Layout Generation as Intermediate Action Sequence Prediction

Huiting Yang [*1], **Danqing Huang** [2], **Chin-Yew Lin** [2], **Shengfeng He** [3, 1†]

[1] South China University of Technology
[2] Microsoft Research Asia
[3] Singapore Management University
berylsheep@gmail.com, dahua@microsoft.com, cyl@microsoft.com, shengfenghe@smu.edu.sg

Abstract

Layout generation plays a crucial role in graphic design intelligence. One important characteristic of the graphic layouts is that they usually follow certain design principles. For example, the principle of repetition emphasizes the reuse of similar visual elements throughout the design. To generate a layout, previous works mainly attempt at predicting the absolute value of bounding box for each element, where such target representation has hidden the information of higher-order design operations like repetition (e.g. copy the size of the previously generated element). In this paper, we introduce a novel action schema to encode these operations for better modeling the generation process. Instead of predicting the bounding box values, our approach autoregressively outputs the intermediate action sequence, which can then be deterministically converted to the final layout. We achieve state-of-the-art performances on three datasets. Both automatic and human evaluations show that our approach generates high-quality and diverse layouts. Furthermore, we revisit the commonly used evaluation metric FID adapted in this task, and observe that previous works use different settings to train the feature extractor for obtaining real/generated data distribution, which leads to inconsistent conclusions. We conduct an in-depth analysis on this metric and settle for a more robust and reliable evaluation setting. Code is available at this website [1].

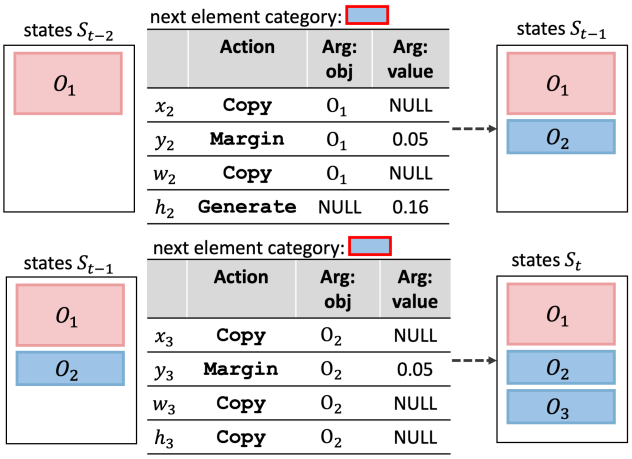

Figure 1: An example of the layout generation process. The actions `copy` and `margin` are introduced for encoding the design principles of repetition and white space. Given state S_{t-2}, our approach generates O_2 using four actions: `copy` the x-axis and width value from O_1 for better alignment, `margin` with 0.05 (normalized) white space below O_1 in the y-axis, and `generate` the absolute values for h_2.

1 Introduction

Layout refers to the arrangements (i.e. position and size) of visual elements on a canvas, which plays an important role in graphic design. A good layout can create a unified composition of elements to make the design more organized and visually appealing. There is a growing interest in the research of automatic layout generation (Li et al. 2019; Jyothi et al. 2019; Kikuchi et al. 2021; Gupta et al. 2021). With or without user specification (e.g. category constraint that a layout must contain one title and two pictures), the goal is to synthesize reasonable layouts by generating the bounding boxes of elements. Different from real images at pixel-level, graphic layouts are more structured which require higher-level modeling by considering attributes of the elements as well as their inter-relations.

Across different graphic design types (e.g. mobile app, article, slide), the creation of layouts usually follows certain design principles. For example, the repetition principle is the act of repeating the same or similar elements and making them more aligned as a cohesive whole. Meanwhile, margin is important to create visual distinction between important elements. These principles provide a very indicative guideline for creating good layouts. However, most of the previous works directly generate the absolute values of bounding boxes for elements, which hides the higher-order design operations such as repetition and margin. Recently several works try to incorporate beautification constraints such as alignment and non-overlap with additional training losses (Lee et al. 2020; Li et al. 2021) or extra penalty function in the latent space optimization (Kikuchi et al. 2021).

[*]Work done during Huiting Yang's internship at Microsoft Research Asia.
[†]Corresponding author

[1]https://github.com/microsoft/KC/tree/main/papers/LayoutAction

They serve as soft regularization which requires careful design of the objective functions.

This paper explores a simpler and more direct solution to encode the design principles by introducing a novel action schema. Specifically, we define three actions: (1) `generate`, to generate the absolute geometry value; (2) `copy`, to repeat the value from a previous generated element; (3) `margin`, to position the element by creating a margin to another element. In this way, layout synthesis can be viewed as action sequence generation. Instead of directly generating the bounding boxes, we generate the intermediate action sequence as output, which can be deterministically converted to the final layout via post-processing. Similar to LayoutTransformer (Gupta et al. 2021), our approach adopts Transformer (Vaswani et al. 2017) as the autoregressive sequence decoder. At each timestep, the model inputs the previously generated sequence and predicts the next token. During inference, we apply nucleus sampling (Holtzman et al. 2019) to synthesize diverse layout samples.

