LayoutDM: Transformer-based Diffusion Model for Layout Generation

Shang Chai, Liansheng Zhuang*
University of Science and Technology of China
chaishang@mail.ustc.edu.cn, lszhuang@ustc.edu.cn

Fengying Yan
Tianjin University
fengying@tju.edu.cn

Abstract

Automatic layout generation that can synthesize high-quality layouts is an important tool for graphic design in many applications. Though existing methods based on generative models such as Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs) have progressed, they still leave much room for improving the quality and diversity of the results. Inspired by the recent success of diffusion models in generating high-quality images, this paper explores their potential for conditional layout generation and proposes Transformer-based Layout Diffusion Model (LayoutDM) by instantiating the conditional denoising diffusion probabilistic model (DDPM) with a purely transformer-based architecture. Instead of using convolutional neural networks, a transformer-based conditional Layout Denoiser is proposed to learn the reverse diffusion process to generate samples from noised layout data. Benefitting from both transformer and DDPM, our LayoutDM is of desired properties such as high-quality generation, strong sample diversity, faithful distribution coverage, and stationary training in comparison to GANs and VAEs. Quantitative and qualitative experimental results show that our method outperforms state-of-the-art generative models in terms of quality and diversity.

1. Introduction

Layouts, *i.e.* the arrangement of the elements to be displayed in a design, play a critical role in many applications from magazine pages to advertising posters to application interfaces. A good layout guides viewers' reading order and draws their attention to important information. The semantic relationships of elements, the reading order, canvas space allocation and aesthetic principles must be carefully decided in the layout design process. However, manually arranging design elements to meet aesthetic goals and user-specified constraints is time-consuming. To aid the design of graphic layouts, the task of layout generation aims to generate design layouts given a set of design components with user-specified attributes. Though meaningful attempts are made [1, 10, 11, 15, 18, 21, 23–25, 33, 44–46], it is still challenging to generate realistic and complex layouts, because many factors need to be taken into consideration, such as design elements, their attributes, and their relationships to other elements.

Over the past few years, generative models such as Generative Adversarial Networks (GANs) [9] and Variational Auto-Encoders (VAEs) [20] have gained much attention in layout generation, as they have shown a great promise in terms of faithfully learning a given data distribution and sampling from it. GANs model the sampling procedure of a complex distribution that is learned in an adversarial manner, while VAEs seek to learn a model that assigns a high likelihood to the observed data samples. Though having shown impressive success in generating high-quality layouts, these models have some limitations of their own. GANs are known for potentially unstable training and less distribution coverage due to their adversarial training nature [4, 5, 27], so they are inferior to state-of-the-art likelihood-based models (such as VAEs) in terms of diversity [28, 29, 34]. VAEs can capture more diversity and are typically easier to scale and train than GANs, but still fall short in terms of visual sample quality and sampling efficiency [22].

Recently, diffusion models such as denoising diffusion probabilistic model (DDPM) [14] have emerged as a powerful class of generative models, capable of producing high-quality images comparable to those of GANs. Importantly, they additionally offer desirable properties such as strong sample diversity, faithful distribution coverage, a stationary training objective, and easy scalability. This implies that diffusion models are well suited for learning models of complex and diverse data, which also motivates us to explore the potential of diffusion-based generative models for graphic layout generation.

Though diffusion models have shown splendid performance in high-fidelity image generation [8, 14, 35, 39, 41], it is still a sparsely explored area and provides unique challenges to develop diffusion-based generative models for

*Corresponding author.

layout generation. First, diffusion models often use convolutional neural networks such as U-Net [36] to learn the reverse process to construct desired data samples from the noise. However, a layout is a non-sequential data structure consisting of varying length samples with discrete (classes) and continuous (coordinates) elements simultaneously, instead of pixels laid on a regular lattice. Obviously, convolutional neural networks are not suitable for layout denoising, which prevents diffusion models from being directly applied to layout generation. Second, the placement and sizing of a given element depend not only on its attributes (such as category label) but also on its relationship to other elements. How to incorporate the attributes knowledge and model the elements' relationship in diffusion models is still an open problem. Since diffusion models are general frameworks, they leave room for adapting the underlying neural architectures to exploit the properties of the data.

Inspired by the above insights, by instantiating the conditional denoising diffusion probabilistic model (DDPM) with a transformer architecture, this paper proposes Transformer-based Layout Diffusion Model (*i.e.*, LayoutDM) for conditional layout generation given a set of elements with user-specified attributes. The key idea is to use a purely transformer-based architecture instead of the commonly used convolutional neural networks to learn the reverse diffusion process from noised layout data. Benefitting from the self-attention mechanism in transformer layers, LayoutDM can efficiently capture high-level relationship information between elements, and predict the noise at each time step from the noised layout data. Moreover, the attention mechanism also helps model another aspect of the data - namely a varying and large number of elements. Finally, to generate layouts with desired attributes, LayoutDM designs a conditional Layout Denoiser (cLayoutDenoiser) based on a transformer architecture to learn the reverse diffusion process conditioned on the input attributes. Different from previous transformer models in the context of NLP or video, cLayoutDenoiser omits the positional encoding which indicates the element order in the sequence, as we do not consider the order of designed elements on a canvas in our setting. In comparison with current layout generation approaches (such as GANs and VAEs), our LayoutDM offers several desired properties such as high-quality generation, better diversity, faithful distribution coverage, a stationary training objective, and easy scalability. Extensive experiments on five public datasets show that LayoutDM outperforms state-of-the-art methods in different tasks.

