities often lead to issues with coherence and consistency, especially when there are competing visual elements.

2.3.3 User Control:

While generative models have made great strides in improving image quality, they still lack the fine-grained control that many users desire. For instance, users may want to specify particular styles, emphasize certain areas of an image, or adjust specific features like colors or textures. Incorporating rich-text inputs that include fonts, sizes, and embedded images helps address this problem by giving users more precise control over the output.

2.4 Recent Advances and Emerging Techniques:

The field of text-to-image generation is rapidly evolving, with several exciting techniques emerging to further improve the quality and flexibility of the generated images. Some of these include:

2.4.1 Cross-modal Models:

These models simultaneously leverage text and image data to improve the interaction between these modalities, resulting in more coherent and contextually accurate image generation. Cross-modal systems enable the alignment of text with visual features in more nuanced ways.

2.4.2 Zero-shot and Few-shot Learning:

Newer models are capable of generating high-quality images from prompts they haven't been explicitly trained on. This capability is particularly useful for reducing the need for large training datasets and allows models to generalize better to unseen prompts, improving their utility in a wider range of applications.

2.4.3 Interactive Editing Systems:

Researchers are also exploring interactive systems that allow users to modify images in real-time by adjusting the input text or directly manipulating the generated output. These systems offer dynamic control over the image generation process, providing a more intuitive and creative user experience.

2.5 Summary

This literature survey provides an overview of the key developments in the field of text-to-image generation. We reviewed early approaches, the evolution of GANs, the rise of diffusion models, and the integration of rich-text inputs for enhanced control. While significant progress has been made in improving the quality and flexibility of generated images, challenges related to precision, scene composition, and user control remain. The exploration of new techniques like cross-modal models and interactive editing systems shows the potential for even more advanced and user-friendly text-to-image generation. By building on these advancements, this project aims to contribute

to the ongoing evolution of text-to-image synthesis by integrating region-specific control and rich-text inputs, providing users with greater customization and accuracy in the generation process.

3 Proposed Work

3.1 Text-to-image models

Text-to-image systems aim to synthesize realistic images according to descriptions (Zhu et al., 2007; Mansimov et al., 2016). Fueled by the large-scale text-image datasets (Schuhmann et al., 2022; Byeon et al., 2022), various training and inference techniques (Ho et al., 2020; Song et al., 2021; Ho et al., 2022; Ho and Salimans, 2021), and scalibility (Ramesh et al., 2022), significant progress has been made in text-to-image generation using diffusion models (Balaji et al., 2022; Ramesh et al., 2022; Nichol et al., 2022; Saharia et al., 2022; Gafni et al., 2022), autoregressive models (Ramesh et al., 2021; Yu et al., 2022; Chang et al., 2023; Ding et al., 2022), GANs (Sauer et al., 2023; Kang et al., 2023), and their hybrids (Rombach et al., 2022). Our work focuses on making these models more accessible and providing precise controls. In contrast to existing work that uses plain text, we use a rich text editor with various formatting options

3.2 Controllable image synthesis with diffusion models

A wide range of image generation and editing applications are achieved through either finetuning pre-trained diffusion models (Ruiz et al., 2023; Kumari et al., 2023; Zhang and Agrawala, 2023; Avrahami et al., 2023; Wu et al., 2023; Kawar et al., 2023; Ma et al., 2023; Li et al., 2023) or modifying the denoising process (Meng et al., 2022; Choi et al., 2021; Hertz et al., 2023; Parmar et al., 2023; Bansal et al., 2023; Chefer et al., 2023; Avrahami et al., 2022; Balaji et al., 2022; Jimenez ´ , 2023; Bar-Tal et al., 2023; Sarukkai et al., 2023; Zhang et al., 2023; Cao et al., 2023; Phung et al., 2024; Xiao et al., 2023; Feng et al., 2023). For example, Prompt-to-prompt (Hertz et al., 2023) uses attention maps from the original prompt to guide the spatial structure of the target prompt. Although these methods can be applied to some richtext-to-image applications, the results often fall short, as shown in Section 4. Concurrent with our work, Mixture-of-diffusion (Jimenez ´ , 2023) and MultiDiffusion (Bar-Tal et al., 2023) propose merging multiple diffusion-denoising processes in different image regions through linear blending. Instead of relying on user-provided regions, we automatically compute regions of selected tokens using attention maps. Gradient (Ho et al., 2022) and Universal (Bansal et al., 2023) guidance control the generation by optimizing the denoised generation at each time step. We apply them to precise color generation by designing an objective on the target region to be optimized.

3.3 Attention in diffusion models

The attention mechanism has been used in various diffusion-based applications such as view synthesis (Liu et al., 2023; Tseng et al., 2023; Watson et al., 2022), image editing (Hertz et al., 2023; Chefer et al., 2023; Patashnik et al., 2023; Parmar et al., 2023; Kumari et al., 2023), and video editing (Liu et al., 2023; QI et al., 2023; Ceylan et al., 2023; Ma et al., 2023). We also leverage the spatial structure in self-attention maps and alignment information between texts and regions in cross-attention maps for rich-text-to-image generation.

3.4 Rich text modeling and application

Exploiting information beyond the intrinsic meanings of the texts has been previously studied (Meng et al., 2019; Sun et al., 2021; Xu et al., 2020; Li et al., 2022). For example, visual information, such as underlining and bold type, have also been extracted for various document understanding tasks (Xu et al., 2020; Li et al., 2022). To our knowledge, we are the first to leverage rich text information for text-to-image synthesis.

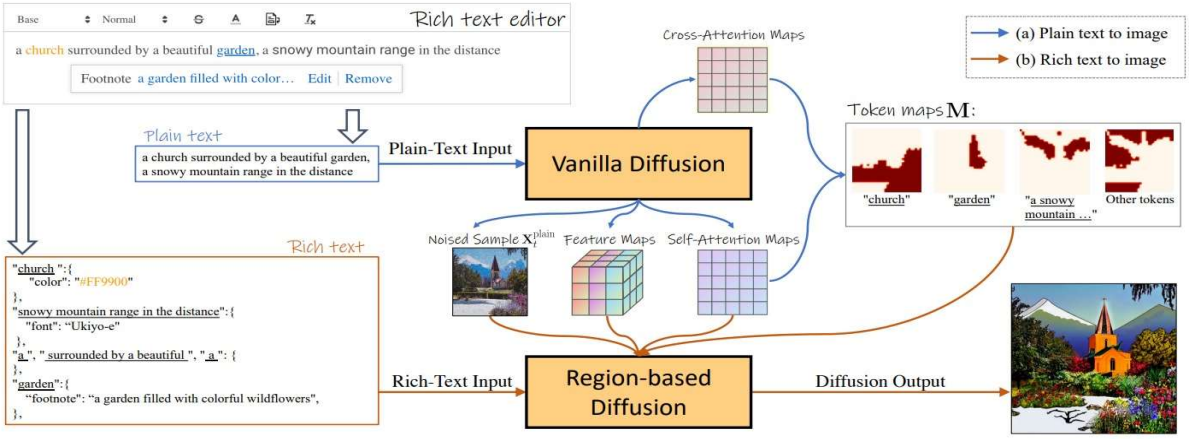


Figure 2. Rich-text-to-image framework.

Fig. 2:Rich-text-to-image framework First, the plain-text prompt is processed by a diffusion model to collect self- and cross-attention maps, noised generation, and residual feature maps at certain steps. The token maps of the input prompt are constructed by first creating a segmentation using the self-attention maps and then labeling each segment using the cross-attention maps. Then the rich texts are processed as JSON to provide attributes for each token span. The resulting token maps and attributes are used to guide our region-based control. We inject the self-attention maps, noised generation, and feature maps to improve fidelity to the plain-text generation.

3.5 Training and Performance Optimization

To enhance performance, the model is optimized using the RMSprop algorithm with binary cross-entropy loss. A learning rate reduction callback is employed to dynamically adjust the learning rate when the validation accuracy plateaus, ensuring steady progress during training. Data augmentation is applied to the training dataset using a generator, introducing variations like rotations and flips to make the model robust to real-world input variability. This design enables the model to automatically learn and classify medical imaging patterns, making it a valuable tool for early and accurate detection of pneumonia in infants, potentially aiding in timely treatment and better healthcare outcomes.

3.6 Training and Performance Optimization

Style transfer (Gatys et al., 2016; Zhu et al., 2017; Luan et al., 2017) and Colorization (Reinhard et al., 2001; Tai et al., 2005; Xu et al., 2013; Levin et al., 2004; Zhang et al., 2016, 2017) for editing real images have also been extensively studied. In contrast, our work focuses on local style and precise color control for generating images from text-to-image models.

3.7 Image generation with complex prompts

To accurately generate the image the users expect, one option is to provide more detailed prompts. Several studies have thus focused on generating images based on complex prompts (Betker et al., 2023; Wang et al., 2024; Wu et al., 2023). DALL-E 3 (Betker et al., 2023) finds that training on highly descriptive synthetic captions reliably improves the alignment between text prompts and generation results. ParaDiffusion (Wu et al., 2023) utilizes pretrained LLM with a larger context window to process complex prompts. Instead, we decouple the complex prompts into multiple detailed prompts that describe local regions. Similar to ours, InstanceDiffusion (Wang et al., 2024) also studies detailed prompts for individual regions, while ours does not require layout as part of the user input.