To evaluate the quality and diversity of the generated layouts, Fréchet Inception Distance metric (FID) (Heusel et al. 2017) is commonly reported. It requires a well-trained feature extractor to obtain the distribution of real/generated data and calculate the FID between them. Different from the natural image evaluation where the feature extractor is a standard model backbone (e.g. Inception-V3 (Szegedy et al. 2016) or ResNet (He et al. 2016)) pre-trained on ImageNet (Deng et al. 2009) classification, the settings of extractor (e.g. network architecture, training objective) in layout evaluation vary in previous works. We show that such inconsistent settings can lead to different FID evaluation conclusions. In this paper, we conduct an in-depth analysis on training the feature extractor and aim to settle for a more robust and reliable FID evaluation setting.

We conduct experiments on three datasets with different types of design, including mobile UI, scientific documents, and slides. Both automatic and human evaluations show that our approach performs consistently better than the baselines. To summarize, the contributions of this paper include:

1. We propose a simple but effective action schema to model the layout generation process with graphic design principles. Instead of directly generating the absolute values of bounding boxes, we use the intermediate action sequence as the target output. Experiment results show that our approach achieves state-of-the-art performance.
2. We revisit the evaluation metric FID for layout generation, where we have some interesting findings and it leads to a more robust evaluation setting.

2 Related Work

2.1 Layout Generation

Nowadays Generative Adversarial Networks (GANs) (Goodfellow et al. 2014; Brock, Donahue, and Simonyan 2018; Karras, Laine, and Aila 2019; Karras et al. 2021), Variational Autoencoders (VAEs) (Kingma and Welling 2014; Higgins et al. 2017) and autoregressive models have been the mainstream generative modeling frameworks, which have been shown promising to generate high-quality samples of image and natural language and other modalities. Recently, they have been adapted to layout generation in graphic design. LayoutGAN (Li et al. 2019) utilizes self-attention modules to generate layouts and proposes a differentiable wireframe renderings layer with a CNN-based discriminator to optimize the layouts in the image space. But it has been shown to suffer from unstable training in later works (Arroyo, Postels, and Tombari 2021; Gupta et al. 2021). LayoutVAE (Jyothi et al. 2019) uses two conditional VAEs to first predict the element category label counts and then generate the bounding boxes. VTN (Arroyo, Postels, and Tombari 2021) follows the VAE framework with a Transformer backbone. NDN (Lee et al. 2020) treats the layout as a directed graph where the edge indicates the relative size and position between two element nodes. They convert the generation task into a graph completion task and also adopts VAE for learning. LayoutTransformer (Gupta et al. 2021) and BLT (Kong et al. 2022) adopt the Transformer decoder to (non-)autoregressively generate the geometry values. The above approaches all use the absolute values of bounding box as target output, which has hide some higher-order design operations such as repetition and margins.

To consider the design principles into learning, previous works mainly focus on latent space constraints. NDN iteratively fine-tunes the predicted bounding boxes with an additional alignment loss in the refinement module. Similarly, Attribute-Conditioned LayoutGAN (Li et al. 2021) adds several losses targeted for alignment, non-overlap and margin area. LayoutGAN++ (Kikuchi et al. 2021), based on a Transformer generator and discriminator, satisfies the beautification constraints (i.e. alignment and non-overlap) by optimizing latent codes with a penalty function. Different from previous works, this paper explores a more direct solution that uses an intermediate action schema for capturing the design operations, which works surprisingly well.

2.2 Autoregressive Sequence Generation

Autoregressive models output the target tokens step-by-step depending on the previous generated sequence.They have been widely used for generating images (Oord, Kalchbrenner, and Kavukcuoglu 2016; van den Oord et al. 2016), natural languages (Brown et al. 2020) and other modalities. For example, PixelCNN (van den Oord et al. 2016) sequentially generates one pixel at a time in an image along the two spatial dimensions. In our task, autoregressive generation is also favored by many previous works. For example, LayoutTransformer generates layouts with the flattened sequence consisting of the category labels and the bounding boxes. Though the non-autoregressive models with parallel decoding enjoy fast inference speed, generally the autoregressive ones perform much better due to the conditionally dependent decoding.

The design of our action tokens is partially inspired from the copy mechanism widely used in seq2seq models, which copies the source tokens to the target sentences. Having observed that human tend to repeat entity names or even long phrases in conversation, CopyNet (Gu et al. 2016) provides the generation and copy modes in the decoding. Similarly in

graphic design, elements tend to repeat the position/size or share the same margin value in the layout. Therefore we propose an action schema to encode such information for better model generalization.

3 Approach

In this section, we first describe the task and introduce our proposed action schema. Next, we present our model architecture and training objective of generating the action sequence for layout generation.

3.1 Task Formulation

In general, a graphic layout L contains n visual elements. Each element e_i holds both the semantic and geometric properties, i.e. (c_i, x_i, y_i, w_i, h_i). $c_i \in C$ represents the element category label such as title and figure, and the remaining represent its bounding box, where (x_i, y_i) are the center coordinate and (w_i, h_i) are the width and height. With or without a set of category labels with counts as input (e.g. one title plus two figures), the goal is to generate layouts with the elements' bounding boxes.