In summary, our main contributions are as follows:

- This paper proposes a novel LayoutDM to generate high-quality design layouts for a set of elements with user-specified attributes. Compared with existing methods, LayoutDM is of desired properties such as high-quality generation, better diversity, faithful distribution coverage, and stationary training. To our best knowledge, LayoutDM is the first attempt to explore the potential of diffusion model for graphic layout generation.
- This paper explores a new class of diffusion models by replacing the commonly-used U-Net backbone with a transformer, and designs a novel cLayoutDenoiser to reverse the diffusion process from noised layout data and better capture the relationship of elements.
- Extensive experiments demonstrate that our method outperforms state-of-the-art models in terms of visual perceptual quality and diversity on five diverse layout datasets.

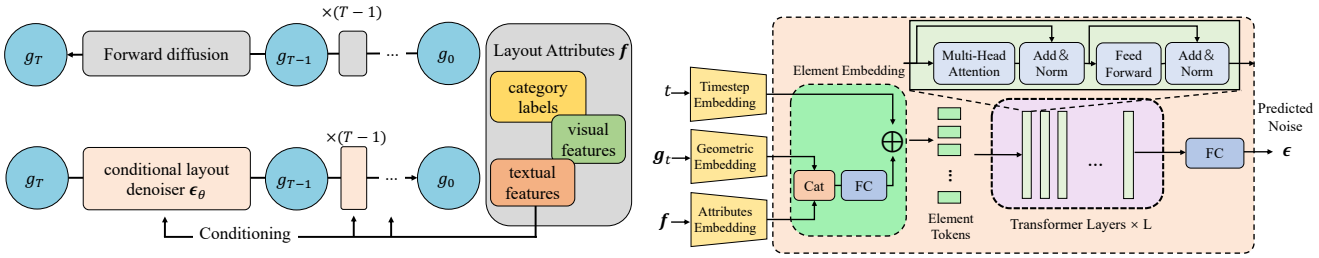
2. Related Work

2.1. Layout Generation

Automatic layout generation has been widely studied for a long time. Early approaches to layout generation [30, 31] embed design rules into manually-defined energy functions. In recent years, generative model based methods are increasingly progressed. LayoutGAN [24] and LayoutVAE [17] are the first attempts to utilize GAN and VAE to generate graphic and scene layouts. NDN [23] represents the relative positional relationship of elements as a graph and uses a graph neural network based conditional VAE to generate graphic layouts. READ [33] uses heuristics to determine the relationships between elements and trains a Recursive Neural Network (RNN) [37, 38] based VAE to learn the layout distribution. CanvasVAE [45] generates vector graphic documents which contain structured information about canvas and elements. VTN [1] and Coarse2fine [16] deploy self-attention based VAEs to generate graphic layouts, making progress in diversity and perceptual quality. LayoutTransformer [11] and BLT [21] define layouts as discrete sequences and exploit the efficiency of transformer and bidirectional-transformer in structured sequence generation. LayoutNet [46], TextLogo3K [44] and ICVT [6] propose conditional layout generative models which can utilize additional attributes about design elements or entire layouts to aid layout generation in different application scenarios. LayoutGAN++ [18] designs a transformer-based generator and discriminator and generates graphic layouts conditioned on the given element category labels.

2.2. Diffusion Model

Diffusion models [14, 40] have proved its capability of generating high-quality and diverse samples [8, 14, 35] and have recently achieved state-of-the-art results on several benchmark generation tasks [8]. The diffusion model uses diffusion processes to model the generation and defines the sampling of data as the process of gradually denoising from complete Gaussian noise. The forward process gradually



(a) LayoutDM architecture

(b) Architecture of conditional layout denoiser $\boldsymbol{\epsilon}_\theta$

Figure 1. a) Architecture of LayoutDM. It consists of a forward diffusion process and a reverse process modeled by a conditional layout denoiser $\boldsymbol{\epsilon}_\theta$. b) Architecture of our transformer-based conditional layout denoiser, cLayoutDenoiser. cLayoutDenoiser predicts the added noise conditioned on the layout attributes $\boldsymbol{f}$ and time step t.

adds Gaussian noise to the data from a predefined noise schedule until time step T. The reverse process uses a neural backbone often implemented as a U-Net [8, 14, 36, 42] to parameterize the conditional distribution $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. In this work, we instantiate a conditional diffusion model to achieve conditional layout generation.

3. Our Method

3.1. Layout Representation

In our model, each layout consists of a set of elements, and each element is described by both geometric parameters (*i.e.* location and size) and its attribute (*e.g.* category label or textual features). Formally, a layout $\boldsymbol{l}$ is denoted as a flattened sequence of integer indices:

$$\boldsymbol{l} = (g_1, f_1, g_2, f_2, \cdots, g_i, f_i, \cdots, g_N, f_N),$$

where N is the number of elements in the layout. $g_i = [x_i, y_i, w_i, h_i]$ is a vector that presents the geometric parameters (center coordinates and size) of the i-th element in the layout. f_i is the attributes of i-th element which might be category label or textual features. For the sake of convenience, the sequence $\boldsymbol{g} = (g_1, g_2, \cdots, g_i \cdots, g_N)$ is named layout geometric parameters, the sequence $\boldsymbol{f} = (f_1, f_2, \cdots, f_i \cdots, f_N)$ is named layout attributes. Note here that, the elements in a layout are unordered, so swapping the items in sequence $\boldsymbol{g}$ and $\boldsymbol{f}$ does not affect the meaning of the sequences. We normalize the geometric parameters, *i.e.*, $[x_i, y_i, w_i, h_i]$, $i = 1, 2, \cdots, N$, to the interval $[-1, 1]$. In this way, layouts have a uniform structured representation.

3.2. The LayoutDM Architecture

Fig. 1a illustrates the architecture of LayoutDM. From a high-level perspective, our LayoutDM is an instance of conditional denoising diffusion probabilistic model (DDPM) with a transformer architecture suitable for layout data. DDPM learns to model the Markov transition from simple distribution to layout data distribution and generates diverse samples through sequential stochastic transitions. To generate desired layouts, LayoutDM uses the input attributes to guide the generative process in DDPM. We follow the method described in Classifier-Free Diffusion Guidance [13] to realize conditional DDPM, and set the guidance strength w to zero for simplicity.

