

A. Dataset Details

A.1. Distribution

In this analysis, we present the distributions of ground truth word lengths within our LenCom-EVAL and MARIO-EVAL datasets. Figure 10 shows that LenCom-EVAL exhibits a higher proportion of lengthy keywords in its composition.

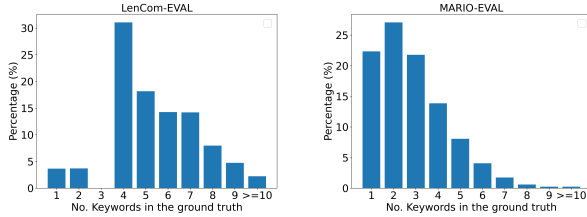


Figure 10. Comparison of LenCom-EVAL and MARIO-EVAL Distributions across Image Subsets by Number of Keywords: LenCom-EVAL exhibits a higher proportion of lengthy keywords in its composition.

A.2. Examples from Datasets

Table 4 displays an example from each subset within the LenCom-EVAL benchmark. In the Aug-MARIO-Hard instance, we employed the splitting augmentation method to divide the word “Amazon” into two parts: “Amaz” and “on”. Our RWC subset is generated using the template “A neon sign that says [Placeholder]”; in this particular case, the placeholder was substituted with “The little Luminous in the garden”.

B. Bounding Box Refinement Algorithm

We selected Simulated Annealing (SA) as our bounding box refinement method based on the empirical findings presented in [4]. The author organized spatio-overlap optimization techniques into six categories: exhaustive search, greedy algorithms, discrete gradient descent, gradient approximation using overlap vectors, mathematical programming, and stochastic search, with SA falling under the stochastic search category. Their experiments demonstrate that SA offers a favorable balance of speed and quality compared to other methods, along with the advantage of easy implementation.

C. Experiments

C.1. Baselines

TextDiffuser: Similar to original Text-Diffuser [3] we used the 2 stage network with maximum keywords length of 8. For the diffusion process we used the provided pretrained check point of runwayml/stable-diffusion-v1-5 and utilized Hugging Face Diffuser API⁸. For inference, we used 30 sampling steps and classifier free guidance with the scale of 7.5. We used a single NVIDIA RTX A6000 GPU with 48GB VRAM. Inference across all the datasets we experimented with was completed within 3 hours.

TextDiffuser-2: The TextDiffuser-2 [2] pipeline consists of two language models: one for converting the input prompt

⁸<https://huggingface.co/docs/diffusers>

into a language-format layout and the other for encoding this layout within the diffusion model to generate images. As described in the experimental setup, we sample with 50 steps during the inference phase of the pipeline. We use the pretrained vicuna-7b-v1.5 model checkpoint of JingyeChen22/textdiffuser2_layout_planner for layout planning. For the diffusion process, we use the pre-trained checkpoint of JingyeChen22/textdiffuser2-full-ft.

Diff-Text: For the Diff-Text [36] model, we use the official GitHub implementation along with the following pre-trained models: runwayml/stable-diffusion-v1-5 and llyasviel/sd-controlnet-canny. Furthermore, since the sketch renderer does not support line breaks, we truncate any text that goes beyond the 512 pixel limit.

AnyText: The AnyText [30] inference pipeline contains a text-control diffusion pipeline with two primary components: an auxiliary latent module and a text embedding module. We use the official implementation on GitHub and use the text-generation mode. Following the experimental setup described in AnyText, we split each input into a maximum of 5 text lines, with each line having no more than 20 characters. Any text exceeding 100 characters was truncated. This input was then sent to the Glyph Builder, and the produced templates were fed into the pipeline to generate the final images with the inscribed text.

C.2. Results for the Overlapped Area of Layout Generation

In addition to accuracy, we evaluate the overlap area and Intersection over Union (IoU) of bounding boxes generated by the layout generator. A smaller overlapped area or IoU indicates better layout generation by the model. From Table 5, we observe that SA-OcrPaint achieves significant reductions in both metrics. Specifically, our method improves the overlapped area and IoU by 91.3

C.3. Accuracy on 1/2/4/6/8/10 words subsets of SA-OcrPaint

We present the OCR accuracy at both the word and character levels across various subsets. From Figure 11, it’s evident that SA-OcrPaint outperforms the baseline TextDiffuser notably when there are two or more keywords, with the improvement becoming more significant as the number of words increases. Conversely, when there’s only one keyword, TextDiffuser performs better. In this scenario, where there’s no overlap of bounding boxes due to a single bounding box in the layout generation, our SA algorithm doesn’t exert any influence. However, during the second OCR in-painting step, multiple iterations actually decrease model performance. A potential remedy for this issue is to introduce a policy for accepting either the newly generated image or the previous one. This policy could be based on OCR word or character accuracy, wherein if the accuracy of the new image surpasses that of the previous one, it is accepted; otherwise, it is rejected.

Subset	Input Text Prompt	Ground Truth
MARIO-Hard	‘Amazon Cloud Player Amazon Cloud Player’ Music	Amazon Cloud Player Amazon Cloud Player
Aug-MARIO-Hard	‘Amaz on Clo ud Player Amazon Cloud Pla yer’ Music	Amaz on Clo ud Player Amazon Cloud Pla yer
WRC	A neon sign that says ‘The little Luminous in the garden’	The little Luminous in the garden

Table 4. Example from each subset in LenCom, the input prompt and the ground truth.

Model	Overlapped Area	IoU
TextDiffuser	1450	0.41
SA-OcrPaint	189	0.04

Table 5. Comparison of TextDiffuser and SA-OcrPaint in terms of Overlapped Area and IoU: SA-OcrPaint demonstrates a notable reduction in the extent of overlapped bounding boxes.

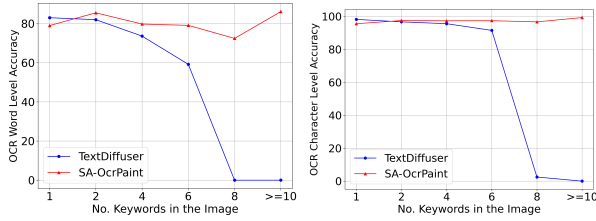


Figure 11. Compare Accuracy of TextDiffuser and SA-OcrPaint across image subsets as the number of keywords increases: SA-OcrPaint outperforms TextDiffuser when keywords number is equal or more than 2.

D. Discussion on Text Font

For our experiments, we utilized the default font, “Arial.ttf,” which aligns with the approach employed in prior work such as TextDiffuser [3]. The choice of this font ensures consistency with the base model and allows us to focus on evaluating the primary objectives of our framework. It is worth noting that the integration of different font families is feasible by updating the model with the desired font file. However, one limitation of the current setup is that all words within a particular image are generated using the same font family. Generating text with mixed fonts across individual words or characters introduces additional complexities, including alignment, aesthetic consistency, and legibility, which require deeper investigation. This aspect represents a promising direction for future research, as incorporating diverse fonts within the same image may enhance the realism and applicability of the generated text in more complex scenarios.