3.2 Action for Layout Generation

Here we give the action definition and how we obtain the action sequence for the layout generation process.

Action Definition To obtain a well-formatted layout, there are several design principles to be followed. Our action language is mainly motivated by the two widely-used principles: (1) **Repetition**: the reuse of design attributes can make a layout more unified. For example, copying the x/y-coordinate or the size of an element produces better alignment. (2) **White Space**: margins between elements create visual distinction. In a layout, most of the elements are not overlapped.

Based on the above observations, we propose a schema with three actions:

- A_g = `generate(NULL, arg:value)`: generate the absolute geometry value.
- A_c = `copy(arg:object, NULL)`: choose an existing element as anchor object to copy its geometry value, which encourages repetition.
- A_m = `margin(arg:object, arg:value)`: choose an existing element as anchor object and create a margin value with the anchor's position. For example, when generating the center x_i for the i-th element with an anchor o_j and a margin value v, we can obtain x_i as $x_j + 0.5 * w_j + v + 0.5 * w_i$. This action encourages non-overlapping and consistent margins.

We show an example of the layout generation process with actions in Figure 1, Given the previous layout state S_{t-1}, the position and size of element O_3 would be determined by four actions. As we can see, O_3 should be aligned with O_2 in the x-axis and thus we apply the action `copy(`O_2`, NULL)` for x_3. Meanwhile O_3 should be placed under O_2 with the same margin value as O_2 to O_1, therefore we use `margin(`O_2`, 0.05)` to obtain y_3. For

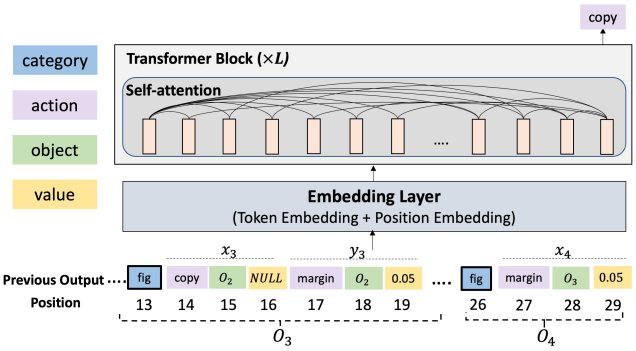

Figure 2: The overview of our approach. At each timestep, the Transformer decoder inputs the previously generated tokens and predicts the next one. The position and size (x, y, w, h) of each element are determined by a subsequence of 13 tokens (one category label plus four actions).

a consistent representation, we pad the empty argument of some actions with a `NULL` token, i.e. `generate(NULL, arg:value)` and `copy(arg:object, NULL)`.

Action Sequence Derivation The layout generation process can be viewed as an action sequence. The i-th element is expressed using a sequence of 13 tokens with a category label and four actions corresponding to x, y, w, h: (c_i, a_i^x, o_i^x, v_i^x, a_i^y, o_i^y, v_i^y, a_i^w, o_i^w, v_i^w, a_i^h, o_i^h, v_i^h), where a, o, v indicate the action name, object argument and value argument respectively. We concatenate all the elements in a flattened sequence and sort them in ascending order according to their (x, y) coordinates. The layout can be denoted by a sequence of $13n + 2$ tokens:

$$\mathbf{s} = (< bos >, c_0, a_0^x, o_0^x, v_0^x, \cdots, a_n^h, o_n^h, v_n^h, < eos >), \tag{1}$$

where $< bos >$ and $< eos >$ denote the beginning and end of a sequence.

In the absence of ground truth labels for the sequence, we define several rules to ascertain the labels as follows: the actions are prioritized in the order of `copy`, `margin` and `generate`. We would like to encourage the use of `copy` and `margin`, and thus they have higher priorities in our derivation rules. Specifically, `copy` is chosen when current element shares the same value of coordinate (x/y) or size (w/h) with one of the previous elements. `margin` is only applied on the coordinates. It is chosen if the element's value in another axis is applied on `copy`. For example, if the x-axis of an element e_p is copied from e_q, then the y-axis of e_p uses the action `margin` with e_q. This heuristic is based on our observation that elements located on the same horizontal or vertical plane usually are non-overlapped and follow frequently-used margin values across the design corpus. At last, when the element does not meet the above conditions, `generate` is selected by default. Please note that one layout can be mapped to multiple correct action sequences (e.g. a sequence consisting of only the `generate` actions). During the training process, one layout is only mapped to only one ground truth sequence.

3.3 Model Architecture and Training

The overview of our approach is shown in Figure 2. Following Gupta *et al.* (Gupta et al. 2021), we adopt the Transformer backbone as an autoregressive decoder. The model's objective can be now changed to maximize the likelihood of the action sequence $\mathbf{s}$, which can be factorized into the following products of conditional distributions:

$$p(\mathbf{s}) = \prod_{i=0}^{13n+1} p(s_{i+1}|s_{0:i}), \tag{2}$$

where s_i is the i-th token in the action sequence.