Specifically, let $q(\boldsymbol{g}_0|\boldsymbol{f})$ be the unknown conditional data distribution, where $\boldsymbol{g}_0$ is the geometric parameters of real layouts, and $\boldsymbol{f}$ is the layout attributes. LayoutDM models the conditional distribution $q(\boldsymbol{g}_0|\boldsymbol{f})$ by two processes: a forward diffusion process and a reverse denoising diffusion process. First, LayoutDM defines the forward diffusion process $q(\boldsymbol{g}_t|\boldsymbol{g}_{t-1})$ which maps layout data to noise by gradually adding Gaussian noise at each time step t:

$$q(\boldsymbol{g}_t|\boldsymbol{g}_{t-1}) = \mathcal{N}(\boldsymbol{g}_t; \sqrt{1-\beta_t}\boldsymbol{g}_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

where $\{\beta_t\}_{t=1}^T$ are forward process variances.

Then, LayoutDM defines the conditional reverse diffusion process $p(\boldsymbol{g}_{t-1}|\boldsymbol{g}_t, \boldsymbol{f})$ which performs iterative denoising from pure Gaussian noise to generate high-quality layouts conditioned on layout attributes $\boldsymbol{f}$:

$$p_\theta(\boldsymbol{g}_{t-1}|\boldsymbol{g}_t, \boldsymbol{f}) = \mathcal{N}(\boldsymbol{g}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f}), \sigma_t^2\mathbf{I}) \tag{2}$$

where σ_t is the constant variance following [14], $\boldsymbol{\mu}_\theta$ is the mean of the Gaussian distribution computed by a neural network, and θ is the parameters of the network. As shown in Ho *et al.* [14], we can reparameterize the mean to make the neural network learn the added noise at time step t instead. In this way, $\boldsymbol{\mu}_\theta$ can be reparameterized as follows:

$$\boldsymbol{\mu}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f}) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{g}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f})) \tag{3}$$

where t is the time step, $\{\beta_t\}_{t=1}^T$ are forward process variances, $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. $\boldsymbol{\epsilon}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f})$ is the neural network to predict the added noise for layout geometric parameters conditioned on elements' attributes at time step t. We also call the neural network $\boldsymbol{\epsilon}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f})$ as *conditional layout denoiser* (cLayoutDenoiser).

3.3. Conditional Layout Denoiser

The inputs to cLayoutDenoiser are layout geometric parameters $\boldsymbol{g}_t$, layout attributes $\boldsymbol{f}$ and time step t. To deal with sequences data, the conditional layout denoiser $\boldsymbol{\epsilon}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f})$ employs a purely transformer-based architecture instead of convolutional neural networks as its backbone. The architecture of cLayoutDenoiser is illustrated in Fig. 1b. Benefitting from the transformer architecture, cLayoutDenoiser can deal with the sequence with various lengths, and capture the relationships among elements. Moreover, cLayoutDenoiser adds "attributes embeddings" to the input "geometric embeddings", so as to guide the reverse diffusion process at each time step t. Formally, the architecture of cLayoutDenoiser can be described as follow:

$$\boldsymbol{h}_f = \mathrm{AttributesEmbedding}(\boldsymbol{f}) \tag{4}$$
$$\boldsymbol{h}_g = \mathrm{GeometricEmbedding}(\boldsymbol{g}_t) \tag{5}$$
$$\mathbf{E} = \mathrm{ElementEmbedding}(\boldsymbol{h}_f, \boldsymbol{h}_g, \mathrm{TE}(t)) \tag{6}$$
$$\mathbf{E}' = \mathrm{TransformerLayers}(\mathbf{E}) \tag{7}$$
$$\boldsymbol{\epsilon} = \mathrm{FC}(\mathbf{E}') \tag{8}$$

where $\boldsymbol{f}$ is the layout attributes, $\boldsymbol{g}_t$ is the noised layout geometric parameters, $\boldsymbol{h}_f$ and $\boldsymbol{h}_g$ are their hidden representations. $\mathbf{E}$ is the element tokens computed by element embedding module, $\mathbf{E}'$ is intermediate feature. $\mathrm{TE}(t)$ denotes the timestep embedding, and $\boldsymbol{\epsilon}$ is the predicted noise. Note here that, since the element order in the sequence makes no sense in our setting, cLayoutDenoiser omits the positional encoding, which is different from existing transformers in the context of NLP or video.

Geometric, attributes and timestep embedding. Three embedding modules are used to learn meaningful representations for noised layout geometric parameters $\boldsymbol{g}_t$, layout attributes $\boldsymbol{f}$ and time step t. Geometric embedding projects layout geometric parameters to a specific dimension, aiming to find a more efficient feature space than the original coordinate space. Feature embedding learns the continuous features of discrete element attributes by embedding them into a specific dimension. Following [14], we condition cLayoutTransformer on time step t by adding a sinusoidal time embedding $\mathrm{TE}(t)$ to make the network aware at which time step it is operating.

Element embedding. Element embedding module calculates the element tokens used as the input of transformer layers. Element tokens should contain geometric, attributes and time step information, so that transformer can efficiently capture the relationship information between elements conditioned on the time step t and layout attributes $\boldsymbol{f}$. We concatenate the geometric embedding and attributes embedding, and then use a fully-connected layer to fuse element representation. We further perform an element-wise plus operation with timestep embedding on the results to finally obtain the element tokens.