3.8 Text-to-image generation benchmark

As text-toimage models develop rapidly, many works have paid attention to the evaluation of these models (Bakr et al., 2023; Hu et al., 2023; Huang et al., 2024; Patel

et al., 2024; Zhao et al., 2024). While these benchmarks focus on different aspects of text-to-image generation such as the text-image alignment (Hu et al., 2023; Bakr et al., 2023) and concept learning (Kumari et al., 2023; Patel et al., 2024), we aim at building a benchmark for evaluating rich text to image generation on several applications, including precise color rendering, local style control, and complex prompt alignment.

4 Method

To utilize rich text annotations, our method consists of two steps, as shown in Figure 2. First, we compute the spatial layouts of individual token spans. Second, we use a new region-based diffusion to render each region's attributes into a globally coherent image.

4.1 Step 1. Token maps for spatial layout

Several works (Tang et al., 2022; Ma et al., 2023; Balaji et al., 2022; Hertz et al., 2023; Chefer et al., 2023; Patashnik et al., 2023; Tumanyan et al., 2023) have discovered that the attention maps in the self- and cross-attention layers of the diffusion UNet characterize the spatial layout of the generation. Therefore, we first use the plain text as the input to the diffusion model and collect self-attention maps of size 32 × 32 × 32 × 32 across different heads, layers, and time steps. We take the average across all the extracted maps and reshape the result into 1024 × 1024. Note that the value at i th row and j th column of the map indicates the probability of pixel i attending to pixel j. We average the map with its transpose to convert it to a symmetric matrix. It is used as a similarity map to perform spectral clustering (Shi and Malik, 2000; Von Luxburg, 2007) and obtain the binary segmentation maps Mc of size K ×32×32, where K is the number of segments.

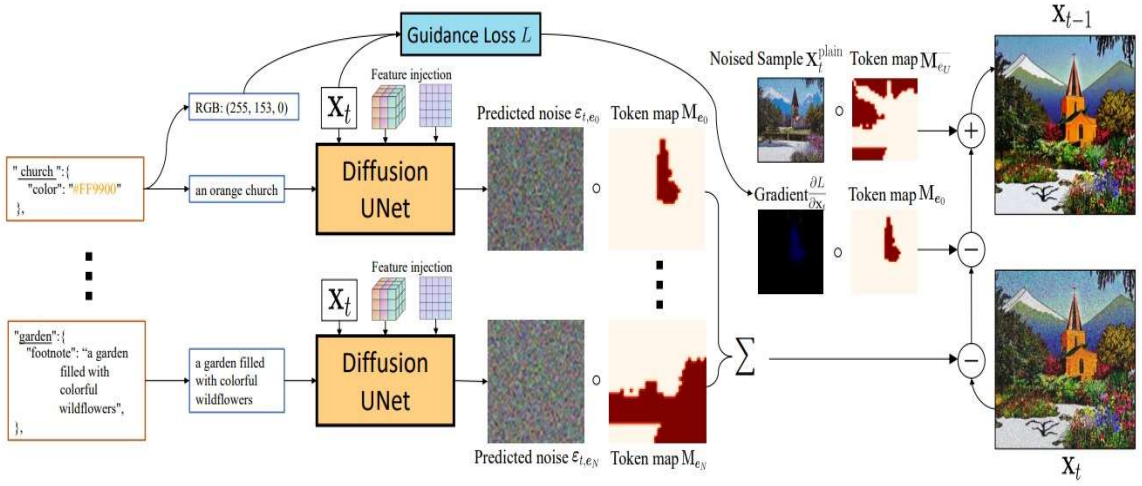


Figure 3. Region-based diffusion.

Fig. 3: For each element of the rich-text input, we apply a separate diffusion process to its region. The attributes are either decoded as a region-based guidance target (e.g. re-coloring the church), or as a textual input to the diffusion UNet (e.g. handling the footnote to the garden). The self-attention maps and feature maps extracted from the

plain-text generation process are injected to help preserve the structure. The predicted noise ϵ_{t,e_i}, weighted by the token map M_{e_i}, and the guidance gradient $\frac{\partial L}{\partial x_t}$ are used to denoise and update the previous generation x_t to x_{t-1}. The noised plain text generation $x_{t,\text{plain}}$ is blended with the current generation to preserve the exact content in those regions of the unformatted tokens.

To associate each segment with a textual span, we also extract cross-attention maps for each token w_j:

$$m_j = \frac{\exp(s_j)}{\sum_k \exp(s_k)}, \tag{1}$$

where sj is the attention score. We first interpolate each cross-attention map mj to the same resolution as Mc of 32 × 32. Similar to the processing steps of the self-attention maps, we compute the mean across heads, layers, and time steps to get the averaged map mb j . We associate each segment with a texture span ei following Patashnik et al. (2023):

$$M_{e_i} = \left\{ M_{c_k} \mid \frac{M_{c_k} \cdot (m_{b_j} - \min(m_{b_j}))}{\max(m_{b_j}) - \min(m_{b_j})} > \epsilon \right\}, \tag{2}$$

$$\forall j \text{ s.t. } w_j \in e_i, \tag{3}$$

where is a hyperparameter that controls the labeling threshold, that is, the segment Mck is assigned to the span ei if the normalized attention score of any tokens in this span is higher than . We associate the segments that are not assigned to any formatted spans with the unformatted tokens eU . Finally, we obtain the token map in Figure 2 as below:

$$M_{e_i} = \frac{\sum_{M_{c_j} \in M_{e_i}} M_{c_j}}{\sum_i \sum_{M_{c_j} \in M_{e_i}} M_{c_j}} \tag{4}$$

4.2 Step 2. Region-based denoising and guidance

As shown in Figure 2, given the text attributes and token maps, we divide the overall image synthesis into several region-based denoising and guidance processes to incorporate each attribute, similar to an ensemble of diffusion models (Kumari et al., 2023; Bar-Tal et al., 2023). More specificially, given the span ei , the region defined by its token map Mei , and the attribute ai , the predicted noise t for noised generation xt at time step t is

$$\epsilon_t = \sum_i M_{e_i} \cdot \epsilon_{t,e_i} = \sum_i M_{e_i} \cdot D(x_t, f(e_i, a_i), t), \tag{5}$$

where D is the pretrained diffusion model, and $f(e_i, a_i)$ is a plain text representation derived from text span e_i and attributes a_i using the following process:

1. Initially, we set $f(e_i, a_i) = e_i$.
2. If an embedded image is available, we convert it into a footnote a_{f_i} using the gradient-based discrete optimization (Wen et al., 2023).

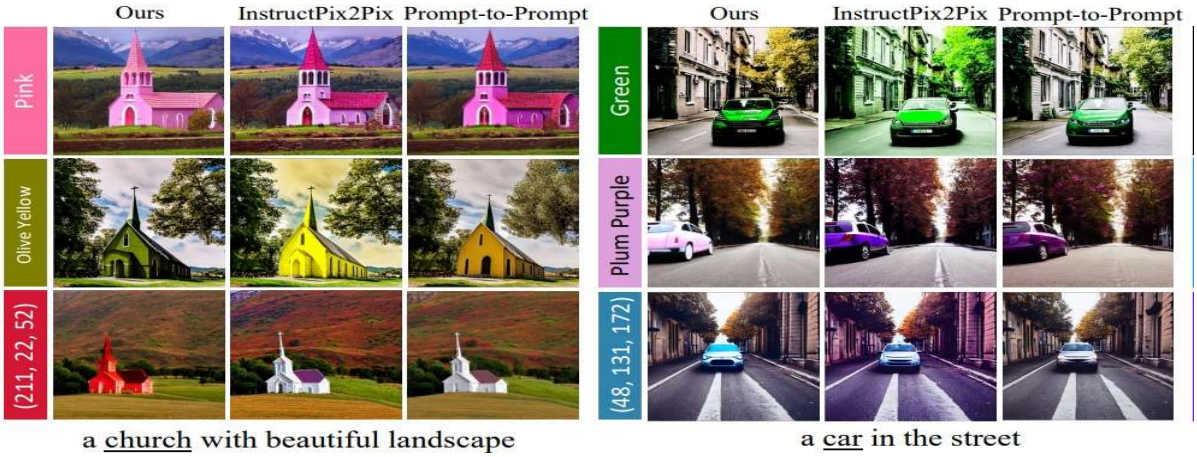


Figure 4. Qualitative comparison on precise color generation.

Fig. 4: We show images generated by Prompt-to-Prompt (Hertz et al., 2023), InstructPix2Pix (Brooks et al., 2023), and our method using prompts with font colors. Our method generates precise colors according to either color names or RGB values. Both baselines generate plausible but inaccurate colors given color names, while neither understands the color defined by RGB values. InstructPix2Pix tends to apply the color globally, even outside the target object.

3. If footnote a_{f_i} is available, we set $f(e_i, a_i) = a_{f_i}$.
4. The style a_{s_i} is appended if it exists. $f(e_i, a_i) = f(e_i, a_i) +$ 'in the style of' $+ a_{s_i}$.
5. The closest color name (string) of font color $\hat{a}c_i$ from a predefined set C is prepended. $f(e_i, a_i) = \hat{a}c_i + f(e_i, a_i)$. For example, $\hat{a}c_i =$ 'brown' for RGB color $ac_i = [136, 68, 20]$.

We use $f(e_i, a_i)$ as the original plain text prompt of Step 1 for the unformatted tokens e_U. This helps us generate a coherent image, especially around region boundaries.