At each timestep i, an embedding layer sums the action token embedding ϵ_i and the position embedding $\mathbf{p}_i$ to obtain the hidden state $\mathbf{h}_i$, which are then fed to Transformer with self-attention to include the information of previous tokens for a contextualized $\mathbf{h}'_i$:

$$\begin{aligned} \mathbf{h}_i &= \mathrm{MLP}(\epsilon_\mathrm{i} + \mathbf{p}_\mathrm{i}; \theta) \\ \mathbf{h}'_i &= \sum_{j=0}^{i} \mathrm{softmax}(\alpha_\mathrm{ij})(\mathbf{W}_\mathrm{V}\mathbf{h}_\mathrm{j}) \\ \alpha_{ij} &= \frac{1}{\sqrt{\mathbf{d_K}}}(\mathbf{W}_Q\mathbf{h}'_i)(\mathbf{W}_K\mathbf{h}'_j)^T, \end{aligned} \tag{3}$$

where ϵ, $\mathbf{p}$, θ, $\mathbf{W}_Q$, $\mathbf{W}_K$ and $\mathbf{W}_V$ are learnable model parameters and $\mathbf{d_K}$ is the dimensions of $\mathbf{W_K}$. On the top is a classification layer implemented as a MLP (Multi-Layer Perceptron) with softmax to predict the token s_{i+1}:

$$\begin{aligned} p(s_{i+1}|s_{0:i}) &= \mathrm{softmax}(\mathrm{MLP}(\mathbf{h}'_\mathrm{i}; \theta)) \\ \mathcal{L} &= \sum_{i=1}^{13n+1} -\log p(s_i), \end{aligned} \tag{4}$$

where $\mathcal{L}$ is the cross entropy loss for classification. We use teacher forcing in the training and validation for a faster and more stable learning process. The vocabulary for input and output includes all types of tokens, i.e. element category labels, actions, object indexes, and geometry values. Specifically, the floating geometry values are discretized using 8-bit uniform quantization and converted to integers for categorical encoding.

4 Experiments

In this section, we first introduce the experiment settings and revisit the current FID metric for layout evaluation. Then we show both the quantitative and qualitative results of our approach compared against other strong baselines.

4.1 Experiment Setting

Datasets We evaluate on three datasets with different types of graphic designs:

- **Rico** (Deka et al. 2017; Liu et al. 2018). The dataset consists of over 66k unique UI layouts from more than 9.3k Android mobile apps spanning 27 categories. Following previous works, we exclude elements whose labels are not in the 13 most frequent labels and exclude layouts with more than 9 elements. After filtering there are 20,507 layouts in total.
- **PubLayNet** (Zhong, Tang, and Yepes 2019). It is a large collection of over 360k scientific document images crawled from PubMed Central™. Each element in the layout is annotated with its category label (text, title, list, table, and figure) and bounding box. Similarly, layouts with more than 9 are excluded, totaling 173,225 layouts in the final set.
- **InfoPPT** (Shi et al. 2022). The dataset includes 23,072 information presentations collected from the Internet, with manually filtering to ensure the diversity and quality[2]. Since the slide layouts are more diverse than the other two datasets, we exclude several categories such as footnote and decorators (e.g. line, arrow), and exclude layouts with elements less than 4 elements as well as more than 20 elements. There are 46,654 final layouts.

For all the datasets, we randomly split the dataset into 85% train, 5% validation, and 10% test. In this paper, we do not consider the Magazine dataset (Zheng et al. 2019) as used in some previous works (Zheng et al. 2019; Lee et al. 2020), since its layouts are highly content-dependent (e.g. the placement of a title cannot be overlapped with the objects in the background image) and we currently do not consider the element contents into modeling.

Baselines We consider the following publicly-available baselines:

- **LayoutVAE** (Jyothi et al. 2019). Given a label set of element categories, the sub-module CountVAE first predicts the counts for each category, which is then used as the input to another sub-module BBoxVAE to generate the bounding boxes sequentially. We use its BBoxVAE for comparison, which has the same input as other baselines.
- **LayoutGAN++** (Kikuchi et al. 2021). Improved from LayoutGAN (Li et al. 2019), it adopts two Transformers as generator and discriminator respectively. Moreover, the model is coupled with latent code optimization as to incorporate beautification constraints such as alignment and non-overlap.
- **LayoutTransformer** (Gupta et al. 2021). This model autoregressively generates the flattened sequence consisting of element category labels and bounding boxes. It adopts a Transformer decoder and obtains the results via nucleus sampling (Holtzman et al. 2019).

4.2 Evaluation Metrics

Heuristic Metrics Several metrics are proposed in previous works[3]:

- **Maximum Intersection over Union (IoU)**. It measures the closeness of the generated layouts to the references using the IoU-based similarity.
- **Alignment**. The calculation is based on the minimum distance between any element pairs in a layout via six

[2]We use the open-source library https://pypi.org/project/python-pptx/ to parse the slides.

[3]We use the implementation of Kotaro *et al.* (Kikuchi et al. 2021) for these metrics.

