DogLayout: Denoising Diffusion GAN for Discrete and Continuous Layout Generation

Zhaoxing Gan, Guangnan Ye,

School of Computer Science,Fudan University,Shanghai, China
zxgan23@m.fudan.edu.cn

Abstract

Layout Generation aims to synthesize plausible arrangements from given elements. Currently, the predominant methods in layout generation are Generative Adversarial Networks (GANs) and diffusion models, each presenting its own set of challenges. GANs typically struggle with handling discrete data due to their requirement for differentiable generated samples and have historically circumvented the direct generation of discrete labels by treating them as fixed conditions. Conversely, diffusion-based models, despite achieving state-of-the-art performance across several metrics, require extensive sampling steps which lead to significant time costs. To address these limitations, we propose **DogLayout** (**D**en**o**ising Diffusion **G**AN **Layout** model), which integrates a diffusion process into GANs to enable the generation of discrete label data and significantly reduce diffusion's sampling time. Experiments demonstrate that DogLayout considerably reduces sampling costs by up to 175 times and cuts overlap from 16.43 to 9.59 compared to existing diffusion models, while also surpassing GAN based and other layout methods. Code is available at https://github.com/deadsmither5/DogLayout.

Introduction

Creating aesthetic layouts is an essential method for conveying important information to viewers and plays a critical role in various applications such as UI design (Deka et al. 2017), advertisement poster synthesis (Guo et al. 2021), and editing in printed media (Zhong, Tang, and Yepes 2019). Traditionally reliant on manual design processes, the scalability and efficiency demands of modern digital content creation necessitate automated solutions using generative models.

In the layout generation task, each layout element is characterized by discrete labels along with continuous size and position attributes. Besides conditioning on the label types, sizes, and positions, there are two other important layout generation tasks: 1) Unconditional generation, which involves generating layouts without predefined constraints or elements' attributes, such as randomly designing a game layout. 2) Completion, which involves filling in missing layout elements based on partially known elements and can be used to complete layouts forgotten by human designers.

In both unconditional generation and completion, generating the discrete label is inevitable. However, previous GAN-based layout models (Li et al. 2020; Kikuchi et al.
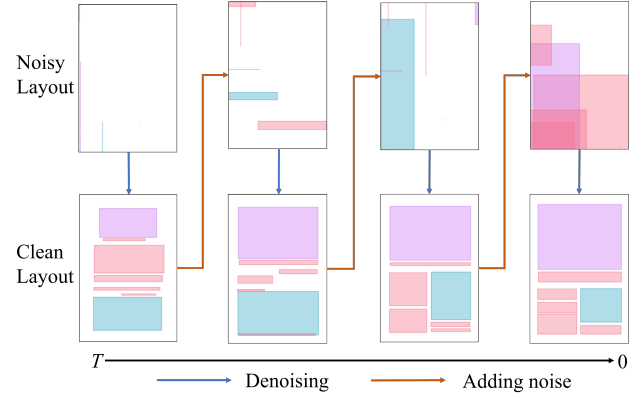

Figure 1: Visualization of DogLayout's inference process. During inference, we first obtain the noisy layout from standard gaussian. Then the generator takes it as input to output the predicted clean layout. Subsequently, we derive the less noisy layout by adding noise to the predicted clean layout. Repeat the above process to achieve the final clean layout.

2021) can only generate the continuous sizes or positions conditioned on the label. Given a one-hot encoded label, GAN-based layout models fail to generate discrete data for two reasons: 1) Given the generator's probabilistic outputs, the discriminator's task is overly simplified as it only needs to identify a single non-zero element in the data's one-hot vector representation. 2) Using the non-differentiable argmax function to convert the generator's probabilistic outputs into discrete formats leads to vanishing gradients.

On the other hand, existing diffusion models (Inoue et al. 2023) focus too much on improving automatic evaluation indicators and ignore the importance of sampling speed in practical applications. To maintain the posterior distribution in a Gaussian form, diffusion models (Ho, Jain, and Abbeel 2020) always involve thousands of sampling timesteps. Although some works like LACE (Chen et al. 2024) use DDIM (Song, Meng, and Ermon 2020) to accelerate the sampling process, the time cost is still significant. In scenarios where quick response is required, the high time cost is unaffordable. Additionally, the layouts generated by diffusion models, such as LayoutDM (Inoue et al. 2023), still suffer from

excessive overlap, which is highly noticeable to humans.

In this study, we propose the DogLayout (shown in Figure 2) to expand Layout GAN models' ability to handle unconditional generation and completion while maintaining a high sampling speed. By adding a diffusion process to GAN, we propose a new method for GANs to deal with discrete label data: 1) All operations on the Generator's output are differentiable. 2) The discriminator does not directly see the generator's output and cannot distinguish real denoised layout from predicted denoised layout based solely on the presence of a single non-zero element. Moreover, by using GAN to fit the non-Gaussian denoising distribution (Xiao, Kreis, and Vahdat 2021), we significantly reduce the number of sampling steps and achieve a sampling speed that is up to 175 times faster than current diffusion-based layout models and also improved the overlap from 16.43 to 9.59.

We summarize our contributions as follows:

- By adding a diffusion process to GANs, we propose a new method to generate discrete label data, which maintains the differentiability of the generator's output and prevents the discriminator from distinguishing real from fake data by detecting a single non-zero element.

