Two-Stage Dual Augmentation with CLIP for Improved Text-to-Sketch Synthesis

Zhenfei Zhang Ming-Ching Chang
University at Albany, State University of New York, NY 12222, USA
zzhang45@albany.edu mchang2@albany.edu

Abstract

We introduce an improved text-to-sketch synthesis method using two-stage dual augmentation based on the large-scale pre-trained CLIP and CLIPDraw models. In the first stage, the input text is fed to CLIPDraw to produce text augmentation adaptively. In the second stage, attention mechanisms and structural images with lower strokes are adopted for image augmentation enhancement. Parameters of the Bezier drawing curves are optimized using global and local loss terms. Our method produces visually plausible drawings with better stroke layouts and improved drawing details. There is no need for model re-training or parameter tuning. We further utilize CLIPScore, a reference-free metric, to evaluate the matching of the generated image against the input text description. Experimental results show that the proposed method produces improved drawing sketches.

1 Introduction

Text-to-sketch synthesis is a technology that creates plausible curve-based drawing images from natural language inputs; see Fig. 1. Understanding the relationship between the natural language descriptions and the scene structure in the generated image is the key, which is relatively simple for humans but non-trivial for AI algorithms.

In the past few years, a tremendous amount of effort has enabled computers to generate or synthesize visually-plausible images based solely on textual descriptions. This line of work has improved significantly with the advent of Deep Neural Networks (DNN). The Generative Adversarial Network (GAN) [7] have boosted several vision-language tasks including text-to-image generation [31], image-to-image translation [10], and image style transfer [16]. GAN consists of two main components, namely the generator and the discriminator. For text-to-image GAN, the generator creates images from the text descriptions, and the discriminator aims to discern generated images from real ones. Recently, *diffusion* based [9, 17] text-to-image models have achieved improved results over GAN-based methods.

Training an end-to-end text-to-image generation model from scratch requires a well-labeled dataset (such as COCO, which contains multiple captions for each image) and significant computation resources. A recent breakthrough deviating from this is to leverage Large Language Models (LLM), for example the large-scale vision-language pre-trained models such as CLIP [20]. Such foundation models are scalable to synthesize arbitrary images not limited by the domains of the training dataset. CLIPDraw [4] and StyleCLIPDraw [23] can produce images with wide contexts through an evaluation perspective based on CLIP, by minimizing the semantic gap (or distance) between the text input and the generated image. Instead of fine-tuning the weights of the generation model, CLIP-based methods iteratively optimize the strokes (from random initialization) to capture general visual features such as object shapes. Such stroke-based methods pay more attention to the larger features and less to the details; they alleviate the drawing deviation issues of GAN-based methods.

Figure 1: Example drawings generated by our method. Improved Bezier curve-based drawings are produced from the input texts without the need of training a large model or parameter tuning.

Despite the breakthrough of text-to-sketch methods based on CLIPDraw, they struggle with the following common challenges: (1) Only weak (global) image cropping augmentations are used, which ignores the local object contents; these methods cannot address the difficulty of *shallow solutions* in the CLIP optimization [23]. (2) Existing CLIPDraw based methods [4, 23] are sensitive to the selected hyper-parameters, in particular, the number of strokes in the drawing. (3) Only image augmentation is considered in the optimization; there is no consideration of text augmentation, despite the fact that there are often many ways to describe a scene using natural language. (4) There is no suitable metric to evaluate the quality or suitability of the gen-

