Dolfin: Diffusion Layout Transformers without Autoencoder

Yilin Wang[1,2*], Zeyuan Chen[2], Liangjun Zhong[1], Zheng Ding[2], and Zhuowen Tu[2]

[1] Tsinghua University, Beijing, 100084, China
[2] University of California, San Diego, California, 92093, USA

Abstract. In this paper, we introduce a new generative model, Diffusion Layout Transformers without Autoencoder (Dolfin), that attains significantly improved modeling capability and transparency over the existing approaches. Dolfin employs a Transformer-based diffusion process to model layout generation. In addition to an efficient bi-directional (non-causal joint) sequence representation, we also design an autoregressive diffusion model (Dolfin-AR) that is especially adept at capturing neighboring objects' rich local semantic correlations, such as alignment, size, and overlap. When evaluated on standard unconditional layout generation benchmarks, Dolfin notably outperforms previous methods across various metrics, such as FID, alignment, overlap, MaxIoU, and DocSim scores. Moreover, Dolfin's applications extend beyond layout generation, making it suitable for modeling other types of geometric structures, such as line segments. Our experiments present both qualitative and quantitative results to demonstrate the advantages of Dolfin.

1 Introduction

Modeling highly-structured geometric scenes such as layout [56] is of both scientific and practical significance in design, 3D modeling, document analysis, and image generation. Layout data are typically highly-geometrically-structured, demonstrating strong correlations between neighboring objects in alignment, location, and size. Traditional approaches modeling joint objects using e.g. constellation models [10, 45, 49] demonstrate interesting results but they fail in accurately capturing the joint relations of different objects. Deep model based layout generation [2, 5, 22, 23, 29, 31] has demonstrated significantly improved quality for the synthesis/generation of layouts.

1. **Main challenges of unconditional layout generation**. Layout generation for documents, designs, and scenes consists of discrete geometric objects/items (e.g. bounding boxes) that are highly structured with special joint spatial arrangement requirements. For example, adjacent tables often observe strict horizontal and vertical alignments. There is a strong interdependence amongst

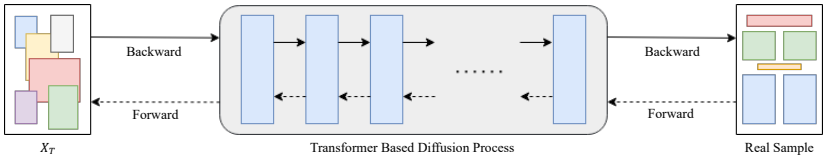
* Work done during internship of Yilin Wang, Liangjun Zhong at UC San Diego.

the items and the generation of an appropriate layout requires having all of them well sized, properly localized, and perfectly aligned. A slight misalignment of one item might destroy the overall layout structure. Valid layouts may form complex manifold that is challenging to learn and explore when searching for the optimization solution.

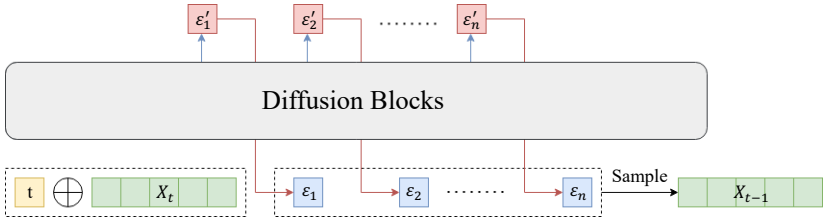
2. **Why applying Transformers in layout generation?** Before the wide adoption of the transformers, generative modeling is primarily focused on image generation using CNN-based architectures (e.g. VAE [25], GAN [12]). Diffusion models for image generation (e.g. Stable Diffusion [38]) commonly adopt the U-Net [39] as the denoising backbone, which is not directly applicable to geometric structural data like layouts.
3. **Significance of removing autoencoders in Transformers for layout generation.** Autoencoders are typically included in image generation [37,38] and layout generation [23], which help reduce the learning complexity at a certain level of increased opacity. In order to deal with the continuous and discrete parts of the data, all existing Transformer-based layout generation methods [2, 5, 22, 29] consist of an encoder and a decoder to operate in the latent space; mapping highly structural data to latent space with autoencoders lead to distortion of the geometrical features (e.g. bounding box alignment). Removing autoencoders in layout generation allows the modeling to stay in the original space, which offers notable advantages in achieving the *transparency and accessibility* of the diffusion process. This 1). allows a better understanding of the generative process, 2). facilitates the direct interaction for the layout with other tasks (if one wants to pull out or add constraints to the intermediate layout for image generation and matching), and 3). makes feasible extensions to other domains like line segment generation, as shown in Sec. 5.4.

In this paper, we present **D**iffusi**o**n **l**ayout Trans**f**ormers w**i**thout Autoe**n**coder (Dolfin), that operates on the original space (the coordinates of the bounding box corners and the corresponding class label) for the geometric structural data. Dolfin is a new diffusion model with the following contributions:

- By removing the autoencoder layer that is typically included in a diffusion model for layout/image generation, the Dolfin model (bi-directional) operates directly on the input space of the geometric objects/items (e.g. rectangles and line segments). Our proposed Dolfin outperforms the competing methods in a number of metrics with reduced algorithm complexity as well as greatly enhanced generalization capability to modeling geometric structures beyond rectangles, such as the line segments.
- In addition to a bi-directional (non-causal joint) representation for Dolfin, we further propose Dolfin-AR, an autoregressive diffusion model especially effective in capturing the rich semantic correlation between objects/items.
- Given the simplicity and generalization capability of the proposed Dolfin model, we experiment on generating geometric structures beyond layout, such as line segments. To the best of our knowledge, this is the first attempt to learn a faithful generative diffusion model for line segment structures that are annotated from natural image scenes.



