DLT: Conditioned layout generation with Joint Discrete-Continuous Diffusion Layout Transformer

Elad Levi, Eli Brosh, Mykola Mykhailych, Meir Perez
Wix.com

Abstract

Generating visual layouts is an essential ingredient of graphic design. The ability to condition layout generation on a partial subset of component attributes is critical to real-world applications that involve user interaction. Recently, diffusion models have demonstrated high-quality generative performances in various domains. However, it is unclear how to apply diffusion models to the natural representation of layouts which consists of a mix of discrete (class) and continuous (location, size) attributes. To address the conditioning layout generation problem, we introduce DLT, a joint discrete-continuous diffusion model. DLT is a transformer-based model which has a flexible conditioning mechanism that allows for conditioning on any given subset of all the layout component classes, locations, and sizes. Our method outperforms state-of-the-art generative models on various layout generation datasets with respect to different metrics and conditioning settings. Additionally, we validate the effectiveness of our proposed conditioning mechanism and the joint continuous-diffusion process. This joint process can be incorporated into a wide range of mixed discrete-continuous generative tasks. More information can be found on our project webpage: https://wix-incubator.github.io/DLT

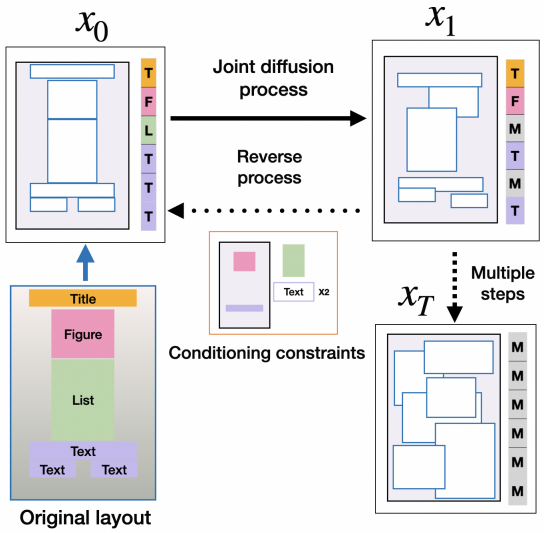

Figure 1: Overview of our Joint Discrete-Continuous diffusion process for layout generation. The diffusion process is applied jointly on both the continuous attributes of the components (size and location) by adding a small Gaussian noise, and on the discrete attributes of the components (class) by adding a noise that adds a mass to the Mask class. The reverse diffusion model can be conditioned on any given subset of the attributes.

1. Introduction

An essential aspect of graphic design is layout creation: the arrangement and sizing of visual components on a canvas or document. A well-designed layout enables users to easily comprehend and efficiently interact with the information presented. The ability to generate high-quality layouts is crucial for a range of applications, including user interfaces for mobile apps and websites [4] and graphic design for information slides [6], magazines [38], scientific papers [40], infographics [28], and indoor scenes [5].

To facilitate graphic design tasks, modelling approaches such as Generative Adversarial Networks (GANs) [18], Variational Autoencoders (VAEs) [14, 1], and masking completion [36, 16, 7] were explored for generating novel layouts. These approaches employ various architectures such as Recurrent Neural Networks (RNNs) [14], and Graph Neural Networks (GNNs) [17]. Recently proposed, Transformer-based approaches [1, 36, 16, 7] have been shown to produce diverse and plausible layouts for a variety of applications and user requirements.

Layouts are often represented as a set of components, each consisting of several attributes, such as class, position, and size [18]. Inspired by recent advancements in NLP, a common practice is to model a component as a sequence of discrete attribute tokens, and a layout as the concatenation of all attribute tokens [36, 7, 16]. The model is trained using a self-supervised objective (either next-word completion or

unmasking) to generate an output sequence. The discretization process is done by binning geometry attributes, such as position and size. However, using discrete tokens fails to take into account the natural representation of (geometry) data, which is continuous, and either result in poor resolution of layout composition, or in a large sparse vocabulary that is not covered well by smaller datasets.