- We expanded the capabilities of previous layout GAN models, which were limited to conditional tasks, to include unconditional and completion tasks, thereby improving the quality of generated layouts by up to 2.5 times compared to LayoutGAN++ (Kikuchi et al. 2021).

- Through extensive experiments, our model outperforms non-diffusion-based layout models in most tasks and reduces the time cost of layout generation by up to 175 times compared to diffusion-based layout models, while maintaining competitive performance. User studies indicate that our model is more favored by real users.

Related Work

Layout Generation

Automatically generating layouts is a long-researched topic in graphic design (Hurst, Li, and Marriott 2009). Early studies optimize layouts by manually designing energy functions with constraints. Recent works have begun to use deep generative models to learn plausible layouts. LayoutVAE (Jyothi et al. 2019) introduces two conditional Variational Autoencoders (VAE). NDN-none (Lee et al. 2020) is also a VAE-based model for conditional layout generation using graph neural networks. BLT (Kong et al. 2022) proposes a hierarchical sampling policy with bidirectional layout transformer. As for the generative adversarial networks (GANs, (Goodfellow et al. 2014)) based models, LayoutGAN (Li et al. 2020) can synthesize graphic layouts conditioned on different element attributes. LayoutGAN++ (Kikuchi et al. 2021) builds a transformer-based gan and formulates the layout generation as a constrained optimization problem. However traditional GAN can't deal with categorical data (Hjelm et al. 2017), previous GAN based layout model is all limited to condition on the discrete labels. Recently, diffusion-based models have begun to be used. LayoutDM (Inoue et al. 2023), LayoutDiffusion (Zhang et al. 2023) and (Hui et al.

2023) use the Discrete Diffusion Models (Austin et al. 2021) in a similar way to handle the structured layout data in the discrete representation, their works also expand the conditional layout generation to unconditional situation. LACE (Chen et al. 2024) proposed a novel align loss and also utilizes unconditional layout generation. Although diffusion-based layout models show strong capabilities in terms of diversity, the time cost and overlap in the generated layouts are non-negligible. Our work proposes a new way to enable GANs to generate discrete labels with minimal time cost.

GANs for Discrete Data

Challenges with Discrete Data in GANs. Generative Adversarial Networks (GANs) face two main challenges when applied to discrete data. The generator transforms a latent vector into an output, while the discriminator evaluates whether this output resembles the actual discrete data, typically represented as a one-hot vector. The first challenge is that discriminator's task becomes straightforward—it only needs to detect the presence of a single non-zero element in the real one-hot discrete data. This simplicity contrasts sharply with generator's output, which often spreads non-zero probabilities across multiple dimensions. The second challenge arises when attempting to address the first: using argmax on generator's output to provide one-hot form input for discriminator leads to a gradient vanishing problem due to the non-differentiability of the argmax operation.

Existing solutions. Several studies seek to address these challenges. Gumbel GAN(Kusner and Hernández-Lobato 2016) employs the Gumbel reparameterization technique (Jang, Gu, and Poole 2016), which enables gradient backpropagation from discriminator to generator. However, this approach introduces a new challenge: discriminator only observes the one-hot transformed output from generator, not the output itself, limiting it's ability to effectively guide generator's gradient optimization. BGAN (Hjelm et al. 2017) addresses this by exploring the boundaries of data distributions, yet accurately defining and identifying these boundaries remains challenging. Seq-GAN (Yu et al. 2017) uses discriminator as a reward function employing policy gradients, but designing an effective reward function that accurately evaluates the quality of generated layouts proves difficult. Figure 5 shows that using LayoutGAN++ directly to model discrete label data results in gradient vanishing.

Preliminary

Problem Formulation. Following previous studies (Chen et al. 2024; Kikuchi et al. 2021), we define a layout l with M elements and as $\{(c_1, b_1), \ldots, (c_M, b_M)\}$, where (c_i, b_i) represents the i-th elements in l. $c_i \in \{0, \ldots, N-1\}$ is the discrete label in a range of N classes such as the Text or Title and $b_i = (x_i, y_i, w_i, h_i) \in [0, 1]^4$ is the corresponding center coordinates (x, y) and size ratio (w, h).

