TextDiffuser-2: Unleashing the Power of Language Models for Text Rendering

Jingye Chen*[13], Yupan Huang* [23], Tengchao Lv[3], Lei Cui[3],
Qifeng Chen[1], Furu Wei[3]
[1]HKUST [2]Sun Yat-sen University [3]Microsoft Research
qwerty.chen@connect.ust.hk,
{yupanhuang,tengchaolv,lecu,fuwei}@microsoft.com, cqf@ust.hk

Fig. 1: Text-to-image results generated by TextDiffuser-2. Alongside accurate text generation, TextDiffuser-2 offers reasonable text layouts and exhibits diversity in text style powered by the strong capability of language models.

Abstract. The diffusion model has been proven a powerful generative model in recent years, yet it remains a challenge in generating visual text. Although existing work has endeavored to enhance the accuracy of text rendering, these methods still suffer from several drawbacks, such as (1) limited flexibility and automation, (2) constrained capability of layout prediction, and (3) restricted diversity. In this paper, we present TextDiffuser-2, aiming to unleash the power of language models for text rendering while taking these three aspects into account. Firstly, we fine-tune a large language model for layout planning. The large language model is capable of automatically generating keywords and placing the text in optimal positions for text rendering. Secondly, we utilize the language model within the diffusion model to encode the position and content of keywords at the line level. Unlike previous methods that employed tight character-level guidance, our approach generates more diverse text images. We conduct extensive experiments and incorporate user studies involving human participants and GPT-4V, validating TextDiffuser-2's capacity to achieve a more rational text layout and generation with enhanced diversity. Furthermore, the proposed methods are compatible with existing text rendering techniques, such as TextDiffuser and GlyphControl, serving to enhance automation and diversity, as well as augment

* Work done during internship at Microsoft Research.

the rendering accuracy. For instance, by using the proposed layout planner, TextDiffuser is capable of rendering text with more aesthetically pleasing line breaks and alignment, meanwhile obviating the need for explicit keyword specification. Furthermore, GlyphControl can leverage the layout planner to achieve diverse layouts without the necessity for user-specified glyph images, and the rendering F-measure can be boosted by 6.51% when using the proposed layout encoding training technique. The code and model are available at `https://aka.ms/textdiffuser-2`.

1 Introduction

In recent years, diffusion models [15,18,40,42,45,56,58] have successfully revolutionized the field of image synthesis. Despite showcasing impressive performance, most existing diffusion models still fall short in rendering visual text. Specifically, existing diffusion models often generate unintended symbols or artifacts during the text rendering process [9], which significantly impairs the visual quality of the generated images. Notably, text is ubiquitous in daily life, encompassing logos, banners, book covers, newspapers, etc. In this case, how to generate images with accurate, visually appealing, and coherent visual text is a crucial problem.

Through investigation, there has been a few research works [1, 3, 8, 11, 19, 29, 31, 32, 43, 48, 52, 55] focusing on visual text rendering. Some works [1, 11, 29, 43] validate that using powerful language models [39,50] as text encoders benefits the text rendering process. However, it is observed that they still can not achieve satisfactory rendering accuracy. Other works utilize explicit text position and content guidance. Although showing impressive rendering accuracy, we have noticed several drawbacks in these methods: (1) ***Limited flexibility and automation***. GlyphControl [52], AnyText [48], and Brush Your Text [55] need users to design glyph images or text region masks/contours to provide layout guidance. This extra step lacks automation as opposed to simply providing prompts. Moreover, despite supporting conversions from text to image, GlyphDraw [31] and TextDiffuser [3] rely on the manual specification of keywords with quotation marks. These requirements hinder the direct conversion of natural user prompts into corresponding images, thereby narrowing the flexibility and automation capabilities. Such constraints may impede future advancements toward more intuitive interfaces, such as converting voice commands into images; (2) ***Constrained capability of layout prediction***. GlyphDraw [31] can only render images with a single text line, constraining its applicability for scenarios, such as posters and book covers, involving multiple text lines. For TextDiffuser [3], the produced text layouts are not visually appealing, which is primarily attributed to the limited capability of the Layout Transformer; (3) ***Restricted diversity***. For TextDiffuser [3], the utilization of character-level segmentation masks as control signals implicitly imposes constraints on the position of each character, thereby restricting the diversity of text styles. For methods that require user-specific text placements, exemplified by GlyphControl [52], Brush Your Text [55], and AnyText [48], there is also a lack of diversity since the position of text regions are predetermined and immutable.

Given these observations, we introduce TextDiffuser-2 in this paper, taking advantage of two language models for text rendering. Firstly, we ***tame a language model into a layout planner*** using the caption-OCR pairs in the MARIO-10M dataset [3]. The language model demonstrates flexibility and automation by inferring keywords from user prompts or incorporating user-specified keywords to determine their positions. Secondly, we ***use the language model in the diffusion model as the layout encoder*** to represent the position and content of text at the line level. Contrary to prior methods that utilized tight character-level guidance, this approach enables diffusion models to generate text images with broader diversity. Some samples are shown in Figure 1.

We primarily evaluate the capabilities of TextDiffuser-2 on two tasks, namely, **text-to-image**, which denotes the conversion from prompts to images without other conditions, and **text inpainting**, which involves the modification or addition of text on given images. Through comprehensive experiments and user studies that engaged both human participants and GPT-4V, we validate that our method can generate reasonable and visually pleasing text layouts, and it enhances the style diversity of the generated text. Additionally, the proposed method is compatible with current text rendering approaches, facilitating increased automation and diversity, as well as boosting rendering accuracy. For example, when replacing the original Layout Transformer with the proposed layout planner, TextDiffuser can create text with better breaking and alignment, and there is no need for users to specify keywords. Additionally, GlyphControl can take advantage of diverse layouts produced by the layout planner, without the need for users to supply any glyph images. Besides, the rendering F-measure scores of TextDiffuser and GlyphControl are boosted by 1.60% and 6.51% when using the layout encoding training technique. The code and model are available at `https://aka.ms/textdiffuser-2` to promote future research.

