A Parse-Then-Place Approach for Generating Graphic Layouts from Textual Descriptions

Jiawei Lin[1]*, Jiaqi Guo[2], Shizhao Sun[2], Weijiang Xu[2], Ting Liu[1],
Jian-Guang Lou[2], Dongmei Zhang[2]
[1]Xi'an Jiaotong University, [2]Microsoft Research Asia
kylelin@stu.xjtu.edu.cn, tingliu@mail.xjtu.edu.cn,
{jiaqiguo, shizsu, weijiangxu, jlou, dongmeiz}@microsoft.com

Abstract

Creating layouts is a fundamental step in graphic design. In this work, we propose to use text as the guidance to create graphic layouts, i.e., ***Text-to-Layout****, aiming to lower the design barriers. Text-to-Layout is a challenging task, because it needs to consider the implicit, combined, and incomplete layout constraints from text, each of which has not been studied in previous work. To address this, we present a two-stage approach, named* ***parse-then-place****. The approach introduces an intermediate representation (IR) between text and layout to represent diverse layout constraints. With IR, Text-to-Layout is decomposed into a parse stage and a place stage. The parse stage takes a textual description as input and generates an IR, in which the implicit constraints from the text are transformed into explicit ones. The place stage generates layouts based on the IR. To model combined and incomplete constraints, we use a Transformer-based layout generation model and carefully design a way to represent constraints and layouts as sequences. Besides, we adopt the pretrain-then-finetune strategy to boost the performance of the layout generation model with large-scale unlabeled layouts. To evaluate our approach, we construct two Text-to-Layout datasets and conduct experiments on them. Quantitative results, qualitative analysis, and user studies demonstrate our approach's effectiveness.*

1. Introduction

Graphic design is ubiquitous in our daily life. *Layout*, the sizes and positions of design elements, is fundamental to a graphic design. However, creating layouts is a complex task that requires design expertise and consumes much time. While we may not have design knowledge and be unfamiliar with design tools, we have a strong linguistic

*Work done during an internship at Microsoft Research Asia.

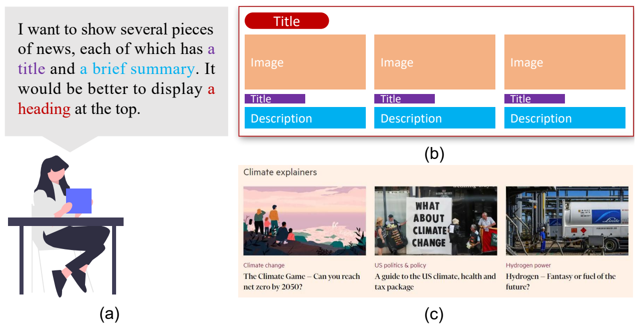

I want to show several pieces of news, each of which has a title and a brief summary. It would be better to display a heading at the top.
Title
Image
Image
Image
Title
Title
Title
Description
Description
Description
Climate explainers
WHAT ABOUT CLIMATE CHANGE
Climate change
The Climate Game – Can you reach net zero by 2050?
US politics & policy
A guide to the US climate, health and tax package
Hydrogen power
Hydrogen – Fantasy or fuel of the future?
(a)
(b)
(c)

Figure 1. An example of Text-to-Layout. (a) A user describes the desired layout in natural language. (b) A visually appealing layout is automatically generated. (c) The layout is a fundamental ingredient of the final graphic design.

ability to express our requirements. That is also what we are doing in the communication with designers. Thus, we propose to use *text* as the guidance to create graphic layouts, i.e., *Text-to-Layout* (see Figure 1). This enables people without expertise to participate in the design. Moreover, it helps professional designers create drafts more efficiently, thereby reducing their workloads. Besides, it also makes the discussion between users and designers much smoother.

While automatic layout generation has been intensively studied [23, 1, 9, 20], Text-to-Layout is rarely investigated and remains a challenging task due to its unique way to specify layout constraints. *(i) Implicit Constraint*. Textual description tends to be abstract and vague, and thus layout constraints from the text are often specified in an implicit and vague way. Consider the description in Figure 1. It describes the elements in a layout by their functional roles (e.g., news title and summary) rather than their primitive element types (e.g., text box). It also states a vague, subjective position constraint (e.g., heading at the top) and implicitly poses a hierarchy constraint (e.g., news should be organized in a list). This characteristic makes Text-to-Layout drastically different from other conditional layout generation tasks where constraints are explicitly specified [24, 21, 20]. *(ii) Combined Constraint*. Various types of layout constraints are often jointly specified in text. For example, the description in Figure 1 poses 4 different kinds of constraints in total, including element type constraint (e.g., news title), size constraint (e.g., brief summary), position (e.g., heading at the top) and hierarchy (e.g., all news arranged in a list). However, existing conditional layout generation approaches only tackle one certain kind of constraint at a time. Hence, how to model the combined constraints and create visually appealing layouts that satisfy all constraints simultaneously remains to be explored. *(iii) Incomplete Constraint*. Users tend not to describe all the elements in a layout, because doing so is extremely tedious. For instance, the description in Figure 1 does not specify the image in each piece of news, but the images are indispensable for engaging audiences' attention. Thus, it is necessary to auto-complete the omitted yet important elements. While previous work [9, 15] has studied the task of layout completion, it has not been jointly considered with other conditional layout generation tasks. In addition to the above challenges, Text-to-Layout also faces the serious problem of scarcity of labeled data. Unlike the Text-to-Image task that has billions of text-image pairs from the Internet [31, 30, 33], it is prohibitively expensive to collect a similarly sized dataset for Text-to-Layout.