Diffusion Models. A standard diffusion model contains a forward diffusion process and reverse diffusion process. In diffusion's forward process, given $x_0 \sim q(x_0)$, each data x_t is corrupted gradually by adding Gaussian noise to x_{t-1}.
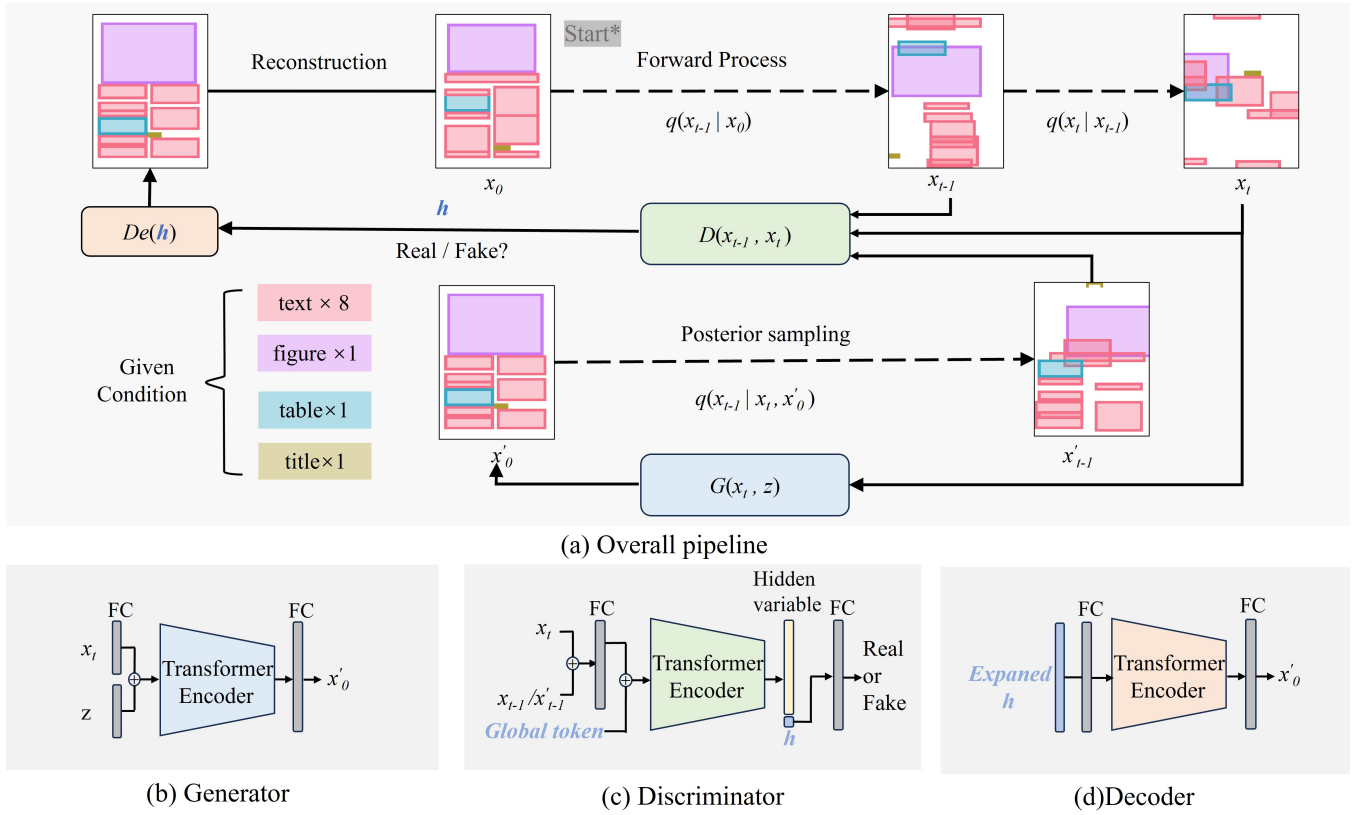
(a) Overall pipeline

(b) Generator   **(c) Discriminator**   **(d)Decoder**

Figure 2: Overview of our method. (a) During training, we first obtain the noisy layout x_{t-1}, then generate x_t by directly adding noise to x_{t-1}. The generator then takes x_t and an additional latent dimension z as inputs to output the predicted clean layout x'_0. Subsequently, we derive the predicted x'_{t-1} using 2. For the real data, the discriminator evaluates the real noisy layout x_t and x_{t-1} to determine whether x_{t-1} is the true denoised layout of x_t. An additional decoder then takes the global context token h from the discriminator and reconstructs x_0, which forces the discriminator to learn the meaningful attributes of the layout. For the fake data, the discriminator assesses the real noisy layout x_t and the predicted layout x'_{t-1} to determine whether x'_{t-1} is the true denoised layout of x_t. The model architectures are shown in (b), (c) and (d).

In reverse process , the goal is to train a network p_θ to predict x_{t-1} from x_t and the whole training objective could be presented by minimizing the Evidence Lower Bound:

$$\mathcal{L}_{\mathrm{ELBO}} = \underbrace{\mathbb{E}_q[D_{\mathrm{KL}}(q(x_T|x_0)||p_\theta(x_T))]}_{\mathcal{L}_T} - \underbrace{\log p_\theta(x_0|x_1)}_{\mathcal{L}_0} +$$

$$\underbrace{\mathbb{E}_q[\sum_{t=2}^{T} D_{\mathrm{KL}}(q(x_{t-1}|x_t)||p_\theta(x_{t-1}|x_t))]}_{\mathcal{L}_{t-1}}. \quad (1)$$

In a standard diffusion process, when forward timestep is large enough $q(x_T|x_0)$ and $p_\theta(x_T)$ will both follows a standard normal distribution $\mathcal{N}(0,1)$ and the $\mathcal{L}_T$ term will be nearly equal to zero. $\mathcal{L}_0$ is the reconstruct loss from x_1 to x_0. $\mathcal{L}_{t-1}$ means to train a neural network $p_\theta(x_{t-1}|x_t)$ to fit the true distribution $q(x_{t-1}|x_t)$. In order to trace $q(x_{t-1}|x_t)$, DDPM (Ho, Jain, and Abbeel 2020) use $q(x_{t-1}|x_t,x_0)$ to alternate $q(x_{t-1}|x_t)$, which could be denoted as:

$$q(x_{t-1}|x_t,x_0) = \frac{q(x_t|x_{t-1},x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)}. \quad (2)$$

Approach

Overview and Model Architecture

DogLayout builds on Diffusion GAN models (Xiao, Kreis, and Vahdat 2021; Gong et al. 2024). In this chapter, we will first introduce the model architecture of DogLayout, then discuss the details of our framework and the methods used to reduce sampling time costs by integrating a GAN into the diffusion process, and explain how this integration enables GANs to handle discrete data.