Providing controllable layout generation is crucial for many real-world graphic design applications that require user interaction. We would like to allow a user to fix certain component attributes, such as component class, or a set of components, and then generate the remaining attributes (position and size) or components. In recent iterative-based masking completion approaches [16], such conditioning is performed only during inference, in a way that does not provide the model with the ability to separate between the conditioned and generated parts of the layout during the iterative refinement. This lack of separation might lead to ambiguity in the generation process which can result in suboptimal performance.

Diffusion models are a generative approach that has gained significant attention recently. Continuous diffusion models have demonstrated remarkable generative capabilities in various tasks and domains, such as text-to-image [29], video generation [11], and audio [39]. While diffusion models were extended to discrete spaces [41, 2], a challenge still remains on how to apply these models to generative tasks whose representation consists of both continuous and discrete features. In this paper, we introduce a Diffusion Layout Transformer (DLT) for layout generation and flexible editing. Different from traditional diffusion models, we propose a novel framework based on a joint continuous-discrete diffusion process and provide theoretical justification for the derived optimization objective. Using a transformer-encoder architecture, we apply a diffusion process jointly both on the continuous attributes of the layout components (size and location) and on the discrete ones (class), as depicted in Figure 1. Our diffusion model addresses an important limitation of transformer-based models that rely on a discrete layout representation, which can result in reduced resolution and limited diversity of layout compositions.

Our framework offers flexible layout editing which allows for conditioning on any subset of the attributes of the layout components provided (class, size and location). Unlike masking completion approaches [36, 16, 7], which perform conditioning only during inference, and inspired by diffusion image inpainting techniques [24], we explicitly train the model to perform layout conditioning. We use condition embedding to control on which layout attributes to apply the diffusion process. In this way, the model is able to separate between the condition and the generation parts during the iterative refinement. Our experiments reveal that these design choices significantly improve the model's performance compared to inference-based conditioning.

We study the effectiveness of our DLT model by evaluating it on three popular layout datasets of diverse graphic design tasks over several common metrics. Using extensive experiments, we demonstrate that our proposed model outperforms state-of-the-art layout generation models on layout synthesis and editing tasks while maintaining runtime complexity on par with these models. By comparing different alternatives, we show that both the joint diffusion process and the conditioning mechanism contribute to the model's strong generative capabilities. Our joint discrete-continuous diffusion framework has a generic design and thus is applicable to other domains such as multi-modal generation tasks (text+image, text + audio, etc').

2. Related work

Layout synthesis. The task of generating high-quality, realistic and diverse layouts using data-driven generative methods has received increased attention in recent years. Several generative approaches have been proposed to address this problem. For example, LayoutGAN [18] trains a generator to map a gaussian/uniform distribution to the locations/categories of layout components by rendering a wireframe and training a pixel-based discriminator. LayoutVAE [14] introduces two types of Variational Auto-Encoders. The first one learns the distribution of category counts of components, while the second one predicts the locations and sizes of components, conditioning on the categories. Motivated by the recent advancements in NLP, transformer-based language models were proposed to solve the layout generation task. In [7], a standard transformer-decoder model was applied to generate the attributes of layout components, where these components are sorted in lexicographic order. Training is performed with a self-supervised objective of predicting the next attribute given the attributes of previous components. In BLT [16], a bidirectional transformer encoder was used to predict the attributes of layout components. In order to improve the quality of the generated layouts, they iteratively refine layout attributes by remasking the attributes with the lowest probability. Unlike previous methods, our generative method is based on a diffusion process (see Section 3.2), which has demonstrated strong generative capabilities in various fields.

Layout conditioning. In order to be applicable to many real-world graphic design applications, where user interaction is required, various methods were proposed for controllable layout generation. LayoutGAN++ [15] improves LayoutGAN [18] by replacing the generator and the discriminator with a transformer-based architecture. They propose to add constraints on the layout generation by defining and solving a nonlinear optimization problem in the la-

tent space. In the case of VAE, Lee et al [17] proposed a method to constraint the layout generation on a set of relative design properties using a graph-based conditional VAE model. VTN [1] is a transformer-based conditional VAE approach in which the conditioning is performed either by an RNN that maps the condition to a prior distribution or by a transformer encoder. Different from these conditioning approaches which impose restrictions in the latent space, in BLT [16] the conditioning is done simply by unmasking the conditioned component attributes. Similar to [16], our approach also allows the same flexible controllable conditioning, and during inference, the model is explicitly informed of the conditioning attributes. However, we propose to perform the optimization of the conditioning scenarios during training, which enhances the model's performance.