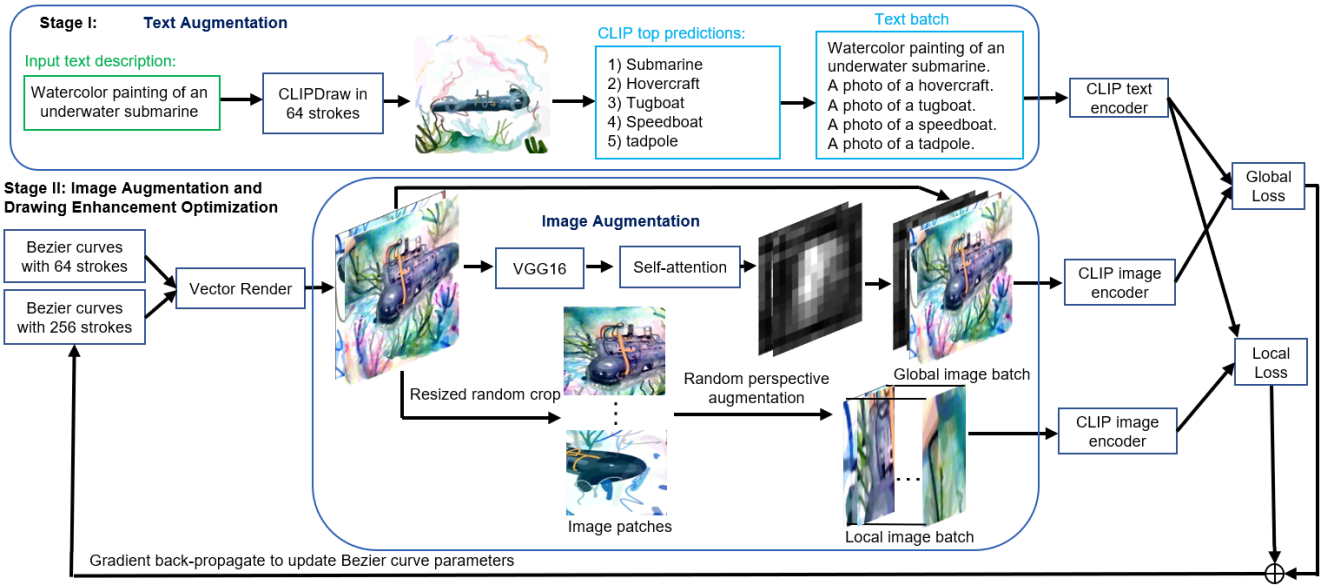

Figure 2: The overall architecture of the proposed method. In **Stage I**, we use CLIPDraw to generate text augmentation. In **Stage II**, both the augmented text batch and the augmented image batch are fed into the CLIP encoder to calculate the global and local loss terms, which are used to optimize the Bezier drawing curve parameters using back-propagation.

erated image that should match the given text description. Existing works only present visual results without quantitative evaluation.

In this paper, we proposed a novel text-to-sketch synthesis approach based on CLIP and CLIPDraw. Our method considers both global and local patch consistencies while performing data augmentation for drawing optimization. Fig. 2 provides an overview. Our method comes with the following advantages. (1) It does not require any elaborate design on manually setting a suitable number of strokes for each text input. (2) In addition to the standard image augmentations, we apply the attention mechanism [11] to enhance the image inputs and generation quality. (3) We introduce a text augmentation method in addition to image augmentation in the drawing optimization. The proposed text augmentation is fully adaptive and has no need for manual weight tuning. (4) We adopt CLIPScore [8] as a quantitative evaluation metric to show that our produced drawings better match the text descriptions over CLIPDraw. Figs. 1, 3, and 4 show visual examples generated from our method.

2 Related Work

Text-to-image synthesis has attracted wide research attentions [5, 1]. The first text-to-image approach [22] based on GAN in 2016 works by replacing the class label with a text embedding. The GAN discriminator distinguishes the generated and realistic images and tells apart the mismatching text embedding and a real image. StackGAN [33] and StackGAN++ [34] are introduced to improve the synthetic image resolution using multiple generators. Under a similar insight, FusedGAN [2] applies two generators that share a common latent space. PPAN [6] overcomes the need for multiple generators in using one pyramid generator to match three independent discriminators.

With the advent of the **attention mechanisms**, AttnGAN [31] can focus on relevant words and generate images more efficiently. SD-GAN [32] combines GAN with the Siamese Network and processes the branches with shared weights. CycleGAN [36] and MirrorGAN [19] use cycle-consistent adversarial networks, where the generator optimizes the synthesized features iteratively. DM-GAN [37] uses dynamic memory networks to achieve high-resolution generation. StyleGAN [13] and TextStyleGAN [27] provide image style transfer to effectively change the style of the original image. Regarding data augmentation, C4Synth [12] utilizes multiple descriptions to depict one image and enhance the semantic information. AVQA-GAN [18] uses question-answer pairs as text descriptions. The layout-to-image [28] comes with an insight that localization labels (bounding boxes) are declared to provide position information to the generator, such that the generator can produce images with more spatial supervision.

