Towards Diverse and Consistent Typography Generation

Wataru Shimoda[1] Daichi Haraguchi[2] Seiichi Uchida[2] Kota Yamaguchi[1]
[1]CyberAgent, Japan [2]Kyushu University, Japan

Abstract

In this work, we consider the typography generation task that aims at producing diverse typographic styling for the given graphic document. We formulate typography generation as a fine-grained attribute generation for multiple text elements and build an autoregressive model to generate diverse typography that matches the input design context. We further propose a simple yet effective sampling approach that respects the consistency and distinction principle of typography so that generated examples share consistent typographic styling across text elements. Our empirical study shows that our model successfully generates diverse typographic designs while preserving a consistent typographic structure.

1. Introduction

In textual communication, typographers carefully express their intent in their typographic work, such as product packages, posters, banner ads, book covers, signboards, and presentation slides. Appropriately designed typography affects how people perceive the impression, legibility, and importance of the text content, yet choosing appropriate typography is surprisingly challenging [3]. Typographic design involves a complex interplay between the message content, background visuals, layout arrangement, and styling consistency across text elements.

In building a practical automatic typography system, we have to take into account the following requirements.

Context awareness: A system should reflect the context of the creative work; e.g., styling should emphasize the word "Sale" for a sale event poster or use serif-style fonts with careful letter spacing for luxury brands to express their authority. Also, typography should match the background visuals; e.g., a bright font color for a dark background.

Fine-grained representation: A system can handle fine-grained typographic attributes beyond font family and color, such as horizontal text alignment, line spacing, letter spacing, or angle, that are important to convey a delicate nuance within the graphic design.

Consistency and distinction: A system should apply consistent style across multiple texts that share the same semantics [40]; e.g., menu items should have uniform styling. On the other hand, typography should have distinct styling to emphasize the content semantics; e.g., a title should be highlighted by a different font family and size.

Diversity: A system should be able to suggest diverse design candidates to the users because there is usually no single optimal typographic design in a real-world creative workflow.

In this paper, we formulate the typography generation task as fine-grained typographic attribute generation and build an autoregressive model that can generate diverse yet consistent typographic styling for the given graphic document. Given a canvas, texts, and their rough positions (Fig. 1a), our model generates fine-grained attributes such as font, color, or letter spacing for each text element.

Our model relies on the attention mechanism of the Transformer architecture to capture the consistency-relationship among texts as well as the relationship between texts and the input context. For generating diverse typography, we propose a simple yet effective sampling approach to enforce consistent styling among text elements, which we refer to as *structure-preserved sampling*. Our sampling approach predicts which text elements share the uniform styling in the first step (Fig. 1b) and samples diverse attributes constrained by the predicted relationships in the second step (Fig. 1c). We also propose metrics to evaluate the quality of typography generation, where we define the typography structure in the form of pairwise consistency relationships among text elements.

We show in experiments that our autoregressive models outperform baseline approaches and successfully generate diverse typography that respects context and consistency. Our user study also confirms that our approach is qualitatively preferred over the baseline. Our attribute-based formulation is readily applicable in a real-world creative workflow, as designers usually work on graphic documents with vector-graphic authoring tools like Adobe Illustrator.

We summarize our main contributions in the following.

- We formulate the typography generation task that aims at jointly generating diverse fine-grained typographic attributes.

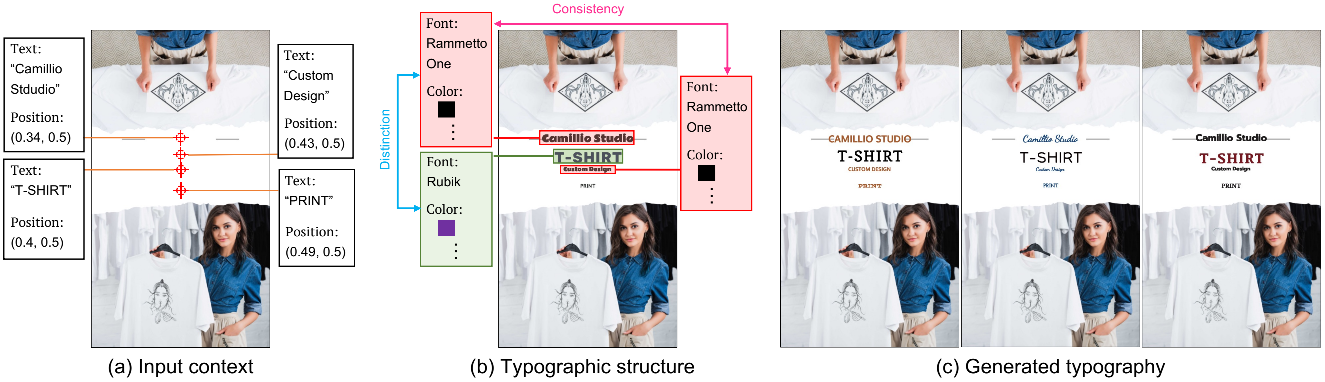

Figure 1. We formulate the fine-grained typography generation task considering the structure of multiple texts. a) An example of an input context: background image, texts, and their corresponding center positions. b) Typographic structure predicted by our model via top-1 sampling. c) Generated typography by our structure-preserved sampling.