Guidance. By default, we use classifier-free guidance (Ho and Salimans, 2022) for each region to better match the prompt $f(e_i, a_i)$. In addition, if the font color is specified, to

exploit the RGB values information further, we apply gradient guidance (Ho et al., 2022; Dhariwal and Nichol; Bansal et al., 2023) on the current clean image prediction:

$$x_{b_0} = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_t}{\sqrt{\bar{\alpha}_t}}, \tag{6}$$

where x_t is the noisy image at time step t, and $\bar{\alpha}t$ is the coefficient defined by noise scheduling strategy (Ho et al., 2020). Here, we compute an MSE loss L between the average color of x_b weighted by the token map Me_i and the RGB triplet a_{c_i}. The gradient is calculated below,

$$\frac{dL}{dx_t} = \frac{d\left\| \frac{\sum_p (M_{e_i} \cdot x_{b_0})}{\sum_p M_{e_i}} - a_{c_i} \right\| 2^2}{\sqrt{\bar{\alpha}_t} dx b_0}, \tag{7}$$

where the summation is over all pixels p. We then update x_t with the following equation:

$$x_t \leftarrow x_t - \lambda \cdot M_{e_i} \cdot \frac{dL}{dx_t}, \tag{8}$$

where λ is a hyperparameter to control the strength of the guidance. We use $\lambda = 1$ unless denoted otherwise.

Token reweighting with font size. Last, to re-weight the impact of the token w_j according to the font size a_{w_j}, we modify its cross-attention maps m_j. However, instead of applying direct multiplication as in Prompt-to-Prompt (Hertz et al., 2023) where $\sum_j a_{w_j} m_j \neq 1$, we find that it is critical to preserve the probability property of m_j. We thus propose the following reweighting approach:

$$m_{b_j} = \frac{a_{w_j} \exp(s_j)}{\sum_k a_{w_k} \exp(s_k)}. \tag{9}$$

We can compute the token map (Equation 4) and predict the noise (Equation 5) with the reweighted attention map.

Preserve the fidelity against plain-text generation. Although our region-based method naturally maintains the layout, there is no guarantee that the

Fig. 6: We report the CLIP similarity between each stylized region and its region prompt. Our method achieves the best stylization.

Details and shape of the objects are retained when no rich-text attributes or only the

Figure 5. Qualitative comparison on style control.

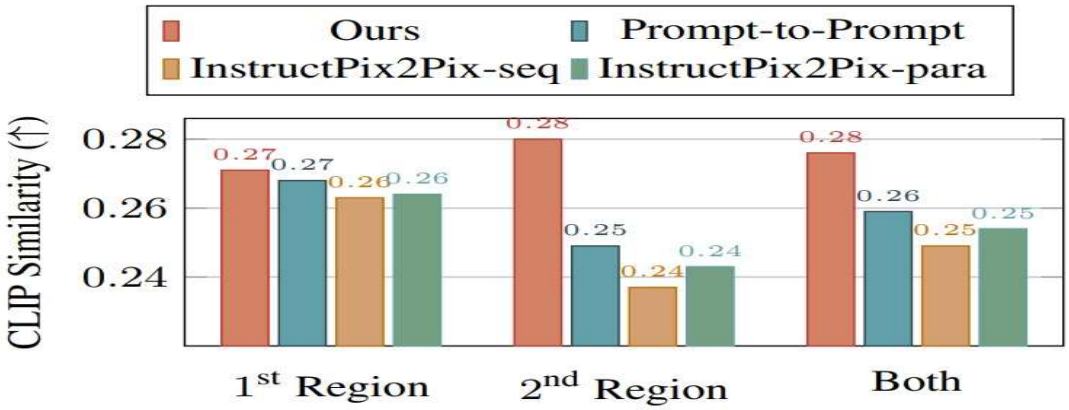


Figure 6. Quantitative evaluation of local style control.

color is specified, as shown in Figure 13. To this end, we follow Plug-and-Play (Tumanyan et al., 2023) to inject the self-attention maps and the residual features extracted from the plain-text generation process when $t > T_{pnp}$ to improve the structure fidelity. In addition, for the regions associated with the unformatted tokens e_U, stronger content preservation is desired. Therefore, at certain $t = T_{blend}$, we blend the noised sample $x_{t,\text{plain}}$ based on the plain text into those regions:

$$x_t \leftarrow M_{e_U} \cdot x_{t,\text{plain}} + (1 - M_{e_U}) \cdot x_t \tag{10}$$

5 Experimental Results Implementation details

We use Stable Diffusion V1-5 (Rombach et al., 2022) for our experiments. To create the token maps, we use the cross-attention layers in all blocks, excluding the first encoder and last decoder blocks, as the attention maps in these highresolution layers are often noisy. We discard the maps at the initial denoising steps with T > 750. We use K = 15, = 0.3, Tpnp = 0.3, Tblend = 0.3, and report the results averaged from three random seeds for all quantitative experiments. More details, such as the running time, can be found in Appendix B. Font style evaluation. We compute CLIP scores (Radford et al., 2021) for each local region to evaluate the stylization quality. Specifically, we create prompts of two objects and styles. We create combinations using 7 popular styles and 10 objects,

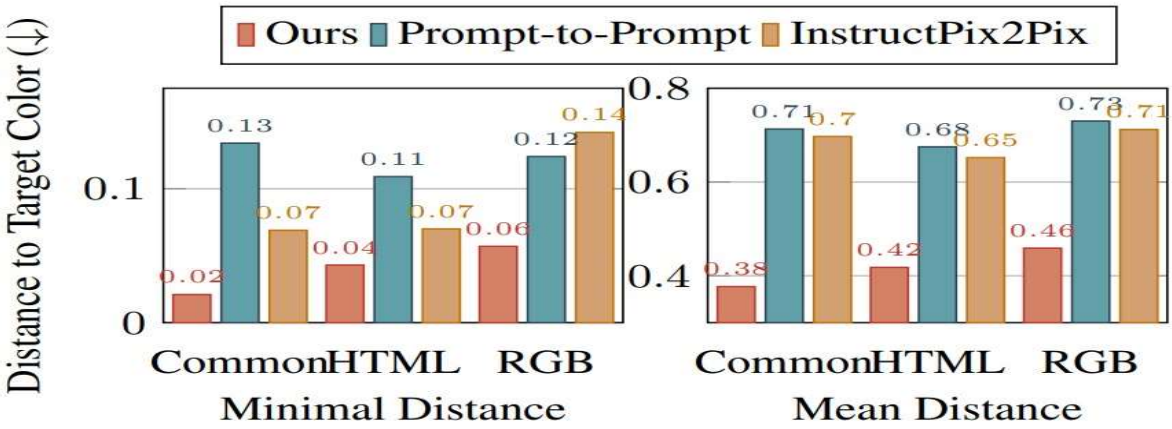


Figure 7. Quantitaive evaluation on precise color generation.

Fig. 7: Distance against target color is reported (lower is better). Our method consistently outperforms baselines.

resulting in 420 prompts. For each generated image, we mask it by the token maps of each object and attach the masked output to a black background. Then, we compute the CLIP score using the regionspecific prompt. For example, for the prompt "a lighthouse (Cyberpunk) among the turbulent waves (Ukiyo-e)", the local CLIP score of the lighthouse region is measured by comparing its similarity with the prompt "lighthouse in the style of cyberpunk." In this example, we refer to "lighthouse" as the first region and "waves" as the second region.

5.0.1 Font color evaluation

We divide colors into three categories to evaluate a method's capacity to understand and generate a specific color. The Common color category contains 17 standard names, such as "red", "yellow", and "pink". The HTML color names are selected from the web

color names 1 used for website design, such as “sky blue”, “lime green”, and “violet purple”. The RGB color category contains 50 randomly sampled RGB triplets to be used as “color of RGB values [128, 128, 128]”. To create a complete prompt, we use 12 objects exhibiting different colors, such as “flower”, “gem”, and “house”. This gives us a total of 1, 200 prompts. We evaluate color accuracy by computing the mean L2 distance between the region and target RGB values. We also compute the minimal L2 distance as sometimes the object should contain other colors for fidelity, e.g., the “black tires” of a “yellow car”.

5.0.2 Footnote evaluation

Most existing studies on textto-image generation (Ramesh et al., 2021; Saharia et al., 2022; Hertz et al., 2023) are only evaluated on the relatively short prompts (Bakr et al., 2023; Hu et al., 2023). However, long text prompts are still useful for accurately describing a scene

To understand how well a model performs when a lengthy text prompt is given, we collect a set of long prompts using the GPT-4 model (OpenAI, 2023). To improve the stability and quality of the collected prompts, we ask the language model first to generate a shorter caption for a scene and then generate detailed descriptions for the objects that appear in the scene. Such a format also makes it easy to convert to richtext prompts. We manually create a few hierarchical descriptions as the in-context examples.

After collecting the results from GPT-4, we then manually filter and edit the individual descriptions. For example, we remove the descriptions of objects not shown in the scene. We collect 100 prompts in total. Then, we use GPT-4 to produce single, long, plain-text prompts and write scripts to produce richtext prompts from these hierarchical prompts. We summarize the statistics of the plain-text prompts in Table 1. Note that the average length of the collected prompts is around 4× longer than the prior arts and closer to the concurrent work (Wu et al., 2023).