(a) Dolfin. The layout tensors are directly fed as input to the transformer-based diffusion block. The model processes the input and generates the desired samples without using an autoregressive decoding process.



(b) Dolfin-AR. The diffusion process starts with the input x_t, and it passes through the transformer-based diffusion block. During each autoregressive step i, the noise ϵ_i is sampled, and both ϵ_i and other inputs are used to sample the next noise ϵ_{i+1}. Finally, the previous sample x_{t-1} is generated based on the sampled noise using DDIM [44].

Fig. 1: The Dolfin model. We directly apply Gaussian noise on the original input space.

We make comparisons of our model with other models. In our experiments, our design achieves high performance across numerous metrics.

To further expand the capabilities of our model into line segment generation, an area not yet explored by diffusion models or GAN-based models, our approach demonstrates promising results.

Furthermore, some alternative models, like the SAM algorithm for instance segmentation, directly utilize input tokens in the original space. This aspect makes our approach versatile and suitable for adaptation to various related domains, aligning with the exploding developments of the future.

2 Background

2.1 Layout Generation

A layout is characterized by the global height H and width W of the entire scene and a sequence of 5-tuples $\{x_i, y_i, h_i, w_i, c_i\}_{i=1}^{N}$, where each tuple comprises the left-bottom coordinates of a bounding box (x_i, y_i), the height and width (h_i, w_i) of the bounding box, and the corresponding category c_i.

In our proposed method, each object in a layout is represented by a 4×4 tensor. The tensor consists of different entries that encode specific information about the bounding box. The 4 entries on the first row represent x, y, h, w of the bounding box, The entries on the second row denote the height and width of the entire layout. All of these values are normalized to the range $[-1, 1]$. The remaining 8 entries are used to indicate the category of the bounding box. Some

of these entries have a value of 1, while others have a value of -1, representing different categories. If all of the entries are -1, it indicates that the predicted item isn't active, which is used to handle layouts with different lengths.

The task of layout generation involves generating plausible layouts based on incomplete information through the use of learned models, and it can be classified into two categories: conditional and unconditional. In conditional layout generation, a portion of the layout information is provided, such as the category, shape (height and width), location (coordinates x and y), and other relevant information. Inspired by the approach used in BLT [27], conditionality can be achieved by selectively unmasking specific parts of the tensors, which allows us to generate layouts based on the provided conditions. On the other hand, unconditional layout generation involves generating the final layout entirely from scratch, without any pre-existing conditions or constraints imposed on the layout.

2.2 Diffusion Model Used in Layout Generation

A diffusion model learns a prior distribution $p_\theta(x)$ of the target distribution x with parameters θ. It has already been employed in modeling various representations, including 2D images [8, 18, 36], videos [17, 19], and 3D radiance fields [4, 42]. The model consists of a forward diffusion process and a backward denoising process. The forward process is modeled by a Markov chain which gradually adds Gaussian noise with scheduled variances to the input data. The noisy data at time step t has the closed form representation as the following:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \mathbf{I}) \tag{1}$$

where $\bar{\alpha}_t$ is a constant determined by t. Based on Eq. (1), we can directly sample x_t from x_0 during training.

In the backward process, a denoising network is learned for predicting and removing the noise of noisy data x_t at each time step. The network is optimized by an L2 denoising loss:

$$L_{\text{diff}} = \|\epsilon_t - \epsilon_\theta(x_t)\|_2^2 \tag{2}$$

where $\epsilon_\theta(x_t)$ is the noise predicted by our model with model parameter θ as well as input x_t and t.

3 Method

We propose Dolfin, a diffusion layout transformer for generative layout and structure modeling. Fig. 1 provides an overview of Dolfin.

3.1 Diffusion Layout Transformers

As mentioned in Sec. 2.1, our input data for layout generation comprises both continuous components (bounding boxes) and discrete components (categories). In contrast to previous methods that encode the input into a continuous latent space or separate the discrete/continuous components and design a specialized discrete diffusion process for discrete parts, our method directly operates on the input geometric objects without any additional modules or modifications on the diffusion model.

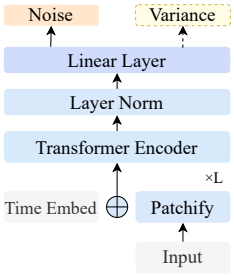


Fig. 2: The transformer structure includes L transformer layers, a layer norm, and a linear layer.

Inspired by the recent work DiT [37], we select a transformer based diffusion model as the base architecture of Dolfin, as transformer-based models are particularly good at modeling sequential data. As described before, we directly process the data in the discrete input space by adding Gaussian noise to the input tensor. The transformer network predicts the noise at each step of the diffusion process.

Starting from the original DiT, we make several modifications to the transformer model. Our model only embeds the timestamp t into the input, as opposed to embedding both the timestamp t and a global category c. Fig. 2 illustrates the structure.

Our model consists of two versions: the non-autoregressive version and the autoregressive version. The key difference between the two is the training approach used for the transformer encoder. In the non-autoregressive version, the transformer encoder is trained in non-autoregressive way. Meanwhile, in the autoregressive version, the transformer encoder is trained in autoregressive way.

3.2 Non-Autoregressive Model

In this version of our method, the transformer operates in a non-autoregressive manner, processing all tokens simultaneously rather than sequentially. Fig. 1a provides an overview of our method.