- We present an autoregressive approach to generate typographic attributes, where we develop the structure-preserved sampling to generate diverse yet consistent typographic designs.
- We propose metrics to evaluate the quality of typography generation that is aware of the consistency among text elements.
- We empirically show the effectiveness of our approach both quantitatively and qualitatively.

2. Related work

2.1. Attribute-based typography generation

While attribute-based representation is commonly observed in commercial design authoring tools, we do not find much literature on attribute-based typography generation. MFC [49] is a notable exception that predicts the font, color, and font size of a single text box from the global image, local image, and auxiliary tag information. AutoPoster [23] recently proposes a poster generation approach that also considers font, color, and font size within the model. While the previous work considers typographic attributes, we consider far more fine-grained attributes including text angle, alignment, letter spacing, and line spacing, and explicitly consider consistency relationships among multiple text elements. Other notable works include the study of Jiang *et al*. [14] on combinatorial preference in font selection for subjects and subtitles in PDF data and Shimoda *et al*. [36] proposing a de-rendering approach to parse rendering parameters from texts in raster images.

2.2. Raster typography generation

Raster typography generators directly render stylized texts in pixels. There are two types of formulations: text style transfer and conditional stylized text generator. Text style transfer aims at generating stylized text images for the specified styles. Awesome Typography is a style transfer method by a patch matching algorithm [44]. Recent literature reports several GAN-based models [2, 6, 28, 38, 45–47]. Wang *et al*. propose a layout-specified text style transfer method [39]. Raster text editing is another branch of the text style transfer task, where the goal is to apply a reference style to the manually edited image [35, 41, 43].

There are several neural network-based glyph renderers without reference images. We refer to these approaches as conditional stylized text generators. Miyazono *et al*. and Gao *et al*. [7] propose a generative model that directly produces stylized texts in the raster format from background images, layouts, and text contents [29]. Recent text-to-image model [33, 34] can draw stylized texts via prompts, but these models tend to corrupt glyphs in the raster format [24]. Some recent works propose fine-tuned text-to-image models [13, 27, 48] that address glyph corruption.

While there are quite a few works on raster generation, attribute-based generation has a clear practical advantage in that the generation result is 1) free from raster artifacts and 2) easily applicable in real-world authoring tools.

2.3. Graphic design generation

Our typography generation task can be regarded as one sub-topic within the broader study of attribute-based graphic design or layout generation. Early work on layout generation utilizes templates [5, 11] or heuristic rules [30]. Recent literature relies on neural networks for generation. LayoutVAE [16] generates scene layouts from label sets using autoregressive VAE. LayoutGAN [22] adopts a GAN-based layout generator via a differentiable wireframe model. VTN [1], LayoutTransformer [9], and CanvasVAE [42] report Transformer-based VAE for graphic de-

Table 1. Context and typographic attributes. Context attributes consist of canvas input and element input.

Type	Name	Modality	Size
Canvas input $\mathbf{x}_{\text{canvas}}$	Background	Image	$256 \times 256 \times 3$
	Aspect ratio	Categorical	40
	Number of text	Categorical	50
Element input $\mathbf{x}_{\text{t}}$	Text	Text	variable
	Left	Categorical	64
	Top	Categorical	64
	Line count	Categorical	50
	Char count	Categorical	50
	Background	Image	$256 \times 256 \times 3$
Typographic attributes (output) $\mathbf{y}_{\text{t}}$	Font	Categorical	261
	Color	Categorical	64
	Alignment	Categorical	3
	Capitalization	Categorical	2
	Font size	Categorical	16
	Angle	Categorical	16
	Letter spacing	Categorical	16
	Line spacing	Categorical	16

signs. LayoutDM [12] adopts a discrete diffusion model to layout generation. Towards finer control on the generation quality, several literature [4,8,12,15,17–20,31,50,51] tackles to generate layouts with constraining and conditional information. While most recent attempts seem to be interested in the layout-level generation, our focus is the unique and explicit modeling of text styling in the typographic design.

3. Approach

Our goal is to generate typography with consistency and diversity from context attributes such as *background image*, *texts*, and their corresponding *center positions*. To this end, our model first predicts typographic structure (Fig. 1b) and then generates typography through a structure-preserved sampling of typographic attributes such as *font* and *color* (Fig. 1c).

3.1. Problem formulation

We define the context attributes by $X \equiv (\mathbf{x}_{\text{canvas}}, \mathbf{x}_1, \dots, \mathbf{x}_T)$, where $\mathbf{x}_{\text{canvas}} \equiv (x_{\text{background}}, x_{\text{aspect}}, \dots)$ denotes a tuple of canvas input and $\mathbf{x}_t \equiv (x^t_{\text{text}}, x^t_{\text{top}}, x^t_{\text{left}}, \dots)$ denotes the t-th element input. We assume there are T text elements in the document. We consider target typographic attributes $Y \equiv (\mathbf{y}_1, \dots, \mathbf{y}_T)$, where $\mathbf{y}_t \equiv (y^t_{\text{font}}, y^t_{\text{color}}, \dots)$ is typographic attributes of the t-th text element. Our goal is to generate typographic attributes Y by a conditional generation model p_θ parametrized by θ:

$$\hat{Y} \sim p_\theta(Y|X). \tag{1}$$

3.2. Typographic attributes

Our context and typographic attributes contain multiple modalities, which we preprocess into feature representation beforehand. We summarize the feature representation of all attributes in Table 1. Our context attributes consist of the canvas input and the element input. We extract a background image for both the global canvas and the region of each text element, resize the image to the fixed resolution with the RGB format, and finally apply a pre-trained CLIP encoder [32] to extract features. We preprocess text content using a pre-trained CLIP encoder [32]. We discretize continuous attributes, such as an aspect ratio or a position, based on the k-means clustering, where we empirically set the appropriate number of clusters.