2 Related work

Visual text rendering. Despite the significant advancements in diffusion models [15, 18, 40, 54, 56], the generation of visual text rendering remains a persistent challenge. Current works can be broadly classified into two distinct approaches according to the input type. **The first approach** involves using solely user-provided prompts, without any additional conditions, to generate text images. Some studies [1, 29, 43] have leveraged large language models [39, 50] to augment the text generation capabilities of generative models. Other works utilize two-stage frameworks. For instance, GlyphDraw [31] initially employs a diffusion model to create single-line text region masks from the given prompts, followed by another diffusion model to generate the final text image outputs. TextDiffuser [3] utilizes a Layout Transformer to convert user-specified keywords into an intermediate representation, and then employs a diffusion model to generate the text images. Some commercial applications such as Midjourney-v6 [32], Ideogram [19], and DALL-E 3 [8] have also optimized the process of text rendering conditioned with prompts. **The second approach** requires users to explicitly specify text

content and placement. For example, GlyphControl [52] requires users to provide a glyph image to guide the position and content of the text to be drawn. Brush Your Text [55] utilizes a pre-trained ControlNet [56] to produce text images based on provided text contours. AnyText [48] demands more detailed inputs, including both the glyph image used in GlyphControl and binary text masks.

Our proposed model, TextDiffuser-2, falls into **the first category**. In practical image generation scenarios, users may find it challenging to predetermine the precise positioning of keywords on the image. TextDiffuser-2 is designed to offer superior flexibility, catering to the users' need for a more intuitive and less constrained image generation process.

Visual text modification Recent years have seen significant advancements in visual text modification. Notably, since generative models often struggle with text rendering, there is a considerable demand for using extra models to modify the text in the generated images. Current research falls into two categories: text editing and text inpainting. Text editing [10, 22, 23, 37, 41, 49, 51] constrain that the edited text should maintain the original style and the background should be identical. By contrast, text inpainting [3, 20, 48] is a relatively new field. It requires that the generated text should harmonize with the surrounding text without strictly enforcing text style and background to be the same. Such property allows the model to produce more varied results. In this paper, we also seek to explore the capability of TextDiffuser-2 for the text inpainting task.

Language model for layout generation. Layout generation [16, 21, 24, 26] has a wide range of applications, including document formatting [17, 33], screen UI design [12], and image synthesis [13,25]. Previous methods [21,24] usually model layout generation as a regression task, representing bounding boxes using continuous coordinates. Recent advancements, such as Pix2Seq [5,6], have explored alternative methods by treating coordinates as discrete language tokens. Another work, LayoutGPT [13], carefully designs prompts to guide GPT-4 [14] generating formatted layout information to assist in image synthesis. Recently, some multimodal large language models [2,30,34,53,57,60] have also adopted this design for grounding specific objects in images. In line with these designs, TextDiffuser-2 aims to leverage language models as layout planners for visual text rendering. It is a **challenging task** since the layout planner should consider the intrinsic properties of text, including its length, line breaking, and alignments.

3 Methodology

The architecture of TextDiffuser-2 is depicted in Figure 2, where the language model $\mathbf{M_1}$ and the diffusion model are trained in two stages. In the following, we introduce the role of two language models, including a language model for layout planning and another language model for layout encoding. At last, we discuss the compatibility of the proposed components for existing works.

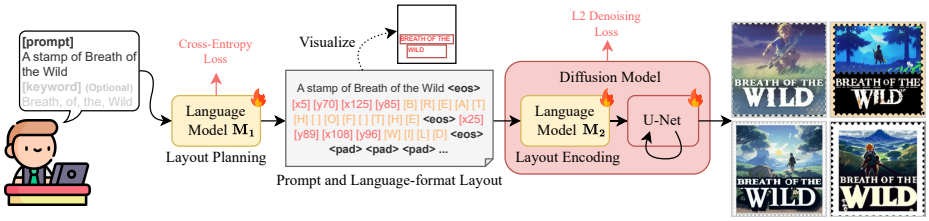

Fig. 2: The architecture of TextDiffuser-2. The language model $\mathbf{M_1}$ and the diffusion model are trained in two stages. The language model $\mathbf{M_1}$ can convert the user prompt into a language-format layout to specify the content and position of the text to be rendered. It also allows users to specify keywords optionally. Further, the prompt and language-format layout is encoded with the trainable language model $\mathbf{M_2}$ within the diffusion model to generate images. $\mathbf{M_1}$ is trained via the cross-entropy loss in the first stage, while $\mathbf{M_2}$ and U-Net are trained using the denoising L2 loss in the second stage.

3.1 Language model for layout planning

Recent research has revealed that benefiting from the extensive training data across various domains, large language models [14, 46, 47] exhibit expertise beyond the language domain, such as layout planning [13, 27]. Inspired by this, we tame a large language model into a layout planner, which can organize the content and position of text to be rendered on an image, based on the provided prompts. Specifically, we seek to fine-tune a pre-trained large language model $\mathbf{M_1}$, which functions as a decoder, using caption-OCR pairs. As demonstrated in Figure 2, TextDiffuser-2 supports two scenarios: (1) If users do not explicitly provide keywords, the language model should infer the text and layout to be drawn on the image; (2) If users provide keywords (marked in gray color), the language model only needs to determine the corresponding layout for the keywords. Specifically, the input follows the format "[**description**] Prompt: [**prompt**] Keywords: [**keywords**]"[1]. Each output follows the format "*textline* x_0, y_0, x_1, y_1", where (x_0, y_0) and (x_1, y_1) represent the coordinates of the top-left corner and bottom-right corner, respectively. Some training samples are shown in Figure 3. We optimize the language model with cross-entropy loss, training simultaneously for scenarios with and without keywords. We use all the text detected in the OCR results as keywords to formulate the input. Please note that the rectangle boxes used in the layout merely specify the text regions and **do not constrain** the generated text to be horizontal. The generated text can be inclined or curve as well, as shown in the last two columns in Figure 1.