To tackle the challenging Text-to-Layout, our intuition is three-fold. First, implicit layout constraints in a text could be transformed into explicit ones. Considering the description in Figure 1, the functional roles of elements can be transformed to corresponding primitive element type constraints. Since this transformation does not require any layout generation capability, we can take advantage of pretrained language models (PLM) that have achieved impressive natural language understanding performance even in a low-data regime [3, 29, 34]. In addition, generating layouts conditioned on explicit constraints is a well-studied setting in prior work [1, 20, 15]. We can learn from their successful experience to address the problem. Second, the state-of-the-art approach [15] has formulated conditional layout generation as a sequence-to-sequence transformation problem and demonstrated the superiority of representing constraints and layouts as sequences. This motivates us to take the same formulation and seek a way to represent combined and incomplete constraints in Text-to-Layout as sequences. Third, graphic designs are ubiquitous and their layouts are often abundantly available [2, 22]. Though these layouts do not have corresponding textual descriptions, their large quantity and rich layout patterns are highly valuable for learning to generate high-quality, diverse layouts, especially when only scarce labeled data is available.

Motivated by the intuitions, we propose a two-stage approach for Text-to-Layout, called *parse-then-place* (see Figure 2). The approach introduces an *intermediate representation* (IR) between text and layout to formally represent diverse layout constraints, such as element type, size, and hierarchy. By introducing IR, our approach decomposes Text-to-Layout into two stages: *parse* and *place*. The *parse* stage takes a textual description as input and outputs IR. Since IR is a representation of layout constraints specified in the text, we formulate the parse stage as a natural language understanding problem and finetune the T5 [29] PLM to map the text to IR. The *place* stage generates layouts according to the constraints stated in IR. Inspired by UniLayout [15], we use a Transformer-based layout generation model and carefully design an input-output sequence format to represent combined, incomplete constraints and layouts. In addition, owing to the two-stage design of our approach, we can leverage large-scale unlabeled layouts to pretrain the layout generation model and then finetune it with labeled data.

To evaluate our approach, we construct two Text-to-Layout datasets: *Web5K* and *RICO2.5K*. Web5K targets Web page layouts and contains 4,790 $\langle$`text`, `IR`, `layout`$\rangle$ samples, while RICO2.5K targets Android UI layouts and includes 2,412 samples. The quantitative and qualitative results on both datasets show that parse-then-place significantly outperforms baseline approaches in terms of perceptual quality and consistency. We also conduct a user study to evaluate our approach more comprehensively. Compared to the baseline approaches, users find that our generated layouts better match textual descriptions in 47.6% and 56.0% of trials, and have higher quality in 52.2% and 62.5% of trials in Web5K and RICO2.5K, respectively.

2. Related Work

Graphic layout generation. The automatic generation of aesthetic layouts has fueled growing interest. Early work in this field primarily studies unconditional layout generation using advanced generative models, such as Generative adversarial Network (GAN) [23] and Variational Autoencoder (VAE) [1, 16, 38]. To enable more practical usages, recent work explores conditional layout generation, where the goal is to generate layouts conforming to certain constraints. Li et al. [24] introduce a GAN-based approach to incorporate the meta attributes of elements, e.g., reading order, expected area, and aspect ratio, for layout generation. Jyothi et al. [17] present LayoutVAE, a two-stage VAE-based framework that generates layouts given a set of element types. Lee et al. [21] propose Neural Design Network, a three-stage VAE-based method that generates layouts based on a set of elements and their partial geometric relations. Gupta et al. [9] propose LayoutTransformer for layout completion. Given an initial layout with only one element, LayoutTransformer leverages self-attention to complete it autoregressively. Kong et al. [20] observe that LayoutTransformer's pre-defined generation order hinders it from per-

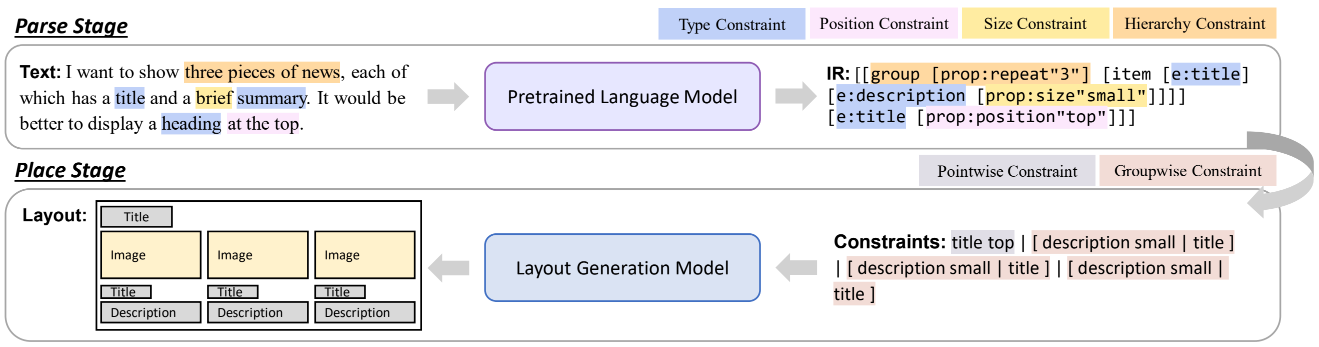

Parse Stage
Type Constraint
Position Constraint
Size Constraint
Hierarchy Constraint
Text: I want to show three pieces of news, each of which has a title and a brief summary. It would be better to display a heading at the top.
Pretrained Language Model
IR: [[group [prop:repeat"3"] [item [e:title] [e:description [prop:size"small"]]]] [e:title [prop:position"top"]]]
Place Stage
Pointwise Constraint
Groupwise Constraint
Layout:
Title
Image
Title
Description
Layout Generation Model
Constraints: title top | [description small | title] | [description small | title] | [description small | title]

Figure 2. An illustration of our approach. We decompose Text-to-Layout into a parse stage and a place stage. The parse stage maps a textual description into an intermediate representation (IR), in which implicit and vague constraints are transformed into explicit ones. The place stage generates visually pleasing layouts according to the combined constraints. Meanwhile, the place stage completes reasonable elements (the three *image*s in the generated layout) automatically.

forming tasks where the constraints disagree with the predefined order. To alleviate this issue, they report Bidirectional Layout Transformer to generate layouts in a non-autoregressive manner. The above work adopts customized model architectures and optimization methods for different conditional layout generation tasks. Jiang et al. [15] instead propose UniLayout, which formulates conditional layout generation as a sequence-to-sequence transformation problem and adopts the Transformer encoder-decoder architecture to address it. They further design an input sequence format to represent diverse constraints in a unified way so that the model can handle various tasks. Albeit effective, these methods cannot be directly adopted to Text-to-Layout. On the one hand, they all require explicit constraints as input, but the constraints from text are implicit and vague. On the other hand, they only tackle one certain kind of constraint at a time, but the constraints from text are often combined and incomplete, requiring an ability to take into consideration various kinds of constraints. To bridge this gap, our approach's parse stage transforms implicit constraints in the text to explicit ones, and the place stage extends UniLayout to enable layout generation conditioned on combined and incomplete constraints.