Algorithm 1: Training LayoutDM

Require: conditional layout denoiser $\boldsymbol{\epsilon}_\theta$
repeat
  Sample $(\boldsymbol{g}_0, \boldsymbol{f}) \sim q_{data}$;
  $t \sim \mathrm{Uniform}(\{1, \cdots, T\})$;
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\boldsymbol{g}_t = \sqrt{\overline{\alpha}_t}\boldsymbol{g}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}$;
  Take gradient descent step on
    $\nabla_\theta \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f})\|^2$;
until *converged*;

Algorithm 2: Sampling

Input: layout attributes $\boldsymbol{f}$
Output: layout geometric parameters $\boldsymbol{g}_0$
$\boldsymbol{g}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
for $t = T, \cdots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$;
  $\boldsymbol{g}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{g}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f})) + \sigma_t \mathbf{z}$
end
return $\boldsymbol{g}_0$

Transformer layers. Generating an effective layout requires understanding the relationships between layout elements. Self-attention mechanism in Transformer [43] has proven effective in capturing high-level relationships between lots of elements in layout generation [1, 11, 15]. In this paper, we adopt multihead attention mechanism to capture relationship information between elements from element tokens. We stack multiple transformer layers (8 in our model) to enable cLayoutDenoiser to capture relationships between layout elements from the element tokens. Positional encoding is omitted because of the unordered nature of elements.

$$\hat{\mathbf{E}} = \mathrm{LayerNorm}(\mathbf{E}^{l-1} + \mathbf{Head}(\boldsymbol{e}_1^{l-1}, \cdots, \boldsymbol{e}_N^{l-1})) \tag{9}$$
$$\mathbf{E}^l = \mathrm{LayerNorm}(\hat{\mathbf{E}} + \mathrm{FFN}(\hat{\mathbf{E}})) \tag{10}$$

where $l = 1, \cdots, L$ denotes the layer index, $\mathbf{Head}$, LayoutNorm and FFN denote multi-head attention layer, Layer Normalization [3] and fully connected feed-forward network. $\mathbf{E}^{l-1} = (\boldsymbol{e}_1^{l-1}, \cdots, \boldsymbol{e}_N^{l-1})$ are the intermediate element tokens used as the input of l-th transformer layer.

3.4. Training and Inference

Following denoising diffusion probabilistic model [14], we optimize random terms L_t which are the KL divergences between $p_\theta(\boldsymbol{g}_{t-1}|\boldsymbol{g}_t, \boldsymbol{f})$ and forward process posteriors. After simplifying the objective function following the method in [14], the final loss function is as follows:

$$\begin{aligned} L_{simple}(\theta) &= \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{g}_t, t, \boldsymbol{f})\|^2 \\ &= \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\boldsymbol{g}_0 + \sqrt{(1-\overline{\alpha}_t)}\boldsymbol{\epsilon}, t, \boldsymbol{f})\|^2 \end{aligned} \tag{11}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\boldsymbol{\epsilon}_\theta$ is our conditional layout denoiser, $\boldsymbol{g}_t \sim \mathcal{N}(\boldsymbol{g}_t; \sqrt{\overline{\alpha}_t}\boldsymbol{g}_0, (1-\overline{\alpha}_t)\mathbf{I})$ is computed using the property of Gaussian distributions and $\boldsymbol{g}_0$ is the real layout geometric parameters. $\{\beta_t\}_{t=1}^T$ are forward process variances, $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{s-1}^{t} \alpha_s$.

The training and sampling algorithm of LayoutDM are illustrated in Algorithm 1 and Algorithm 2 respectively.

4. Experiments

4.1. Experimental Settings

Datasets. We evaluate our method on five public datasets of layouts for documents, natural scenes, magazines, text logos and mobile phone UIs. **Rico** [7] is a dataset of user interface designs for mobile applications, containing 72,219 user interfaces from 9,772 Android apps. **PublayNet** [47] contains 330K samples of machine-annotated scientific documents crawled from the Internet with five categories (text, title, figure, list, table). **Magazine** [46] is a magazine layout dataset that covers a wide range of magazine categories. It consists of semantic layout annotations. **COCO** [26] is a large-scale labeled image dataset that contains images of natural scenes with instance segmentation. **TextLogo3K** [44] is a recently released dataset of text logos. It consists of 3,470 text logo images with manually annotated bounding boxes and pixel-level masks for each character.

Evaluation metrics. To validate the effectiveness of LayoutDM, we employ four metrics in the literature to measure the perceptual quality of layouts. To be a fair comparison, we follow the guidance in [18] to evaluate these metrics. Specifically, **FID** [12] measures the distribution distance between real layouts and generated layouts. We use the pre-trained classifiers in [18] to compute FID. **Max. IoU** [18] measures the similarity between the generated layouts and the reference layouts. It is designed to find the best match for each layout from the generated set to the reference set. **Overlap** and **Alignment** measure the perceptual quality of generated layouts. Overlap measures the total overlapping area between any pair of bounding boxes inside the layout. Additionally, we measure the Alignment by computing an alignment loss proposed in [25].

Implementing details. The max time step T in LayoutDM is set to 1000. We set the forward process variances to constants increasing linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$ following Ho's design [14]. We use eight transformer layers which use 8-head attention in our model. Adam optimizer [19] with a learning rate of 1×10^{-5} is used to optimize learnable parameters. The batch size is set to 1024. We implement our model with PyTorch [32] and PyTorch Lightning. All experiments are performed on a single NVIDIA Quadro RTX 6000 GPU device.