[1] Task description: Given a prompt that will be used to generate an image, plan the layout of visual text for the image. The size of the image is 128x128. Therefore, none of the properties of the positions should exceed 128, including the coordinates of the top, left, right, and bottom. You don't need to specify the details of font styles. At each line, the format should be textline left, top, right, and bottom. So let us begin.

3.2 Language model for layout encoding

Based on the layouts generated by $\mathbf{M_1}$, we leverage the latent diffusion models [40] for image generation. Different from TextDiffuser [3] which incorporates text information using segmentation masks and GlyphControl [52] which duplicates backbone parameters to accommodate the glyph image conditions, we introduce a simple and parameter-free strategy by combining the prompt and the layout for the language model $\mathbf{M_2}$, *i.e.*, the text encoder within the latent diffusion model. In contrast to character-level segmentation masks that regulate the position of individual characters, the line-level bounding box offers greater flexibility during generation and does not constrain the diversity of styles.

Previous work [29] demonstrates that fine-grained tokenization (*e.g.,* using ByT5 [50]) can enhance the spelling capability of diffusion models. Inspired by this, we design a hybrid-granularity tokenization method that utilizes conventional approaches (*e.g.,* BPE tokenization method [44]) to encode the prompt, followed by position and character tokens of each keyword. For example, the word "WILD" is decomposed into tokens "[W]", "[I]", "[L]", "[D]". Additionally, we introduce new coordinate tokens to encode the position of each keyword. For instance, the tokens "[x5]" and "[y70]" correspond to an x-coordinate of 5 and a y-coordinate of 70, respectively. Each keyword information is separated by the end-of-sentence token "⟨eos⟩", and any remaining space to the maximum length L will be filled with padding tokens "⟨pad⟩". We train the whole diffusion model, including the language model $\mathbf{M_2}$ and U-Net, using the L2 denoising loss.

3.3 Compatibility with existing text render methods

The proposed layout planning and layout encoding techniques can be used for existing text rendering methods. For instance, the results of the layout planner can be utilized for methods that do not directly support text-to-image conversion, such as GlyphControl [52] and Brush Your Text [48]. Moreover, diverse layouts can be obtained when using different seeds for sampling from the layout planner. Visualizations are shown in Figure 6. Additionally, we observe that the existing methods benefit from the layout encoding strategy during training. For example, GlyphControl [52] obtains significant improvement regarding rendering accuracy (12.65%↑). We will provide more details in the experiment section.

4 Experiments

Implementation details. For ***layout planning***, we fine-tune the vicuna-7b-v1.5 [7] model based on the FastChat framework [59]. The training data of the caption-OCR pairs are derived from the MARIO-10M dataset [3], and we use 5k samples for fine-tuning (some samples are shown in Figure 3). We normalize the positions to the range of 0∼128 to increase the compactness of the coordinate feature space. The learning rate is set to 2e-5, and we conduct a total of 6 epochs of fine-tuning with a batch size of 256. It takes one day to train with 8 A100

Input Prompt	Output Layout	Visualization
Comprehensive guide to Credit Appraisal Process in Banks	Comprehensive 50,33,114,46 Guide 70,51,94,62 Credit Appraisal Process 8,94,124,109	
The Spellmans Strike Again (Izzy Spellman) Cover	THE 58,10,70,17 SPELLMANS 24,17,103,39 STRIKE 25,39,102,67 AGAIN 24,64,104,100	

Fig. 3: Samples for training the layout planner $\mathbf{M_1}$. The training dataset derives from the MARIO-10M dataset [3]. More samples are shown in the Appendix A.

Table 1: Ablation studies on the amount of fine-tuning data. The "0k-2shot" setting denotes the use of two examples for few-shot learning, without any additional fine-tuning. When using 5k data, the language model $\mathbf{M_1}$ performs better. The percentage sign is omitted, as is consistent with the following tables. 'Pre', 'Rec', and 'F' denote precision, recall, and f-measure, same as follows.

#Data	Acc↑	Pre↑	Rec↑	F↑	IOU↓
0k-2shot	49.65	84.18	69.69	76.25	19.69
2.5k	61.10	82.20	85.18	83.67	**3.21**
5k	**64.85**	84.98	**86.38**	**85.67**	3.25
10k	**64.85**	84.38	86.23	85.29	4.27
50k	63.72	**85.32**	85.78	85.55	3.68
100k	62.87	85.26	85.98	85.62	4.31

GPU cards. During the inference stage, when using a single A100 GPU card, the average time to generate a layout for each prompt is 1.1 seconds. For ***layout encoding***, we utilize SD 1.5[2] [40] and use the built-in CLIP text encoder with base size [38]. The whole model consists of 922M parameters. We incorporate special tokens, including 256-coordinate tokens and 95-character tokens. The alphabet contains 26 uppercase and 26 lowercase letters, 10 numbers, 32 punctuation marks, and a space. The size of the input image is 512×512. The model is trained for 6 epochs on the MARIO-10M dataset [3] with a learning rate of 1e-4 and a batch size of 576. MARIO-10M [3] has already been filtered to remove noisy samples. The maximum length L is set to 128. More details about the choice of L are shown in Appendix B. It takes one week to train the whole diffusion model with 8 A100 GPU cards. When sampling with 50 steps, the generation for a single image costs 6 seconds.