Text-to-Layout in other scenarios. There has been several attempts at using text as the guidance to create other types of layouts. House Plan Generative Model (HPGM) is a pioneering work on generating building layouts from textual descriptions [4]. Given a description, HPGM first parses it into a structural graph representation and then predicts room attributes according to the graph. Tan et al. [35] study scene layout generation from text and propose Text2Scene, an end-to-end approach that sequentially generates objects and their attributes. Radevski et al. [28] target the same problem and present SR-BERT. It is built upon BERT [6], and it generates layouts in an iterative and non-autoregressive manner. Hong et al. [12] introduce semantic layout generation from text and regard it as an intermediate step for Text-to-Image. Similarly, Huang and Canny [13] propose to generate sketch layouts from text. They use the layouts as bottleneck representations for Text-to-Sketch. These methods are highly customized for their targeted scenarios. Adapting them to graphic layouts is non-trivial and of inferior performance, as shown in our experiments.

3. Methodology

In this section, we elaborate on our parse-then-place approach for generating a layout y from a textual description x. As illustrated in Figure 2, our approach introduces an intermediate representation (IR) and decomposes the generation into the following two stages:

1. **Parse Stage**: Translate the textual description x into IR z. As IR is a formal representation of layout constraints, we formulate this stage as a natural language understanding problem and take advantage of pretrained language models (PLM) to address it.
2. **Place Stage**: Generate a layout y conditioned on IR z. We formulate this stage as a sequence-to-sequence transformation problem and use a Transformer-based layout generation model. To support combined and incomplete constraints, we carefully design an input-output sequence format to represent constraints and layouts. In addition, we adopt a pretrain-then-finetune strategy to improve the layout generation model with large-scale unlabeled layout data.

In what follows, we first introduce the IR and then elaborate on the parse and place stages, respectively.

3.1. Intermediate Representation

As a bridge between text and layout, IR should meet the following two requirements. *(i) Expressiveness*. IR should be expressive to represent diverse user constraints on layouts. Otherwise, the mapping from text to IR will incur undesired information loss, and consequently, the generated layouts will fail to satisfy user constraints. *(ii) For-*

#	Textual Description	Intermediate Representation
1	a news heading	`[e:title]`
2	a news heading at the top	`[e:title [prop:position"top"]]`
3	a brief news summary	`[e:description [prop:size"small"]]`
4	3 news pieces. each has a title and summary	`[group [prop:repeat"3"] [item[e:title][e:description]]]`

Table 1. Illustrative examples of intermediate representation.

malism. IR should have a good formalism so that existing natural language understanding techniques can exhibit accurate understanding performance. This is because previous studies [8, 11] show that the formalism of formal languages could significantly impact understanding performance.

To this end, we design an IR based on the above requirements. Table 1 shows 4 illustrative examples of our designed IR. In our preliminary study on Text-to-Layout,[1] we found that users tend to describe layouts by specifying the functional roles (Table 1 #1), positions (#2), sizes (#3), and hierarchies (#4) of elements. Hence, our IR currently supports these constraints[2]. In terms of formalism, the IR follows the hierarchical representation scheme [10], which has exhibited accurate understanding performance [32]. More details and examples of the IR can be found in the appendix.

3.2. Parse Stage

The parse stage aims to map a description x to IR z, so that implicit constraints in text are transformed into explicit constraints. We use T5 [29], a Transformer encoder-decoder based PLM to address this natural language understanding problem. While T5 is pretrained on text data, it has been shown to be effective at translating natural language to formal language even in a low-data regime [37, 34]. Specifically, the encoder takes a description $x = \{x_1, \cdots, x_n\}$ as input, where x_i is the i-th token in the description, and it outputs the contextualized representations $h^x = \{h^x_1, \cdots, h^x_n\}$. Then the decoder autoregressively predicts the tokens of IR $P_\theta(z_j|z_{<j}, x)$ by attending to previously generated tokens $z_{<j}$ (via self-attention) and the encoder outputs h^x (via cross-attention). During training, we fine-tune T5 parameters θ by minimizing the negative log-likelihood of labeled IR z given input text x:

$$\mathcal{L}_\theta = -\frac{1}{|\mathcal{D}|} \sum_{(x,z,y)\in\mathcal{D}} \sum_{j=1}^{|z|} \log P_\theta(z_j|z_{<j}, x). \quad (1)$$

3.3. Place Stage

The goal of the place stage is to generate a layout y according to the combined constraints specified in IR z. Inspired by UniLayout [15], we formulate this stage as a sequence-to-sequence transformation problem and adopt a Transformer encoder-decoder based layout generation model to solve it. Given an IR z, we first deterministically transform it into a constraint sequence $s = \pi(z)$. The model then takes the sequence s as input and generates a layout sequence y. The parameters ϕ of the model are trained to minimize the negative log-likelihood of the layout sequence y given constraint sequence s:

$$\mathcal{L}^{\text{FT}}_\phi = -\frac{1}{|\mathcal{D}|} \sum_{(x,z,y)\in\mathcal{D}} \sum_{j=1}^{|y|} \log P_\phi(y_j|y_{<j}, \pi(z)). \quad (2)$$

To support combined and incomplete constraints, we carefully design the constraint and layout sequences, as introduced below.

Constraint Sequence. To represent combined constraints, we serialize each constraint as a sub-sequence and concatenate all of them in a certain order.