Training Given an initial layout represented by a vector x_{start} and a diffusion timestep t, we randomly sample a Gaussian noise ϵ_t and apply **Equation (1)** to add noise to x_{start}, resulting in x_t. Next, we input x_t and t into a transformer model to predict the noise $\epsilon_\theta(x_t)$ and the variance of the step (only required for DDPM [18], not required for DDIM in the DiT [37] framework) $\hat{\sigma}_t$. The mean squared error (MSE) loss is then calculated between ϵ_0 and $\epsilon_\theta(x_t)$, while the Kullback-Leibler (KL) divergence loss is calculated (if we use DDPM) between the actual and predicted mean and variance μ_t, σ_t and $\hat{\mu}_t, \hat{\sigma}_t$, where $\hat{\mu}_t$ can be sampled by $\hat{\sigma}_t$ and $\epsilon_\theta(x_t)$ in DDPM.

$$\mathcal{L}_{MSE} = \|\epsilon_t - \epsilon_\theta(x_t)\|_2^2 \tag{3}$$

$$\mathcal{L}_{KL} = \sum_t D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right) = \sum_t D_{KL}(\mu_t, \sigma_t \| \hat{\mu}_t, \hat{\sigma}_t) \tag{4}$$

Sampling Let T be the number of diffusion steps. At each time step t from $T-1$ to 0, we first input x_{t+1} and timestamp $t+1$ into the pre-trained transformer to obtain the predicted noise $\epsilon_\theta(x_{t+1})$. We then use DDPM to compute x_t from x_{t+1} and $\epsilon_\theta(x_{t+1})$. This process is repeated until the final sample x_0 is obtained.

3.3 Autoregressive Model

In contrast to the non-autoregressive model that samples $\hat{\epsilon}$ using a single step, our proposed method adopts a recursive sampling strategy repeated for N times, where N corresponds to the number of tokens. This approach allows for more comprehensive sampling and captures the dependencies among tokens. The training and sampling procedures are outlined in Algorithm 1 and Algorithm 2 respectively, and Fig. 1b offers a visual representation of this approach.

For the k-th noise prediction $\epsilon_{k,t}$ corresponding to x_t, it's based on x_{t+1} and the previous predicted $\{\epsilon_{0,t}, \cdots, \epsilon_{k-1,t}\}$ by the Transformer-based denoising backbone. ϵ_θ in the standard version corresponds to the combination of $\{\epsilon_{0,t}, \cdots, \epsilon_{n,t}\}$ here. In Algorithm 1 and Algorithm 2, *noise* represents the output of the denoising backbone, referring to the mean of the Gaussian diffusion step. f, η are the parameters of the denoising backbone and the learning rate.

Algorithm 1 Autoregressive Training

Input: input x_0, timestamp t
Sample x_t from x_0 and t
for $t = 0$ to $N-1$ **do**
    $n\_in \leftarrow \text{concat}(x_t, \text{noise}[0 \dots t-1])$
    $\epsilon_\theta(x_t) \leftarrow \text{Model}(n\_in, t)$
    $Loss \leftarrow \text{MSE}(\epsilon_\theta(x_t), \text{noise}[t])$
    Update the parameters $f \leftarrow f - \eta \nabla_f Loss$
end for

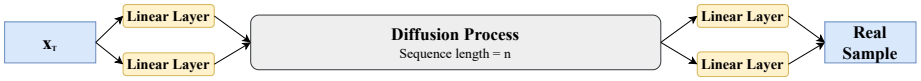
Algorithm 2 Autoregressive Sampling

Input: input x_t, timestamp t
Sample x_t from x_0 and t
for $t = 0$ to $N-1$ **do**
    $noise[t] \leftarrow Model(x, t)$
    $x \leftarrow concat(x, noise[t])$
end for
$\epsilon_\theta(x_t) \leftarrow noise[0 \cdots N-1]$
Sample x_{t-1} by x_t, t, $\epsilon_\theta(x_t)$ using DDIM [44]

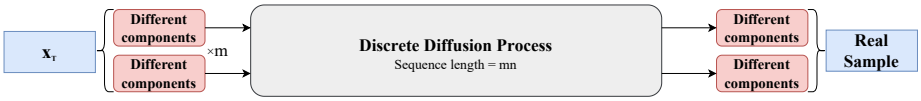
3.4 Comparison of Our Method with Other Approaches

Our model has several advantages over other methods in the field. One of the main benefits is that it employs a relatively simple model, which makes it computationally efficient. In particular, unlike the works PLay [5] and DLT [29], in which the inputs are encoded to a latent space, diffusion is directly applied to the original space. Mapping the highly structural data to latent space with autoencoders may lead to the loss of geometrical features (e.g. bounding box alignment). What's more, this approach eliminates the need to separate each

(a) In the redrawing of PLay [5], the input layout is encoded using a pretrained Variational Autoencoder (VAE) as a first-stage model to map it into a latent space. The diffusion process is then applied, and the final result is generated by decoding the latent representation using a VAE decoder.



(b) In the redrawing of DLT [29], the attributes of a layout are divided into discrete and continuous parts, and they are processed separately through different linear layers before being concatenated and fed into the diffusion process. The output of the diffusion process is then passed through separate linear layers to obtain the final sample.



(c) In the redrawing of LayoutDM [22], a layout is divided into multiple discrete components, creating a token sequence of length mn, where m is the number of components and n is the number of objects in a layout. This differs from other methods, which typically have a token sequence length equal to the number of items in a layout.