[2] Please note that our design is also compatible with various text-conditioned generative models, such as SD-XL [36] and StableCascade [35], by training with additional tokens indicating the text content and position. Limited by computing resources, our experiments are confined to the SD 1.5 model.

Table 2: Ablation studies on the representation of coordinates and the tokenization level. 'L', 'T', 'R', and 'B' denote left, top, right, and bottom. "Char" refers to tokenizing keywords into individual characters, whereas "Subword" refers to the use of BPE for tokenizing into subwords. Using the top-left and bottom-right corners and character-level tokenization achieves better performance.

Representation	Acc↑	Pre↑	Rec↑	F↑
Center (Char)	35.19	61.75	62.71	62.23
LT (Char)	28.32	54.94	55.64	55.29
LT+RB (Subword)	15.48	41.74	42.53	42.13
LT+RB (Char)	**57.58**	**74.02**	**76.14**	**75.06**

4.1 Ablation studies

How much data is needed for fine-tuning the layout planner $\mathbf{M_1}$*?* As illustrated in Table 1, we conduct experiments with different data amount, including 0k, 2.5k, 5k, 10k, 50k, and 100k. Particularly, in the 0k setting, we provide two examples of few-shot learning inspired by LayoutGPT [13] and LayoutPrompter [26]. In the absence of examples, the result often fails to conform to the appropriate format. We evaluate our approach using the MARIO-Eval benchmark [3], which consists of prompt and keyword pairs. Besides, the quotation marks in the prompt are removed for evaluation. Since the LAIONEval subset within the MARIO-Eval benchmark depends on OCR results to infer keywords for prompts, it contains some noise and is unreliable for accurate assessments. Therefore, we decided not to use it in this keyword extraction experiment. For evaluation, we use accuracy, precision, recall, and F-measure to assess the model's ability to extract keywords. Additionally, we introduce an IoU metric to measure the maximum IoU value between the generated boxes for each sample (only those samples with more than one predicted box are calculated). The experimental results showcase that the model achieves optimal performance in the majority of metrics when fine-tuned with 5k data. We speculate the main reason that fine-tuning with more than 5k data did not yield better performance is that the LLM has been pre-trained on large-scale datasets, and a small dataset curated for our text rendering task is adequate for good performance. Therefore, we employ the model fine-tuned on 5k data for layout planning in subsequent experiments. We visualize some samples in Figure 4. We notice that the language model exhibits flexibility in generating keywords, such as determining the case of the keyword or introducing appropriate words beyond the provided prompt. More samples are shown in Appendix C.

How to represent the position of text lines? Apart from utilizing the top-left and bottom-right corners to represent a text line, we also investigate alternative single-point representations, such as employing the top-left point or the center point. Intuitively, using a single point to represent a text line provides more flexibility, enabling the generated text to exhibit greater diversity in angles and

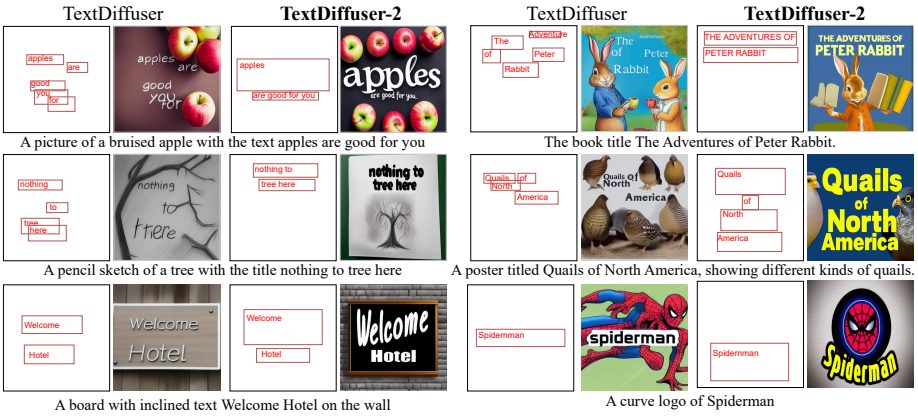

Fig. 4: Visualizations of layout predictions. We explicitly specify the keywords using quotation marks for TextDiffuser. It is observed that TextDiffuser-2 generates more visually pleasing and rational layouts compared with TextDiffuser. Specifically, the layout planner of TextDiffuser-2 exhibits enhanced capabilities in terms of text line breaking and alignment. Furthermore, according to the last row, it can predict boxes with suitable sizes that accommodate the text style, such as inclined and curve.

sizes. In Appendix D, visualizations are shown to validate the diversity of the generation using single-point conditions. However, as shown in Table 2, we notice that there is a considerable decline in the OCR accuracy of the single-point representation on the MARIO-Eval benchmark [3]. For example, compared with the LT-RB setting, the accuracy of the center and LR settings declined by 22.39% and 29.26%. We also experiment without using newly added positional tokens. In this case, the text encoder tokenizes coordinate numerals into individual digits (for instance, '127' is tokenized as '1', '2', and '7'). Experimental results indicate that, compared to the incorporation of new positional tokens, this approach results in a 10.89% reduction in the OCR F-measure. Hence, we leverage the top-left and bottom-right corners with introduced positional tokens to represent the box in the following experiments. We also explore the inclusion of angle information in Appendix E.

Should text be tokenized at the character or subword level? We also explore Byte Pair Encoding (BPE) to tokenize keywords into the subword level. As shown in Table 2, we observe that using subword-level tokenization significantly underperforms character-level representation, *i.e.*, it is lower by 42.1% on the accuracy metric. When using subword-level tokenization, the model becomes insensitive to the spelling of each token, which poses significant challenges to the text rendering process.