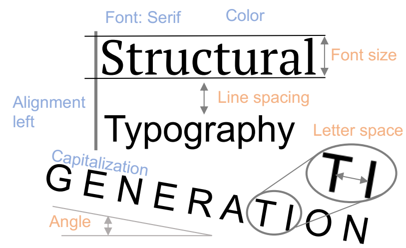

Figure 2. An illustration of typographic attributes. We handle semantic quantities including *font*, *color*, *alignment*, and *capitalization* and geometric quantities including *font size*, *angle*, *letter spacing*, and *line spacing*.

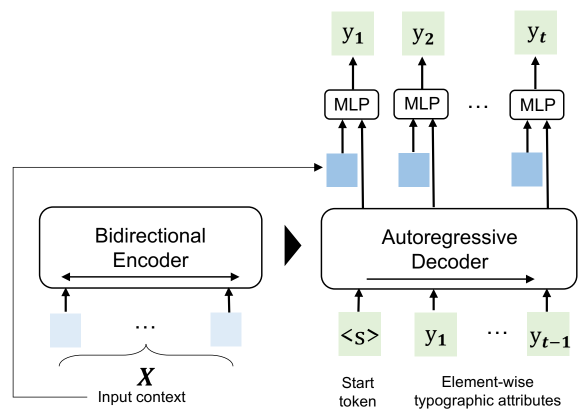

Figure 3. Model architecture.

In this work, we consider the following typographic attributes as outputs: *font*, *color*, *font size*, *alignment*, *capitalization*, *angle*, *letter spacing*, and *line spacing* for each text element. Our typographic attributes have semantic and geometric quantities. We show the illustration of the typographic attributes in Fig. 2. We also discretize typographic attributes based on the k-means clustering.

3.3. Typography generation

We build an encoder-decoder architecture based on Transformer [37] to effectively capture the interaction among the inputs and the target attributes within the attention mechanism. Fig. 3 illustrates the overall architec-

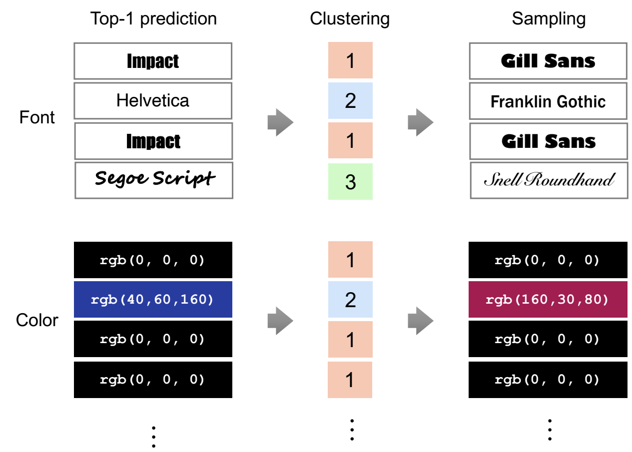

Figure 4. Our structure-preserved sampling first clusters elements by top-1 prediction, then draws samples per cluster so that the result maintains the most likely typographic structure while being capable of generating various design.

ture. Our architecture combines BART-style Transformer blocks [21] with skip connections between the input and output of each element. We project input features into fixed-size embeddings and feed into the Transformer encoder blocks.

We adopt an autoregressive decoder to model the joint distribution of typographic attributes:

$$p_\theta(Y|X) = \prod_t^T p_\theta(\mathbf{y}_t|\mathbf{y}_{t-1}, \dots, \mathbf{y}_1, X), \tag{2}$$

and we apply element-wise autoregressive sampling to generate attribute k at the t-th element:

$$\hat{y}_k^t \sim p_\theta(y_k^t|\mathbf{y}_{t-1}, \dots, \mathbf{y}_1, X). \tag{3}$$

Here, we apply top-p sampling [10] to draw attributes. Top-p sampling has a hyper-parameter $p_k \in [0, 1]$ that controls the diversity for each attribute k. In our experiments, we fix $p_k = 0.1$ for geometric attributes (font size, angle, letter spacing, and line spacing) to avoid visually disturbing generation, and vary p_k for other attributes depending on the experimental setup.

To train the model, we minimize the following objective:

$$\sum_t \sum_k \mathcal{L}_{\text{entropy}}^k(y_k^t, \tilde{y}_k^t) + \lambda_{\text{reg}}|\theta|^2, \tag{4}$$

where $\mathcal{L}_{\text{entropy}}^k$ is the standard cross entropy for the attribute k, $\tilde{y}_k^t$ is the ground truth, and λ_{reg} is the L2 weight decay.