Conditional and Unconditional Generation. Conditional generation involves creating an entire layout from a partially known layout x_p. Let m represent the mask, where 1 and 0 indicate known and unknown layout attributes, respectively. The conditional information is incorporated as follows: $x_{t-1} = (1-m) \odot \tilde{x}_{t-1} + m \odot x_p$, where $\tilde{x}_{t-1} \sim p_\theta(x_{t-1}|x_t)$. Unconditional generation refers to the process of generating a layout initially from standard Gaussian.

Generator. To process the input noise layout x_t, we utilize a fully-connected layer to expand its dimensions to the em-

bedding dimension. The latent variable z is initially sampled from a standard Gaussian distribution, subsequently resized to the specified latent dimension through a fully-connected layer. while temporal embedding is not explicitly incorporated. The core processing unit comprises a transformer-encoder (Vaswani et al. 2017) Finally, the transformer-encoder's output is adjusted back to the input's dimensions using another fully-connected layer:

$$z \sim \mathcal{N}(0, I), \ h_1 = f_{FC}(z),$$
$$h_2 = f_{FC}(x_t),$$
$$h_3 = f_{TF-ENC}([h_1, h_2]),$$
$$x_0' = f_{FC}(h_3). \quad (3)$$

In the above notations, f_{FC} represents the fully-connected layer, and f_{TF-ENC} represents the transformer-encoder layer. x_t is sampled from $\mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t \mathbf{I})$ and condition is injected from known layout x_p.

Discriminator. The discriminator's input is formed by concatenating x_t with either x_{t-1} or x_{t-1}', depending on whether the data is real or generated. This combined input is then passed through a fully-connected layer to expand its dimensions to match the embedding dimension. Position embedding is injected via a trainable embedding layer, while time embedding is not included. The core unit consists a transformer-encoder which includes a learnable special token h_s to get global context token h. Then a fully-connected layer processes h to produce the probability logits:

$$h_1 = f_{FC}([x_{t-1}, x_t]) \ or \ f_{FC}([x_{t-1}', x_t]),$$
$$[h, h_2] = f_{TF-ENC}(h_1, h_s),$$
$$p = f_{FC}(h). \quad (4)$$

Here, p presents the probability that whether x_{t-1} or x_{t-1}' is the true denoised layout of x_t.

Decoder. When the discriminator processes real inputs x_t and x_{t-1}, it employs a transformer-encoder and a fully-connected layer to reconstruct the initial layout x_0 from the extracted global context h:

$$h_1 = f_{TF-ENC}(h),$$
$$x_{d0}' = f_{FC}(h_1). \quad (5)$$

x_{d0}' is the reconstruction results of decoder. This reconstruction process enables the discriminator to learn the meaningful attributes of the layout effectively, which enables the discriminator to effectively distinguish between real and generated layouts based on their meaningful attributes.

DogLayout

The key to reducing sampling time in the diffusion process is to decrease the timesteps. Using Bayes' rule, the real denoising distribution $q(x_{t-1}|x_t) = q(x_t|x_{t-1})q(x_{t-1})/q(x_t)$, when T is sufficiently large, the noise added between each adjacent step is small enough that the ratio $q(x_{t-1})/q(x_t) \approx 1$. Consequently, both $q(x_t|x_{t-1})$ and $q(x_{t-1}|x_t)$ can be assumed to follow Gaussian distributions.

To reduce the timestep T to a smaller number (e.g., $T = 4$), we can use a GAN to match the non-Gaussian distribution $q(x_{t-1}|x_t)$. When T is small, DDGAN (Xiao, Kreis, and Vahdat 2021) proposes using a conditional generative adversarial network to minimize the distance between these two distributions instead of the original KL Divergence described in Equation 1. Given the noisy layout x_t to both the generator and discriminator, the generator $p_\theta(x_{t-1}|x_t)$ aims to reconstruct the cleaner layout x_{t-1} that is indistinguishable from the real x_{t-1}. The discriminator aims to maximize its ability to distinguish between the real cleaner layout x_{t-1} and the predicted $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$. This training process can be regarded as minimizing the following expression, where D_{adv} represents a metric for calculating the distance between two distributions (e.g., Wasserstein distance (Arjovsky, Chintala, and Bottou 2017)):

$$\min_\theta \sum_{t \geq 1} \mathbb{E}_{q(x_t)}[D_{adv}(q(x_{t-1}|x_t), p_\theta(x_{t-1}|x_t))]. \quad (6)$$

We choose the softened reverse KL as the D_{adv}. We propose not to inject time into the generator and discriminator due to the fact that the timestep t is implicitly included in the noise strength of the given x_t. The generator will take an additional N-dimensional latent variable z to enhance diversity and directly output the predicted version of the layout $x_0 = G_\theta(x_t, z)$. Then, x_{t-1} is sampled using Equation 2. The denoising distribution $p_\theta(x_{t-1}|x_t)$ can be written as:

$$p_\theta(x_{t-1}|x_t) := \int p_\theta(x_0|x_t)q(x_{t-1}|x_t, x_0) \, dx_0 =$$
$$\int p(z)q(x_{t-1}|x_t, x_0 = G_\theta(x_t, z)) \, dz. \quad (7)$$