Table 3: Quantitative results of the text-to-image task on the MARIO-10M benchmark. 'LP' and 'LE' denote the layout planner and layout encoding. TextDiffuser-2 outperforms other text-to-image methods in terms of the FID, CLIPScore, and OCR accuracy metrics. The proposed components can also be used for existing works. It is observed that TextDiffuser and GlyphControl gain significant improvement when trained with the layout encoding. *The training script of AnyText is not available during submission so we cannot obtain the re-training results with the LE strategy.

#	Methods	FID↓	CLIPScore↑	OCR(F-measure)↑	OCR(Acc)↑
Text-to-image without additional conditions					
1	SD 1.5 [40]	51.30	30.15	0.02	0.00
2	SD 2.1 [40]	51.40	31.03	5.00	0.04
3	SD-XL [36]	62.54	31.31	3.66	0.31
4	StableCascade [35]	70.37	30.67	13.11	2.18
5	Deepfloyd [11]	34.90	32.67	17.62	2.62
6	PixArt-α [4]	87.09	27.88	0.03	0.02
7	TextDiffuser [3]	38.76	34.36	**78.24**	56.09
8	**TextDiffuser-2**	**33.66**	**34.50**	75.06	**57.58**
Proposed components for existing adaptive-text-style methods					
9	TextDiffuser [3]+LP	37.05	34.56	85.70	62.02
10	TextDiffuser [3]+LP+LE	$36.41_{\downarrow 0.64}$	$34.47_{\downarrow 0.09}$	$87.30_{\uparrow 1.60}$	$62.25_{\uparrow 0.23}$
11	GlyphControl [52]+LP	50.82	34.56	64.07	32.56
12	GlyphControl [52]+LP+LE	$43.22_{\downarrow 7.60}$	$34.62_{\uparrow 0.06}$	$70.58_{\uparrow 6.51}$	$45.21_{\uparrow 12.65}$
13	AnyText [48]+LP*	59.26	33.90	72.27	37.46
Proposed components for existing fixed-text-style training-free methods					
14	ControlNet [56]+LP	46.52	34.72	75.54	35.16
15	Brush Your Text [55]+LP	44.74	30.55	78.91	40.07

4.2 Experimental results

Quantitative results. As shown in Table 3, we conduct quantitative experiments on the MARIO-Eval benchmark [3] to evaluate the text-to-image capability (#1∼#10) compared with open-source methods. The detail of each method is shown in Appendix F. The experimental results demonstrate that TextDiffuser-2 outperforms other methods in terms of the FID, CLIPScore, and OCR accuracy metrics. It is noteworthy that TextDiffuser mainly renders text in a stand font (see Figure 7), thereby reducing the complexity of the rendering process. This strategy sacrifices font style diversity to enhance the accuracy of text rendering. By contrast, while maintaining the ability to generate accurate text, TextDiffuser-2 can generate text with greater diversity.

We also explore the usage of proposed components for existing text rendering methods (#9∼#15). The details about the incorporation of each method are in the Appendix G. Specifically, the layout planner can replace the original low-capability Layout Transformer in TextDiffuser, or serve as a plug-in module for methods that do not support image generation conditioned only with prompts (*e.g.,* GlyphControl, AnyText, ControlNet, and Brush Your Text). Moreover, we validate that the proposed layout encoding technique can further boost the

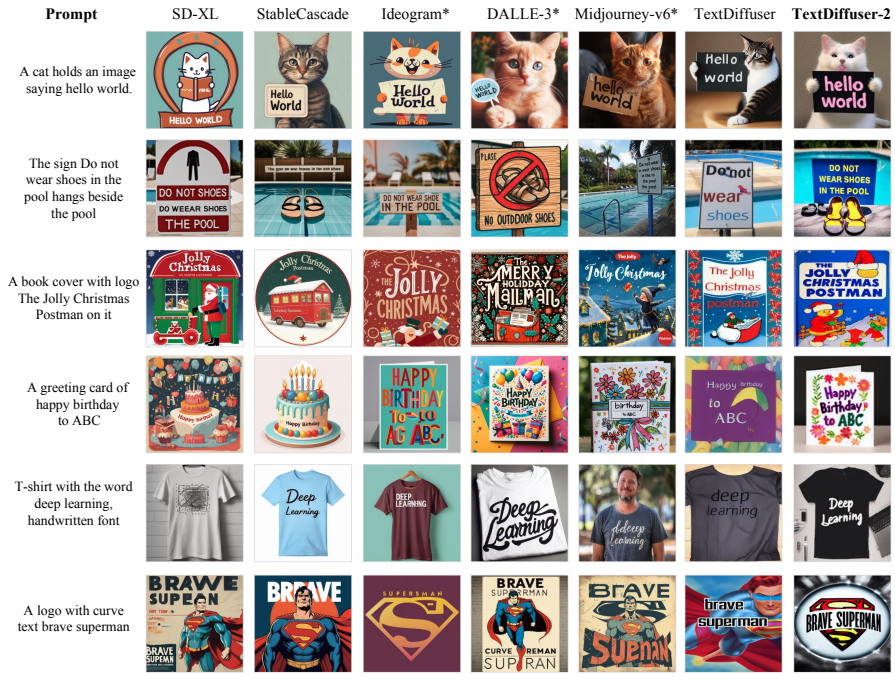

Fig. 5: Visualizations of text-to-image results compared with open-source and closed-source (marked with '*') works. Note that we use additional quotation marks to specify keywords for other methods. Specifically, TextDiffuser-2 can automatically extract keywords from prompts with specifications. Additionally, TextDiffuser-2 demonstrates better rendering accuracy and follows the font styles specified in the prompts. The compared methods are hindered by the spelling errors. Although TextDiffuser mitigates this issue, it encounters limitations in text style diversity and exhibits disorganized layouts.

existing methods. For example, the rendering F-measure scores of TextDiffuser and GlyphControl are boosted by 1.60% and 6.51%, respectively. It is also worthy mentioning that, utilizing identical layout configurations as facilitated by the same Layout Planner (LP) for fair comparison, TextDiffuser-2 exhibits significantly better performance in the most challenging metric, exactly matching OCR(Acc), compared with training-free methods (#14 and #15) guided by strong contour-level conditions.