3.4. Structure-preserved sampling

While autoregressive sampling can adjust sampling hyper-parameter for each attribute, we find the plain autoregressive approach sometimes corrupts the consistency and distinction among element styling (Sec. 1), especially when we increase the parameter p_k of top-p sampling. Here, we propose the *structure-preserved sampling*, which is a simple two-step inference approach that effectively controls the diversity while preserving the typography structure. The general steps are the following.

1. Infer initial prediction $\check{Y}$ via top-1 sampling:

$$\check{y}_k^t = \operatorname{argmax} p_\theta(y_k^t|\mathbf{y}_{t-1}, \dots, \mathbf{y}_1, X). \tag{5}$$

2. For each attribute k, cluster text elements $\mathcal{T} \equiv \{1, \dots, T\}$ by label linkage $\check{y}_k^t = \check{y}_k^{t'}$ for any pair $t \neq t'$.

3. Autoregressively sample $\hat{y}_k^t$ again but assign the same label if any element in the same cluster is already assigned a label.

In both inference steps, we keep the same raster scan order of elements (left-to-right, top-to-bottom). Basically, we autoregressively sample over clusters instead of all the elements in the second sampling step. Fig. 4 illustrates the above steps. The intuition is that top-1 sampling gives the best typographic structure, and the second sampling generates diverse examples while forcing the consistent structure from the initial inference. Our approach is heuristic but generates visually plausible typography without significant overhead.

It is possible to replace the initial top-1 sampling with other sampling approaches if we need to generate a typographic design with a different structure. In this work, we assume a typical typographic design does not require a diverse structure in the application scenario; e.g., design suggestion in an authoring tool.

We split the clustering step for each attribute, but it is also possible to consider joint clustering across attributes. The challenge here is that a different attribute has a different perception in the final visualization. It is not straightforward to define a unified cluster affinity across typographic attributes; e.g., humans perceive the difference in a font more than the different alignments. In our dataset, we often observe texts that share the same font but with different sizes. We leave the optimal design of typographic clusters for our future work.

4. Evaluation Metrics

There is no standardized evaluation metric for typography generation. We adopt several metrics to evaluate typography generation performance.

4.1. Attribute metrics

In our setting, we handle several typographic attributes, but the format of each attribute is not the same. Here, we introduce evaluation metrics for measuring the fidelity of attribute prediction.

Accuracy: We evaluate categorical attributes (*font*, *align*, *capitalization*) by the standard accuracy metric between the prediction and the ground truth.

Mean absolute error: We evaluate the geometric attributes by the absolute difference in their respective unit. We measure *font size* in points, *angle* in degree, *letter spacing* in points, and *line spacing* in a relative scale centered at 1.0.

Color difference: We employ CIEDE2000 color difference [26] to measure the similarity between colors, which is known to well reflect the human perception of color difference.

4.2. Structure score

The structure score examines whether the use of the same attribute pairs matches the ground truth. That is, if a pair of texts share the same attribute, we assign `1`, and if the pair differs, we assign `0`, then measure the accuracy between the prediction and the truth. Formally, for attribute k, we consider the set of binary labels over any pair of text elements:

$$S_k(Y) \equiv \{\delta(y_k^i, y_k^j) | i \in \mathcal{T}, j \in \mathcal{T}, i \neq j\}, \tag{6}$$

where $\delta(y_k^i, y_k^j)$ is an indicator function that measures the condition $y_k^i = y_k^j$. The structure score is the accuracy of prediction $S_k(\hat{Y})$ against the ground truth $S_k(Y)$ for each document.

4.3. Diversity score

We evaluate how diverse the generated typography attributes are. Assuming we generate N samples, we count the average number of unique labels over elements in the generated samples:

$$\frac{1}{T}\sum_{t}^{T}\frac{N^t_{\text{uniq},k}}{N}, \tag{7}$$

where $N^t_{\text{uniq},k}$ is the unique count of attribute k at the t-th element.

5. Experiments

We evaluate typography generation performance as well as top-1 prediction performance for fair comparison.

5.1. Dataset

We evaluate the generation task using the Crello dataset [42], which includes various design templates in vector format. Since the original dataset does not contain all of the necessary typographic information for visualization, we collect additional resources like `ttf` files. We parsed and compiled the typographic details of each template, and finally obtained 23,475 templates that contain text elements in the vector format. We split the Crello dataset to *train*:*test*:*val* with an 8:1:1 ratio (i.e., 18,780, 2,347, 2,347).

5.2. Implementation details

We set the dimension of feature embeddings to 256. We set the feed-forward dimension to 512 and the number of the head to 8 in the Transformer blocks. We stack 8 Transformer blocks in our model. We use AdamW [25] optimizer with a 0.0002 learning rate and 30 epochs to train our model.

5.3. Prediction evaluation

Here, we evaluate the performance of the top-1 prediction for a fair comparison with the previous work. We compare the following baselines.

Mode always predicts the most frequent category, which shows the bias of each attribute in the dataset.