Inspired by the self-supervised learning method of (Liu et al. 2020), when trained with real x_{t-1} and x_t, another decoder takes the global context token h from the discriminator and reconstructs the layout $x_0 = De(h)$. With such a constraint, we can ensure that the discriminator has learned effective layout features. The training objective for the discriminator is described as:

$$\min \sum_{t \geq 1} \mathbb{E}_{q(x_t)}[\mathbb{E}_{p_\theta(x_{t-1}|x_t)}[-\log(1 - D(x_{t-1}, x_t))]+$$
$$\mathbb{E}_{q(x_{t-1}|x_t)}[-\log(D(x_{t-1}, x_t))]+$$
$$\mathbb{E}_{q(x_0|x_t)}[\mathcal{L}_{rec}(x_0, De(h))]]. \quad (8)$$

DogLayout for Discrete Data. We are the first to discover that adding a diffusion process to GANs enables the generation of discrete data. The introduction of the diffusion process addresses two challenges that GANs face with discrete data, for two specific reasons:

1. All operations on the generator's output $x_0 = G(x_t, z)$ are differentiable. Instead of applying an argmax to x_0, we use Equation 2 to compute the predicted noisy layout x_{t-1}. Meanwhile, the operations of the discriminator are all on x_{t-1}, ensuring that the gradient flows normally towards the generator after back-propagation.

2. The discriminator no longer directly sees the output of the generator, except when $T = 1$. Since all noisy layouts

x_{t-1} are defined in a continuous space, the discriminator cannot distinguish real noisy x_{t-1} from predicted noisy x_{t-1} by simply detecting whether they contain a single non-zero element.

Technically, we can now directly treat the one-hot label l_0 in whole layout x_0 as a class prototype. After directly applying T step noise to l_0, we can use the generator $p_\theta(l_{t-1}|l_t)$ to gradually reconstruct l_0 from l_T. The final discrete layout label L is sampled by the following equation:

$$L = \arg\max_i(p(l = i|l_0)),$$
$$\text{where } p(l = i|l_0) = \frac{\exp(l_0[i])}{\sum_{j=1}^K \exp(l_0[j])}. \quad (9)$$

Variable Length Generation. Diffusion GAN models are designed to produce data with fixed dimensions; therefore, they do not naturally extend to layout generation, where the number of elements in each layout can vary. We add an additional padding element with an extra noise class $c = N$ and $b = (0, 0, 0, 0)$ to manage variable length generation for up to a maximum of E elements.

Experiments

Experimental Setup

Datasets. To evaluate our method, We apply two widely-used datasets in recent studies about layout generation. **Rico** (Deka et al. 2017) contains 72K mobile app screens collected from over 97K Android apps with 25 label types such as Advertisement, Text Button and so on. **PubLayNet** (Zhong, Tang, and Yepes 2019) contains 330K document layout annotations sourced from PubMed Central with 5 label types such as table, figure and list. We set the maximum number of elements to 25. The dataset split of training, validation and test is 35, 851/2, 109/4, 218 for Rico and 315, 757/16, 619/11, 142 for PubLayNet.

Evaluation Metrics. We adapt the five metrics to measure the quality of the generated layouts. These include: **Fréchet Inception Distance (FID)** (Heusel et al. 2017), which measures the distance between feature vectors calculated for real and generated layouts; for this, we use the same feature extraction model employed in LayoutDM (Inoue et al. 2023). **Alignment**, which evaluates the aesthetic and functional arrangement of elements within the layout. **Maximum Intersection over Union (IoU)**, which quantifies the overlap accuracy between the predicted and actual bounding boxes of layout elements. **Overlap**, which quantifies the intersection area among elements within a layout. However, given that overlapping elements are a prevalent pattern within the Rico dataset, this metric is not appropriate for evaluating performance on Rico. Therefore, it is exclusively employed to assess generation quality on the PubLayNet dataset. **Generation Time per Sample (T/sample)**, which compares the time cost per sample across models.

Tasks. Our research explores multiple tasks in layout generation: 1) Conditional Generation by Class **(C→S+P)**:

Given a class of elements, generates size and position. 2) Conditional Generation by Class and Size **(C+S→P)**: Conditions on both the class and size of elements to generate positions. 3) **Completion** with Attributes for a Subset of Elements: Employs a binary condition mask, as described in LayoutDM (Inoue et al. 2023), by randomly selecting 0% to 20% of elements from real samples and generate entire layout based on the masked layout. 4) Unconditional Generation **(Uncond)**: Generates layouts without any conditions.

Baselines. We compare DogLayout against a suite of models on layout tasks using the PubLayNet and Rico datasets, categorized into: **Task-Specific models** include Layout-VAE (Jyothi et al. 2019), NDN-none (Lee et al. 2020), and LayoutGAN++ (Kikuchi et al. 2021), which could prove DogLayout's improvements on GAN-based model. **Task-Agnostic models** comprise BLT (Kong et al. 2022), MaskGIT (Chang et al. 2022) and LayoutDM (Inoue et al. 2023), which is the state-of-art diffusion-based model.