Fig. 3: The comparison between different layout generating models

object in the layout into several different tokens like the method provided in LayoutDM [22], which in turn requires less computation.

To visually depict this diffusion sampling process, we present it in the supplementary material, which showcases the gradual refinement of the layout from its initial random state to its final coherent form. The figure serves as an illustration of the step-by-step evolution of the layout during the diffusion process, highlighting the transition from noise to a structured and visually pleasing arrangement. This visual representation enhances our understanding of the generative process and provides insights into the underlying mechanisms driving the layout generation.

3.5 Beyond Layout Generation

Dolfin can be extended beyond simple application of layout generation. As the transformer in Dolfin directly operates on the input geometric objects, it can manage different types of geometric structures.

We present one application on line segments generation. Dolfin is able to learn distributions of line segments from the images in the training set. Similar to layout generation, we take two endpoint coordinates of each line as one input token to the transformer, directly adding noise to the tokens and performing denoising. By sampling from the learned distribution, the model generates novel line segments that are natural and plausible.

4 Related Work

Layout Generation. The task of layout generation involves using generative models to create layouts (vectorized data) for various purposes such as houses, rooms, posters, documents, and user interfaces [6, 11, 14, 24, 34, 35, 40, 43, 51, 52, 55].

LayoutGAN [31] and LayoutVAE [23] are two representative methods of layout generation. In LayoutGAN, a generator is trained to produce layouts using a differentiable wireframe rendering layer, while a discriminator is trained to assess the alignment. This approach allows for the generation of realistic layouts. LayoutVAE employs a variational autoencoder (VAE) [26] to generate layouts. The VAE utilizes a Long Short-Term Memory (LSTM) network [20] to extract relevant information for generating layouts. Recent works, such as BLT [27], introduce a bidirectional transformer for layout generation, which further enhances the model capacity of extracting relationships between different layout objects. The work LT [15] introduces an autoregressive model without diffusion, tokens in a sequence are sampled one by one, allowing each token to incorporate information from the previously sampled tokens. DLT [29], LayoutDM [22], and PLay [5], introduce transformer-based [47] diffusion processes, resulting in improved performance and layout generation capabilities. Methods such as LayoutPrompter [32] and LayoutGPT [9] utilize large language models (LLMs) to perform text-to-layout generation, which are not applicable for unconditional layout generation.

In real-world applications, the generated layouts are often required to meet customized constraints like categories or partial layouts, which yields the task of conditional layout generation [1, 2, 22, 24, 27, 29]. Particular conditions include categories, bounding box numbers, fixed bounding box positions and sizes, etc. Conditions are added by using some encoders to encode the constraints or simply by applying simple masks.

Layout Generation Using Diffusion Models. The diffusion model [18, 44] is applied to layout generating tasks in recent days. Works such as LayoutDiffusion [53] and HouseDiffusion [41] employ diffusion models to generate geometrical data. Currently, researchers employ three main approaches that employ Transformer-based models to address this problem. In PLay [5] (Fig. 3a), an encoder is used to map the layout into a continuous latent space. Diffusion models are then applied to this latent space instead of the original space. In DLT [29] (Fig. 3b), separate linear layers are used for the continuous and discrete parts of the input tensor, both before and after the transformer block. When adding noise during the diffusion process, Gaussian noise is applied to the continuous parts, while for the discrete parts, a discrete diffusion process is defined to introduce noise. In LayoutDM [22] (Fig. 3c), layouts are considered as compositions of distinct discrete elements, such as category, size, and position. To enable diffusion modeling in this context, a discrete version of the diffusion process is devised. What's more, the sequences that enter the transformer have length far larger than the number of items in the layouts, which is computational consuming.

In our method, as shown in Fig. 1, we simply view each element as a whole and directly add Gaussian noise on the original vector space without projecting it to a latent space.

5 Experiments

5.1 Implementation Details

Datasets Our model is trained using two widely used layout datasets on document and user interface, PublayNet [56] and RICO [7]. The PublayNet is a large dataset of document layouts which includes around 330,000 samples extracted from published scientific papers. The dataset is categorized into five classes: text, title, figure, list, and table. It is divided into training, validation, and testing sets as described in [56]. The RICO dataset consists of about 70,000 layouts of user interface designs, classified into 25 categories. We adopt the 85%-5%-10% split for training, validation, and testing as in [22, 29].

Training Details The transformer encoder in Dolfin consists of 4 attention layers and 8 attention heads with a hidden size of 512. This parameter setting makes our model size similar to other baselines, ensuring a fair comparison. The

Fig. 4: Unconditional generation on the PublayNet dataset. The first 6 columns are from paper LayoutDM [22], each including 5 visual results of unconditional generation by LT [15], MaskGIT [3], BLT [27], BART [30], VQDiffusion [13] and LayoutDM [22] respectively. The last 2 columns contain the visual results of our Dolfin and Dolfin-AR.