In addition to GAN-based methods, recent text-to-image generators are based on the large-scale Contrastive Language-Image Pre-training (CLIP) model [20] and Diffusion Probabilistic Model [9]. CLIP-GEN [29] introduces a self-supervised text-to-image generator using CLIP to reduce the cost of paired text-image data. In [21], a two-stage model consisting of CLIP and a decoder improves the diversity of visual expression. The DDPM [9] and GLIDE [17] are based on powerful diffusion models. All these techniques generate photo-realistic images other than a sketch

or drawing. They demand much well-labeled data and a huge amount of training computation. In contrast, CLIPDraw [4] and StyleCLIPDraw [23] are proposed based on the pre-trained CLIP model, such that plausible, realistic drawings can be generated. The drawing is based on parameterizing a set of Bezier curves (with random initialization) to characterize the colors, opacity, and coordinates of the drawing strokes. The Bezier curves are then controlled and optimized via carefully crafted loss terms to iteratively improve the similarity between the text-image pairs estimated using the CLIP cosine distance. The drawing image is generated via vector rendering [15] of the resulting Bezier curves.

Data augmentation has a major impact on Natural Language Processing (NLP) and Computer Vision (CV) tasks. The issue of insufficient training data can be alleviated by applying random transformations to the data. Basic geometric image augmentation [25] include flipping, rotation, shearing, cropping, *etc.* Typical text augmentation [30] includes text replacement, insertion, swapping, deletion, *etc.*

Image enhancement has been extensively used in multiple CV tasks to improve image quality and analysis [24]. Image enhancement can also mitigate the negative impact of low image quality. Typical image enhancement methods rely on numerical filters, such as Gaussian [3] filter. Inspired by [11], we apply the attention mechanism to enhance images during drawing generation.

3 Method

A main novelty in our text-to-sketch approach is the **two-stage dual augmentation**, that we perform both *text* and *image* augmentations based on CLIP and CLIPDraw in two stages to improve the drawing synthesis process. Fig. 2 shows the overall pipeline. Given a text description as input, we first use CLIPDraw to generate an initial drawing with 64 strokes with a text embedding, which is then fed to CLIP to obtain text augmentations. We keep the top 5 augmented texts and add a prefix "a photo of" to the objects provided in the augmented texts. The augmented texts and the original input text constitute our augmented text batch in the **Stage I** processing (§ 3.1). We next perform drawing image augmentation and again use CLIP to estimate the similarity of the resulting drawing and the augmented text batch. Such curve drawing, augmentation, and parameter optimization are performed in turns, which iteratively improve the drawing via back-propagation.

For the curve-based image drawing, the number of drawing strokes has a direct and significant impact on the drawing quality. Drawings with high strokes typically have rich, realistic details. In contrast, drawing with low strokes are typically salient in the overall structure. We use two settings, and refer to the drawing with **64** Bezier curves to be **low** strokes and **256** curves to be **high** strokes, respectively.

Algorithm 1 The Proposed Text-to-Sketch Algorithm

Input text T; # Bezier curve strokes: low N_l, high N_h; pre-trained CLIP, VGG, CLIPDraw.
Begin:
// Stage I: text aug. using the top 5 CLIPDraw text predictions
$I_{CLIPDraw}$ = CLIPDraw (T, N_{low})
$T^{aug}_{0,\dots,4}$ = CLIP.ImgTextFeatureRanking $(I_{CLIPDraw})$
$T^{enc}_{0,\dots,4}$ = CLIP.TextEnc $(T^{aug}_{0,\dots,4})$ // Encode aug. texts
// Bezier curve initialization:
$C_{0,\dots,N_l}, C_{0,\dots,N_h}$ = RandomBezierCurve()
$L_{overall} = 0$
// Stage II: image aug. and drawing optimization
while $e = 0$ to epoch E **do**
  while resolution $r \in$ { low/high strokes l, h} **do**
    $B_g = \emptyset$ // Global image batch set
    I_r = RenderCurvesToImage $(C_{0,\dots,N_r})$
    I^{aug}_r = ImageAugmentation (I_r)
    // Encode local image batch:
    B^{enc}_c = CLIP.ImgEnc(I^{aug}_r)
    $L_c = -\frac{1}{N}\sum_{i=1}^{N} L_{CLIP}(B^{enc}_c, T^{enc}_{0,\dots,4}))$ //Eq. 3
    M_r = VGG (I_r) // Self-attention map in Eq. 1
    M^c_r = GrayscaleToColor (M_r)
    B_g.append (I_r, M^c_r)
    // Encode global image batch:
    B^{enc}_g =CLIP.ImgEnc(B_g)
    $L_g = -L_{CLIP}(B^{enc}_g, T^{enc}_{0,\dots,4})$ // Eq. 2
    $L_{overall} += \lambda_g L_g + \lambda_c L_c$ // Eq. 4
  end while
  // Back-propagation to optimize drawing curves:
  $C^*_{0,\dots,N_l}, C^*_{0,\dots,N_h} \leftarrow$ Minimize $(L_{overall})$
end while
Output the high-stroke drawing I^*_h = Render $(C^*_{0,\dots,N_h})$