First, we divide layout constraints into the following two categories, which have distinct characteristics and thus require different serialization strategies. *(i) Pointwise constraint* refers to the constraint specific to a single element. The element type, position and size constraints in Figure 2 all fall into this category. *(ii) Groupwise constraint* denotes the hierarchy between a group of elements. It includes containment (e.g., a toolbar with two icons and a textbox) and arrangement (e.g., three news lists) relationships between elements. It is worth noting that the layout constraints studied in prior work (see Section 2) are all within our consideration.

For pointwise constraint, we denote it using a predefined constraint token $k \in \Sigma$, where Σ is a vocabulary that varies with different constraints (e.g., the vocabulary for position constraint could be $\Sigma_{pos} = \{\text{left}, \cdots, \text{top}\}$). Multiple pointwise constraints on an element are represented as a concatenation of constraint tokens: $C^{po} = \{k_{type}\ k_{pos}\ k_{size}\}$, where k_{type}, k_{pos} and k_{size} are element type, position and size constraints, respectively. For example, we use $\underline{\text{image left large}}$ to express "a large image on the left". In terms of groupwise constraint, we take inspiration from recent work that represents structural knowledge bases (e.g., knowledge graph and table) as sequences [37, 25], and use brackets to denote groups C^{gp}. For instance, the hierarchy "a news piece with an image and a title at the top" can be represented as $\underline{\text{[image | title top]}}$. With these seri-

[1]We invited people without professional graphic design skills to describe a set of layouts and studied their descriptions' characteristics.

[2]Notably, it is easy to extend the IR for other types of constraints.

alization strategies, we transform the combined constraints into a sequence by sorting the sub-sequences in the alphabetic order of element types and concatenating them with a separator $|$:

$$s = \{C_1^{po}|\cdots|C_p^{po}|C_1^{gp}|\cdots|C_q^{gp}\}\,.$$

We provide some examples of constraint sequences in the appendix. Since IR is a formal language, it can be deterministically translated into corresponding constraint sequences.

Layout Sequence. To facilitate the model learning to complete omitted elements given incomplete constraints, we introduce a categorical attribute $a \in \{\text{complete}, \text{null}\}$ for each element to indicate whether it is auto-completed or not. It helps the model better distinguish the auto-completed elements. Each element e_i now has 6 attributes, including the newly-introduced attribute a_i, element type c_i, left coordinate l_i, top coordinate t_i, width w_i and height h_i. Following previous work [15, 9, 20], we represent an element as a sequence with 6 discrete tokens $e_i = \{a_i c_i l_i t_i w_i h_i\}$, in which the continuous attributes l_i, t_i, w_i and h_i are discretized into integers between $[0, n_{bins} - 1]$. Then, we represent a layout by sorting all the elements in the alphabetic order and concatenating all their tokens using a separator $|$:

$$y = \{a_1 c_1 l_1 t_1 w_1 h_1|\cdots|a_m c_m l_m t_m w_m h_m\}\,.$$

Exploiting Unlabeled Layouts. Learning to generate high-quality, diverse layouts from combined, incomplete constraints is challenging, especially when only a small volume of labeled data is available. But fortunately, graphic layouts are easily accessible in the wild. Though they do not have corresponding textual descriptions, their wealthy layout patterns are tremendously helpful for acquiring layout generation skills. Therefore, we attempt to pretrain the layout generation model with large-scale unlabeled layouts, and then finetune it with labeled data.

For pretraining, we curate a synthetic dataset $\mathcal{D}_s = \{(\hat{z}_i, y_i)\}_i^{|\mathcal{D}_s|}$ from unlabeled layouts, where IR $\hat{z}_i$ is synthesized from layout y_i, and we use it to pretrain the model:

$$\mathcal{L}_\phi^{\text{PT}} = -\frac{1}{|\mathcal{D}_s|}\sum_{(\hat{z},y)\in\mathcal{D}_s}\sum_{j=1}^{|y|}\log P_\phi(y_j|y_{<j}, \pi(\hat{z})). \tag{3}$$

This dataset is built upon the layout's structural nature, making it feasible to extract desired constraints from unlabeled layouts through some heuristic rules. For example, we can infer position constraints from the position property of elements in the layout source code. In addition, the hierarchical characteristics of source code also facilitate the extraction of hierarchy constraints. With these constraints, we can synthesize an IR from a layout and then build the synthetic dataset. The details of IR synthesis are given in the appendix.

After pretraining, we finetune the model with labeled data (see Equation 2) to close the distribution gap between synthetic and labeled data. There are two main reasons for this gap. First, the characteristics of elements that are commonly omitted by users cannot be perfectly described in the synthesized IR. Second, position and size constraints are somewhat subjective, so there is an inevitable gap between the constraints extracted by rules and human perception.

3.4. Inference

At inference time, our approach can generate multiple layouts for a textual description. The parse stage takes as input a description and generates an IR with the largest likelihood (*argmax sampling*). The place stage deterministically transforms the IR into a constraint sequence and generates multiple layouts via *Top-K sampling*, which samples the next token from the top k most probable choices [7].

4. Experiments

4.1. Setups

Datasets. We conduct experiments on two graphic layout datasets, including **RICO** [5] and **WebUI**. RICO is a public dataset of Android UI layouts without textual descriptions. As introduced in Section 3, our method uses $\langle$`text`, `IR`$\rangle$ labeled data during the parse stage. In the place stage, it uses unlabeled `layout` data for pretraining and $\langle$`IR`, `layout`$\rangle$ labeled data for finetuning. Thus, we labeled 2,412 $\langle$`text`, `IR`, `layout`$\rangle$ triplets (denoted as *RICO2.5K*) for the RICO dataset, and left the other 40k layouts in the dataset as unlabeled data (denoted as *RICO40K*). WebUI is a dataset of web UI layouts crawled from the Internet by ourselves. Similarly, we labeled 4,790 data (denoted as *Web5K*) and left 1.5 million layouts as unlabeled data (denoted as *Web1.5M*). Moreover, we split the labeled data (i.e., RICO2.5K and Web5K) into training, validation and test sets by ratios of 80%, 10% and 10%.