Implementation Details. We set the transformer-encoder in generator and decoder with 4 layers, 8 attention heads, and a feed-forward network width of 2048, while 8 layers, 4 heads for discriminator. Embedding dims for label and bounding box are both 256. We find $T = 4$ is enough for the c→s+p and c+s→p task. But generator needs more timesteps to match the distribution of discrete label. Thus the timestep for unconditional and completion task is $T = 8$ for PubLayNet and $T = 12$ for Rico. We train our model on a single NVIDIA RTX 4090 with 24 GB memory and Adam optimizer with learning rate 10^{-5}. The batch size is 512. The model on each task is trained for up to 200 epochs.

Quantitative Comparisons

We report the overlap results on the PubLayNet dataset in Table 1. Our method demonstrates a significant improvement in the overlap metric compared to LayoutGAN++ and LayoutDM. It is crucial to emphasize the significance of the overlap metric, as overlaps are more noticeable to humans than FID scores and can significantly affect the aesthetics and comfort of the layout. The layouts generated by our model exhibit considerably less overlap, which is particularly important in the context of document layout design.

	Task	C → S+P	C+S → P	Completion	Uncond
Model	Metric	Overlap	Overlap	Overlap	Overlap
LayoutGAN ++		22.8	17.6	-	-
LayoutDM		16.43	18.91	15.0	**13.43**
DogLayout(Ours)		**9.59**	**12.5**	**11.9**	16.3

Table 1: Overlap comparison of our model with Layout-GAN++ and LayoutDM on PubLayNet dataset.

We summarize other quantitative comparison results in Table 2. Compared to LayoutGAN++, our model not only expands GAN's capabilities for completion and unconditional tasks but also significantly improves the FID and MaxIoU scores. Specifically, in the C→S+P task on Pub-LayNet, we reduced LayoutGAN++'s FID score from 24.0

Task	C → S+P				C+S → P				Completion				Uncond			
Dataset	Rico		PubLayNet		Rico		PubLayNet		Rico		PubLayNet		Rico		PubLayNet	
Model	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Align. ↓	FID ↓	Align. ↓
LayoutVAE	33.3	0.249	26.0	0.316	30.6	0.283	27.5	0.315	-	-	-	-	-	-	-	-
NDN-none	28.4	0.158	61.1	0.162	62.8	0.219	69.4	0.222	-	-	-	-	-	-	-	-
LayoutGAN++	6.84	0.267	24.0	0.263	6.22	0.348	9.94	0.342	-	-	-	-	-	-	-	-
MaskGIT	26.1	0.262	17.2	**0.319**	8.05	0.320	5.86	0.380	33.5	0.533	19.7	**0.484**	52.1	**0.015**	27.1	**0.101**
BLT	17.4	0.202	72.1	0.215	4.48	0.340	5.10	**0.387**	117	0.471	131	0.345	88.2	1.030	116	0.153
LayoutDM	4.63	**0.274**	**8.96**	0.308	**2.26**	**0.390**	**4.29**	0.379	10.5	**0.576**	**8.47**	0.376	**8.96**	0.162	15.5	0.195
DogLayout(Ours)	**4.44**	0.268	9.62	0.287	2.86	0.362	9.47	0.345	**10.3**	0.473	16.8	0.356	10.9	0.212	**14.7**	0.190
Real data	1.85	0.691	6.25	0.438	1.85	0.691	6.25	0.438	1.85	0.691	6.25	0.438	1.85	0.109	6.25	0.021

Table 2: Quantitative results on the PubLayNet and Rico datasets for four generation tasks. Task-specific models include LayoutVAE, NDN-none, and LayoutGAN++, which must be conditioned on class labels. The best results are highlighted in **bold**.

Model	LayoutGAN++	LayoutDM	DogLayout(Ours)		
Timesteps	1	50	4	8	12
T/sample (ms)	0.0327	23.3	0.133	0.255	0.377

Table 3: Sampling time comparison of our model with LayoutGAN++ and LayoutDM. ($T = 4$ for C→S+P and C+S→P on both dataset. $T = 8/12$ for uncond and completion on PubLayNet/Rico respectively.)

to 9.62, achieving nearly a 2.5 times improvement. Our model outperforms all non-Diffusion models, with a few exceptions in tasks involving MaskGIT and BLT. Regarding diffusion models, our model exhibits comparable performance . DogLayout also outperforms LayoutDM in several FID metrics and demonstrates significant improvements in sampling speed, as detailed in Table 3. Additionally, we observed suboptimal performance in the completion task on PubLayNet, which we suspect is due to the larger continuous space compared to LayoutDM.

Sampling Time Comparison. In Table 3, we illustrate the sampling time efficiency of our DogLayout compared to LayoutGAN++ and LayoutDM. The experiments were conducted over 50 epochs with a batch size of 128. Our results indicate that DogLayout significantly outperforms the diffusion models, being 175 times faster than LayoutDM in conditional generation tasks. Additionally, it demonstrates substantial speed improvements in other tasks as well. Compared to LayoutGAN++, our model incurs only a minimal increase in sampling time while extending capabilities to include unconditional and completion tasks, thereby significantly enhancing performance in conditional tasks. This improvement in efficiency across tasks positions our model as particularly suitable for real-world applications, where rapid response times are crucial and can significantly conserve resources while maintaining high output quality.