In the **Stage II** processing (§ 3.2), the Bezier curve parameters in both low and high strokes are iteratively improved.

During the iterative optimization process, the Bezier drawing parameters are updated based on a set of loss terms from the CLIP text-image cosine similarity (§ 3.3). The learning optimization will push the high-stroke images to show structural details from low-stroke samples, while low-stroke samples will obtain detailed supervision. As a result, the randomly initialized Bezier curves will be refined iteratively into the final drawing as in Fig. 4. Algorithm 1 shows the detailed steps of the proposed text-to-sketch algorithm.

3.1 (Stage I) Adaptive Text Augmentation

We aim to develop an automatic, effective text augmentation to improve the CLIP-based text-to-sketch synthesis capability. Our idea is partly motivated from that CLIPDraw drawing can be tuned manually by adding *undesired* text descriptions with negative weights [4], which can be viewed as a kind of *passive* text augmentation. In our case, we develop an *adaptive* text augmentation based on the relevant text embedding ordering provided by the CLIP pre-trained model. Our method does not require any user intervention and can introduce positive text prompts adaptively solely based on CLIP.

Starting with the input text description, CLIPDraw is

used to produce an initial drawing with low 64 strokes, such that less computation is required. Then, the drawing images are generated with top text predictions from CLIP. We view these top CLIP text predictions as extensions (or augmentation) of the input text, as the generated images are most likely related to these predictions ranked by CLIP. We add the prefix "**a photo of**" (which is the primary and frequently-used text prefix in CLIP) to each top-ranked predicted text and form the text augmentation.

3.2 (Stage II) Image Augmentation/Enhancement

For each rendering image generated from the Bezier drawing curves, we randomly crop the image into n patches. We then apply random perspective augmentation for each patch. Such mild augmentation is proven to be effective in CLIPDraw [4] with improved drawing quality. Next, we feed the augmented patches to the image encoder and text augmentation to the text encoder for local loss computation. For the global perspective, we first feed the global images as inputs to a VGG16 [26] backbone to calculate global spatial attention maps. It is shown in [11] that feeding a global spatial attention map to networks can enhance the quality of the generated images. Since most drawing contains a single main object thus the scene is not complex, a straightforward spatial self-attention [35] is sufficient. The attention map M_{att} calculation is based on the following spatial average pooling:

$$M_{att} = \frac{\sum_{i=0}^{c} F_i}{c}, \tag{1}$$

where $F \subseteq R^{H \times W \times C}$ is a feature map from the last convolutional layer of VGG16, H and W are the height and width of the feature map, C is channel number. By averaging pixels from different channels at the same location, the self-attention map M_{att} can express spatial information of the feature map very efficiently. We input the global attention maps as well as entire images to the image encoder and text augmentation to the text encoder for global loss computation, which are detailed in § 3.3.

3.3 Loss Functions

We consider two overall loss terms: (1) cosine similarity between the global images and texts in a *global* perspective and (2) cosine similarity between the cropped patches from the image augmentation and the corresponding texts in a *local* perspective. Denote the augmented input text batch as T and the rendered image batch as B. Let B_g and B_c represent the global and local image batches, respectively. Denote the total number of patches as N, and let B_c^i denote the i^{th} local patch. Let E denote the CLIP embedding function, and L_{CLIP} denote the original CLIP cosine similarity distance. The goal of the proposed method is to minimize the overall loss $L_{overall}$. Note that the CLIP similarity L_{CLIP} increases during optimization; thus, we multiply -1 to L_{CLIP}. The global loss L_g and local loss L_c are defined as:

$$L_g = -L_{CLIP}\left(E(B_g), E(T)\right), \tag{2}$$

$$L_c = -\frac{1}{N}\sum_{i=1}^{N} L_{CLIP}\left(E(B_c^i), E(T)\right). \tag{3}$$

The overall loss $L_{overall}$ is the combination of the local and global losses, which is minimized for the drawing optimization:

$$L_{overall} = \lambda_g L_g + \lambda_c L_c, \tag{4}$$

where the weighting parameters λ_g and λ_c default to 1.