Specifically, to ensure the coverage and quality of the labeled datasets, we create them by following steps: 1) sampling from the original unlabeled set, 2) training annotators, 3) labeling the data by annotators, and 4) examining annotation quality by experts. Please see the appendix for examples of labeled data and details of the annotation process.

Baselines. While there is no existing work in our scenario (i.e., graphic layouts), there are strong methods in relevant scenarios (e.g., scenes). We adapt them as our baselines. **MockUp** [14] encodes a description with BERT [6] and adopts Transformer decoder to generate a layout autoregressively. **Text2Scene** [35] leverages an RNN-based text encoder to encode a description, and recursively predicts the next element in a layout and its corresponding bounding box by the proposed convolutional recurrent module. **SR-BERT** [28] first obtains a description's contextual rep-

	WebUI							
Method	FID ↓	Align. ↓	Overlap ↓	mIoU ↑	UM ↑	Type Cons. ↑	Pos & Size Cons. ↑	Hierarchy Cons. ↑
Mockup [14]	37.0123	0.0059	0.4348	0.1927	0.4299	0.6851	0.5508	-
Text2Scene [35]	27.1612	0.0042	0.4455	0.1899	0.4164	0.7515	0.5377	-
SR-BERT [28]	10.1640	0.0032	0.6501	0.2322	0.4127	0.8551	0.6423	-
Ours	**2.9592**	**0.0008**	**0.1380**	**0.6841**	**0.5080**	**0.8864**	**0.8086**	**0.4622**
Real data	-	0.0007	0.1343	-	-	-	-	-

	RICO							
Method	FID ↓	Align. ↓	Overlap ↓	mIoU ↑	UM ↑	Type Cons. ↑	Pos & Size Cons. ↑	Hierarchy Cons. ↑
Mockup [14]	29.5170	0.0096	0.5416	0.1868	0.2892	0.7548	0.5724	-
Text2Scene [35]	23.4324	0.0072	0.4558	0.1972	0.3512	0.8093	0.6061	-
SR-BERT [28]	13.1268	0.0055	0.6373	0.2927	0.3306	0.9295	0.6425	-
Ours	**4.9256**	**0.0015**	**0.2918**	**0.5267**	**0.3661**	**0.9539**	**0.7145**	**0.7841**
Real data	-	0.0029	0.2714	-	-	-	-	-

Table 2. Quantitative results. The *perceptual quality* of generated layouts is measured in columns 2-6. The *consistency* between the generated layout and the description is measured in columns 7-9, including Type Cons., Pos & Size Cons. and Hierarchy Cons.

resentations from BERT, and then generates a layout in an iterative and non-autoregressive manner. They all require the description and layout pairs as the training data.

Evaluation Metrics. We evaluate generation performance from two aspects. The first aspect is the **perceptual quality**, reflecting whether the generated layout looks aesthetically pleasing and diverse. Following previous work, we consider five metrics. *Fréchet Inception Distance (FID)* [18] measures the distance between the distribution of generated layouts and that of real layouts, which reflects both aesthetic quality and diversity. *Alignment (Align.)* [24] indicates whether elements in the generated layout are well-aligned. *Overlap* [24] measures the overlap area between two arbitrary elements in the generated layout. *Maximum IoU (mIoU)* [18] measures the maximum IoU between the elements in the generated layout and those in the real layout. Align., Overlap and mIoU mainly measure aesthetic quality. *Unique Match (UM)* [1] uses DocSim [27] to retrieve the most similar layout from the training set for each generated layout and then computes the ratio of the number of distinct retrieved layouts to the total number of generated layouts. It mainly indicates diversity.

The second aspect is the **consistency**, reflecting whether the generated layout matches the description. We measure it by the satisfaction rate of constraints [21]. Specifically, we subdivide the constraints into three groups based on the discussion in Section 3.1 and compute the satisfaction rate for each of them, denoted as *Type Consistency*, *Position & Size Consistency* and *Hierarchy Consistency*[3] respectively.

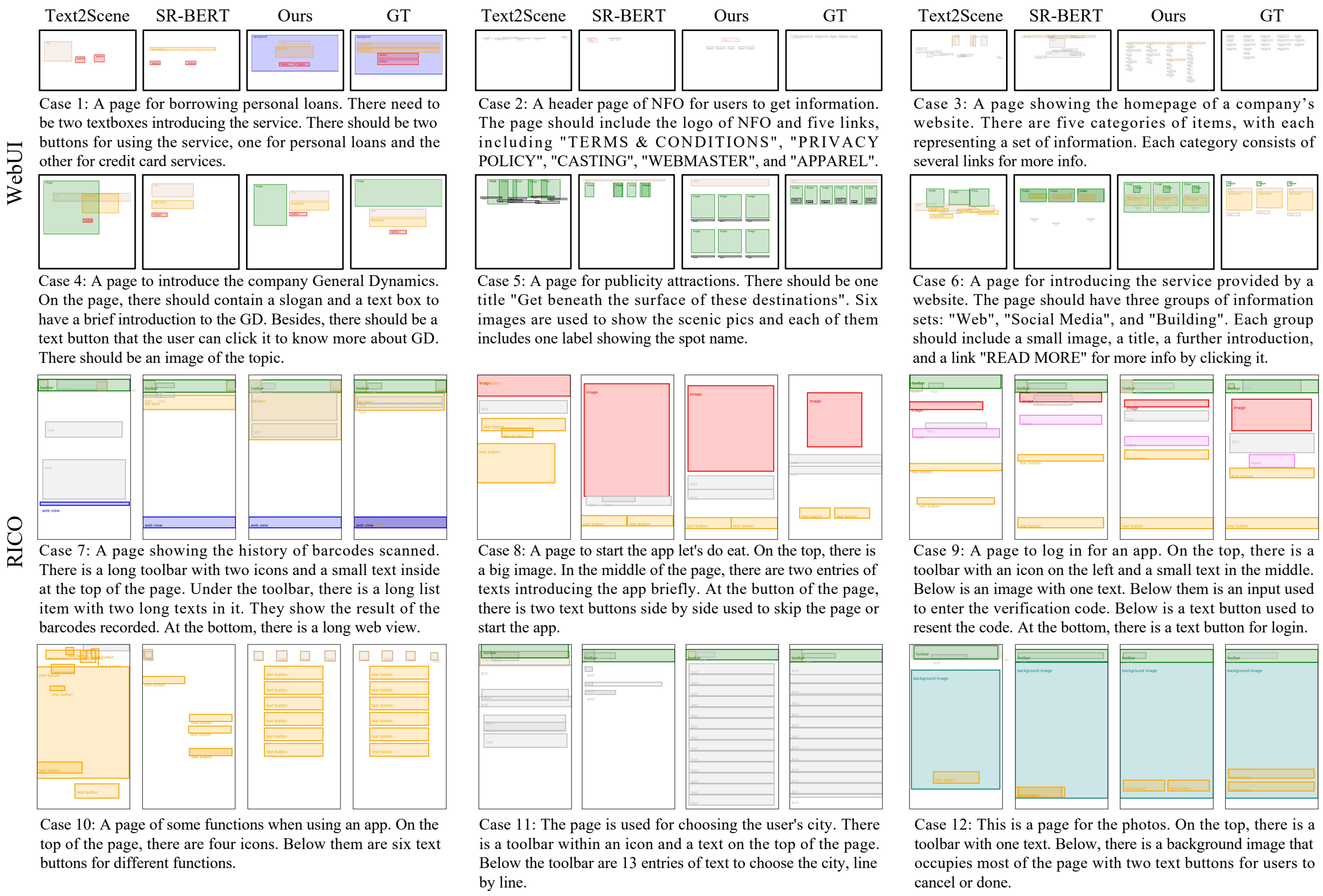
Implementation Details. Our approach is implemented with PyTorch [26] and Huggingface [36]. In the parse stage, we use T5-base. In the place stage, the layout generation model has 12 encoder layers and 12 decoder layers. The hidden dimension is set to 768, and the number of attention heads is set to 8. All the models are optimized with Adam optimizer [19] on NVIDIA V100 GPUs. The layouts in WebUI and RICO are proportionally scaled to 120×120 and 144×256 respectively. See the appendix for more details.