Model Architecture for ϕ	Transformer (Dec)		Transformer (Enc-Dec)		GMN		ResNet-18	
Pre-training Objective for ϕ (cls acc %)	cls. (94.26)	w/ cl. (97.48)	cls. (86.43)	w/ cl. (84.07)	cls. (89.01)	w/ cl. (92.25)	cls. (100)	w/ cl. (94.87)
LayoutVAE	1221.97 (4)	4121.78 (3)	148.96 (4)	140.79 (4)	679.24 (4)	339.56 (4)	481.40 (4)	36.22 (4)
LayoutGAN++	1221.01 (3)	4824.31 (4)	**30.76** (1)	51.77 (3)	425.28 (3)	187.29 (3)	28.20 (3)	4.16 (3)
LayoutTransformer	**39.21** (1)	866.04 (2)	42.80 (3)	46.33 (2)	195.14 (2)	184.07 (2)	13.04 (2)	3.69 (2)
Ours	47.61 (2)	**812.74** (1)	36.74 (2)	**37.92** (1)	**146.29** (1)	**124.59** (1)	**8.49** (1)	**2.38** (1)
Real Data	24.13	405.53	3.61	4.02	96.94	99.09	0.28	1.93
Gauss Neg.	1473.45	1550.76	143.67	97.12	519.87	189.24	141.11	5.93
Shuffle Neg.	1550.76	5378.29	265.04	212.12	464.60	225.75	53.36	35.72

Table 1: Different settings of the feature function ϕ (model architecture, pre-training objective) can lead to different FID comparison results. The experiment is conducted on the Rico dataset. cls. means read/fake layout classification training objective. w/ cl. means auxiliary contrastive learning.

Datasets	Rico				PubLayNet				InfoPPT			
	FID↓	IoU	Align	Ovp	FID↓	IoU	Align	Ovp	FID↓	IoU	Align	Ovp
LayoutVAE	6.56	0.24	0.98	66.25	36.21	0.28	0.69	**8.98**	22.03	0.20	1.11	68.07
LayoutGAN++	4.16	**0.36**	0.60	59.85	45.77	0.36	0.19	22.80	19.01	0.09	0.32	127.02
LayoutTransformer	3.69	**0.36**	0.06	71.79	66.37	**0.44**	0.32	15.40	12.99	**0.21**	0.36	53.35
Ours	**2.38**	0.33	**0.12**	**52.31**	**25.88**	0.32	**0.10**	9.28	**12.38**	0.16	**0.23**	**45.55**
Real data	1.93	0.68	0.27	51.31	1.78	0.53	0.04	0.22	0.40	0.75	0.14	17.20

Table 2: Overall results on three datasets. For the three heuristic metrics (IoU, Align and Overlap), the closer values to real data, the better is the performance.

types of alignment, including left, x-center, right (x-axis), top, y-center, and bottom (y-axis). The smaller score indicates the layout is more likely to be well-aligned.

- **Overlap**. It calculates the overlapping area between any element pairs by assuming that well-designed layouts typically avoid overlapping elements.

Please note that the above metrics are based on simple heuristics, which suffer from different limitations. For example, a model good at memorizing all training samples would score perfectly in terms of the IoU metric since this metric calculates the maximum IoU between generated layouts and the training ones.

Seeking for a robust FID Evaluation FID (Heusel et al. 2017) is a popular learning-based metric that measures both the quality and diversity of the generated results. Given a feature function ϕ, the metric calculates the Fréchet distance between two Gaussian distributions $\phi(\mathbb{P}_r)$ and $\phi(\mathbb{P}_g)$. For natural image evaluation (Karras, Laine, and Aila 2019), the feature function is by default pre-trained on the ImageNet (Deng et al. 2009) using the Inception network (Szegedy et al. 2016). While being compared, the setting of pre-training the feature function for layout generation is rather inconsistent, varying from network architecture to data set and training objective. In this paper, we conduct an empirical study to answer the question: **What is the reasonable setting for pre-training a robust FID evaluator for layouts?** We explore some most relevant options to pre-train the feature function ϕ:

- **Model architecture**: (1) Transformer Decoder, which is the same as the LayoutTransformer backbone (Gupta et al. 2021); (2) Transformer Encoder-Decoder, which is the same as LayoutGAN++ (Kikuchi et al. 2021); (3) Layout Matching Network (GMN) (Patil et al. 2021), which uses the graph structure to represent a layout for similarity learning and retrieval. (4) ResNet-18 (He et al. 2016), a CNN-based model which takes the layout image as input.
- **Negative training data (fake layouts)**: (1) Gaussian 0.01 noise added to the bounding boxes; (2) In-batch shuffle that randomly interchanges elements of two layouts in a batch, which has larger perturbation to the layouts than the Gaussian noise.
- **Pre-training objective**: (1) by default real/fake layout classification as used in previous works; (2) auxiliary contrastive learning: we constrain the distance of real data distribution with the shuffle negatives to be farther than with the Gaussian negatives (smaller perturbation), which is well-correlated with human perception.

We pre-train the feature function ϕ with the above different settings, which is later used to evaluate the layout generation systems. Table 1 shows the results on Rico dataset with the following observations:

1. The classification accuracy (cls. acc) of training ϕ in different settings can all reach over 85% on average, which means that ϕ can generally capture discriminative representations to distinguish real and fake layouts to a certain degree.