Qualitative Comparisons

We also present a qualitative comparison of our model with LayoutGAN++ and LayoutDM in Figure 4. Compared to LayoutGAN++, our model produces layouts with superior spatial arrangement. It can also generate aesthetic layouts for
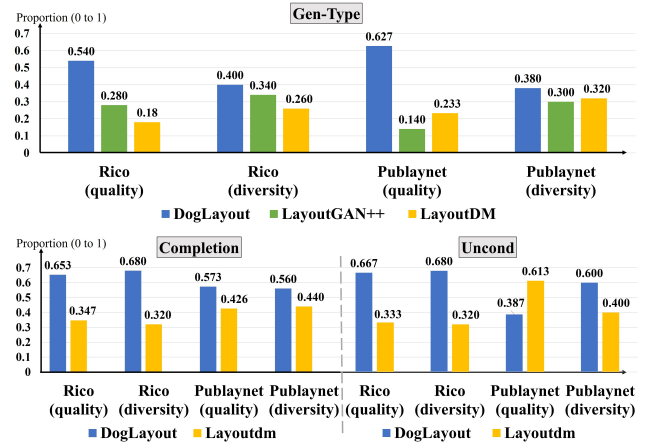

Figure 3: Results of the user studies for Gen-Type(C→S+P and C+S→P), Completion and Uncond. For every selection, We place samples from multiple models and count how many people prefer the layouts generated from each model.

completion and unconditional tasks, a feat not achievable by LayoutGAN++. Notably, while LayoutDM produces layouts with significant overlap areas on PubLayNet, DogLayout generates layouts with much less overlap, enhancing their visual appeal and harmony. We hypothesize that this is because the discriminator can easily detect instances of layout overlap and guide the generator to improve.

User Study. We conducted a user study, as shown in Figure 3, aimed at assessing the aesthetics and diversity of generated layouts, attributes that are often not fully captured by automated metrics such as FID. We designed two types of evaluations: quality and diversity, and invited 25 participants. For each task, samples were randomly selected from each model's outputs generated in the same batch. For the quality test, participants were asked to select the most aesthetically pleasing layout from a set of anonymized layouts. For the diversity assessment, participants were shown 20 layouts generated in the same batch and asked to identify the set with the greatest diversity. Results suggest that DogLayout is more favored by real users.
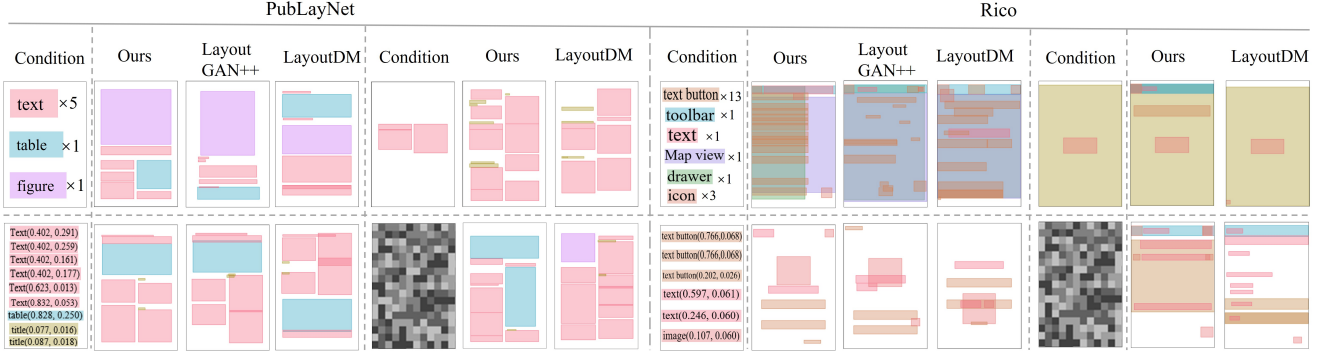
Figure 4: Qualitative comparison results on Rico and PubLyNet for four generation tasks with LayoutGAN++ and LayoutDM. Different colors represent different label classes and the decimals in parentheses are the values of width (w) and height (h).
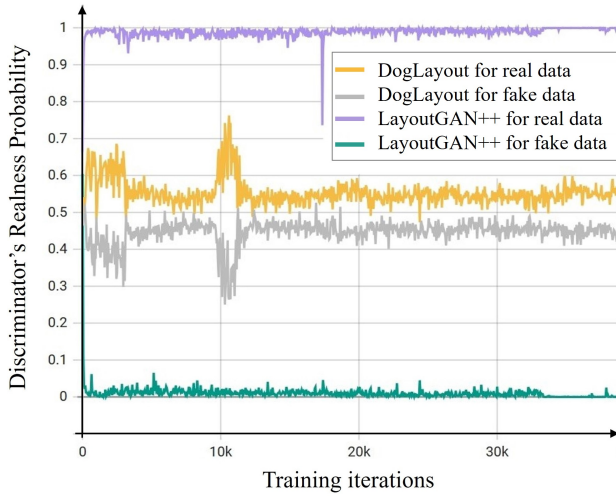

Figure 5: Comparative analysis of discriminator's ability to distinguish real and fake/generated data during training for LayoutGAN++ and DogLayout.

Timestep	FID↓	Align↓
T = 2	199	0.0619
T = 4	53.3	0.450
T = 6	37.4	0.207
T = 8	**14.7**	0.190
T = 10	19.9	**0.179**

Table 4: Timestep effect on PubLayNet dataset with unconditional generation.