[3]Hierarchy Consistency is only computed in our method since hierarchies are hard to parse from the layouts generated by baseline methods.

4.2. Main Results

Quantitative Analysis. Table 2 shows the quantitative results. Our method significantly outperforms the baselines on all metrics, indicating that it can generate layouts that are more visually pleasing and consistent with the descriptions. Moreover, our method can generate layouts with similar or even better Align. and Overlap compared to real data.

Qualitative Analysis. Figure 3 and Figure 4 show qualitative results. The results demonstrate that the layouts from our approach are of high quality, while the layouts from baselines frequently contain misalignment, incorrect overlap, and weird spacing. Besides, the layouts from our approach are more consistent with the descriptions. For example, in case 2 of Figure 3, the baselines fail to generate five links, while our approach succeeds. Our approach can also complete omitted yet important elements. For example, in case 1 of Figure 3, it completes a background image, which is common on web pages. Figure 4 shows qualitative results for layout diversity, where each method generates four layouts for each description. The results indicate that our approach can generate diverse layouts with rich patterns and high quality, while other methods either suffer from unsatisfactory quality or homogeneous layout designs. For example, in case 1 of Figure 4, the eight groups are arranged in different ways by our approach. However, in SR-BERT, they are all arranged in a two-column manner with misalignment and incorrect overlap.



Case 1: A page for borrowing personal loans. There need to be two textboxes introducing the service. There should be two buttons for using the service, one for personal loans and the other for credit card services.

Case 2: A header page of NFO for users to get information. The page should include the logo of NFO and five links, including "TERMS & CONDITIONS", "PRIVACY POLICY", "CASTING", "WEBMASTER", and "APPAREL".

Case 3: A page showing the homepage of a company's website. There are five categories of items, with each representing a set of information. Each category consists of several links for more info.

Case 4: A page to introduce the company General Dynamics. On the page, there should contain a slogan and a text box to have a brief introduction to the GD. Besides, there should be a text button that the user can click it to know more about GD. There should be an image of the topic.

Case 5: A page for publicity attractions. There should be one title "Get beneath the surface of these destinations". Six images are used to show the scenic pics and each of them includes one label showing the spot name.

Case 6: A page for introducing the service provided by a website. The page should have three groups of information sets: "Web", "Social Media", and "Building". Each group should include a small image, a title, a further introduction, and a link "READ MORE" for more info by clicking it.

Case 7: A page showing the history of barcodes scanned. There is a long toolbar with two icons and a small text inside at the top of the page. Under the toolbar, there is a long list item with two long texts in it. They show the result of the barcodes recorded. At the bottom, there is a long web view.

Case 8: A page to start the app let's do eat. On the top, there is a big image. In the middle of the page, there are two entries of texts introducing the app briefly. At the button of the page, there is two text buttons side by side used to skip the page or start the app.

Case 9: A page to log in for an app. On the top, there is a toolbar with an icon on the left and a small text in the middle. Below is an image with one text. Below them is an input used to enter the verification code. Below is a text button used to resent the code. At the bottom, there is a text button for login.

Case 10: A page of some functions when using an app. On the top of the page, there are four icons. Below them are six text buttons for different functions.

Case 11: The page is used for choosing the user's city. There is a toolbar within an icon and a text on the top of the page. Below the toolbar are 13 entries of text to choose the city, line by line.

Case 12: This is a page for the photos. On the top, there is a toolbar with one text. Below, there is a background image that occupies most of the page with two text buttons for users to cancel or done.

Figure 3. Qualitative comparison with the best two baselines (zoom-in for better view). The real layout corresponding to the description is denoted as GT. The generated layout does not have to be the same as GT, as long as it is visually pleasing and matches the description.

Dataset	Method	Perceptual Quality ↑	Consistency ↑
WebUI	Text2Scene	0.226	0.250
	SR-BERT	0.252	0.274
	Ours	**0.522**	**0.476**
RICO	Text2Scene	0.170	0.200
	SR-BERT	0.205	0.240
	Ours	**0.625**	**0.560**

Table 3. Results of user study. For each model, we count how many people prefer the layouts it generates.