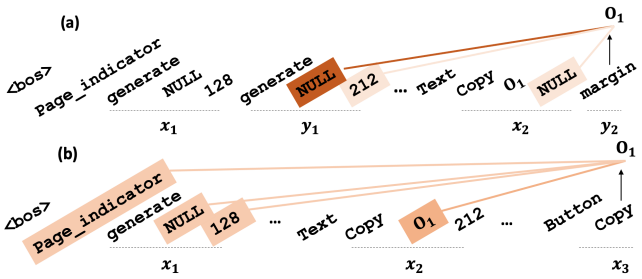

Figure 3: Two examples of attention visualization at a timestep. Darker color indicates higher attention weight.

	Rico		PubLayNet		InfoPPT	
	Real data	ours	Real data	ours	Real data	ours
grammar correctness (%)	100	99.92	100	100	100	99.98
generate (%)	54.25	53.72	62.98	60.18	40.67	47.78
copy (%)	33.63	34.06	26.06	27.78	49.35	41.98
margin (%)	22.24	24.28	24.92	24.09	19.96	20.44

Table 3: Statistics on the action sequence, including grammar correctness and trigger rate of different actions.

2. We observe similar performance rankings of systems under almost all FID settings: LayoutVAE performs the worst and our model performs the best. This indicates the robustness of the FID measurement, and our model does perform consistently better than the baselines.
3. FID can be biased when ϕ has the same model architecture with the system to be evaluated. FID using pre-trained Transformer (Dec) is biased to LayoutTransformer as they share the same backbone. Similar bias occurs for the FID using Transformer (Enc-Dec) to LayoutGAN++. This is likely because ϕ and the system to be evaluated obtain similar feature distributions with the same model architecture, which is not fair to other systems.
4. Contrastive learning (w/ cl.) helps regularize the learned distribution of ϕ. In the last two rows, the FID score of the Gaussian negatives is higher than the shuffle negatives using the model architecture of GMN and ResNet-18, which is counter-intuitive since the Gaussian noise has smaller perturbation than shuffling and its data distribution should be closer to the real data. After adding the contrastive loss, the FID scores become more reasonable.

Based on the above findings, we use the pre-trained feature function ϕ using ResNet-18 with the auxiliary contrastive learning for the FID evaluation in the following experiments.

4.3 Implementation Details

We set the Transformer with 6 layers of hidden size 512. The number of attention heads is set to 8. We use AdamW optimizer (Loshchilov and Hutter 2017) with initial learning rate of 3e-4, $\beta_1 = 0.9$, $\beta_1 = 0.95$ and l_2 weight decay of 5e-4. We also apply early stopping, gradient clipping (Pascanu, Mikolov, and Bengio 2013), and warm up over the initial 1% training iterations. The dropout rate is set to 0.1. Models are trained with maximum of 50 epochs with batch size 64. During inference, we sample from the multinomial distribution and the top-k sampling (k = 5) for all models.

	Rico		PubLayNet		InfoPPT	
	ranking↓	Kappa	ranking↓	Kappa	ranking↓	Kappa
LayoutVAE	6	-0.069	5	0.175	5	0.068
LayoutGAN++	4	-0.069	4	0.168	4	0.147
LayoutTransformer	2	0.026	2	0.2	2	0.131
Ours	**1**	**0.016**	**1**	**0.228**	**1**	**0.249**
Gaussian Neg.	2	0.012	3	0.145	3	0.273
Shuffle Neg.	5	-0.04	6	0.139	6	0.141

Table 4: Human evaluation results: rankings by merging the annotators' results.

4.4 Quantitative Results

Overall Performance We show the overall system performances on all three datasets in Table 2. Regarding FID, the classification accuracy of pre-training the feature function on Rico, PubLayNet, and InfoPPT are 94.87%, 99.92%, and 84.25% respectively. As we can see, our approach performs consistently better than the baselines on three datasets. Meanwhile, we use a separate set of real data to compute the metrics, which are displayed in the last row. For the three heuristic metrics (i.e. IoU, Align, and Ovp), we expect a good system to be closer to the real data performance, following Gupta *et al.* (Gupta et al. 2021). The results show that our method outperforms all baselines in terms of FID and alignment. For overlap, we are the best on Rico and InfoPPT, while achieving comparable results with LayoutVAE that exceeds the other two baselines by a large margin. For IoU, our scores are slightly lower than other methods. We argue that this metric mainly measures the closeness to the references but ignores diversity, since high IoU may be likely due to the model being good at memorizing training data. To summarize, our approach achieves state-of-the-art results in most of the metrics, which demonstrates the effectiveness of our proposed action schema.