Ablation Study

Discrete Layout Generation. Figure 5 illustrates the discriminator's probability of recognizing real and fake data during the training process. In the absence of a diffusion process for discrete data, LayoutGAN++ fails to generate discrete labels. Initially, the discriminator of LayoutGAN++ swiftly approaches a realness probability close to 1 for real data and 0 for fake data, indicating that the gradients quickly vanish. In contrast, DogLayout exhibits remarkable training stability, ensuring consistent convergence and performance improvements across epochs.

Number of Denoising Timesteps. As illustrated in Table 4, the FID score significantly improves with increasing timesteps, reaching a minimum at $T = 8$. Beyond $T = 8$, the FID score slightly increases, We hypothesize that when t is too large, both x_t and x_{t-1} are close to pure noise, making it difficult for the decoder to reconstruct x'_0. Consequently,

this leads to challenges for the discriminator in learning meaningful layout attributes. When $T = 2$, the alignment is low because most layout samples are blank. When T is too low, DogLayout is unable to fit the non-Gaussian distribution in unconditional generation. These results underscore the importance of selecting an optimal number of timesteps to balance generation quality and computational efficiency.

Conclusion

In this work, we propose DogLayout and introduce a new method for GANs to generate discrete label data. We expand the capabilities of previous conditional-only layout GAN models to unconditional and completion tasks, while significantly reduce the diffusion sampling timesteps. Our model achieves speeds up to 175 times faster and better overlap than current diffusion models while maintaining high quality and diversity. We foresee wide-ranging applications of our discrete handling method for GANs, potentially extending beyond the layout domain into other areas where discrete data is prevalent, such as structured data synthesis.

Limitation

While our model has several advancements compared to previous works, there are still several limitations. First, there is still room for improvement in automatic evaluation metrics. Second, most existing layout works lack content awareness, which means the generated layout elements have limited interaction with the background image. In future work, we will expand our model's ability and add image-layout cross attention to improve the current model's content awareness.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223. PMLR.

Austin, J.; Johnson, D. D.; Ho, J.; Tarlow, D.; and Van Den Berg, R. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34: 17981–17993.

Chang, H.; Zhang, H.; Jiang, L.; Liu, C.; and Freeman, W. T. 2022. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11315–11325.

Chen, J.; Zhang, R.; Zhou, Y.; and Chen, C. 2024. Towards Aligned Layout Generation via Diffusion Model with Aesthetic Constraints. *arXiv preprint arXiv:2402.04754*.

Deka, B.; Huang, Z.; Franzen, C.; Hibschman, J.; Afergan, D.; Li, Y.; Nichols, J.; and Kumar, R. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, 845–854.

Gong, M.; Xie, S.; Wei, W.; Grundmann, M.; Batmanghelich, K.; Hou, T.; et al. 2024. Semi-Implicit Denoising Diffusion Models (SIDDMs). *Advances in Neural Information Processing Systems*, 36.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Guo, S.; Jin, Z.; Sun, F.; Li, J.; Li, Z.; Shi, Y.; and Cao, N. 2021. Vinci: an intelligent graphic design system for generating advertising posters. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, 1–17.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hjelm, R. D.; Jacob, A. P.; Che, T.; Trischler, A.; Cho, K.; and Bengio, Y. 2017. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Hui, M.; Zhang, Z.; Zhang, X.; Xie, W.; Wang, Y.; and Lu, Y. 2023. Unifying layout generation with a decoupled diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1942–1951.

Hurst, N.; Li, W.; and Marriott, K. 2009. Review of automatic document formatting. In *Proceedings of the 9th ACM symposium on Document engineering*, 99–108.

Inoue, N.; Kikuchi, K.; Simo-Serra, E.; Otani, M.; and Yamaguchi, K. 2023. Layoutdm: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10167–10176.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jyothi, A. A.; Durand, T.; He, J.; Sigal, L.; and Mori, G. 2019. Layoutvae: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9895–9904.

Kikuchi, K.; Simo-Serra, E.; Otani, M.; and Yamaguchi, K. 2021. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, 88–96.

Kong, X.; Jiang, L.; Chang, H.; Zhang, H.; Hao, Y.; Gong, H.; and Essa, I. 2022. Blt: Bidirectional layout transformer for controllable layout generation. In *European Conference on Computer Vision*, 474–490. Springer.

Kusner, M. J.; and Hernández-Lobato, J. M. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*.

Lee, H.-Y.; Jiang, L.; Essa, I.; Le, P. B.; Gong, H.; Yang, M.-H.; and Yang, W. 2020. Neural design network: Graphic layout generation with constraints. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 491–506. Springer.

Li, J.; Yang, J.; Zhang, J.; Liu, C.; Wang, C.; and Xu, T. 2020. Attribute-conditioned layout gan for automatic graphic design. *IEEE Transactions on Visualization and Computer Graphics*, 27(10): 4039–4048.

Liu, B.; Zhu, Y.; Song, K.; and Elgammal, A. 2020. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*.

Song, J.; Meng, C.; and Ermon, S. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xiao, Z.; Kreis, K.; and Vahdat, A. 2021. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Zhang, J.; Guo, J.; Sun, S.; Lou, J.-G.; and Zhang, D. 2023. Layoutdiffusion: Improving graphic layout generation by discrete diffusion probabilistic models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7226–7236.

Zhong, X.; Tang, J.; and Yepes, A. J. 2019. Publaynet: largest dataset ever for document layout analysis. In *2019 International conference on document analysis and recognition (ICDAR)*, 1015–1022. IEEE.