User Study. In the user study, we compare with the best two baselines (measured by quantitative metrics), SR-BERT and Text2Scene. For each method, we randomly sample 500 and 200 generated layouts on WebUI and RICO test sets, respectively. Then, we invite 7 participants to the study. Each participant is asked to evaluate 100 groups, each group containing a description and three layouts generated by different models. They need to answer two questions: 1) which layout has the best perceptual quality, and 2) which layout is the most consistent with the description. Table 3 shows the results. Our approach surpasses the baselines in both perceptual quality and consistency.

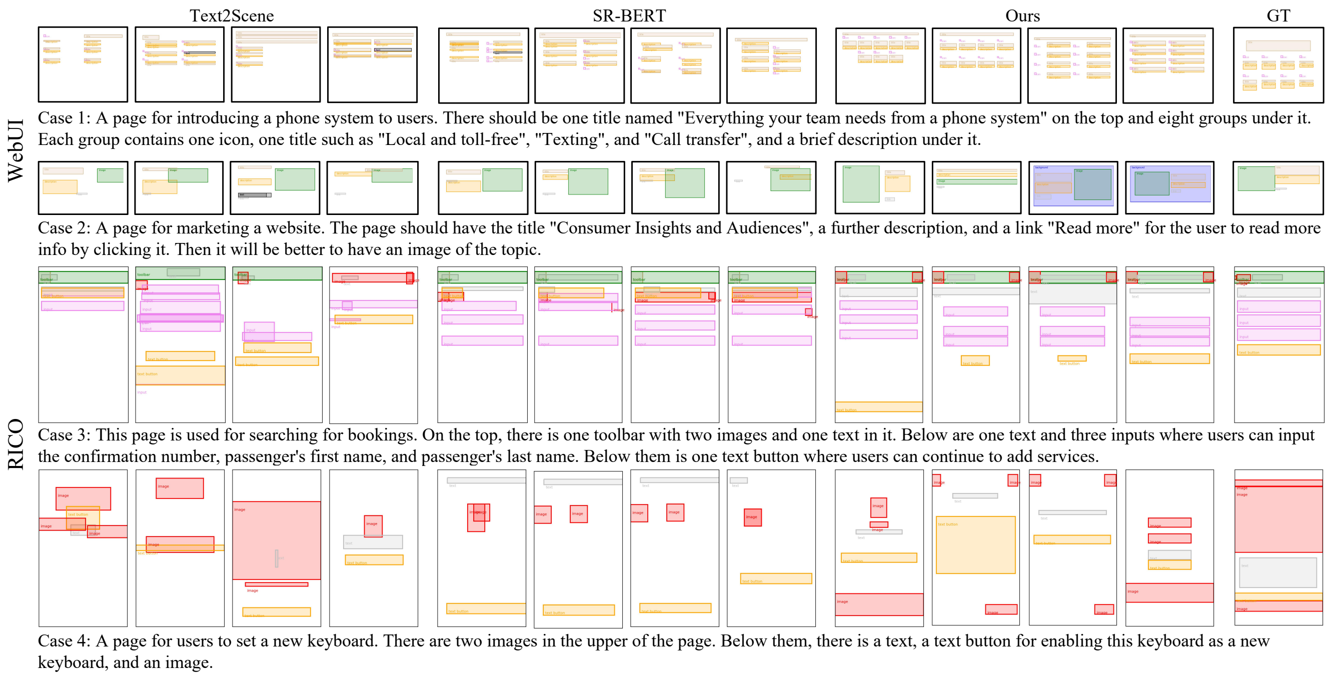
Dataset	Random Initialization	T5 Initialization
WebUI	35.5%	76.2%
RICO	9.9%	56.6%

Table 4. IR accuracy in the parse stage.

4.3. Ablation Studies

Our parse stage utilizes a pretrained language model. To understand its effect, we compare the performance of the network initialized by T5 pretrained weights and the same network initialized randomly. To avoid the influence of the place stage, we only evaluate the performance of the parse stage. We use IR accuracy as the metric, where a predicted IR is considered correct if all its constraints are the same as those of the golden IR. Table 4 shows the results. On both datasets, IR accuracy improves significantly with T5 pretrained weights initialization. This suggests that PLM offers great help in transforming implicit constraints in the text into explicit ones.

To study the effect of unlabeled data used in the place stage, we conduct experiments on various volumes of unlabeled layouts: 0, 50K, 250K, 500K, 1M and 1.5M (ours).



Case 1: A page for introducing a phone system to users. There should be one title named "Everything your team needs from a phone system" on the top and eight groups under it. Each group contains one icon, one title such as "Local and toll-free", "Texting", and "Call transfer", and a brief description under it.

Case 2: A page for marketing a website. The page should have the title "Consumer Insights and Audiences", a further description, and a link "Read more" for the user to read more info by clicking it. Then it will be better to have an image of the topic.

Case 3: This page is used for searching for bookings. On the top, there is one toolbar with two images and one text in it. Below are one text and three inputs where users can input the confirmation number, passenger's first name, and passenger's last name. Below them is one text button where users can continue to add services.

Case 4: A page for users to set a new keyboard. There are two images in the upper of the page. Below them, there is a text, a text button for enabling this keyboard as a new keyboard, and an image.

Figure 4. Qualitative comparison of layout diversity with the best two baselines. We show four layouts for each description.

	FID ↓	Align. ↓	Overlap ↓	mIoU ↑	UM ↑	Type Cons. ↑	Pos & Size Cons. ↑	Hierarchy Cons. ↑
0 (*w/o pretrain*)	3.8141	0.0010	0.2453	0.5773	0.6304	0.8727	0.7737	0.4447
50K	3.3341	0.0009	0.1792	0.6623	0.5191	0.8863	0.8114	0.4155
250K	3.2846	0.0009	0.1597	0.6783	0.5055	0.8844	0.8229	0.4480
500K	3.1259	0.0009	0.1400	0.6843	0.4998	0.8780	0.8086	0.4291
1M	2.9408	0.0008	0.1504	0.6836	0.4776	0.8837	0.8103	0.4339
1.5M (Ours)	2.9592	0.0008	0.1380	0.6841	0.5080	0.8864	0.8086	0.4622
w/o two stages	4.3163	0.0010	0.3010	0.3168	0.5236	0.8620	0.7691	0.4350
w/ golden IR	2.6555	0.0008	0.1301	0.6866	0.5047	0.9532	0.8286	0.5058

Table 5. Ablation studies on WebUI.