MFC [49] is a fill-in-the-single-blank model tailored for typography. This model predicts three attributes: *font*, *font size*, and *color*. MFC learns to predict embedding for font representation by minimizing L2 loss and adversarial loss, a scalar value for font size by minimizing the L1 loss, and a discretized token for color. The embedding for font representation is obtained by a simple autoencoder. Since this model cannot produce multiple outputs, we repeatedly apply the model to generate multiple outputs in an autoregressive manner. We do not consider external contexts (HTML tags and design tags) used in [49] since the Crello dataset does not contain such resources.

CanvasVAE* [42] is a Transformer-based variational autoencoder model for structured elements, including layout and canvas information [42]. Since CanvasVAE is an unconditional model, we adapt CanvasVAE to accept input contexts and predict typographic attributes. For the prediction task, we fix the bottleneck latent of the VAE to the mean vector.

Ours is the initial autoregressive prediction of our model (Sec 3).

Table 2 and Table 3 summarize the quantitative prediction performance. Our model achieves the best scores in all structure scores, though not always the best in attribute metrics. Interestingly, while our model shows moderate improvement over baselines in attribute metrics like *font size*, our model shows significant improvement in terms of the structure score. We observe that our model outperforms MFC even if MFC designs a dedicated loss for each attribute. Our model also outperforms CanvasVAE, perhaps because CanvasVAE has a limited model capacity due to the global latent that is regularized to follow the normal distribution. In distinction, our autoregressive models have sufficient capacity to model rich conditions across attributes and

Table 2. Attribute metrics. Acc: accuracy, MAE: mean absolute error, and Diff: CIEDE2000 color difference.

Method	Font	Color	Align	Capitalize	Size	Angle	Letter space	Line height
	Acc (%) ↑	Diff (-) ↓	Acc (%) ↑	Acc (%) ↑	MAE (pt) ↓	MAE (°) ↓	MAE (pt) ↓	MAE (-) ↓
Mode	16.6	53.2	91.9	54.1	45.1	0.30	2.31	0.102
MFC	10.4 ±6.87	54.9±5.06	-	-	28.0±8.82	-	-	-
CanvasVAE*	27.7±9.56	53.3±1.41	92.3±0.43	66.0±9.07	32.5±3.97	0.30±0.01	2.23±0.09	0.095±0.006
Ours	40.9±0.76	53.7±1.96	93.8±0.74	75.3±0.67	20.9±0.66	0.26±0.02	2.16±0.16	0.065±0.003
Ours w/o layout	41.4	57.1	93.9	69.0	23.4	0.37	2.11	0.067

Table 3. Structure scores (%).

Method	Font	Color	Align	Capitalize	Font size	Angle	Letter space	Line height
Mode	61.9	58.0	66.8	85.7	22.4	83.5	57.4	56.6
MFC	59.8±4.22	58.9±3.40	-	-	63.4±6.19	-	-	-
CanvasVAE*	62.0±0.56	59.6±1.84	66.5±0.59	85.7±0.22	43.7±18.40	83.9±0.42	58.7±1.48	60.1±3.80
Ours	68.6±0.44	66.9±0.65	68.1±0.58	86.3±0.55	71.3±0.55	86.0±0.37	63.8±0.77	78.9±1.06
Ours w/o layout	68.3	66.1	66.7	86.6	71.5	83.2	64.1	79.1

elements.

5.4. Generation evaluation

We generate 10 samples for each test input for evaluation. We compare the following baselines.

CanvasVAE* is the same model we evaluate in Sec 5.3. We control the generation diversity by scaling the coefficient of standard deviation in the latent space.

Ours is our model with a plain top-p sampling and without our structure-preserved sampling. We control the generation diversity by the hyper-parameter $p_k \in [0, 1]$ of top-p sampling except for geometric attributes.

Ours+SS applies the structure-preserved sampling to the above model.

Quantitative results Fig. 5 plots the attribute metrics and the structure score of font and color as we increase the diversity hyper-parameter. We observe that our models show a good quality-diversity trade-off compared to CanvasVAE. While a plain top-p approach clearly degrades the structure score as we increase the diversity, our structure-preserved sampling keeps the constant score in the highly diverse regime. Note that our structure-preserved sampling can slightly drop the attribute metrics compared to the plain autoregressive sampling due to the cases when the initial structure prediction fails.

Qualitative results Fig. 6 shows qualitative results. We set the diversity hyper-parameter of CanvasVAE to $std = 100$, Ours to $p = 0.9999$, and Ours+SS to $p_k = 0.99999$, which yields similar diversity scores. CanvasVAE tends to ignore the input context. We suspect CanvasVAE suffers from learning a good single latent space for a complex task like typography generation. Besides, CanvasVAE cannot independently control the diversity of different attributes, which causes poor overall appearance. Our models generate sufficiently diverse typography for individual attributes in each element, and with the structure-preserved sampling, the results hold consistent styling across elements. We show more generation examples by our model in Fig. 7. The first row, the second row, and the third row show examples that have only a few elements but have sufficient contrast. The fourth and fifth rows show that our model consistently generates diverse yet plausible typography even when a document has many text elements.

Limitation We show some failure cases of our approach in Fig. 8. Our model does not explicitly handle the appearance of typography and sometimes generates unintentional spatial overlaps between texts (Fig. 8a), colors that are difficult to see (Fig. 8b), and overflow of a text element due to the unawareness of the final text width (Fig. 8c). Further, if our model fails to capture plausible structure, generated results corrupt (Fig. 8d).