4 Experimental Results

Our method takes an input text string and produces an output sketch in about 5 minutes for each run. No training or validation datasets are used, and no hyperparameter tuning is required. This makes our method very suitable for taking arbitrary descriptive texts as input and producing a satisfactory, meaningful drawing; see Fig. 3 for examples.

4.1 Implementation Details

VGG16 is used as the feature extractor for spatial attention calculation. We only use the feature maps from the last convolutional layer to generate the self-attention map and resize it to 224×224, which is the same size as the rendered image. For **image augmentation**, we use the cropping parameter $(0.2, 0.9)$ for `RandomResizedCrop` from `torchvision.transforms`, and the perspective augmentation of `RandomPerspective` with $(\texttt{fill} = 1, \texttt{p} = 1, \texttt{distortion\_scale} = 0.5)$. Both cropping and distortion produce 8 augmented instances for subsequent use. For **adaptive text augmentation** in Stage I (§ 3.1), we select the top 5 predictions from CLIP, excluding the keywords in the original text. We add the commonly-used prefix of "a photo of" from CLIP as additional keywords for augmentation. Each text augmentation batch contains 5 sentences.

For **Bezier drawing curves**, we use 64 and 256 strokes for low-stroke and high-stroke curves, respectively. There is no need to fine-tune the number of strokes for each text to obtain the desired image. We use CLIP to generate the similarity loss to optimize the parameters of the differentiable Vector Render [15] (the module on the left side of Fig. 2). The Bezier drawing curves are thus improved to match the text input. We optimize 3 parameters for each Bezier drawing curve: (1) the control point positions (each curve contains 3 to 5 control points), (2) the curve width, and (3) the curve color. We use Adam to optimize these 3 parameters using fixed learning rates of 1.0, 0.1 and 0.01, respectively.

The proposed model was trained using a Tesla 100 GPU for 1,000 optimization iterations. We take the drawing with 256 (high) strokes as the output, as it contains richer structural information with more details.

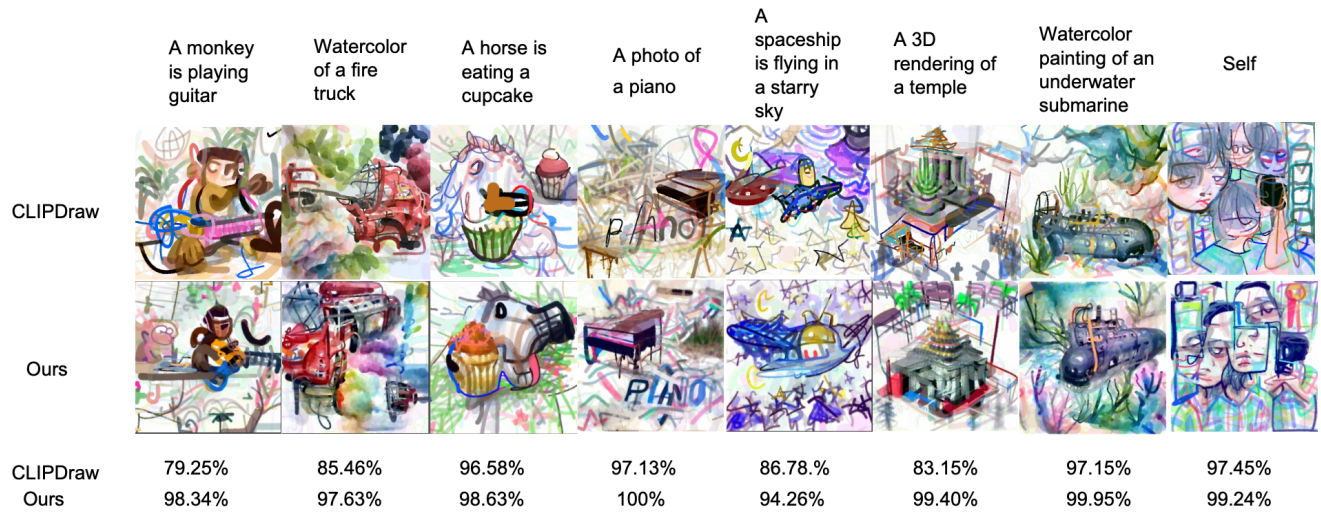

Figure 3: Comparison of the drawings from CLIPDraw and our method in terms of visual quality and the CLIPScore.