To evaluate the alignment between the generated images and the lengthy text prompts, we follow the process in TIFA (Hu et al., 2023) to sample questionanswer pairs for both scene and object prompts using language models. For example, for the scene prompt “a nightstand next to a bed in the bedroom,” the question-and-answer pair could be “Q: Is there a bed? A: Yes.” Through this process, we collect 2921 pairs. Then, either a VQA model or human annotators can answer the question according to the generated image and check whether it is consistent with the ground truth answer

We manually modify the questions based on the object prompts to reflect that the object is part of the scene. For example, for the object prompt “a nightstand with some books,” we modify the question “Are there books?” to “Are there books on the night-

stand?" We manually review and drop low-quality, irrelevant, or repeated questions. For example, we drop the pair "Q: What is in the bedroom? A: A bed." since "a nightstand" or other reasonable objects could be the correct answer. Instead, we prefer to ask the more concrete question, "Is there a bed in the bedroom?" After the manual processing, we obtain 1974 question-answer pairs in total

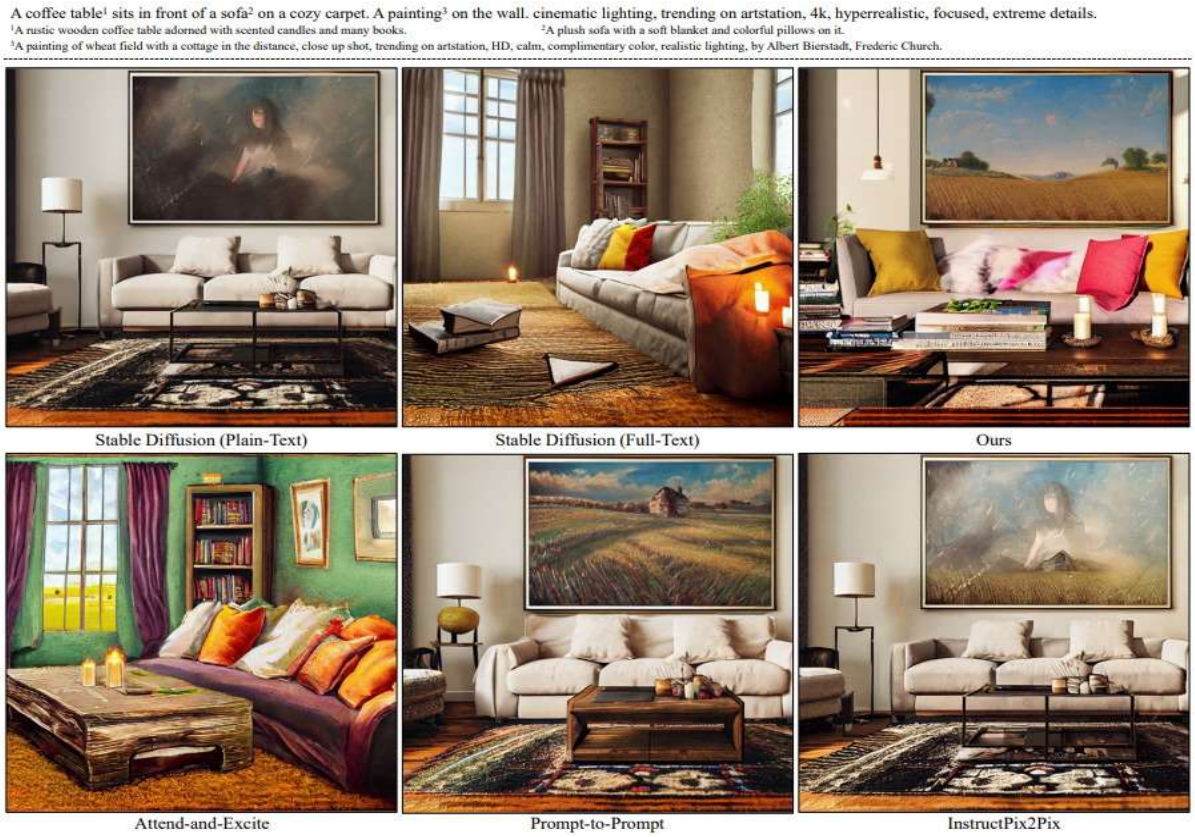


Figure 8.Qualitative comparison on detailed description generation.

Fig. 8: We show images generated by Attend-and-Excite, Prompt-to-Prompt, InstructPix2Pix, and our method using complex prompts. Our method is the only one that can generate all the details faithfully.

5.0.3 Baselines

For font color and style, we quantitatively compare our method with two strong baselines, Prompt-to-Prompt (Hertz et al., 2023) and InstructPix2Pix (Brooks et al., 2023). When two instructions exist for each image in our font style experiments, we apply them in parallel (InstructPix2Pixpara) and sequential manners (InstructPix2Pix-seq). More details can be found in Appendix B. We also perform a human evaluation on these two methods in Appendix Table 1. For re-weighting token importance, we visually compare with Prompt-to-Prompt (Hertz et al., 2023) and two heuristic methods,

repeating and adding parentheses. For complex scene generation with footnotes, we also compare with Attend-andExcite (Chefer et al., 2023).

5.1 Quantitative Comparison

We report the local CLIP scores computed by a ViTB/32 model in Figure 6. Our method achieves the best overall CLIP score compared to the two baselines. This demonstrates the advantage of our region-based diffusion method for localized stylization. To further understand each model's capacity for generating multiple styles, we report the metric on each region. Prompt-to-Prompt and InstructPix2Pix-para achieve a decent score on the 1st Region, i.e., the region first occurs in the sentence. However, they often fail to fulfill the style in the 2nd Region. We conjecture that the Stable Diffusion model tends to generate a uniform style for the entire image, which can be attributed to single-style training images. Furthermore, InstructPix2Pix-seq performs the worst in the 2nd Region.

Details and shape of the objects are retained when no rich-text attributes or only the color is specified, as shown in Figure 13. To this end, we follow Plug-and-Play (Tumanyan et al., 2023) to inject the self-attention maps and the residual features extracted from the plain-text generation process when $t > T_{pnp}$ to improve the structure fidelity. In addition, for the regions associated with the unformatted tokens e_U, stronger content preservation is desired. Therefore, at certain $t = T_{blend}$, we blend the noised sample $x_{t,\text{plain}}$ based on the plain text into those regions:

$$x_t \leftarrow M_{e_U} \cdot x_{t,\text{plain}} + (1 - M_{e_U}) \cdot x_t \tag{11}$$

Fig. 9: We show images generated by our method and Prompt-to-Prompt using token weight of 13 for 'mushrooms'. Prompt-to-Prompt suffers from artifacts due to the large weight. Heuristic methods like repeating and parenthesis do not work well.

Table 2: Quantitative evaluation of long prompt generation. We report the percentage of the time that the VQA output is aligned with the answer to a question regarding the generation. Our method consistently improves Stable Diffusion models

5.2 Region

This is because the first instruction contains no information about the second region, and the second region's content could be compromised when we apply the first instruc-

Figure 9.Qualitative comparison on token reweighting.

Quantitative evaluation of long prompt generation.

Method	**Stable Diffusion**	**Stable Diffusion XL**
Plain-text generation	72.86 ± 1.13	79.05 ± 0.41
Rich-text generation	73.24 ± 1.00	81.08 ± 0.66

tion. We show quantitative results of precise color generation in Figure 7. The distance of HTML color is generally the lowest for baseline methods, as they provide the most interpretable textual information for text encoders. This aligns with our expectation that the diffusion model can handle simple color names, whereas they struggle to handle the RGB triplet. Our rich-textto-image generation method consistently improves on the three categories and two metrics over the baselines. In Table 2, we follow TIFA (Hu et al., 2023) to evaluate the long prompt generation using either Stable Diffusion or Stable Diffusion XL as the base models. Specifically, we generate 5 images for each text prompt using different random seeds and use the mplug-large model (Li et al., 2022) to generate the answers to each benchmark question. We report how often the generated answers are consistent with the benchmark answers by averaging across different seeds. We also report the standard deviation. Our rich-text method consistently improves the generation quality with different base models when a lengthy prompt is given. Moreover, we find that the Stable Diffusion XL significantly outperforms the Stable Diffusion model.

Fig. 10: We underline the modified concepts in the prompts and display the reference image on the top right of each rich-text generation result. Our method is able to synthesize the image with the object according to the reference image without changing

Figure 10. Qualitative results on customized concept generation.

the overall structure.

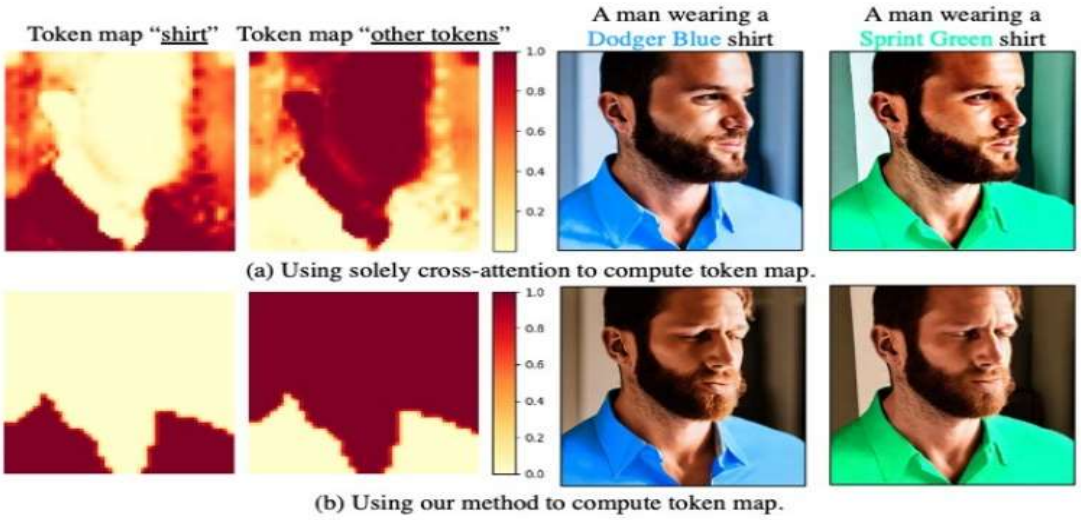


Figure 11.Ablation of token maps.