Model Analysis To better understand the behavior of our model, we conduct the following analyses. We first visualize the attention weight (Equation 3) in the last Transformer layer. It measures the importance degree of the current i-th token attending to the previous j-th token. As we can see from Figure 3(a), when choosing the object index for the action `margin` of y_2, the model is highly attended to the correct token related to y_1. Similarly in Figure 3(b), the attention weight is correctly distributed to the relevant tokens to copy x_1 for x_3. Moreover, Table 3 shows some statistics on the action sequences derived from real data as well as our model outputs. The grammar correctness indicates whether the sequence complies with the grammar rules (i.e. action

token followed by the object index token and the value token). The accuracy of our model outputs is all over 99.9% in these datasets. Meanwhile, we calculate the triggering rate of `generate`, `copy` and `margin`. In real data, the action `copy` is about 30% in average use while the rate of `margin` is approximately 20%. In our predictions, the rates of actions are in similar distribution, which indicates our model's capability to leverage the actions of `copy` and `margin` for modeling design principles in the generation process.

4.5 Qualitative Results

To further compare the generation qualities, here we conduct a human evaluation and show some cases.

Human Evaluation We recruit 5 human annotators to rate the generated results. Given the same label set input, we randomly select the output from each system to form a group. For a better comparison, we also include two distorted layouts with Gaussian and shuffle noise (described in Section 4.2) respectively. There are 30 groups per dataset. We ask the annotators to rank the samples based on the design principles[4] and the degree of visual aesthetic, with the system information hidden. Table 4 reports the human evaluation results. As we can see, our approach is consistently ranked as the highest in three datasets, while LayoutVAE is ranked as the lowest. The rankings are well aligned with the quantitative results in Table 1 and Table 2. Also, most systems are ranked higher than the shuffle negatives, which ensures the lower bound of the generated outputs is better than the shuffle noise. As for the agreement between annotators, we calculate the Kappa (Randolph 2010)[5]. The Kappa on our system is relatively high compared to other baselines, indicating stronger agreement among annotators for voting our approach as the best.

Case Study We show some generated samples in Figure 4. Given the same category label input, we can see our generated samples are more visually aesthetic. Elements are better aligned and non-overlapped, and the widths/heights of some elements are repeatedly used. This further proves the effectiveness and generalization ability of our approach to capture the graphic design principles. Figure 5 demonstrates the diversity of our generated layouts. For one label set input, our approach can generate different layouts with good quality (1st row in each subfigure). In Figure 5, given the input condition of one picture and five textboxes, the three generated slide layouts are visually pleasing with alignments and appropriate margins. The pictures in the last two samples are considered as background images instead of undesired wrong overlapping. Moreover, the real layouts from the references with maximum IoU to our generated layouts (2nd row in each subfigure) are not identical to ours, which indicates that our approach can generate novel and diverse layouts instead of only memorizing the training data.

[4]https://en.wikibooks.org/wiki/Graphic_Design/Principles_of_Design

[5]As the number of categories and subjects will affect the magnitude of the Kappa, the absolute values here are less meaningless.

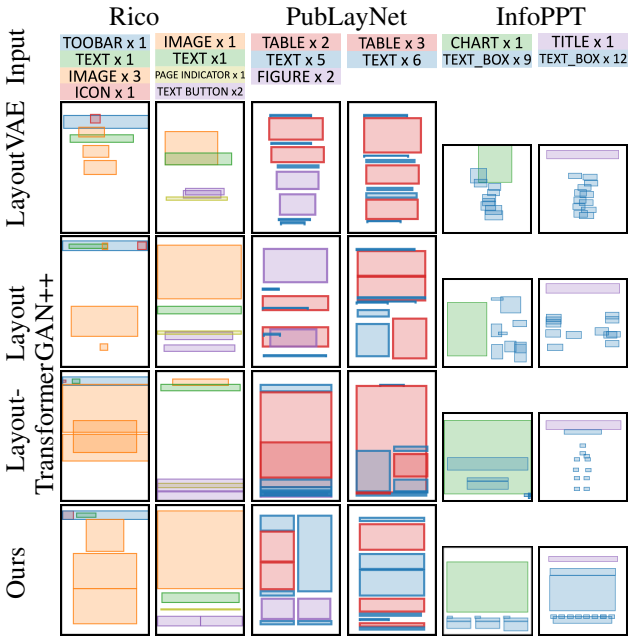

Figure 4: Conditional layout generation for mobile UI, scientific article and slide.

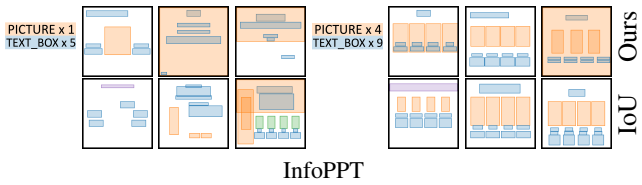

Figure 5: Diversity results given the same label set. The row with IoU indicators contains real layouts of max. IoU in the training data with our generated samples.

5 Conclusions

In this paper, we design an action schema to explicitly encode the graphic design principles such as repetition and white space. The layout generation process can be viewed as an action sequence. Instead of directly generating the absolute values of bounding boxes, our approach outputs the intermediate action sequence and obtains the final layout with deterministic post-processing. Moreover, we revisit the FID metric for layout evaluation with respect to the settings of its pre-trained feature function and obtain a more robust and reliable conclusion. Experiment results on three datasets show that our approach can synthesize layouts with higher quality and diversity. In the future, we will refine and expand the action schema to better include more design principles. Also, we would like to explore and propose more robust metrics for layout evaluation beyond current simple heuristics.