5.5. User study

To verify that our evaluation metrics accurately reflect human perception, we conducted pilot user studies. We asked ten participants to choose which generated design groups they preferred in a pairwise comparison between the two methods. We compared the generation quality of our model with the CanvasVAE and our model without the structure-preserved sampling. Each user study comprises 100 questions, resulting in 1000 responses in total.

As the diversity hyper-parameter affects generation quality, we choose the hyper-parameter of each approach to be comparable. Specifically, we set the diversity hyper-parameter to have the diversity score within 49.8-51.5% for font and 33.3-35.2% for color in the CanvasVAE comparison, and the diversity score within 70.4-73.3% for the font

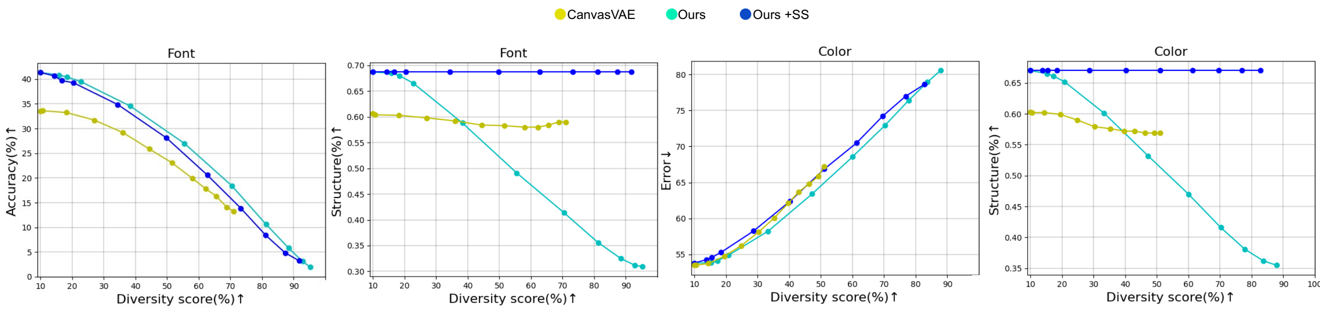

Figure 5. Generation performance in terms of attribute metrics vs. diversity score for font and color attributes. Our models outperform the CanvasVAE baseline by a large margin. Our structure-preserved sampling further keeps the constant structure score regardless of the sampling parameter p_k.

Figure 6. Qualitative comparison of typography generation. Our models generate sufficiently diverse typography with appropriate color tones to the background. With the structure-preserved sampling, our model further enforces consistent styling like fonts to multiple texts (Ours+SS).

and 60.0-61.3% for the color in the plain sampling baseline. We pick the diversity scores from Fig. 5.

Fig. 9 summarizes the user preference. We confirm that participants clearly prefer our model compared to CanvasVAE. The results support the hypothesis that our quantitative results indeed reflect human perception. On the other hand, our structure-preserved sampling does not make a difference in user preference. While unexpected, we suspect that our sampling hyper-parameter was too diverse to give appropriate colors to texts and that made the pairwise comparison difficult for users. In the future, we wish to continue on studying how to suggest the most comfortable designs.

6. Conclusion

In this paper, we formulate the task of typography generation where we have to generate diverse yet compelling typography given the input contexts. We build a fine-grained typographic attribute generation model and propose a sampling technique to generate diverse typography with consistency and distinction among texts. The empirical study confirms our approach successfully generates diverse yet consistent typography and outperforms the baselines.