Fig. 11: Using solely cross-attention maps to create token maps leads to inaccurate segmentations, causing the background to be colored in an undesired way.

5.3 Visual Comparison

5.3.1 Precise Color Generation

Figure 4 illustrates a qualitative comparison of precise color generation. Existing methods like InstructPix2Pix (Brooks et al., 2023) often affect colors across the entire image rather than just the specified region. For instance, when changing the color of a flower in a vase, the method might unintentionally alter the vase and the background color as well. Similarly, Prompt-to-Prompt (Hertz et al., 2023) offers better control over the target region but still struggles to produce the exact color requested.

In contrast, our approach excels at accurately applying the precise colors specified in the prompt, without unintended alterations to other parts of the image. This demonstrates our system's ability to localize and apply color modifications effectively.

5.3.2 Local Style Generation

Figure 5 compares how different methods handle local style generation. When using InstructPix2Pix-seq, the style from the first instruction often dominates the entire image, overshadowing subsequent regional styling requests. Adjusting parameters like classifier-free guidance helps only minimally, as demonstrated in Figure 13 (Appendix). Both Prompt-to-Prompt and InstructPix2Pix tend to apply a globally uniform style, which fails to differentiate between the styles intended for specific regions.

Our method, however, successfully synthesizes distinct styles for different parts of the image, maintaining a clear separation between regions. Attempts to apply baseline methods independently for each region and then merge results often lead to visible artifacts along the boundaries, as shown in Figure 12 (Appendix). By contrast, our approach avoids these artifacts while preserving stylistic accuracy.

5.3.3 Complex Scene Generation

Figure 8 highlights how our method outperforms others in generating complex scenes. For instance, Attend-and-Excite (Chefer et al., 2023) tries to fix missing objects in the generated scene (e.g., a coffee table or carpet in a living room). However, it often fails to include finer details like books, paintings, or blankets. Similarly, Prompt-to-Prompt and InstructPix2Pix can edit individual objects, such as paintings, but struggle to render smaller details like colorful pillows or items on a table.

Our system, on the other hand, faithfully captures all the details specified in the prompt, ensuring a more complete and accurate representation of the scene.

5.3.4 Token Importance Control

Figure 9 demonstrates the importance of controlling token weights in text-to-image generation. For example, when emphasizing "mushroom" with a large weight, Prompt-to-Prompt often creates artifacts due to unbounded attention probabilities, resulting in distorted or unrealistic features. Heuristic approaches fail to add more mushrooms while maintaining image quality.

Our method not only successfully increases the number of mushrooms but also preserves overall image quality, showing that our approach manages token importance effectively without compromising realism.

5.3.5 Customized Concept Generation

Figure 10 showcases how our system handles reference-driven image generation. For example, when embedding a reference image to specify a concept—like a particular street scene or a dog's pose—our method accurately incorporates the customized concept into the generated image. Importantly, this is done without introducing unwanted changes to the rest of the scene. This ability makes our approach highly effective for tasks requiring personalization or precise adjustments.

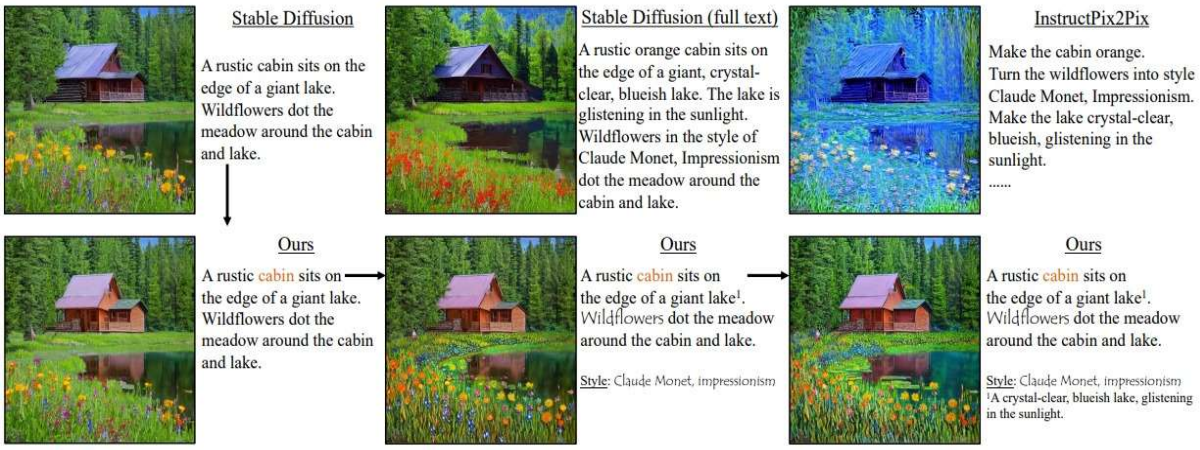


Figure 12. Our workflow.

Fig. 12: (top left) A user begins with an initial plain-text prompt and wishes to refine the scene by specifying the color, details, and styles. (top center) Naively inputting the whole description in plain text does not work. (top right) InstructPix2Pix Brooks et al. (2023) fails to make accurate editing. (bottom) Our method supports precise refinement with region-constrained diffusion processes. Moreover, our framework can naturally be integrated into a rich text editor, enabling a tight, streamlined UI.

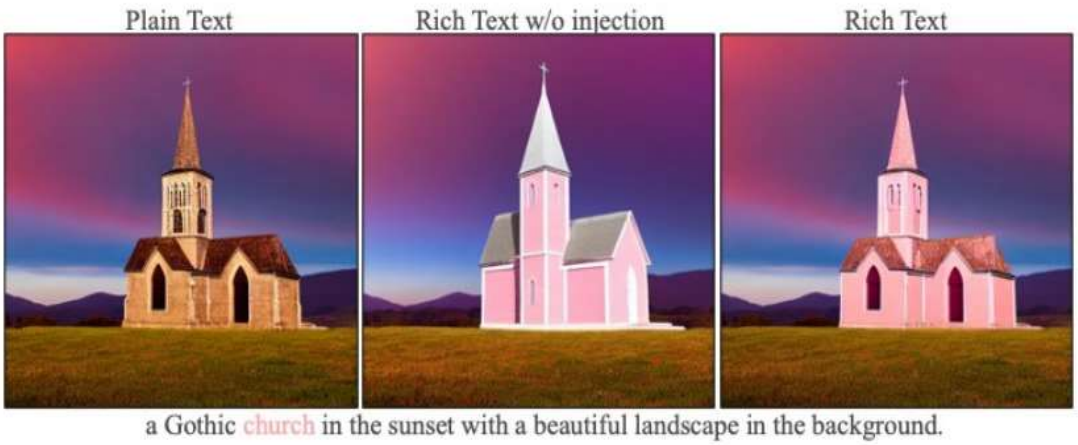


Figure 13. Ablation of injection method.

Fig. 13: We show images generated based on plain text and rich text with or without injection methods. Injecting features and noised samples help preserve the structure of the church and unformatted token regions.

6 Simulation And Results

6.1 Dataset Used:

- The project relies on pre-trained models like **Stable Diffusion v1.5** and **Stable Diffusion XL**.
- These models are trained on **LAION-5B**, a large-scale, publicly available dataset of image-text pairs.

6.1.1 Stable Diffusion v1.5

- **Stable Diffusion v1.5** is part of the foundational series of Stable Diffusion models developed by Stability AI. It represents an incremental improvement over version 1.4 and focuses on delivering high-quality, semantically accurate images from textual prompts.
- Trained on the **LAION-5B dataset**, which includes billions of image-text pairs scraped from the internet, ensuring diversity and robustness.
- **Capabilities:**Generates images at resolutions like 512x512 pixels with a focus on clarity and prompt adherence.

6.1.2 Stable Diffusion XL (SDXL)

- **Stable Diffusion XL (SDXL)** represents a major advancement in the Stable Diffusion series. Released in 2023, it significantly enhances both image quality and the complexity of images generated, making it a strong choice for professional and creative applications.
- Features a larger parameter set and more layers than v1.5, enabling deeper understanding of prompts and richer image details.
- **Capabilities:**Natively supports 1024x1024 resolution, with better upscaling options for larger images. Reduces artifacts and improves fine details, such as textures, shadows, and reflections.

6.2 Simulation and Training:

- The project incorporates Transformer-based architectures to process the textual input and GAN-based models (Generative Adversarial Networks) for image syn-

thesis.

- Training is performed using the RTI datasets, involving a process where text embeddings guide the image generation pipeline to produce visually coherent outputs.
- **Evaluation Metrics:** The project likely uses common metrics for text-to-image tasks, such as FID (Frechet Inception Distance) and IS (Inception Score), to validate the quality and relevance of generated images.

6.3 Data Preprocessing

Data preprocessing ensures that the input text and image data are clean, consistent, and ready for model training.

6.3.1 Text Data Preprocessing:

- **Tokenization:**

  The textual descriptions (rich captions) are tokenized into sequences of words or subwords using libraries like NLTK, SpaCy, or HuggingFace Tokenizers.

- **Embedding Conversion:**

  Text is converted into embeddings using pre-trained language models such as BERT, GPT, or CLIP, which map text into a high-dimensional vector space.

- **Normalization:**

  Text is lowercased, punctuation is removed, and stop words are handled to standardize the input format.

- **Parsing Rich Text:**

  For rich captions, additional semantic structures like part-of-speech tags, dependency trees, or scene graphs may be extracted and stored as auxiliary information.

6.3.2 Image Data Preprocessing:

- **Resizing:**

  Images are resized to a fixed resolution (e.g., 256x256 or 128x128) to ensure consistent input to the GAN model.