Acknowledgements

This project is supported by the National Natural Science Foundation of China (No. 61972162); Guangdong International Science and Technology Cooperation Project (No.

2021A0505030009); Guangdong Natural Science Foundation (No. 2021A1515012625); Guangzhou Basic and Applied Research Project (No. 202102021074); Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2023B1515020097). The corresponding author is Shengfeng He.

References

Arroyo, D. M.; Postels, J.; and Tombari, F. 2021. Variational Transformer Networks for Layout Generation. In *CVPR*.

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *ICLR*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *NeurIPS*, volume 33, 1877–1901. Curran Associates, Inc.

Deka, B.; Huang, Z.; Franzen, C.; Hibschman, J.; Afergan, D.; Li, Y.; Nichols, J.; and Kumar, R. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *ACM UIST*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*, 248–255.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In *NeurIPS*, volume 27. Curran Associates, Inc.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*, 1631–1640. Association for Computational Linguistics.

Gupta, K.; Lazarow, J.; Achille, A.; Davis, L. S.; Mahadevan, V.; and Shrivastava, A. 2021. Layouttransformer: Layout generation and completion with self-attention. In *ICCV*, 1004–1014.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. *CVPR*, 770–778.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*, volume 30.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C. P.; Glorot, X.; Botvinick, M. M.; Mohamed, S.; and Lerchner, A. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*.

Holtzman, A.; Buys, J.; Forbes, M.; and Choi, Y. 2019. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

Jyothi, A. A.; Durand, T.; He, J.; Sigal, L.; and Mori, G. 2019. LayoutVAE: Stochastic Scene Layout Generation From a Label Set. In *ICCV*, 9894–9903.

Karras, T.; Aittala, M.; Laine, S.; Härkönen, E.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2021. Alias-free generative adversarial networks. In *NeurIPS*, volume 34.

Karras, T.; Laine, S.; and Aila, T. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. In *CVPR*.

Kikuchi, K.; Simo-Serra, E.; Otani, M.; and Yamaguchi, K. 2021. Constrained graphic layout generation via latent optimization. In *ACM MM*, 88–96.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. *CoRR*, abs/1312.6114.

Kong, X.; Jiang, L.; Chang, H.; Zhang, H.; Hao, Y.; Gong, H.; and Essa, I. 2022. BLT: Bidirectional Layout Transformer for Controllable Layout Generation. In *ECCV*, 474–490.

Lee, H.-Y.; Jiang, L.; Essa, I.; Le, P. B.; Gong, H.; Yang, M.-H.; and Yang, W. 2020. Neural design network: Graphic layout generation with constraints. In *ECCV*, 491–506. Springer.

Li, J.; Yang, J.; Hertzmann, A.; Zhang, J.; and Xu, T. 2019. LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators. In *ICLR*.

Li, J.; Yang, J.; Zhang, J.; Liu, C.; Wang, C.; and Xu, T. 2021. Attribute-Conditioned Layout GAN for Automatic Graphic Design. *IEEE Transactions on Visualization and Computer Graphics*, 27(10): 4039–4048.

Liu, T. F.; Craft, M.; Situ, J.; Yumer, E.; Mech, R.; and Kumar, R. 2018. Learning Design Semantics for Mobile Apps. In *ACM UIST*.

Loshchilov, I.; and Hutter, F. 2017. Fixing Weight Decay Regularization in Adam. *ArXiv*, abs/1711.05101.

Oord, A. V.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel Recurrent Neural Networks. In *ICML*, volume 48 of *Proceedings of Machine Learning Research*, 1747–1756. PMLR.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the Difficulty of Training Recurrent Neural Networks. In *ICML*, ICML'13, III–1310–III–1318.

Patil, A. G.; Li, M.; Fisher, M.; Savva, M.; and Zhang, H. 2021. LayoutGMN: Neural Graph Matching for Structural Layout Similarity. In *CVPR*, 11048–11057.

Randolph, J. 2010. Free-Marginal Multirater Kappa (multirater kfree): An Alternative to Fleiss Fixed-Marginal Multirater Kappa. *Advances in Data Analysis and Classification*, 4.

Shi, D.; Cui, W.; Huang, D.; Zhang, H.; and Cao, N. 2022. Reverse-Engineering Information Presentations: Recovering Hierarchical Grouping from Layouts of Visual Elements. *ArXiv*, abs/2201.05194.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2818–2826.

van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; kavukcuoglu, k.; Vinyals, O.; and Graves, A. 2016. Conditional Image Generation with PixelCNN Decoders. In *NeurIPS*, volume 29.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *NeurIPS*, volume 30.

Zheng, X.; Qiao, X.; Cao, Y.; and Lau, R. W. H. 2019. Content-Aware Generative Modeling of Graphic Design Layouts. *ACM Trans. Graph.*, 38(4).

Zhong, X.; Tang, J.; and Yepes, A. J. 2019. PubLayNet: largest dataset ever for document layout analysis. *ArXiv*, abs/1908.07836.