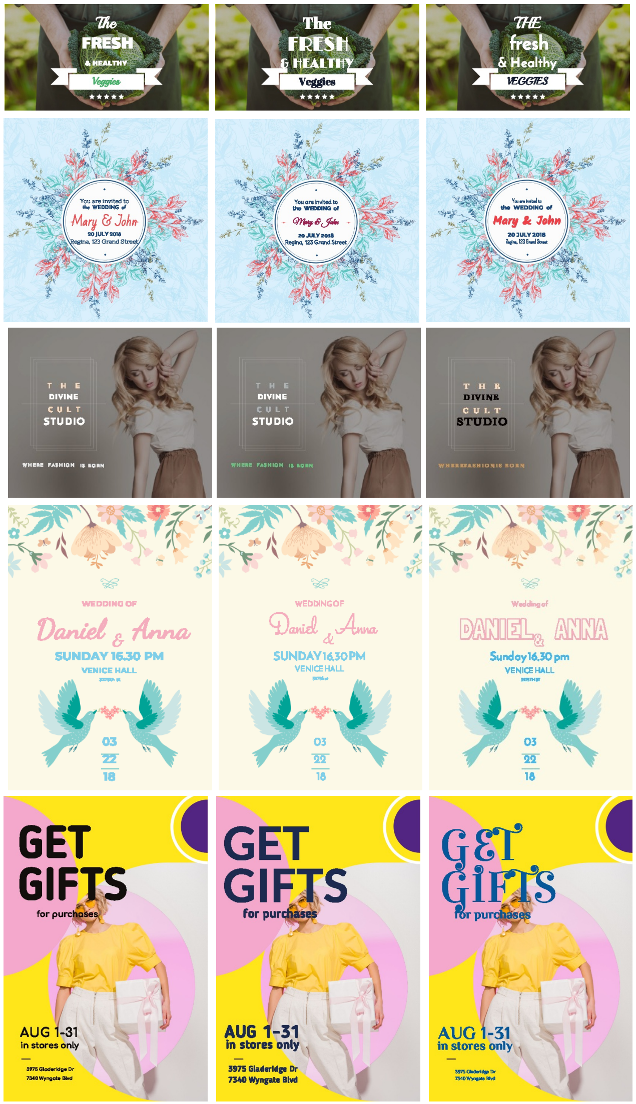

Figure 7. Diverse generation examples. Each row shows three generated examples for the same input.

There are remaining research questions we wish to explore. We hope to analyze the relationship between attributes to human perception, as we identify that the fidelity of colors to the given background somehow dominates the first impression of the design. We also hope to study to what degree of diversity users prefer in the generated results, for building a practical typography generation system.

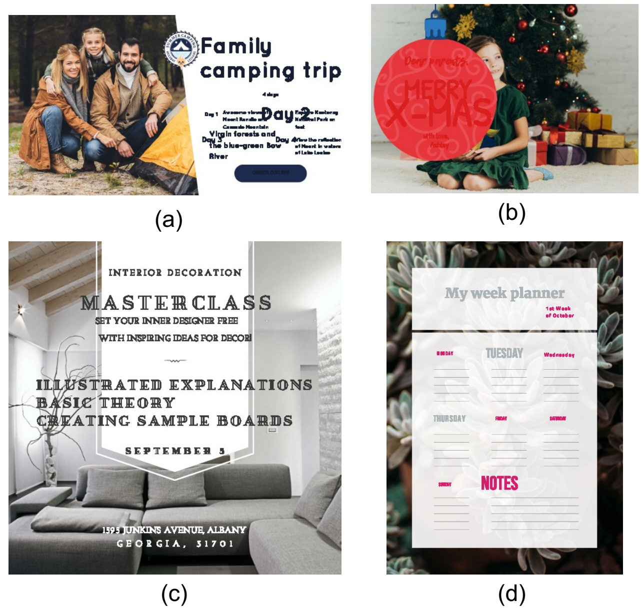

Figure 8. Failure cases.

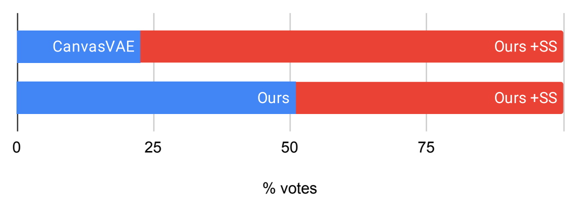

Figure 9. User preference in pairwise comparison.

References

[1] Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[2] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content GAN for few-shot font style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[3] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, 1992. 1

[4] Yunning Cao, Ye Ma, Min Zhou, Chuanbin Liu, Hongtao Xie, Tiezheng Ge, and Yuning Jiang. Geometry aligned variational transformer for image-conditioned layout generation. In *ACM International Conference on Multimedia (ACMMM)*, 2022. 3

[5] Niranjan Damera-Venkata, José Bento, and Eamonn O'Brien-Strain. Probabilistic document model for automated document composition. In *Proceedings of the 11th ACM Symposium on Document Engineering*, 2011. 2

[6] Yue Gao, Yuan Guo, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics (ToG)*, 2019. 2

[7] Yifan Gao, Jinpeng Lin, Min Zhou, Chuanbin Liu, Hongtao Xie, Tiezheng Ge, and Yuning Jiang. Textpainter: Multimodal text image generation with visual-harmony and text-comprehension for poster design. *arXiv preprint arXiv:2308.04733*, 2023. 2

[8] Shunan Guo, Zhuochen Jin, Fuling Sun, Jingwen Li, Zhaorui Li, Yang Shi, and Nan Cao. Vinci: An intelligent graphic design system for generating advertising posters. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2021. 3

[9] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layout-Transformer: Layout generation and completion with self-attention. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1004–1014, 2021. 2

[10] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *International Conference on Learning Representations (ICLR)*, 2020. 4

[11] Nathan Hurst, Wilmot Li, and Kim Marriott. Review of automatic document formatting. In *Proceedings of the 9th ACM Symposium on Document Engineering*, 2009. 2

[12] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete diffusion model for controllable layout generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[13] Jiabao Ji, Guanhua Zhang, Zhaowen Wang, Bairu Hou, Zhifei Zhang, Brian Price, and Shiyu Chang. Improving diffusion models for scene text editing with dual encoders. *arXiv preprint arXiv:2304.05568*, 2023. 2

[14] Shuhui Jiang, Zhaowen Wang, Aaron Hertzmann, Hailin Jin, and Yun Fu. Visual font pairing. In *IEEE Transaction on Multimedia*, 2019. 2