- **Normalization:**

  Pixel values are normalized to a range of [0, 1] or [-1, 1], depending on the neural network requirements.

- **Data Alignment:**

  Images are aligned with their corresponding rich text captions to maintain coherence during training.

6.4 Data Augmentation

Data augmentation artificially increases the size and diversity of the training data, helping to improve the model's robustness and generalization.

6.4.1 Text Data Augmentation:

- **Synonym Replacement:**

  Words in the captions are replaced with synonyms using tools like WordNet to create variations of the same sentence.

- **Paraphrasing:**

  Sentences are rephrased using paraphrasing models (e.g., T5 or Pegasus) to introduce linguistic variety.

- **Noise Injection:** Random typos, missing words, or grammatical errors are introduced to simulate real-world text data.

- **Sentence Expansion:** Add details to captions by enriching them with contextual or descriptive phrases.

6.4.2 Image Data Augmentation:

- **Geometric Transformations:**

  Random cropping, rotation, flipping, and scaling to simulate different perspectives.

- **Color Adjustments:**

  Alter brightness, contrast, saturation, and hue to mimic various lighting conditions.

- **Noise Addition:** Apply Gaussian noise or other distortions to simulate real-world image imperfections.
- **CutMix or MixUp:** Combine parts of two images along with their captions to encourage the model to learn blended contexts.

6.5 Results

Figure 14. A girl with big eyes, skin, and long hair, t-shurt, bursting with vivid color.

Figure 15. A girl with big eyes, skin, and long hair , t-shurt, bursting with vivid color

Figure 16. A girl with big eyes, skin, and long hair , t-shurt, bursting with vivid color

Figure 17. Here a male Lycan wolf wearing jet trench coat playing a guitar, high fantasy, concept art.

Figure 18. Here a male Lycan wolf wearing jet trench coat playing a ***guitar***, high fantasy, concept art.

Figure 19. Here a male Lycan wolf wearing jet trench coat playing a **guitar**, high fantasy, concept art.

6.6 Discussion and Limitations

In this paper, we have expanded the controllability of text-to-image models by incorporating rich-text attributes as the input. We have demonstrated the potential for generating images with local styles, precise colors, different token importance, reference images, and complex descriptions. Nevertheless, numerous formatting options remain unexplored, such as bold/italic, hyperlinks, spacing, and bullets/numbering. Also, there are multiple ways to use the same formatting options. For example, one can use font style to characterize the shape of the objects. We hope this paper encourages further exploration of integrating accessible user interfaces into text-based content creation tasks, even beyond images.

Data availability statement. The Rich-Text Benchmark data, such as the plain-text prompts, rich-text prompts, and evaluation questions, are available at `https://docs.google.com/spreadsheets/d/127exMcICDh7Lrde91eRhvh-5itn0ZWYsTfT6qAfm0O0/edit?gid=0#gid=0`. Our code to reproduce the figures containing rich-text generation results can be found at `https://github.com/siddafda001/High-Level-Text-to-Image-Transforming-Des`

7 Deployment

To deploy the "High-Level-Text-to-Image-Transforming-Descriptions-into-Visual-Art" project from the GitHub repository High-Level-Text-to-Image-Transforming-Descriptions-into-Visual-Art, you can follow the deployment steps below. This process involves setting up the environment, creating a web interface, and deploying the model for users to interact with it.

7.1 Clone the Repository

Start by cloning the repository to your local machine or server where you intend to deploy the project.

```
git clone https://github.com/siddafda001/High-Level-Text-to-Image-Transforming-Descri
cd High-Level-Text-to-Image-Transforming-Descriptions-into-Visual-Art
```

7.2 Set Up the Environment

Ensure that you have the necessary Python environment. The repository requires specific libraries to function. Follow these steps:

7.2.1 Install Python and dependencies

You can create a virtual environment to keep your dependencies isolated:

```
python -m venv venv
source venv/bin/activate  # For Linux/MacOS
venv\Scripts\activate     # For Windows
```

Install the required libraries by running:

```
pip install -r requirements.txt
```

7.3 Download Pretrained Models

You will likely need pretrained models for this project to function properly. Check the repository or the project documentation for links to download the models. You can often find links in the README or in a script that downloads the models automatically.

7.4 Set Up the Server (Optional for Production)

For production, consider using cloud services like AWS EC2, Google Cloud VM, or Microsoft Azure to deploy the model.

If deploying on your local machine or server, you can use Flask or FastAPI for setting up a simple backend API to serve your model.

Example for setting up a Flask server:

```
from flask import Flask, request, jsonify
import torch
from your_model import YourModelClass  # Replace with actual model import

app = Flask(__name__)

# Initialize model
model = YourModelClass.load_from_checkpoint('path_to_checkpoint')

@app.route('/generate', methods=['POST'])
def generate_image():
    data = request.json
    text = data.get('text', '')

    # Use your model to generate an image
    generated_image = model.generate_image_from_text(text)

    # Convert the image to a format suitable for returning in the response
    # Example, save the image and return the file path or image URL
    generated_image.save('generated_image.png')

    return jsonify({"image_url": "/path_to_generated_image.png"})

if __name__ == '__main__':
    app.run(debug=True)
```

7.5 Build a User Interface (Optional)

To make the project user-friendly, you can use Gradio or Streamlit to build a web interface for users to interact with the text-to-image generation model.

Here's how you can set it up with Gradio:

```
import gradio as gr
from your_model import YourModelClass  # Replace with actual model import

# Initialize model
model = YourModelClass.load_from_checkpoint('path_to_checkpoint')

def generate_image(text):
    # Generate image from text using the model
    return model.generate_image_from_text(text)

# Define Gradio interface
interface = gr.Interface(fn=generate_image, inputs="text", outputs="image")

interface.launch()
```

7.6 Deploy the Web Application

After setting up the interface, you can deploy it to a cloud service. If using Heroku, you will need to:

- Install Heroku CLI.
- Initialize a Git repository if not already initialized.
- Create a `Procfile` for Heroku to understand how to run the application:

  ```
  web: python app.py
  ```

- Push the code to Heroku:

  ```
  git init
  heroku create your-app-name
  git add .
  git commit -m "Initial commit"
  git push heroku master
  ```

- Heroku will automatically detect the dependencies and run the app for you.

7.7 Testing the Deployment

Once deployed, you can visit the web interface via the URL provided by your cloud service or Heroku. Ensure that the model is generating accurate images based on the input rich-text prompts.

By following these steps, you will be able to deploy the Rich-Text-to-Image project to a cloud server or local machine and provide an accessible platform for users to generate images using rich-text input.

8 Conclusion And Future Work

8.1 Conclusion

In conclusion, this work presents a significant advancement in the field of text-to-image generation by leveraging rich-text inputs to provide users with more precise control over the generated images. By incorporating attributes like font style, size, color, and embedded images, the proposed method allows for a level of customization that is difficult to achieve with traditional plain-text prompts. The region-based diffusion process ensures that these rich-text features are accurately represented in the generated images, leading to improved image quality, more distinct details, and better alignment with user intent. This approach not only enhances the flexibility and usability of text-to-image synthesis but also lays the foundation for more personalized and intuitive creative tools in the future.

8.2 Future Work

The proposed method for rich-text-to-image generation has shown promising results, but there are several areas for improvement and expansion. In this section, we outline potential directions for future work that could further enhance the approach and its applications.

8.2.1 Expanding Rich-Text Attributes

Currently, the system supports basic rich-text attributes such as font size, color, style, and embedded images. A promising avenue for future work is to expand the types of attributes that can be used to guide image generation. For instance, incorporating additional multimedia elements, such as video clips, animations, or 3D models, could provide users with even greater flexibility and control over the generated images. This would allow for more dynamic content creation in fields like virtual reality (VR) and augmented reality (AR), where high-quality, contextually accurate visuals are essential.

8.2.2 Handling Complex Attribute Interactions

While the current region-based diffusion process works well for simple prompts, handling more complex interactions between multiple text attributes remains an open chal-

lenge. Future research could focus on improving the system's ability to combine multiple font styles, colors, and other attributes within the same region. By enhancing the model's understanding of how these attributes interact, the system could generate more sophisticated and visually coherent images that maintain the fidelity of user instructions even in highly detailed or abstract prompts.

8.2.3 Semantic Understanding for Better Contextual Accuracy

One key limitation of current text-to-image models is the lack of a deep understanding of the semantic relationships between different parts of the prompt. In the future, incorporating advanced natural language processing (NLP) techniques that enable a better grasp of the semantic context of each word could lead to more accurate and coherent image generation. For instance, recognizing how different objects in the scene relate to one another based on context, instead of just individual attributes, would allow the system to generate more meaningful images that align with the user's intended narrative or story.

8.2.4 Real-Time Image Editing and Interaction

Another exciting area of future work lies in developing real-time image editing capabilities. Allowing users to interactively modify specific regions of an image—such as adjusting colors, styles, or adding new elements—and instantly seeing the changes could greatly enhance the usability of the system. This could be particularly useful in creative fields such as graphic design, advertising, and game development, where rapid iteration and precision are essential.

8.2.5 Rich-Text Editing for Existing Images

Expanding the rich-text generation method to edit existing images is another promising direction. By leveraging diffusion inversion techniques and segmentation methods, future research could allow users to modify specific parts of an image using rich-text instructions. This would make it possible to apply localized edits (e.g., changing the color of an object or altering its style) without disturbing the overall structure of the image. Such capabilities could revolutionize workflows in industries that require frequent image adjustments, such as digital marketing and visual content creation.