Dataset	Rico			
Model	FID↓	Max. IoU↑	Alignment↓	Overlap↓
LayoutGAN-W [24]	162.75±0.28	0.30±0.00	0.71±0.00	174.11±0.22
LayoutGAN-R [24]	52.01±0.62	0.24±0.00	1.13±0.04	69.37±0.66
NDN-none [23]	13.76±0.28	0.35±0.00	0.56±0.03	**54.75±0.29**
LayoutGAN++ [18]	14.43±0.13	0.36±0.00	0.60±0.12	59.85±0.59
VTN [1]	9.31±0.21	0.36±0.00	0.88±0.11	59.31±0.45
LayoutDM(Ours)	**3.03±0.06**	**0.49±0.00**	**0.36±0.06**	57.55±0.48
Real data	4.47	0.65	0.26	50.58
Dataset	PublayNet			
Model	FID↓	Max. IoU↑	Alignment↓	Overlap↓
LayoutGAN-W [24]	195.38±0.46	0.21±0.00	1.21±0.01	138.77±0.21
LayoutGAN-R [24]	100.24±0.61	0.24±0.00	0.82±0.01	45.64±0.32
NDN-none [23]	35.67±0.35	0.31±0.00	0.35±0.01	16.5±0.29
LayoutGAN++ [18]	20.48±0.29	0.36±0.00	0.19±0.00	22.80±0.32
VTN [1]	13.07±0.47	0.37±0.00	0.30±0.01	13.15±0.24
LayoutDM(Ours)	**4.04±0.08**	**0.44±0.00**	**0.15±0.00**	**3.73±0.08**
Real data	9.54	0.53	0.04	0.22
Dataset	Magazine			
Model	FID↓	Max. IoU↑	Alignment↓	Overlap↓
LayoutGAN-W [24]	159.2±0.87	0.12±0.00	**0.74±0.02**	188.77±0.93
LayoutGAN-R [24]	100.66±0.35	0.16±0.00	1.90±0.02	111.85±1.44
NDN-none [23]	23.27±0.09	0.22±0.00	1.05±0.03	**30.31±0.77**
LayoutGAN++ [18]	13.35±0.41	0.26±0.00	0.80±0.02	32.40±0.89
VTN [1]	12.34±0.39	0.25±0.00	1.07±0.03	39.97±0.62
LayoutDM(Ours)	**9.11±0.15**	**0.29±0.00**	0.77±0.03	32.53±0.72
Real data	12.13	0.35	0.43	25.64

Table 1. Quantitative comparison conditioned on element category labels. For reference, the FID and Max. IoU computed between the validation and test data, and the Alignment and Overlap computed with the test data are shown as real data. LayoutGAN-W and LayoutGAN-R denote LayoutGAN with wireframe rendering discriminator and with relation-based discriminator.

4.2. Quantitative Evaluation

Comparison with state-of-the-art models. We quantitatively evaluate the quality of conditional layout generation results on benchmark datasets: Rico, PublayNet and Magazine. Since most methods are designed for unconditional layout generation and have no available public implements, we compare our method with conditional LayoutGAN [24], NDN-none [23] and LayoutGAN++ [18] by citing the results in [18]. Moreover, we also implement a conditional VTN [1] as our comparison baseline model, so that we can compare with methods based on both VAEs and GANs. All the metrics are computed on the full test splits of the datasets. We report the mean and standard deviation over five independent evaluations for each experiment.

The comparison results are reported in Tab. 1. From this table, we can observe that: (1) Our method outperforms SOTA methods on both FID and Max. IoU on all three benchmark datasets, which indicates that the generated layouts by our method are more similar to the real layouts than those by SOTA methods. This verifies that our LayoutDM can produce higher-quality and more diverse layouts than SOTA methods, since FID captures both diversity and fidelity. (2) Layouts generated by our method have a lower

	IoU↓ [21]	Overlap↓ [24]	Alignment↓ [23]
L-VAE [17]	0.45±1.3%	0.15±0.9%	0.37±0.7%
NDN [23]	0.34±1.8%	0.12±0.8%	0.39±0.4%
VTN [1]	0.21±0.6%	0.06±0.2%	0.33±0.4%
Trans. [11]	0.19±0.3%	0.06±0.3%	0.33±0.3%
BLT [21]	0.19±0.2%	0.04±0.1%	0.25±0.7%
Ours	**0.0053**±0.5%	**0.01**±0.1%	**0.22**±1.2%

Table 2. Comparison with extended methods on PublayNet. Qualitative results are cited from [21]. "Trans." denotes "LayoutTransformer" and "L-VAE" denotes "LayoutVAE".

FID score than the real ones in validation splits. This is because the generated layouts have identical attributes to those in the test split, while the real layouts in the validation split have different attributes from those in the test split. We provide more analysis on this point in the supplementary material. (3) With regards to Alignment and Overlap, LayoutDM is slightly weaker on Rico and Magazine. Because our method lacks a discriminator that guides the generator to generate layouts with better alignment and overlap properties as in GANs and does not introduce the layout refine module in NDN. This is likely to cause LayoutDM to be inferior to the state-of-the-art on these two metrics.

Note here that, we don't compare LayoutDM to other state-of-the-art methods such as LayoutTransformer [11] and Coarse2fine [16], because these methods focus on unconditional layout generation problem which is different to our setting. Recently, Kong *et al*. [21] reimplement the conditional version of VTN and LayoutTransformer, and compare their proposed BLT model with these models under the settings of conditional layout generation. However, they compute the metrics (IoU [21], Alignment [23], Overlap [24]) in different ways. As an extended comparison, we adopt the metrics used by BLT [21], and compare our LayoutDM with VTN, LayoutTransformer, LayoutVAE, NDN and BLT on PublayNet. The comparison results are reported in Tab. 2. As shown in this table, our model also outperforms SOTA methods on all metrics.

Effect of transformer layers. We conduct ablation experiments to demonstrate the effectiveness of the transformer layers in LayoutDM. Quantitative results are reported in Tab. 3 and qualitative comparisons are shown in Fig. 3. We have the following observation: After replacing the transformer structure in LayoutDM with a sequence of FC layers, the model can still predict a suitable size for each element, but the positional relationships between elements can not be handled, resulting in significant overlapping and misalignment. This proves that the transformer layers play an essential role and can efficiently capture and utilize the high-level relationships between elements to generate high-quality layouts which follow design rules and aesthetic principles.