Qualitative results. The text-to-image visualizations are demonstrated in Figure 5. We compare our method with some representative open-source works including SD-XL [36], StableCascade [35], TextDiffuser [3], and some closed-source works including Ideogram [19], DALLE-3 [8], Midjourney-v6 [32]. It is observed that while SD-XL and StableCascade show good performance in rendering short text like "hello world", they still face challenges in rendering longer text like "happy birthday to ABC". Besides, although closed-source works showcase superior image quality, they do not perform as well in rendering text, sometimes

Fig. 6: The layout planner for existing text rendering methods. By replacing the original Layout Transformer used in TextDiffuser, the layout aesthetics is improved. The layout planner can be incorporated into other methods that do not support text-to-image generation. We use two seeds to sample different layouts from the layout planner.

omitting keywords or introducing extraneous, unwarranted text. Compared to TextDiffuser, our method generates more aesthetically pleasing layouts, avoiding misalignment or discordant font sizes. We conduct comparisons with some methods that are neither open-source nor offer APIs, such as GlyphDraw [31] and Character-Aware Model [29] using the samples shown in their corresponding papers in Appendix H.

We also investigate the contributions of our proposed layout planner to existing text rendering methods, with visualizations shown in Figure 6. Initially, we can substitute the Layout Transformer of TextDiffuser with our layout planner, resulting in a significant improvement in the aesthetic quality of the text layouts. Subsequently, for methods that are incapable of direct text-to-image generation (*i.e.*, need users to manually specify text positions), our layout planner can be utilized to autonomously determine text locations from a given prompt, which can then be integrated into their original frameworks for generation.

We further explore the generation diversity in Figure 7. When rendering "Winter", our method demonstrates greater diversity in terms of inclined angle and font style compared to other methods. In contrast, TextDiffuser, which uses character-level guidance and standard font to obtain intermediate masks, mainly renders rigid font styles. Furthermore, training-free Brush Your Text heavily depends on the contour of given templates and exhibit limited style diversity.

Text inpainting. Similar to TextDiffuser, the architecture of TextDiffuser-2 adapts well for training on text inpainting tasks. We only need to modify the channel of the input convolution kernel in the U-Net. Specifically, we augment the original 4-dimensional latent feature with another 5 additional dimensions, including 4 dimensions of non-inpaint area features and 1 dimension for the mask. Moreover, only the text position and content from the inpaint area are required as conditions for the diffusion model. In Figure 9, we qualitatively compare TextDiffuser-2 with text inpainting methods DiffSTE [20] and TextDiffuser [3]. It is worth noting that DiffSTE can not tackle multiple texts simultaneously so an iterative process is needed. We notice that the inpainting results are not in harmony with the surrounding text, and the image quality suffers from degradation. Besides,

Fig. 7: Visualization of diversity in generating three images under the same prompt: "A logo of Winter in artistic font, made by snowflake". TextDiffuser-2 can generate artistic fonts with diverse character positioning. See more samples in Appendix J.

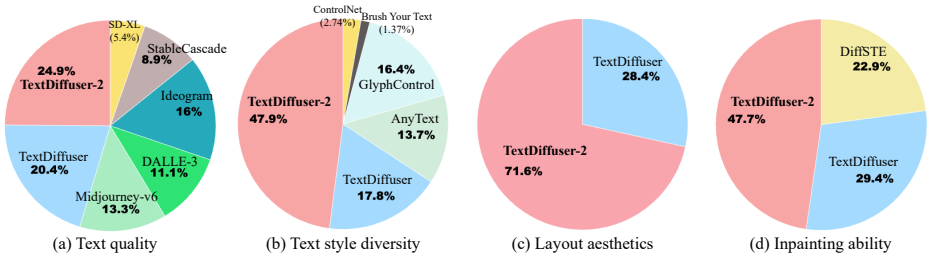

Fig. 8: User studies. TextDiffuser-2 achieves the best across four metrics.

TextDiffuser requires a text mask as a condition to specify the position of each character, which can be cumbersome in practical applications. Additionally, the text mask may limit the style of the generated results. For example, when rendering the word "Curve", the generated result cannot produce a visually curved effect due to the constraints of the character-level segmentation mask. In contrast, the inpainting process of TextDiffuser-2 is more flexible, thus resulting in a better user experience. For quantitative results, we sample 10,000 cases from the test set of MARIO-10M to evaluate the text inpainting task compared with DiffSTE. Note that since TextDiffuser requires an additional mask to specify the position of each character as input, we were unable to conduct such large-scale quantitative experiments. We achieve 76.42% OCR accuracy, better compared with DiffSTE (72.25%). More details about the training and evaluation process are shown in Appendix I.

User studies. As shown in Figure 8, we design questions covering four aspects: text quality, text style diversity, layout aesthetics, and text inpainting ability. Each aspect contains six questions. We involved a total of 21 human participants in our study. Based on the results, TextDiffuser-2 has achieved the best performance in all metrics in studies. We also use GPT-4V to conduct this user study and results reveal that GPT-4V generally favors the results of TextDiffuser-2. Details about the user study and GPT-4V evaluation are shown in Appendix K.