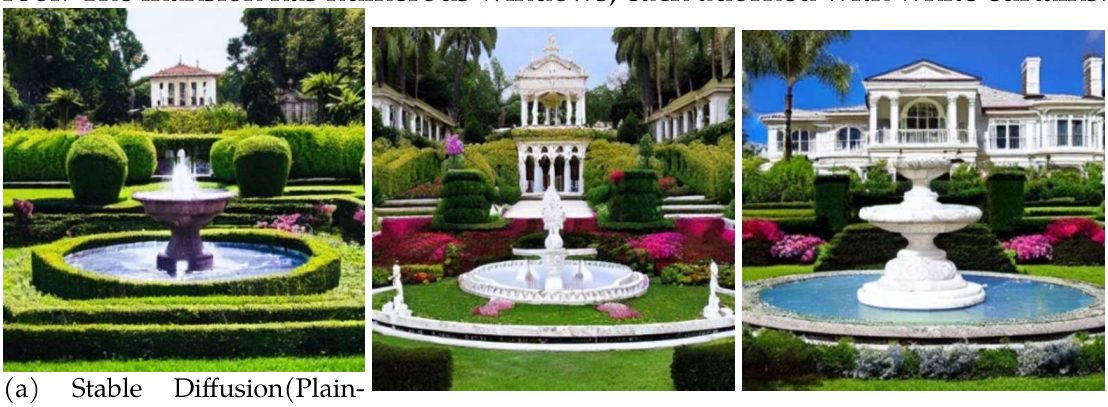
9 Appendix

In this appendix, we provide additional experimental results and details. In section A, we show the images generated by our model, Attend-and-Excite Chefer et al. (2023), Prompt-to-Prompt Hertz et al. (2023), and InstructPix2Pix Brooks et al. (2023) with various RGB colors, local styles, and detailed descriptions via footnotes. In section B, we provide additional details on the implementation and evaluation.

9.1 Appendix A Additional Results

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.



(a) Stable Diffusion(Plain-Text) (b) Stable Diffusion (Full-Text) (c) Ours

(d) Attend-and-Excite (e) Prompt-to-Prompt (f) InstructPix2Pix

Figure 20. Comparison of different methods.

9.2 Appendix A.1 Additional Results

A small chair[1] sits in front of a table[2] on the wooden floor. There is a bookshelf[3] nearby the window[4].

[1]A black leather office chair with a high backrest and adjustable arms.
[2]A large wooden desk with a stack of books on top of it
[3]A bookshelf filled with colorful books and binders.
[4]A window overlooks a stunning natural landscape of snow mountains.

(a) Stable Diffusion(Plain-Text)

(b) Stable Diffusion (Full-Text)

(c) Ours

(d) Attend-and-Excite

(e) Prompt-to-Prompt

(f) InstructPix2Pix

Figure 21. Comparison of different methods.