Table 5 shows the results. Without using any unlabeled layout, our approach already outperforms all baselines (see Table 2) on every metric, suggesting that our overall algorithm design is very effective. When more unlabeled layouts are used, the performance is further boosted, especially on perceptual quality metrics (such as FID, Overlap and mIoU), indicating that the rich patterns of unlabeled layouts indeed improve generation skills.

Moreover, since we represent the layouts as sequences, it is feasible to train an end-to-end model using the description and layout pairs. Specifically, we finetune T5 on Web5K (denoted as *w/o two stages*). The results show that the end-to-end variant performs worse than our method under the same condition (w/o pretrain), again confirming the effectiveness of our method. Finally, we input the golden IR into the place stage (denoted as *w/ golden IR*) to study the effect of the parse stage on the overall approach performance. We see that almost all metrics are significantly improved in this setting. This leads us to believe that utilizing more powerful PLM can further boost the performance of our approach.

5. Conclusion

In this work, we propose to use text as guidance to create graphic layouts, i.e., Text-to-Layout. To tackle this challenging task, we present a two-stage approach, parse-then-place. It introduces an intermediate representation between text and layout to decompose the problem into a parse and place stage. Experiments demonstrate the effectiveness of our approach. However, our approach still has some limitations. First, it does not perform as well on some high-level descriptions (e.g., a layout exhibiting products) as on detailed descriptions in the datasets. Second, it currently does not support arbitrarily nested hierarchy constraints (e.g., a modal with three list items, each containing two texts). In the future, we plan to generalize our approach to more types of graphic design. We will also investigate how to extend our intermediate representation to support user constraints in other modalities, e.g., images.

References

[1] Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13642–13652, 2021. 1, 2, 6

[2] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Aguera y Arcas. Uibert: Learning generic multimodal representations for ui understanding, 2021. 2

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

[4] Qi Chen, Qi Wu, Rui Tang, Yuhan Wang, Shuai Wang, and Mingkui Tan. Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only, 2020. 3

[5] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854, 2017. 5

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 3, 5

[7] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. 5

[8] Jiaqi Guo, Qian Liu, Jian-Guang Lou, Zhenwen Li, Xueqing Liu, Tao Xie, and Ting Liu. Benchmarking meaning representations in neural semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1540, Online, Nov. 2020. Association for Computational Linguistics. 4

[9] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014, 2021. 1, 2, 5

[10] Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. 4

[11] Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. Unlocking compositional generalization in pre-trained models using intermediate representations. *arXiv preprint arXiv:2104.07478*, 2021. 4

[12] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[13] Forrest Huang and John F. Canny. Sketchforme: Composing sketched scenes from text descriptions for interactive applications. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 209–220, New York, NY, USA, 2019. Association for Computing Machinery. 3

[14] Forrest Huang, Gang Li, Xin Zhou, John F Canny, and Yang Li. Creating user interface mock-ups from high-level text descriptions with deep-learning models. *arXiv preprint arXiv:2110.07775*, 2021. 5, 6

[15] Zhaoyun Jiang, Huayu Deng, Zhongkai Wu, Jiaqi Guo, Shizhao Sun, Vuksan Mijovic, Zijiang Yang, Jian-Guang Lou, and Dongmei Zhang. Unilayout: Taming unified sequence-to-sequence transformers for graphic layout generation. *arXiv preprint arXiv:2208.08037*, 2022. 2, 3, 4, 5

[16] Zhaoyun Jiang, Shizhao Sun, Jihua Zhu, Jian-Guang Lou, and Dongmei Zhang. Coarse-to-fine generative modeling for graphic layouts. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):1096–1103, Jun. 2022. 2

[17] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9895–9904, 2019. 2

[18] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 88–96, 2021. 6

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[20] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. Blt: Bidirectional layout transformer for controllable layout generation. *arXiv preprint arXiv:2112.05112*, 2021. 1, 2, 5

[21] 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*, pages 491–506. Springer, 2020. 2, 6

[22] Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. *arXiv preprint arXiv:2210.03347*, 2022. 2

[23] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with

wireframe discriminators. *arXiv preprint arXiv:1901.06767*, 2019. 1, 2

[24] Jianan Li, Jimei Yang, Jianming Zhang, Chang Liu, Christina Wang, and Tingfa Xu. Attribute-conditioned layout gan for automatic graphic design. *IEEE Transactions on Visualization and Computer Graphics*, 27(10):4039–4048, 2020. 2, 6

[25] Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. Leveraging slot descriptions for zero-shot cross-domain dialogue StateTracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5640–5648, Online, June 2021. Association for Computational Linguistics. 4

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[27] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. Read: Recursive autoencoders for document layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 544–545, 2020. 6

[28] Gorjan Radevski, Guillem Collell, Marie-Francine Moens, and Tinne Tuytelaars. Decoding language spatial relations to 2D spatial arrangements. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4549–4560, Online, Nov. 2020. Association for Computational Linguistics. 3, 5, 6

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020. 2, 4

[30] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2

[31] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 2

[32] Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. In *Proceedings of The Web Conference 2020*, WWW '20, page 2962–2968, New York, NY, USA, 2020. Association for Computing Machinery. 4

[33] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 2

[34] Nathan Schucher, Siva Reddy, and Harm de Vries. The power of prompt tuning for low-resource semantic parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 148–156, Dublin, Ireland, May 2022. Association for Computational Linguistics. 2, 4

[35] Fuwen Tan, Song Feng, and Vicente Ordonez. Text2scene: Generating compositional scenes from textual descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3, 5, 6

[36] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. 6

[37] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *EMNLP*, 2022. 4

[38] Kota Yamaguchi. Canvasvae: Learning to generate vector graphic documents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5481–5489, October 2021. 2