	Rico			
Architecture	FID↓	Max.IoU↑	Alignment↓	Overlap↓
w/o transformer	52.64	0.29	1.08	58.13
Full model	3.03	0.49	0.36	57.55
	PublayNet			
Architecture	FID↓	Max.IoU↑	Alignment↓	Overlap↓
w/o transfomer	99.60	0.27	0.89	63.87
Full model	4.04	0.44	0.15	3.73

Table 3. The quantitative results of ablations on transformer layers. "Full" denotes our full model. "w/o transformer" denotes model without transformer layers.

4.3. Qualitative Comparisons

Generation quality comparison. To qualitatively compare the generation performance of different models, we compare with the state-of-the-art method LayoutGAN++ and our implemented conditional VTN. We randomly sample layouts from the test dataset and use the element category labels as conditional inputs. Fig. 2 shows the qualitative comparison results. As one can see, LayoutDM can arrange elements in a reasonable and complicated way, generating higher quality layouts than the other two, with fewer overlapping and better alignment.

Generation diversity comparison. The diversity of results is also an important factor in evaluating the method. We compare the diversity of layouts generated by conditional VTN, LayoutGAN++, and LayoutDM. Fig. 4 show the comparison results. One can see that conditional VTN and LayoutDM generate more diverse results. The *Figure* element in the results floats on the page. Compared to the other two models, LayoutGAN++ captures less diversity, which places a large *Figure* element on the top of the pages in columns 2,4, and 5. LayoutDM performs well in diversity because it breaks the generation into a series of conditional diffusion steps which are relatively easy to model. This alleviates the mode collapse problem in strongly conditional generation task that can lead to the generation of similar modes. We provide more qualitative comparisons on diversity in the supplementary material.

Rendering results comparison. We render graphic pages for better visualization using the generated layouts. Fig. 5 show the rendering results comparison on PublayNet. We find the layouts generated by LayoutDM follow design rules well and reasonably allocate page space. Compared to the results generated by LayoutGAN++, our results are better in alignment and have no overlapping between elements. Note that we crop the elements from the original document and then render the pages using a simple resize-to-fit method, so the text areas and figures will suffer some distortion. In real design scenarios, this problem can be solved by element customization (*e.g*., adjust the font size).

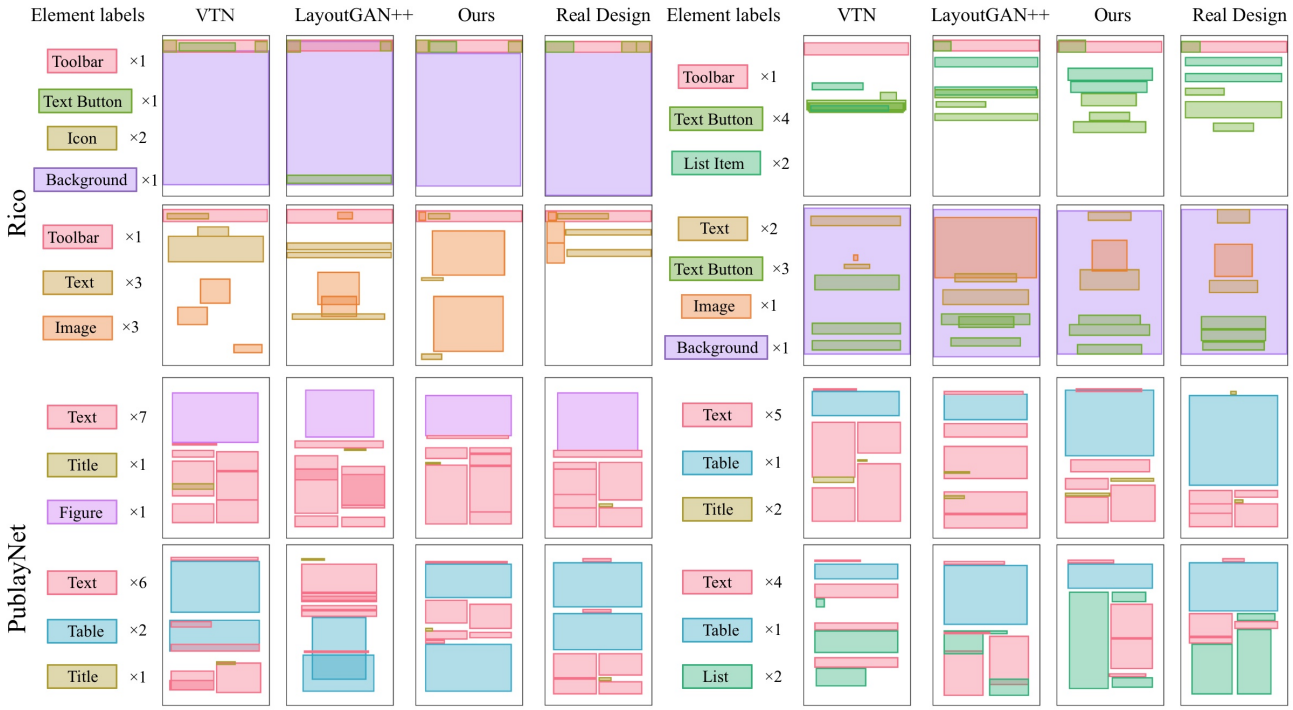


Figure 2. Qualitative comparison on Rico and PublayNet. Element labels indicate the labels of elements used as conditional inputs.