9.3 Appendix B Additional Details

Quantitative Evaluation of 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 Rombach et al. (2022) can generate the correct combination, as our goal is not to evaluate the compositionality of the generation as in DrawBench Saharia et al. (2022). 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
    '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 contains color names that are selected from the HTML color names[2]. 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. The complete list is below.

```
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],
    '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 Hertz et al. (2023) and InstructPix2Pix Brooks et al. (2023). 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 Brooks et al. (2023) 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 3060 GPU, each sampling based on the plain text takes around 11.06 seconds, while sampling an image with two styles takes around 18 seconds, and sampling an image with our color optimization takes around 29.05 seconds.

10 References

1. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: ICML, pp. 8821–8831 (2021)

2. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Ayan, B.K., Mahdavi, S.S., Lopes, R.G., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic text-to-image diffusion models with deep language understanding. In: NeurIPS (2022)

3. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)

4. Kang, M., Zhu, J.-Y., Zhang, R., Park, J., Shechtman, E., Paris, S., Park, T.: Scaling up GANs for text-to-image synthesis. In: CVPR (2023)

5. Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Kreis, K., Aittala, M., Aila, T., Laine, S., Catanzaro, B., et al.: ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. arXiv preprint arXiv:2211.01324 (2022)

6. Gafni, O., Polyak, A., Ashual, O., Sheynin, S., Parikh, D., Taigman, Y.: Make-a-scene: Scene-based text-to-image generation with human priors. In: ECCV (2022)

7. Zhang, L., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: ICCV (2023)

8. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: CVPR (2023)

9. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-or, D.: Prompt-to-prompt image editing with cross-attention control. In: ICLR (2023)

10. Colorado State University, T.A.P.: Tutorial: Rich Text Format (RTF) from Microsoft Word - The ACCESS Project (2012). `http://accessproject.colostate.edu/udl/modules/word/tutorials/rtf.cfm`

11. Witten, I.H., Bainbridge, D., Nichols, D.M.: How to Build a Digital Library, (2009)

12. Chefer, H., Alaluf, Y., Vinker, Y., Wolf, L., Cohen-Or, D.: Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. ACM Transactions on Graphics (TOG) 42(4), 1–10 (2023)

13. Ge, S., Park, T., Zhu, J.-Y , Huang, J.-B.: Expressive text-to-image generation with rich text. In: ICCV (2023)

14. Zhu, X., Goldberg, A.B., Eldawy, M., Dyer, C.R., Strock, B.: A text-to-picture synthesis system for augmenting communication. In: AAAI (2007)

15. Mansimov, E., Parisotto, E., Ba, J.L., Salakhutdinov, R.: Generating images from captions with attention. In: ICLR (2016)

16. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. In: NeurIPS (2022)

17. Byeon, M., Park, B., Kim, H., Lee, S., Baek, W., Kim, S.: COYO-700M: Image-Text Pair Dataset. `https://github.com/kakaobrain/coyo-dataset` (2022)

18. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. NeurIPS (2020)

19. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: ICLR (2021)

20. Ho, J., Saharia, C., Chan, W., Fleet, D.J., Norouzi, M., Salimans, T.: Cascaded diffusion models for high fidelity image generation. Journal of Machine Learning Research 23(47), 1–33 (2022)

21. Ho, J., Salimans, T.: Classifier-free diffusion guidance. In: NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications (2021)

22. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)

23. Nichol, A.Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., Mcgrew, B., Sutskever, I., Chen, M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In: ICML (2022)

24. Yu, J., Xu, Y., Koh, J.Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B.K., et al.: Scaling autoregressive models for content-rich text-to-image generation. Transactions on Machine Learning Research (2022)

25. Chang, H., Zhang, H., Barber, J., Maschinot, A., Lezama, J., Jiang, L., Yang, M.-H., Murphy, K., Freeman, W.T., Rubinstein, M., et al.: Muse: Text-to-image generation via masked generative transformers. arXiv preprint arXiv:2301.00704 (2023)

26. Ding, M., Zheng, W., Hong, W., Tang, J.: Cogview2: Faster and better text-to-image generation via hierarchical transformers. arXiv preprint arXiv:2204.14217 (2022)

27. Sauer, A., Karras, T., Laine, S., Geiger, A., Aila, T.: Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. arXiv preprint arXiv:2301.09515 (2023)

28. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In: CVPR (2023)

29. Kumari, N., Zhang, B., Zhang, R., Shechtman, E., Zhu, J.-Y.: Multi-concept customization of text-to-image diffusion. In: CVPR (2023)

30. Avrahami, O., Hayes, T., Gafni, O., Gupta, S., Taigman, Y., Parikh, D., Lischinski, D., Fried, O., Yin, X.: Spatext: Spatio-textual representation for controllable image generation. In: CVPR (2023)

31. Wu, J.Z., Ge, Y., Wang, X., Lei, S.W., Gu, Y., Shi, Y., Hsu, W., Shan, Y., Qie, X., Shou, M.Z.: Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation, 7623–7633 (2023)

32. Kawar, B., Zada, S., Lang, O., Tov, O., Chang, H., Dekel, T., Mosseri, I., Irani, M.: Imagic: Text-based real image editing with diffusion models. In: CVPR (2023)

33. Ma, Y., He, Y., Cun, X., Wang, X., Shan, Y., Li, X., Chen, Q.: Follow Your Pose: Pose-Guided Text-to-Video Generation using Pose-Free Videos (2023)

34. Li, Y., Liu, H., Wu, Q., Mu, F., Yang, J., Gao, J., Li, C., Lee, Y.J.: Gligen: Open-set grounded text-to-image generation. In: CVPR (2023)

35. Meng, C., Song, Y., Song, J., Wu, J., Zhu, J.-Y., Ermon, S.: Sdedit: Image synthesis and editing with stochastic differential equations. In: ICLR (2022)

36. Choi, J., Kim, S., Jeong, Y., Gwon, Y., Yoon, S.: Ilvr: Conditioning method for denoising diffusion probabilistic models. In: ICCV (2021)

37. Parmar, G., Singh, K.K., Zhang, R., Li, Y., Lu, J., Zhu, J.-Y.: Zero-shot image-to-image translation. In: SIGGRAPH (2023)

38. Bansal, A., Chu, H.-M., Schwarzschild, A., Sengupta, S., Goldblum, M., Geiping, J., Goldstein, T.: Universal guidance for diffusion models. arXiv preprint arXiv:2302.07121 (2023)

39. Avrahami, O., Fried, O., Lischinski, D.: Blended latent diffusion. arXiv preprint arXiv:2206.02779 (2022)

40. Jimenez, A.B.: Mixture of diffusers for scene composition and high resolution image generation. arXiv preprint arXiv:2302.02412 (2023)

41. Bar-Tal, O., Yariv, L., Lipman, Y., Dekel, T.: Multidiffusion: Fusing diffusion paths for controlled image generation. In: ICML (2023)

42. Sarukkai, V., Li, L., Ma, A., R'e, C., Fatahalian, K.: Collage diffusion. ArXiv abs/2303.00262 (2023)

43. Zhang, Q., Song, J., Huang, X., Chen, Y., Liu, M.-Y.: Diffcollage: Parallel generation of large content with diffusion models. (2023)

44. Cao, M., Wang, X., Qi, Z., Shan, Y., Qie, X., Zheng, Y.: Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In: ICCV (2023)

45. Phung, Q., Ge, S., Huang, J.-B.: Grounded text-to-image synthesis with attention refocusing. In: CVPR (2024)

46. Xiao, G., Yin, T., Freeman, W.T., Durand, F., Han, S.: Fastcomposer: Tuning-free multi-subject image generation with localized attention. arXiv preprint arXiv:2305.10431 (2023)

47. Feng, W., He, X., Fu, T.-J., Jampani, V., Akula, A.R., Narayana, P., Basu, S., Wang, X.E., Wang, W.Y.: Training-free structured diffusion guidance for compositional text-to-image synthesis. In: ICLR (2023)

48. Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., Fleet, D.J.: Video diffusion models. In: NeurIPS (2022)

49. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3D object . In: ICCV (2023)

50. Tseng, H.-Y., Li, Q., Kim, C., Alsisan, S., Huang, J.-B., Kopf, J.: Consistent view synthesis with pose-guided diffusion models. In: CVPR (2023)

51. Watson, D., Chan, W., Martin-Brualla, R., Ho, J., Tagliasacchi, A., Norouzi, M.: Novel View Synthesis with Diffusion Models (2022)

52. Patashnik, O., Garibi, D., Azuri, I., Averbuch-Elor, H., Cohen-Or, D.: Localizing object-level shape variations with text-to-image diffusion models. In: ICCV (2023)

53. Liu, S., Zhang, Y., Li, W., Lin, Z., Jia, J.: Video-p2p: Video editing with cross-attention control. arXiv:2303.04761 (2023)

54. QI, C., Cun, X., Zhang, Y., Lei, C., Wang, X., Shan, Y., Chen, Q.: Fatezero: Fusing attentions for zero-shot text-based video editing. In: ICCV (2023)

55. Ceylan, D., Huang, C.-H., Mitra, N.J.: Pix2video: Video editing using image diffusion. arXiv:2303.12688 (2023)

56. Ma, W.-D.K., Lewis, J., Kleijn, W.B., Leung, T.: Directed diffusion: Direct control of object placement through attention guidance. arXiv preprint arXiv:2302.13153 (2023)

57. Meng, Y., Wu, W., Wang, F., Li, X., Nie, P., Yin, F., Li, M., Han, Q., Sun, X., Li, J.: Glyce: Glyph-vectors for Chinese character representations. NeurIPS 32 (2019)

58. Sun, Z., Li, X., Sun, X., Meng, Y., Ao, X., He, Q., Wu, F., Li, J.: Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. ACL (2021)

59. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: Layoutlm: Pre-training of text and layout for document image understanding. In: SIGKDD (2020)

60. Li, J., Xu, Y., Lv, T., Cui, L., Zhang, C., Wei, F.: Dit: Self-supervised pre-training for document image transformer. In: MM (2022)

61. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR (2016)

62. Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017)

63. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: CVPR (2017)

64. Reinhard, E., Adhikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer Graphics and Applications 21(5), 34–41 (2001)

65. Tai, Y.-W., Jia, J., Tang, C.-K.: Local color transfer via probabilistic segmentation by expectation-maximization. In: CVPR (2005)

66. Xu, L., Yan, Q., Jia, J.: A sparse control model for image and video editing. ACM Transactions on Graphics (TOG) 32, 1–10 (2013)

67. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. SIG (2004)

68. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016)

69. Zhang, R., Zhu, J.-Y., Isola, P., Geng, X., Lin, A.S., Yu, T., Efros, A.A.: Real-time user-guided image colorization with learned deep priors. ACM Transactions on Graphics (TOG) 9(4) (2017)

70. Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., et al.: Improving image generation with better captions. Computer Science. https://cdn.openai.com/papers/dall-e-3.pdf 2(3), 8 (2023)

71. Wang, X., Darrell, T., Rambhatla, S.S., Girdhar, R., Misra, I.: Instancediffusion: Instance-level control for image generation. arXiv preprint arXiv:2402.03290 (2024)

72. Wu, W., Li, Z., He, Y., Shou, M.Z., Shen, C., Cheng, L., Li, Y., Gao, T., Zhang, D., Wang, Z.: Paragraph-to-image generation with information-enriched diffusion model. arXiv preprint arXiv:2311.14284 (2023)

73. Bakr, E.M., Sun, P., Shen, X., Khan, F.F., Li, L.E., Elhoseiny, M.: Hrs-bench: Holistic, reliable and scalable benchmark for text-to-image models. In: ICCV (2023)

74. Hu, Y., Liu, B., Kasai, J., Wang, Y., Ostendorf, M., Krishna, R., Smith, N.A.: Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In: ICCV (2023)

75. Huang, K., Sun, K., Xie, E., Li, Z., Liu, X.: T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. In: NeurIPS (2024)

76. Patel, M., Gokhale, T., Baral, C., Yang, Y.: Conceptbed: Evaluating concept learning abilities of text-to-image diffusion models. In: AAAI (2024)

77. Zhao, L., Zhao, T., Lin, Z., Ning, X., Dai, G., Yang, H., Wang, Y.: Flasheval: Towards fast and accurate evaluation of text-to-image diffusion generative models. arXiv preprint arXiv:2403.16379 (2024)

78. Sahuguet, A., Azavant, F.: Wysiwyg web wrapper factory (w4f) (1999)

79. Litt, G., Lim, S., Kleppmann, M., Hardenberg, P.: Peritext: A crdt for collaborative rich text editing. Proceedings of the ACM on Human-Computer Interaction (PACMHCI) (2022)

80. Ignat, C.-L., Andre, L., Oster, G.: Enhancing rich content wikis with real-time collaboration. Concurrency and Computation: Practice and Experience 33(8), 4110 (2021)

81. Agrawala, M.: Unpredictable Black Boxes are Terrible Interfaces. https://magrawala.substack.com/p/unpredictable-black-boxes-are-terrible (2023)

82. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: CVPR (2015)

83. Johnson, J., Karpathy, A., Fei-Fei, L.: Densecap: Fully convolutional localization networks for dense captioning. In: CVPR (2016)

84. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML, pp. 8748–8763 (2021). PMLR

85. Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A.H., Chechik, G., Cohen-or, D.: An image is worth one word: Personalizing text-to-image generation using textual inversion. In: ICLR (2023). https://openreview.net/forum?id=NAQvF08TcyG

86. Chen, X., Huang, L., Liu, Y., Shen, Y., Zhao, D., Zhao, H.: Anydoor: Zero-shot object-level image customization. arXiv preprint arXiv:2307.09481 (2023)

87. Li, D., Li, J., Hoi, S.C.: Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing. arXiv preprint arXiv:2305.14720 (2023)

88. Jia, X., Zhao, Y., Chan, K.C., Li, Y., Zhang, H., Gong, B., Hou, T., Wang, H., Su, Y.-C.: Taming encoder for zero fine-tuning image customization with text-to-image diffusion models. arXiv preprint arXiv:2304.02642 (2023)

89. Chen, W., Hu, H., Li, Y., Rui, N., Jia, X., Chang, M.-W., Cohen, W.W.: Subject-driven text-to-image generation via apprenticeship learning. arXiv preprint arXiv:2304.00186 (2023)

90. Gal, R., Arar, M., Atzmon, Y., Bermano, A.H., Chechik, G., Cohen-Or, D.: Encoder-based domain tuning for fast personalization of text-to-image models. ACM Transactions on Graphics (TOG) 42(4), 1–13 (2023)

91. Shi, J., Xiong, W., Lin, Z., Jung, H.J.: Instantbooth: Personalized text-to-image generation without test-time finetuning. arXiv preprint arXiv:2304.03411 (2023)

92. Tang, R., Pandey, A., Jiang, Z., Yang, G., Kumar, K.V.S.M., Lin, J., Ture, F.: What the daam: Interpreting stable diffusion using cross attention. ArXiv abs/2210.04885 (2022)

93. Tumanyan, N., Geyer, M., Bagon, S., Dekel, T.: Plug-and-play diffusion features for text-driven image-to-image translation. In: CVPR (2023)

94. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)

95. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing 17, 395–416 (2007)

96. Wen, Y., Jain, N., Kirchenbauer, J., Goldblum, M., Geiping, J., Goldstein, T.: Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. In: NeurIPS (2023)

97. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)

98. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. In: NeurIPS

99. OpenAI: GPT-4 Technical Report (2023)

100. Li, C., Xu, H., Tian, J., Wang, W., Yan, M., Bi, B., Ye, J., Chen, H., Xu, G., Cao, Z., et al.: mplug: Effective and efficient vision-language learning by cross-modal skip-connections. In: EMNLP (2022)

101. Huberman-Spiegelglas, I., Kulikov, V., Michaeli, T.: An edit friendly ddpm noise space: Inversion and manipulations. arXiv preprint arXiv:2304.06140 (2023)

102. Wu, C.H., Torre, F.: Unifying diffusion models' latent space, with applications to cyclediffusion and guidance. arXiv preprint arXiv:2210.05559 (2022)

103. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.-Y., Dollar, P., Girshick, R.: Segment anything. In: ICCV (2023)

104. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499 (2023)

105. Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., Chen, J., Huang, X., Chen, Y., Yan, F., Zeng, Z., Zhang, H., Li, F., Yang, J., Li, H., Jiang, Q., Zhang, L.: Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks (2024)