Method	PublayNet Align. ↓	PublayNet Overlap ↓	PublayNet FID ↓	PublayNet MaxIoU ↑	RICO Align. ↓	RICO FID ↓	RICO MaxIoU ↑
Real Data	0.021	4.2	-	-	0.109	-	-
PLay [5]	-	-	13.71	-	-	**13.00**	-
VTN [1]	-	2.6	19.80	-	-	18.80	-
LayoutDM [22]	0.195	-	-	-	0.162	-	-
LT-fixed [15]	0.084	-	-	-	0.133	-	-
LT [15]	0.127	**2.4**	-	-	0.068	-	-
MaskGIT [3]	0.101	-	-	-	**0.015**	-	-
BLT [27]	0.153	2.7	-	-	1.030	-	-
BART [30]	0.116	-	-	-	0.090	-	-
VQDiffusion [13]	0.193	-	-	-	0.178	-	-
DLT [29]	0.110	2.6	14.25	0.34	0.210	17.80	0.27
LayoutGAN-W [31]	-	-	-	0.21	-	-	0.24
LayoutGAN-R [31]	-	-	-	0.24	-	-	0.30
NDN-none [13]	-	-	-	0.31	-	-	0.35
LayoutGAN++ [24]	-	-	-	**0.36**	-	-	0.36
Dolfin (**ours**)	0.074	5.0	**9.12**	0.35	0.094	15.38	**0.42**
Dolfin-AR (**ours**)	**0.042**	**2.4**	16.88	**0.36**	0.148	26.44	0.17

Table 1: Unconditional generation results. Some baseline results are missing because they are not open-source or lack details for implementation and metric computations, making comparison with the numbers reported in their original papers difficult.

number of diffusion steps is set to 100 for sampling and 1000 for training. The AdamW optimizer is employed with a learning rate of 1e-4. The batch size is set to 10000 for the training of non-autogressive model and 6000 for the training of autogressive model. Dolfin is trained on 8 NVIDIA RTX A5000 GPUs. It takes approximately 48 hours to train the non-autoregressive model and 96 hours to train the autoregressive model.

Metrics We employ five different metrics to evaluate the quality of the generated layouts on the PublayNet dataset: Fréchet Inception Distance (FID) score [16], overlap score, alignment score, MaxIoU score and DocSim score. For the RICO dataset, we do not calculate the overlap score as the data samples in RICO typically contain large regions of overlap.

The main models to compare with are three recent SOTA methods (PLay [5], LayoutDM [22], DLT [29]). The evaluation metrics are highly sensitive to the hyper-parameters, but the source code for evaluation is not provided in these three papers. As the details vary, the results will vary a lot. E.g., if we use 1024 samples to compute the scores, the FID of conditional generation (conditioned on category) on PublayNet is 1.73. However, if we use 512 samples, the score will be 4.21. [3] To compute the FID score, we first render the generated layouts into images. Subsequently, we utilize a widely accepted Inception model [46] to measure the similarity between the generated images and the real images from the dataset. The computation process follows the established methodology presented in the work PLay [5], ensuring a fair comparison with their results.

[3] We emailed the authors for the evaluation details. But only the author of PLay replied for computing the FID score and the author of LayoutDM(1) replied for MaxIoU.

Method	PublayNet Cate. ↑	PublayNet Cate.+Size ↑	Rico Cate. ↑	Rico Cate.+Size ↑
LayoutVAE [23]	0.316	0.315	0.249	0.283
NDN-none [28]	0.162	0.222	0.158	0.219
LayoutGAN++ [24]	0.263	0.342	0.267	0.348
LT [15]	0.272	0.320	0.223	0.323
MaskGIT [3]	0.319	0.380	0.262	0.320
BLT [27]	0.215	**0.387**	0.202	0.340
BART [30]	0.320	0.375	0.253	0.334
VQDiffusion [13]	0.319	0.374	0.252	0.331
LayoutDM [22]	0.310	0.381	0.277	0.392
LayoutDM(1) [2]	**0.440**	-	0.490	-
Dolfin (ours)	0.336	0.339	**0.529**	**0.550**

Table 2: MaxIoU scores for models conditioning on category / category+size on PublayNet and RICO. Dolfin achieves superior performance and outperform the competing methods in most cases. This validates the efficacy of including Transformer based architecture into Dolfin, allowing it to directly operating on the original space of rectangles (geometric structures).

Method	FID ↓	Difference ↓
LT [15]	317.4	30.86
LayoutGAN++ [24]	400.0	51.90
Dolfin (Ours)	**92.1**	**16.04**

Table 3: Quantitative results of line segment generation. Dolfin performs better than the competing methods.

Method	PublayNet Cate. ↓	PublayNet Cate.+Size ↓	Rico Cate. ↓	Rico Cate.+Size ↓
Real Data	0.02	0.02	0.11	0.11
VTN [1]	0.29	0.09	0.43	0.44
BLT [27]	**0.10**	0.09	**0.12**	0.30
LT [15]	0.41	0.14	0.58	0.41
DLT [29]	0.11	0.09	0.18	0.28
LayoutDM(1) [2]	0.15	-	0.36	-
Dolfin (**ours**)	0.41	**0.08**	0.17	**0.14**

Table 4: Alignment scores for models conditioning on category / category+size on PublayNet and RICO. Our method performs well when comparing with other baselines.

Method	PublayNet Cate. ↑	PublayNet Cate.+Size ↑	Rico Cate. ↑	Rico Cate.+Size ↑
LayoutVAE [23]	0.07	0.09	0.13	0.19
NDN-none [28]	0.06	0.09	0.15	0.21
LT [15]	0.11	-	0.20	-
VTN [1]	0.10	-	0.20	-
BLT [27]	0.11	0.18	0.21	0.30
Dolfin (ours)	**0.42**	**0.42**	**0.57**	**0.59**

Table 5: DocSim scores for models conditioning on category / category+size on PublayNet and RICO. We observe a significant boost of performance by our method when compared with existing approaches.

For overlap score and alignment score, we adopt similar evaluation protocols as those employed in previous works. Specifically, we utilize data from the LayoutGAN++ [24], DLT [29] and LayoutDM [22] papers to compute the scores. These metrics provide objective measures of the quality and accuracy of the generated layouts.