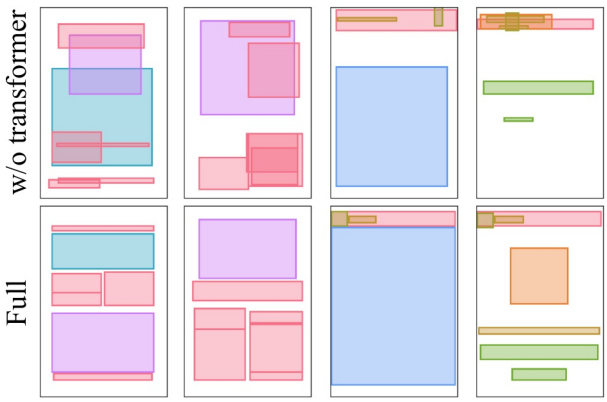


Figure 3. Ablation study on the effect of transformer layers. "w/o transformer" denotes model without transformer layers. "Full" denotes our full model.

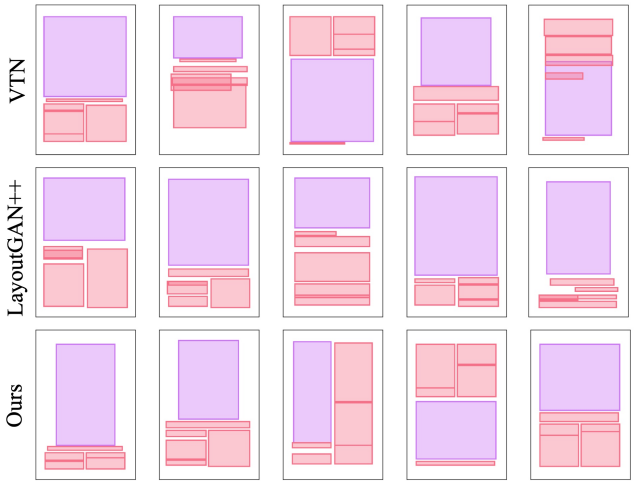


Figure 4. Diversity comparison on PublayNet. We show five samples generated by giving the same category attributes as condition: one *Figure* and five *Texts*. The *Figure* is drawn in purple and the *Texts* are drawn in red.

4.4. Extended Layout Generation Tasks

Text logo layout generation. TextLogo3K [44] dataset contains character and word embedding of texts in the logos. Although the dataset does not provide any label information, we can still generate logo layouts conditioned on the provided textual features. Positional encoding is added to LayoutDM to make the transformer structure aware of the reading order in textual feature sequences. We compare the logo layout generation results of LayoutDM and those generated by the logo layout generator provided by TextLogo3K [44]. The qualitative results are shown in Fig. 6. As one can see, our model generates reasonable logo layouts while maintaining the correct reading order and aesthetic principles. Compared to LogoGAN, the style of the logos generated by our model is more flexible, not simply arranging the text from left to right. Our model also performs better when there are large numbers of characters in the layout, where LogoGAN fails to generate reasonable results.

Scene layout generation. Our model can also generate

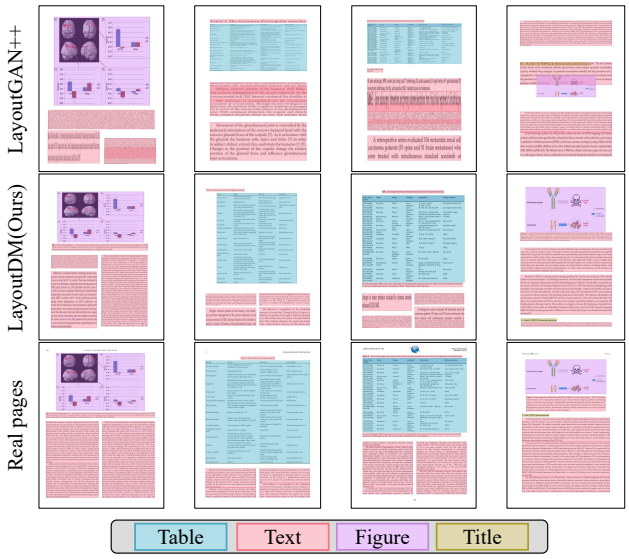


Figure 5. Rendering results comparison. Top: Rendered pages using layouts generated by LayoutGAN++. Middle: Rendered pages using layouts generated by LayoutDM. Bottom: Real paper pages in PublayNet.

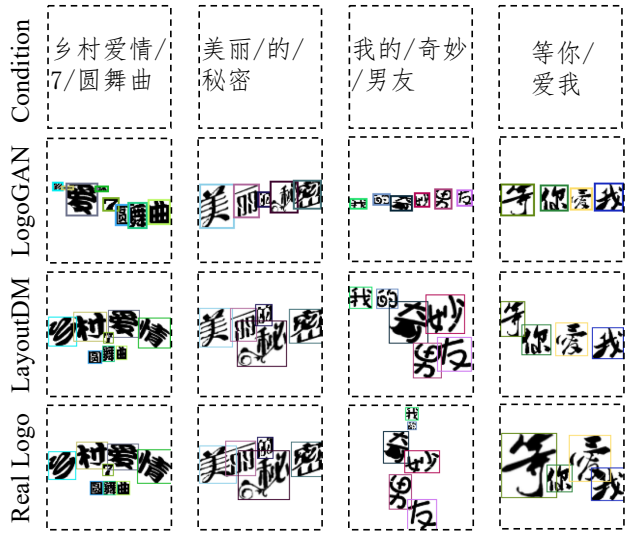


Figure 6. Text logo generation results. "LogoGAN" denotes the text logo generation model proposed in [44]. "Condition" represents the textual features (including character and word embeddings) used as conditional input in LogoGAN and LayoutDM. "/" is the symbol for splitting tokens.

natural scene layouts. We illustrate the qualitative results of the scene layout generation on COCO in Fig. 7. Given the labels of scene elements in a scene, our model generates reasonable scene layout proposals. We then use a downstream layout-to-image generation application [2] to finally render natural scene images. The results show that our model learns the principle of scene layouts well and can understand the complex relationships between elements in