4.2 Evaluation Metric

Due to the nature of a generative model, it is hard to define or quantify how good a generated image is, or determine how well the sketch matches the input text. To the best of our knowledge, there is no relevant evaluation metric in the text-to-sketch synthesis literature due to the lack of ground truth for the synthesized images. On the other hand, the Amazon-Turk-like user study for evaluating the drawing image is labor extensive, prone to errors and inconsistencies.

To this end, we provide a quantitative evaluation using CLIPScore [4]. CLIPScore is shown to yield similar tendencies as the human subject study in [8, 14]; thus, it is suitable to evaluate the compatibility between the input text and the generated drawing image. Since there is no evaluation dataset, and we have no access to the training text data for CLIP, we grab all examples reported in the papers of CLIPDraw [23] and StyleCLIPDraw [23]. We thus obtain a total of 36 text inputs as Evaluation Samples (I). Meanwhile, we randomly generate 164 text inputs in the format CLIP recommends, which constitutes Samples (II). Finally, we combine the two sample sets and obtain 200 text descriptions as Sample (III). For a fair comparison, we set 256 as the default strokes in CLIPDraw. We then calculate the CLIPScores for the generated drawings from the three sample sets. Table 1 reports the average CLIPScore from these sample sets. Observe that our method consistently outperforms CLIPDraw in terms of CLIPScore (*i.e.*, text-image compatibility) in all experiments.

4.3 Qualitative Results

Fig. 3 provides visual comparisons of our generated drawings with CLIPDraw. Our drawings are more plausible, meaningful and with richer visual contexts. Our drawing contains richer structures over CLIPDraw; for example, see the drawing of "A 3D rendering of a temple" in Fig. 3. Fig. 4 illustrates the optimization process for drawing "Forest Temple as 3D Rendered in Unreal Engine".

Table 1: Comparison of the average CLIPScore of our method *vs.* CLIPDraw (the best scores are shown in bold).

Sample Sets	CLIPDraw	Ours
(I) 36 samples from	83.46	**92.35**
(II) 164 random samples	70.89	**81.54**
(III) The combination of (I) and (II)	66.06	**75.33**

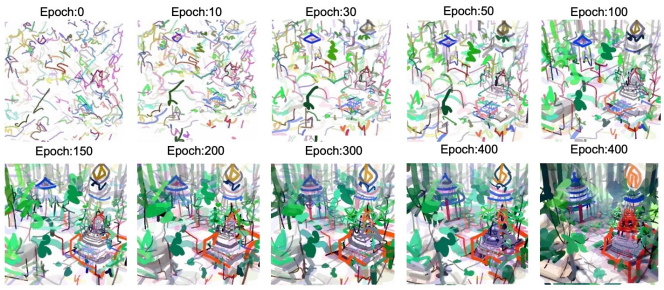

Figure 4: The drawing visualization in the iterations during our text-to-sketch optimization for "Forest Temple as 3D Rendered in Unreal Engine". Observe that the initial random curves are progressively improved to match the input text description.

5 Conclusion

We presented an improved text-to-sketch synthesis method using two-stage dual augmentation as an extension of CLIPDraw. The proposed method is very robust and does not require hyperparameter tuning. It takes advantage of text augmentation based on CLIP predictions. The image augmentation batch is further enhanced via self-attention. Experimental results show that our method can generate better-drawing images in terms of human aesthetics and CLIPScore text-to-image compatibility.

Future Work includes developing solutions to address the known limitations for drawing based on CLIP and CLIPDraw. The bias of CLIP causes a maj limitation in handling the drawing of multiple objects as well as complex drawing descriptions. The lack of a suitable evaluation metric or a way to ground-truthing the aesthetics and suitability of the drawing is another bottleneck. How best to measure or evaluate the drawing image quality without using ground truth is still an open question. If a powerful and efficient evaluation approach exists, then such a metric can be used for loss function design and optimization. A feasible approach might be to create a large enough text-to-sketch dataset such that existing metrics, including IS and FID can be used to evaluate the generated image quality.