Notably, the datasets exhibit different characteristics in terms of alignment and overlap. While the PublayNet dataset demonstrates excellent alignment, it has relatively little overlap between objects. In contrast, the RICO dataset contains numerous instances of overlap between objects. As a result, we do not use overlap as a metric on the dataset RICO.

5.2 Experimental Results

We compare the results of Dolfin and Dolfin-AR with several other methods list in the tables. We present the comparisons of unconditional generation performances in Tab. 1, PublayNet on the left and RICO on the right. Some baselines do not provide the results on all the metrics, so we only report the available values included in their original papers. The results shows that the proposed Dolfin model, including both the autoregressive and non-autoregressive versions, consistently outperforms other SOTAs on various tasks, except for the results of unconditional generation on the RICO dataset. We also present the time cost for our model in the supplementary material.

For conditional generation, we utilize the MaxIoU score, alignment score and DocSim score as the metrics. The corresponding results are shown in Tab. 2, Tab. 4 and Tab. 5 respectively. We test on sampling with conditions include category and category+size (height and width of the bounding boxes) on both PublayNet and RICO datasets. The data shows that even on dataset with large regions of overlap, our method can still achieve nice performance.

We illustrate qualitative generations results in Fig. 4 and Fig. 5, which shows that our methods can efficiently generate satisfying layouts. We also provide

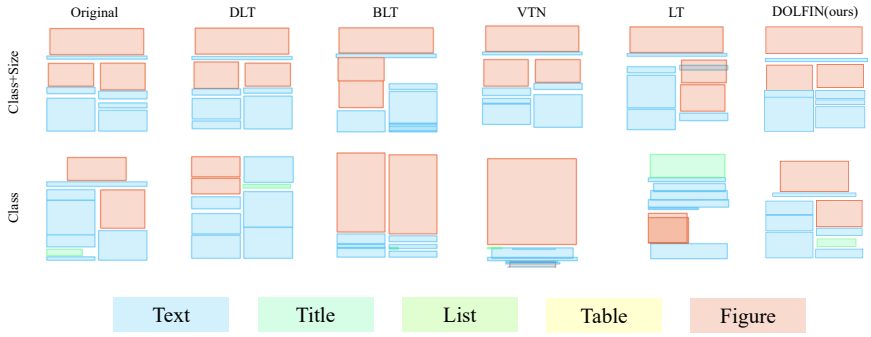


Fig. 5: Conditional generation on the PublayNet dataset. The first 5 columns are from paper DLT [29], each including the visual results of onditional generation (conditioned on Class+Size and Class) by DLT [29], BLT [27], VTN [1] and LT [15] respectively. The last column contains the visual results of our Dolfin method.

visualizations for cate.-conditioned generation in the supplementary material, which shows more diverse results under the same conditions in comparison with others. For more results of conditional and unconditional generation, including datasets COCO [33], Magazine [50] and TextLogo3K [48], please refer to the supplementary material.

5.3 Layout Generation User Study

We perform a user study to assess the qualitative results (see Tab. 6) based on Fig. 4 and Fig. 5. 53 participants are asked to rank the top three generated layouts from various methods. Clearly, Dolfin is the most favourable.

	Unconditional Generation							Conditional Generation				
Method	LT	MaskGIT	BLT	BART	VQDiff	LayoutDM	Dolfin	LT	BLT	VTN	DLT	Dolfin
% of participants	5.7	11.3	1.9	11.3	3.8	22.6	**43.4**	1.9	9.4	3.8	26.4	**58.5**

Table 6: User Study. % of participants who rank the model as Top 1.

5.4 Line Segments Generation

We train our non-autoregressive Dolfin model on the ShanghaiTech Wireframe dataset [21], which consists of 5000 training images. We extract the line segments from each image and use them directly as the input of Dolfin for training.

We present line segments generation results in the supplementary material. We also show the output of ControlNet [54] based on the line segments for better illustration in the supplementary material. For quantitative results and

comparisons, we have diligently searched for relevant works in the same domain, but regrettably, we could not identify any. To make comparisons, we reimplement other models to handle the task as our baseline models, specifically LT [15] and LayoutGAN++ [24]. Our proposed model demonstrates superior line segment generation results compared to these baselines. For a comprehensive evaluation, we utilize the FID score [16] and an additional metric (Difference) as detailed in supplementary material. The numerical results are shown in Tab. 3.

6 Ablation Study

To show the effectiveness of our method Dolfin as well as to reveal the impacts of some parameters, we conduct several ablation studies. (1) Compare the results of encoding the input to latent space and directly operating on input space. (2) Compare the results of using different transformer architectures. (3) Measure the time cost of sampling a layout using our model with different number of tokens.

6.1 Direct operate on the input space

We extend our model by incorporating a two-stage architecture, where we introduce a Multilayer Perceptron (MLP) as an autoencoder. This MLP is trained to map the input tensor into a latent space representation. By leveraging this two-stage model, our approach operates in the latent space, similar to the methodology proposed in the PLay [5] framework. Instead of directly sending the tensor into the transformer block, we initially encode it using the MLP, obtaining a latent representation, which is subsequently fed into the transformer block for further processing.

We evaluate the performance of this extended model through quantitative analysis, as presented in Tab. 7. The results indicate that the model with the MLP performs significantly worse compared to the model without it. This suggests that the introduction of the MLP autoencoder does not yield improvements in terms of the evaluated metrics.