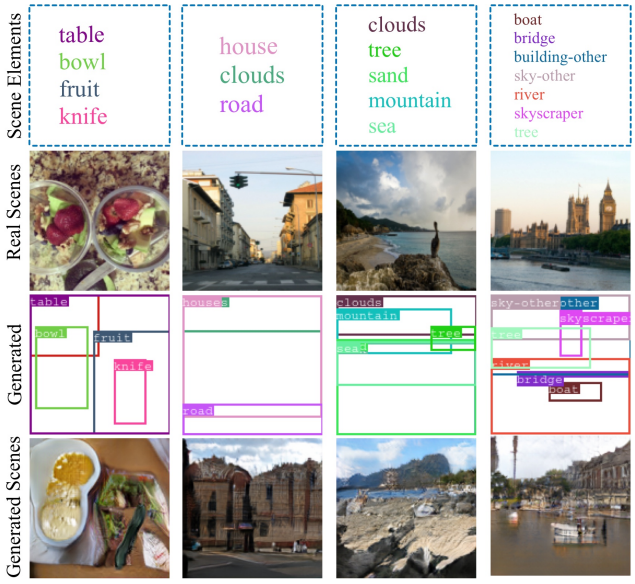


Figure 7. Natural scene layout generation results on COCO. Our model uses scene element labels as conditional input to generate reasonable scene layouts.

natural scenes. For example, the boat should be in the river and the cloud should be in the upper part of the scene.

4.5. Limitations

Although our method shows impressive results in the conditional layout generation problem in comparison to existing methods, it still has limitations. For example, like other layout generation methods, our approach treats design elements as being on a single-layer canvas. This can not model a layout with multiple layers occluding each other. Our method also has no advantage over other generative models in generation speed because the generation of the diffusion model requires an iterative denoising process. We leave the solution to the above problems for future work.

5. Conclusion

This paper proposes a transformer-based diffusion model LayoutDM to address conditional layout generation. We introduce a purely transformer-based Layout Denoiser to model the diffusion reverse process. Benefitting from both DDPM and transformer, in comparison to existing methods, LayoutDM can generate high-quality generation with desired properties such as better diversity, faithful distribution coverage, and stationary training. Quantitative and qualitative results demonstrate that our model outperforms the state-of-the-art methods in terms of visual perceptual quality and diversity.

Acknowledgment. This work was supported in part to Dr. Liansheng Zhuang by NSFC under contract No.U20B2070 and No.61976199, in part to Dr. Fengying Yan by NSFC under contract No.42341207.

References

[1] Diego Martín Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13637–13647, 2021. 1, 2, 4, 5, 6

[2] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4560–4568, 2019. 8

[3] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016. 4

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *ArXiv*, abs/1809.11096, 2019. 1

[5] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *ArXiv*, abs/1609.07093, 2017. 1

[6] Yunning Cao, Ye Ma, Min Zhou, Chuanbin Liu, Hongtao Xie, Tiezheng Ge, and Yuning Jiang. Geometry aligned variational transformer for image-conditioned layout generation. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 1561–1571, New York, NY, USA, 2022. Association for Computing Machinery. 2

[7] 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 Symposium on User Interface Software and Technology*, UIST '17, 2017. 5

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021. 1, 2, 3

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1

[10] Shunan Guo, Zhuochen Jin, Fuling Sun, Jingwen Li, Zhaorui Li, Yang Shi, and Nan Cao. Vinci: An intelligent graphic design system for generating advertising posters. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery. 1

[11] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 984–994, 2021. 1, 2, 4, 6

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. 5

[13] Jonathan Ho. Classifier-free diffusion guidance. *ArXiv*, abs/2207.12598, 2022. 3

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. 1, 2, 3, 4, 5

[15] 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*, 36:1096–1103, 06 2022. 1, 4

[16] Zhao Chun Jiang, Shizhao Sun, Jihua Zhu, Jian-Guang Lou, and D. Zhang. Coarse-to-fine generative modeling for graphic layouts. In *AAAI*, 2022. 2, 6

[17] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9894–9903, 2019. 2, 6

[18] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *ACM International Conference on Multimedia*, MM '21, pages 88–96, 2021. 1, 2, 5

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014. 1

[21] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. Blt: bidirectional layout transformer for controllable layout generation. In *European Conference on Computer Vision*, pages 474–490. Springer, 2022. 1, 2, 6

[22] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016. 1

[23] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 491–506, Cham, 2020. Springer International Publishing. 1, 2, 5, 6

[24] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2388–2399, 2021. 1, 2, 5, 6

[25] 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, 2021. 1, 5

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence

Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 5

[27] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018. 1

[28] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations. *ArXiv*, abs/2103.03841, 2021. 1

[29] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 1

[30] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Designscape: Design with interactive layout suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 1221–1224, New York, NY, USA, 2015. Association for Computing Machinery. 2

[31] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning layouts for single-pagegraphic designs. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1200–1213, 2014. 2

[32] 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. 5

[33] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. Read: Recursive autoencoders for document layout generation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2316–2325, 2020. 1, 2

[34] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 1

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 1, 2

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 2, 3

[37] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 129–136, Madison, WI, USA, 2011. Omnipress. 2

[38] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. 2

[39] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2256–2265. JMLR.org, 2015. 1

[40] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2

[41] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1

[42] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 3

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 4

[44] Yizhi Wang, Gu Pu, Wenhan Luo, Pengfei Wang, Yexin ans Xiong, Hongwen Kang, Zhonghao Wang, and Zhouhui Lian. Aesthetic text logo synthesis via content-aware layout inferring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2, 5, 7, 8

[45] Kota Yamaguchi. Canvasvae: Learning to generate vector graphic documents. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5461–5469, 2021. 1, 2

[46] Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson W. H. Lau. Content-aware generative modeling of graphic design layouts. *ACM Trans. Graph.*, 38(4), jul 2019. 1, 2, 5

[47] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE, Sep. 2019. 5