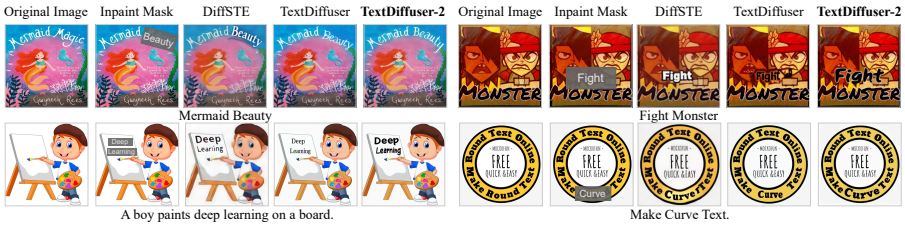

Fig. 9: Visualizations of the text inpainting task compared with DiffSTE and TextDiffuser. TextDiffuser-2 can generate more coherent text.

4.3 Discussions

Natural image generation without text. By omitting the text position and content guidance, TextDiffuser-2 can generate images without text. We randomly select 10,000 prompts from the Microsoft COCO dataset [28] for generation and compare the results with those generated by SD 1.5 [40]. The visualization and quantitative results are Appendix L. Although fine-tuned on domain-specific data, it still maintains its generative capabilities in the original domain.

Compatibility of more control signals. It is also feasible if users already specify the placement of text. The second stage can work independently by directly starting from user-provided layouts. Furthermore, TextDiffuser-2 is also compatible with ControlNet. We present samples in the Appendix M demonstrating that we can employ an off-the-shelf ControlNet to offer fine-grained guidance, as well as additionally train a ControlNet to represent the fine-grained text regions.

Generation based on overlapping layouts. Occasionally, we notice that there exist overlapping boxes during the layout prediction stage. We present TextDiffuser-2, as well as the results generated by GlyphControl and TextDiffuser using overlapping layouts in Appendix N. Experimental results indicate that TextDiffuser-2 demonstrates greater robustness towards overlapping boxes.

5 Conclusion

In this paper, we introduce TextDiffuser-2, aiming to unleash the power of language models for the text rendering task. Specifically, we attempt to tame two language models, one for layout planning and the other for layout encoding. Experimental results validate that TextDiffuser-2 is capable of generating more diverse images while maintaining the accuracy of the generated text. Meanwhile, the proposed components in TextDiffuser-2 can support existing methods with better accuracy, automation, and layout aesthetics. For the *limitation*, TextDiffuser-2 currently lacks the capability to layer the generated text, a feature that is commonly utilized in practical graphic design applications.

Acknowledgements

This research was supported by the Research Grant Council of the Hong Kong Special Administrative Region under grant number 16203122.

References

1. 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)
2. Chen, C., Qin, R., Luo, F., Mi, X., Li, P., Sun, M., Liu, Y.: Position-enhanced visual instruction tuning for multimodal large language models. arXiv preprint arXiv:2308.13437 (2023)
3. Chen, J., Huang, Y., Lv, T., Cui, L., Chen, Q., Wei, F.: Textdiffuser: Diffusion models as text painters. In: NeurIPS (2023)
4. Chen, J., Yu, J., Ge, C., Yao, L., Xie, E., Wu, Y., Wang, Z., Kwok, J., Luo, P., Lu, H., et al.: Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. arXiv preprint arXiv:2310.00426 (2023)
5. Chen, T., Saxena, S., Li, L., Fleet, D.J., Hinton, G.: Pix2seq: A language modeling framework for object detection. In: ICLR (2021)
6. Chen, T., Saxena, S., Li, L., Lin, T.Y., Fleet, D.J., Hinton, G.E.: A unified sequence interface for vision tasks. In: NeurIPS (2022)
7. Chiang, W.L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J.E., Stoica, I., Xing, E.P.: Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality (2023), https://lmsys.org/blog/2023-03-30-vicuna/
8. DALLE-3: Link: https://openai.com/dall-e-3 (2023), https://openai.com/dall-e-3
9. Daras, G., Dimakis, A.G.: Discovering the hidden vocabulary of dalle-2. arXiv preprint arXiv:2206.00169 (2022)
10. Das, A., Roy, P., Bhattacharya, S., Ghosh, S., Pal, U., Blumenstein, M.: Fast: Font-agnostic scene text editing. arXiv preprint arXiv:2308.02905 (2023)
11. DeepFloyd: Github link: https://github.com/deep-floyd/if (2023), https://github.com/deep-floyd/IF
12. Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., Kumar, R.: Rico: A mobile app dataset for building data-driven design applications. In: UIST (2017)
13. Feng, W., Zhu, W., Fu, T.j., Jampani, V., Akula, A., He, X., Basu, S., Wang, X.E., Wang, W.Y.: Layoutgpt: Compositional visual planning and generation with large language models. In: NeurIPS (2023)
14. GPT-4: Link: https://openai.com/gpt-4 (2023), https://openai.com/gpt-4
15. Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., Guo, B.: Vector quantized diffusion model for text-to-image synthesis. In: CVPR (2022)
16. Gupta, K., Lazarow, J., Achille, A., Davis, L.S., Mahadevan, V., Shrivastava, A.: Layouttransformer: Layout generation and completion with self-attention. In: ICCV (2021)
17. He, L., Lu, Y., Corring, J., Florencio, D., Zhang, C.: Diffusion-based document layout generation. In: ICDAR (2023)

18. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)
19. ideogram: Link: https://ideogram.ai/ (2023), https://ideogram.ai/
20. Ji, J., Zhang, G., Wang, Z., Hou, B., Zhang, Z., Price, B., Chang, S.: Improving diffusion models for scene text editing with dual encoders. Transactions on Machine Learning Research (2024)
21. Jyothi, A.A., Durand, T., He, J., Sigal, L., Mori, G.: Layoutvae: Stochastic scene layout generation from a label set. In: ICCV (2019)
22. Krishnan, P., Kovvuri, R., Pang, G., Vassilev, B., Hassner, T.: Textstylebrush: Transfer of text aesthetics from a single example. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
23. Lee, J., Kim, Y., Kim, S., Yim, M., Shin, S., Lee, G., Park, S.: Rewritenet: Reliable scene text editing with implicit decomposition of text contents and styles. arXiv preprint arXiv:2107.11041 (2021)
24. Li, J., Yang, J., Hertzmann, A., Zhang, J., Xu, T.: Layoutgan: Generating graphic layouts with wireframe discriminators. In: ICLR (2019)
25. 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)
26. Lin, J., Guo, J., Sun, S., Yang, Z., Lou, J.G., Zhang, D.: Layoutprompter: Awaken the design ability of large language models. NeurIPS (2024)
27. Lin, J., Guo, J., Sun, S., Yang, Z.J., Lou, J.G., Zhang, D.: Layoutprompter: Awaken the design ability of large language models. In: NeurIPS (2023)
28. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
29. Liu, R., Garrette, D., Saharia, C., Chan, W., Roberts, A., Narang, S., Blok, I., Mical, R., Norouzi, M., Constant, N.: Character-aware models improve visual text rendering. In: ACL (2023)
30. Lv, T., Huang, Y., Chen, J., Cui, L., Ma, S., Chang, Y., Huang, S., Wang, W., Dong, L., Luo, W., et al.: Kosmos-2.5: A multimodal literate model. arXiv preprint arXiv:2309.11419 (2023)
31. Ma, J., Zhao, M., Chen, C., Wang, R., Niu, D., Lu, H., Lin, X.: Glyphdraw: Learning to draw chinese characters in image synthesis models coherently. arXiv preprint arXiv:2303.17870 (2023)
32. Midjourney-v6: (2023), https://www.midjourney-v6.com/
33. Patil, A.G., Ben-Eliezer, O., Perel, O., Averbuch-Elor, H.: Read: Recursive autoencoders for document layout generation. In: CVPRW (2020)
34. Peng, Z., Wang, W., Dong, L., Hao, Y., Huang, S., Ma, S., Wei, F.: Kosmos-2: Grounding multimodal large language models to the world. In: ICLR (2024)
35. Pernias, P., Rampas, D., Richter, M.L., Pal, C., Aubreville, M.: Würstchen: An efficient architecture for large-scale text-to-image diffusion models. In: ICLR (2024)
36. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: Sdxl: improving latent diffusion models for high-resolution image synthesis. arXiv preprint arXiv:2307.01952 (2023)
37. Qu, Y., Tan, Q., Xie, H., Xu, J., Wang, Y., Zhang, Y.: Exploring stroke-level modifications for scene text editing. In: AAAI (2023)
38. 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 (2021)
39. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research (2020)

40. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)
41. Roy, P., Bhattacharya, S., Ghosh, S., Pal, U.: Stefann: scene text editor using font adaptive neural network. In: CVPR (2020)
42. Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., Norouzi, M.: Palette: Image-to-image diffusion models. In: SIGGRAPH (2022)
43. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al.: Photorealistic text-to-image diffusion models with deep language understanding. In: NeurIPS (2022)
44. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: ACL (2016)
45. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: ICLR (2021)
46. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
47. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
48. Tuo, Y., Xiang, W., He, J.Y., Geng, Y., Xie, X.: Anytext: Multilingual visual text generation and editing. arXiv preprint arXiv:2311.03054 (2023)
49. Wu, L., Zhang, C., Liu, J., Han, J., Liu, J., Ding, E., Bai, X.: Editing text in the wild. In: ACM MM (2019)
50. Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., Raffel, C.: Byt5: Towards a token-free future with pre-trained byte-to-byte models. Transactions of the Association for Computational Linguistics (2022)
51. Yang, F., Su, T., Zhou, X., Di, D., Wang, Z., Li, S.: Self-supervised cross-language scene text editing. In: ACM MM (2023)
52. Yang, Y., Gui, D., Yuan, Y., Ding, H., Hu, H., Chen, K.: Glyphcontrol: Glyph conditional control for visual text generation. In: NeurIPS (2023)
53. You, H., Zhang, H., Gan, Z., Du, X., Zhang, B., Wang, Z., Cao, L., Chang, S.F., Yang, Y.: Ferret: Refer and ground anything anywhere at any granularity. arXiv preprint arXiv:2310.07704 (2023)
54. Yu, Y., Zeng, Z., Hua, H., Fu, J., Luo, J.: Promptfix: You prompt and we fix the photo. arXiv preprint arXiv:2405.16785 (2024)
55. Zhang, L., Chen, X., Wang, Y., Lu, Y., Qiao, Y.: Brush your text: Synthesize any scene text on images via diffusion model. arXiv preprint arXiv:2312.12232 (2023)
56. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: ICCV (2023)
57. Zhang, S., Sun, P., Chen, S., Xiao, M., Shao, W., Zhang, W., Chen, K., Luo, P.: Gpt4roi: Instruction tuning large language model on region-of-interest. arXiv preprint arXiv:2307.03601 (2023)
58. Zhao, S., Chen, D., Chen, Y.C., Bao, J., Hao, S., Yuan, L., Wong, K.Y.K.: Uni-controlnet: All-in-one control to text-to-image diffusion models. In: NeurIPS (2023)
59. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.P., Zhang, H., Gonzalez, J.E., Stoica, I.: Judging llm-as-a-judge with mt-bench and chatbot arena (2023)
60. Zhou, Q., Yu, C., Zhang, S., Wu, S., Wang, Z., Wang, F.: Regionblip: A unified multi-modal pre-training framework for holistic and regional comprehension. arXiv preprint arXiv:2308.02299 (2023)