6.2 Adjusting Transformer Architectures

We conducted experiments with different transformer structures, including 4×384 and 8×512, in addition to the original 4×512 transformer block. The purpose was to explore the impact of varying the number of layers and hidden dimensions

Method	Align. ↓	FID ↓
Dolfin (MLP Autoencoder)	0.129	15.50
Dolfin (Regular)	**0.074**	**9.12**

Table 7: Unconditional Generation On PublayNet (Use MLP as autoencoder).

Transformer	4 × 384	4 × 512	8 × 512
FID Score	9.49	9.12	9.45
Alignment	0.062	0.074	0.057

Table 8: Unconditional generation on PublayNet with different architectures.

	FID ↓	Align. ↓	Unique Match ↑
Original	**9.12**	**0.074**	**762**
Separate	10.92	0.097	748

Table 9: Comparision of whether to treat different parts separately. (PubLayNet)

	FID ↓	Align. ↓	Unique Match ↑
Original	**15.38**	**0.094**	**873**
Separate	19.18	0.122	868

Table 10: Comparision of whether to treat different parts separately. (RICO)

on the performance of our method. By evaluating the unconditional generation FID and alignment score on PublayNet dataset, we observed that increasing the number of transformer parameters does not necessarily lead to improved performance. The results are presented in Tab. 8.

6.3 Process discrete and continuous categories separately

We do additional experiments by replacing Dolfin's input (original geometric items) with encoded discrete and continuous features in DLT. Results are in Tab. 9 and Tab. 10, which show the benefits of operating in original space.

7 Conclusion

In this paper, we propose Dolfin, a novel diffusion layout transformer model for layout generation. Dolfin directly operates on the discrete input space of geometric objects, which reduces the complexity of the model and improves efficiency. We also present an autoregressive diffusion model which helps capture semantic correlations and interactions between input transformer tokens. Our experiments demonstrate the effectiveness of Dolfin that improves current state-of-the-arts across multiple metrics. In addition, Dolfin can be extended beyond layout generation, e.g. to modeling other geometric structures such as line segments.
Limitation Dolfin has limitations with complex structured data. Training is costly since we employ DiT as our base model. Additionally, overlapping exists for small items in conditional cases, as shown in Fig. 6.

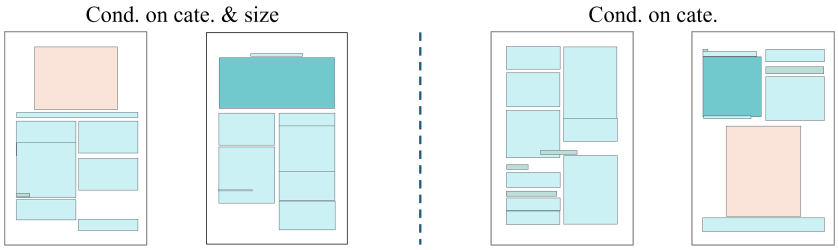


Fig. 6: Failure cases. Overlapping exists for small items in conditional cases.

Acknowledgements

This work is supported by NSF Award IIS-2127544. We thank Saining Xie for the helpful discussion. We also thank Shang Chai and Chin-Yi Cheng for providing instructions about metric computations, and Zhizhou Sha for the help in rebuttal.

References

1. Arroyo, D.M., Postels, J., Tombari, F.: Variational transformer networks for layout generation (2021)
2. Chai, S., Zhuang, L., Yan, F.: Layoutdm: Transformer-based diffusion model for layout generation (2023)
3. Chang, H., Zhang, H., Jiang, L., Liu, C., Freeman, W.T.: Maskgit: Masked generative image transformer (2022)
4. Chen, H., Gu, J., Chen, A., Tian, W., Tu, Z., Liu, L., Su, H.: Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. arXiv preprint arXiv:2304.06714 (2023)
5. Cheng, C.Y., Huang, F., Li, G., Li, Y.: Play: Parametrically conditioned layout generation using latent diffusion (2023)
6. Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., Kumar, R.: Rico: A mobile app dataset for building data-driven design applications. In: UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology. pp. 845–854. UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, Association for Computing Machinery, Inc (Oct 2017). https://doi.org/10.1145/3126594.3126651, publisher Copyright: © 2017 Copyright held by the owner/author(s).; 30th Annual ACM Symposium on User Interface Software and Technology, UIST 2017 ; Conference date: 22-10-2017 Through 25-10-2017
7. Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., Kumar, R.: 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. p. 845–854. UIST '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3126594.3126651, https://doi.org/10.1145/3126594.3126651
8. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in Neural Information Processing Systems **34**, 8780–8794 (2021)
9. Feng, W., Zhu, W., Fu, T.j., Jampani, V., Akula, A., He, X., Basu, S., Wang, X.E., Wang, W.Y.: Layoutgpt: Compositional visual planning and generation with large language models. arXiv preprint arXiv:2305.15393 (2023)
10. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: CVPR (2003)
11. Fu, T.J., Wang, W.Y., McDuff, D., Song, Y.: Doc2ppt: Automatic presentation slides generation from scientific documents. Proceedings of the AAAI Conference on Artificial Intelligence **36**(1), 634–642 (Jun 2022). https://doi.org/10.1609/aaai.v36i1.19943, https://ojs.aaai.org/index.php/AAAI/article/view/19943
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)

13. Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., Guo, B.: Vector quantized diffusion model for text-to-image synthesis (2022)
14. Guo, S., Jin, Z., Sun, F., Li, J., Li, Z., Shi, Y., Cao, N.: 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, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3411764.3445117, https://doi.org/10.1145/3411764.3445117
15. Gupta, K., Lazarow, J., Achille, A., Davis, L., Mahadevan, V., Shrivastava, A.: Layouttransformer: Layout generation and completion with self-attention. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 984–994 (2021). https://doi.org/10.1109/ICCV48922.2021.00104
16. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium (2018)
17. Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D.P., Poole, B., Norouzi, M., Fleet, D.J., et al.: Imagen video: High definition video generation with diffusion models. arXiv preprint arXiv:2210.02303 (2022)
18. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020)
19. Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., Fleet, D.J.: Video diffusion models. arXiv preprint arXiv:2204.03458 (2022)
20. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
21. Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., Ma, Y.: Learning to parse wireframes in images of man-made environments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 626–635 (2018)
22. Inoue, N., Kikuchi, K., Simo-Serra, E., Otani, M., Yamaguchi, K.: Layoutdm: Discrete diffusion model for controllable layout generation (2023)
23. Jyothi, A.A., Durand, T., He, J., Sigal, L., Mori, G.: Layoutvae: Stochastic scene layout generation from a label set (2021)
24. Kikuchi, K., Simo-Serra, E., Otani, M., Yamaguchi, K.: Constrained graphic layout generation via latent optimization. In: ACM International Conference on Multimedia. pp. 88–96. MM '21 (2021). https://doi.org/10.1145/3474085.3475497
25. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013)
26. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2022)
27. Kong, X., Jiang, L., Chang, H., Zhang, H., Hao, Y., Gong, H., Essa, I.: Blt: Bidirectional layout transformer for controllable layout generation. arXiv preprint arXiv:2112.05112 (2021)
28. Lee, H.Y., Jiang, L., Essa, I., Le, P.B., Gong, H., Yang, M.H., Yang, W.: Neural design network: Graphic layout generation with constraints (2020)
29. Levi, E., Brosh, E., Mykhailych, M., Perez, M.: Dlt: Conditioned layout generation with joint discrete-continuous diffusion layout transformer. arXiv preprint arXiv:2303.03755 (2023)
30. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension (2019)
31. Li, J., Yang, J., Hertzmann, A., Zhang, J., Xu, T.: Layoutgan: Generating graphic layouts with wireframe discriminators (2019)
32. Lin, J., Guo, J., Sun, S., Yang, Z.J., Lou, J.G., Zhang, D.: Layoutprompter: Awaken the design ability of large language models. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)

33. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft coco: Common objects in context (2015)
34. Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G., Furukawa, Y.: House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In: European Conference on Computer Vision. pp. 162–177. Springer (2020)
35. Nauata, N., Hosseini, S., Chang, K.H., Chu, H., Cheng, C.Y., Furukawa, Y.: House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13632–13641 (2021)
36. Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021)
37. Peebles, W., Xie, S.: Scalable diffusion models with transformers. arXiv preprint arXiv:2212.09748 (2022)
38. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)
39. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. pp. 234–241 (2015)
40. Shabani, M.A., Hosseini, S., Furukawa, Y.: Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising (2022)
41. Shabani, M.A., Hosseini, S., Furukawa, Y.: Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5466–5475 (2023)
42. Shue, J.R., Chan, E.R., Po, R., Ankner, Z., Wu, J., Wetzstein, G.: 3d neural field generation using triplane diffusion. arXiv preprint arXiv:2211.16677 (2022)
43. Singh, J., Zheng, L., Smith, C., Echevarria, J.: Paint2pix: Interactive painting based progressive image synthesis and editing (2022)
44. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models (2022)
45. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Learning hierarchical models of scenes, objects, and parts. In: ICCV. vol. 2 (2005)
46. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-resnet and the impact of residual connections on learning. Proceedings of the AAAI Conference on Artificial Intelligence **31**(1) (Feb 2017). https://doi.org/10.1609/aaai.v31i1.11231, https://ojs.aaai.org/index.php/AAAI/article/view/11231
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017)
48. Wang, Y., Pu, G., Luo, W., Wang, Y., Xiong, P., Kang, H., Lian, Z.: Aesthetic text logo synthesis via content-aware layout inferring (2022)
49. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: ECCV (2000)
50. Xinru Zheng, Xiaotian Qiao, Y.C., Lau, R.W.: Content-aware generative modeling of graphic design layouts. ACM Transactions on Graphics (Proc. of SIGGRAPH 2019) **38** (2019)
51. Yamaguchi, K.: Canvasvae: Learning to generate vector graphic documents. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5461–5469 (2021). https://doi.org/10.1109/ICCV48922.2021.00543
52. Yang, X., Mei, T., Xu, Y.Q., Rui, Y., Li, S.: Automatic generation of visual-textual presentation layout. ACM Transactions on Multimedia Computing, Communications, and Applications **12**, 1–22 (02 2016). https://doi.org/10.1145/2818709

53. Zhang, J., Guo, J., Sun, S., Lou, J.G., Zhang, D.: Layoutdiffusion: Improving graphic layout generation by discrete diffusion probabilistic models. In: ICCV (2023)
54. Zhang, L., Agrawala, M.: Adding conditional control to text-to-image diffusion models. arXiv preprint arXiv:2302.05543 (2023)
55. Zheng, X., Qiao, X., Cao, Y., Lau, R.W.H.: Content-aware generative modeling of graphic design layouts. ACM Trans. Graph. **38**(4) (jul 2019). https://doi.org/10.1145/3306346.3322971, https://doi.org/10.1145/3306346.3322971
56. Zhong, X., Tang, J., Yepes, A.J.: Publaynet: largest dataset ever for document layout analysis (2019)