References

[1] N. Bhise, Z. Zhang, and T. D. Bui. Improving text to image generation using mode-seeking function. *arXiv preprint arXiv:2008.08976*, 2020.

[2] N. Bodla, G. Hua, and R. Chellappa. Semi-supervised fusedgan for conditional image generation. In *ECCV*, 2018.

[3] G. Deng and L. Cahill. An adaptive gaussian filter for noise reduction and edge detection. In *NSS*, 1993.

[4] K. Frans, L. B. Soros, and O. Witkowski. CLIPDraw: Exploring text-to-drawing synthesis through language-image encoders. In *arXiv:2106.14843*, 2021.

[5] S. Frolov, T. Hinz, F. Raue, J. Hees, and A. Dengel. Adversarial text-to-image synthesis: A review. In *Neural Network Journal*, 2021.

[6] L. Gao, D. Chen, J. Song, X. Xu, D. Zhang, and H. T. Shen. Perceptual pyramid adversarial networks for text-to-image synthesis. In *AAAI*, 2019.

[7] I. J. Goodfellow, J. P. Abadie, M. Mirza, B. Xu, D. W. Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[8] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

[9] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[11] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J.Yang, P. Zhou, and Z. Wang. EnlightenGAN: Deep light enhancement without paired supervision. In *TIP*, 2019.

[12] K. J. Joseph, A. Pal, S. Rajanala, and V. N. Balasubramanian. C4Synth: Cross-caption cycle-consistent text-to-image synthesis. In *WACV*, 2018.

[13] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2018.

[14] G. Kwon and J. Ye. Clipstyler: Image style transfer with a single text condition. In *CVPR*, 2022.

[15] T. M. Li, M. Lukac, M. Gharbi, and J. R. Kelley. Differentiable vector graphics rasterization for editing and learning. In *ACM Tran. on Graphics*, 2020.

[16] H. Liu, P. N. Michelini, and D. Zhu. Artsy-GAN: A style transfer system with improved quality, diversity and performance. In *ICPR*, 2018.

[17] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In *arXiv:2112.10741*, 2022.

[18] T. Niu, F. Feng, L. Li, and X. Wang. Image synthesis from locally related texts. In *ICMR*, 2020.

[19] T. Qiao, J. Zhang, D. Xu, and D. Tao. MirrorGAN: Learning text-to-image generation by redescription. In *CVPR*, 2019.

[20] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *arXiv:2103.00020*, 2021.

[21] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with CLIP latents. In *arXiv:2204.06125*, 2022.

[22] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[23] P. Schaldenbrand, Z. Liu, and J. Oh. StyleCLIPDraw: Coupling content and style in text-to-drawing translation. In *NeurIPS workshop*, 2021.

[24] V. Sharma, A. Diba, D. Neven, M. S. Brown, L. V. Gool, and R. Stiefelhagen. Classification-driven dynamic image enhancement. In *CVPR*, June 2018.

[25] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *CVPR*, 2013.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large scale image recognition. In *ICLR*, 2015.

[27] D. Stap, M. Bleeker, S. Ibrahimi, and M. ter Hoeve. Conditional image generation and manipulation for user-specified content. In *CVPR*, 2020.

[28] W. Sun and T. Wu. Image synthesis from reconfigurable layout and style. In *ICCV*, 2019.

[29] Z. Wang, W. Liu, Q. He, X. Wu, and Z. Yi. CLIP-GEN: Language-free training of a text-to-image generator with CLIP. In *arXiv:2203.00386*, 2022.

[30] J. Wei and K. Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP*, 2019.

[31] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2017.

[32] G. Yin, B. Liu, L. Sheng, N. Yu, X. Wang, and J. Shao. Semantics disentangling for text-to-image generation. In *CVPR*, 2019.

[33] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2016.

[34] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. StackGAN++: Realistic image synthesis with stacked generative adversarial networks. In *PAMI*, 2017.

[35] Z. Zhang and T. D. Bui. Attention-based selection strategy for weakly supervised object localization. In *ICPR*, 2021.

[36] J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[37] M. Zhu, P. Pan, W. Chen, and Y. Yang. DM-GAN: Dynamic memory generative adversarial networks for text-to-image synthesis. In *CVPR*, 2019.