[15] Zhaoyun Jiang, Huayu Deng, Zhongkai Wu, Jiaqi Guo, Shizhao Sun, Vuksan Mijovic, Zijiang Yang, Jian-Guang Lou, and Dongmei Zhang. Layoutformer++: Conditional graphic layout generation via constraint serialization and decoding space restriction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[16] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. LayoutVAE: Stochastic scene layout generation from a label set. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[17] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *ACM International Conference on Multimedia (ACMMM)*, 2021. 3

[18] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. Blt: bidirectional layout transformer for controllable layout generation. In *European Conference on Computer Vision (ECCV)*, pages 474–490. Springer, 2022. 3

[19] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. Blt: Bidirectional layout transformer for controllable layout generation. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[20] Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B. Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. Neural design network: Graphic layout generation with constraints. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[21] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *North American Chapter of the Association for Computational Linguistics(NAACL)*, 2019. 4

[22] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. LayoutGAN: Generating graphic layouts with wireframe discriminators. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[23] Jinpeng Lin, Min Zhou, Ye Ma, Yifan Gao, Chenxi Fei, Yangjian Chen, Zhang Yu, and Tiezheng Ge. Autoposter: A highly automatic and content-aware design system for advertising poster generation. *arXiv preprint arXiv:2308.01095*, 2023. 2

[24] Rosanne Liu, Dan Garrette, Chitwan Saharia, William Chan, Adam Roberts, Sharan Narang, Irina Blok, RJ Mical, Mohammad Norouzi, and Noah Constant. Character-aware models improve visual text rendering. *arXiv preprint arXiv:2212.10562*, 2022. 2

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. 5

[26] M. Ronnier Luo, Guihua Cui, and Bryan Rigg. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research & Application*, 2001. 5

[27] Jian Ma, Mingjun Zhao, Chen Chen, Ruichen Wang, Di Niu, Haonan Lu, and Xiaodong Lin. Glyphdraw: Learning to draw chinese characters in image synthesis models coherently. *arXiv preprint arXiv:2303.17870*, 2023. 2

[28] Yifang Men, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Dyntypo: Example-based dynamic text effects transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[29] Taiga Miyazono, Brian Kenji Iwana, Daichi Haraguchi, and Seiichi Uchida. Font style that fits an image – font generation based on image context. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2021. 2

[30] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning Layouts for Single-Page Graphic Designs. *IEEE Transactions on Visualization and Computer Graphics*, 2014. 2

[31] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. Read: Recursive autoencoders for document layout generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020. 3

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 3

[33] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 2

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 2

[35] Prasun Roy, Saumik Bhattacharya, Subhankar Ghosh, and Umapada Pal. Stefann: Scene text editor using font adaptive neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[36] Wataru Shimoda, Daichi Haraguchi, Seiichi Uchida, and Yamaguchi Kota. De-rendering stylized texts. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NeurIPS)*, 2017. 3

[38] Wenjing Wang, Jiaying Liu, Shuai Yang, and Zongming Guo. Typography with decor: Intelligent text style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[39] Yizhi Wang, Guo Pu, Wenhan Luo, Yexin Wang, Pengfei Xiong, Hongwen Kang, and Zhouhui Lian. Aesthetic text logo synthesis via content-aware layout inferring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[40] Robin Williams. *Non-Designer's Design Book*. Peachpit Press, 2014. 1

[41] Liang Wu, Chengquan Zhang, Jiaming Liu, Junyu Han, Jingtuo Liu, Errui Ding, and Xiang Bai. Editing text in the wild. In *ACM International Conference on Multimedia (ACMMM)*, 2018. 2

[42] Kota Yamaguchi. CanvasVAE: Learning to generate vector graphic documents. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2, 5

[43] Qiangpeng Yang, Hongsheng Jin, Jun Huang, and Wei Lin. Swaptext: Image based texts transfer in scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[44] Shuai Yang, Jiaying Liu, Zhouhui Lian, and Zongming Guo. Awesome typography: Statistics-based text effects transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[45] Shuai Yang, Jiaying Liu, Wenjing Wang, and Zongming Guo. Tet-gan: Text effects transfer via stylization and destylization. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2019. 2

[46] Shuai Yang, Jiaying Liu, Wenhan Yang, and Zongming Guo. Context-aware text-based binary image stylization and synthesis. *IEEE Trans. on Image Processing*, 2018. 2

[47] Shuai Yang, Zhangyang Wang, Zhaowen Wang, Ning Xu, Jiaying Liu, and Zongming Guo. Controllable artistic text style transfer via shape-matching GAN. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[48] Yukang Yang, Dongnan Gui, Yuhui Yuan, Haisong Ding, Han Hu, and Kai Chen. Glyphcontrol: Glyph conditional control for visual text generation. *arXiv preprint arXiv:2305.18259*, 2023. 2

[49] Nanxuan Zhao, Ying Cao, and Rynson W.H. Lau. Modeling fonts in context: Font prediction on web designs. In *Pacific Conference on Computer Graphics and Applications(PG)*, 2018. 2, 5

[50] Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson W. H. Lau. Content-aware generative modeling of graphic design layouts. *ACM SIGGRAPH*, 2019. 3

[51] Min Zhou, Chenchen Xu, Ye Ma, Tiezheng Ge, Yuning Jiang, and Weiwei Xu. Composition-aware graphic layout gan for visual-textual presentation designs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022. 3