Diffusion Generative Models. Diffusion models are a family of neural generative models which parameterize a reverse Markov noising process [10]. In the continuous diffusion case, a small amount of Gaussian noise is added to the sample. Continuous diffusion models have shown strong generative capabilities across diverse tasks and modalities such as text-to-image [24, 29, 31, 30], video generation [32, 9], text-to-3D [27, 19] and audio [39, 20, 13]. Recently, transformer-based diffusion models have demonstrated state-of-the-art performance in various tasks such as text-to-image generation [26] and human motion generation [35]. Inspired by these works, our diffusion model is also based on a transformer-encoder architecture.

Discrete diffusion process. In order to generalize the diffusion process to discrete cases, Austin et al. [2] suggest applying the Markov noising process on the probability vector of categories. The effectiveness of this method was demonstrated in [41] on the image captioning task, where the stationary distribution has all the mass on the [MASK] absorbing class. The natural representation of the layout generation task contains both continuous (size and location) and discrete (category) component attributes. Different from previous approaches which apply either continuous or discrete diffusion process, our approach applies the diffusion process on the joint distribution.

3. Method

Our goal is to generate layouts conditioning on constraints c. The layout is represented by a set of N components $\{B_i\}_{i=1}^{N}$, where each component B_i consists of a bounding box describing the location and size of the component in the layout, and the component class. Similar to [16], the condition is a subset of all locations, sizes and classes of the layout components. In particular, unconditional layout generation is also supported by setting $c = \emptyset$.

3.1. Diffusion model

Following the framework in [10], a diffusion process is modeled as a Markov noising process with T steps $\{\bar{x}_t\}_{t=0}^{T}$, where $\bar{x}_0 \sim q(\bar{x}_0)$ is sampled from the real data distribution, and the forward process is sampled according to $\bar{x}_{t+1} \sim q(\bar{x}_{t+1}|\bar{x}_t)$, where $q(\bar{x}_{t+1}|\bar{x}_t)$ is a predefined (and known) distribution. A diffusion model defines the parameterized reverse diffusion process as:

$$p_\theta(\bar{x}_{0:T}) = p(\bar{x}_T)\prod_{t=1}^{T} p_\theta(\bar{x}_{t-1}|\bar{x}_t)$$

Fitting $p_\theta(\bar{x}_0)$ to match the real data distribution $q(x_0)$ is done by optimizing a variational bound on the negative log-likelihood. This variational bound can be expressed as a sum of three terms (see eq.(5) in [10]):

$$\begin{cases} L_T = D_{KL}(q(\bar{x}_T|\bar{x}_0)||p(\bar{x}_T)) \\ L_t = D_{KL}(q(\bar{x}_{t-1}|\bar{x}_t,\bar{x}_0)||p_\theta(\bar{x}_{t-1}|\bar{x}_t)) \quad 1 \le t \le T-1 \\ L_0 = -log(p_\theta(\bar{x}_0|\bar{x}_1)) \end{cases}$$

Since L_T is constant it can be ignored in the optimization process.

Continuous diffusion models. We denote the functions in the continuous process with the superscript c. Following [10], the forward process is performed by adding a Gaussian noise

$$q^c(\bar{x}_t|\bar{x}_{t-1}) = \mathcal{N}(\bar{x}_t; \sqrt{1-\beta_t}\bar{x}_{t-1}, \beta_t \cdot I),$$

where a constant variance $\beta_t \in (0,1)$ (a hyper-parameter) controls the sizes of the steps. When $T \to \infty$ we can approximate $\bar{x}_T \sim \mathcal{N}(0, I)$. In this case, $p_\theta^c(\bar{x}_{t-1}|\bar{x}_t)$ is parameterized by a family of Gaussian distributions, and L_t^c can be computed in a closed form.

In our setting, we apply the diffusion process on a set of all the bounding boxes in the layout: $\bar{x}_0 \in \mathbb{R}^{NX4}$, where N is the maximal number of components. For the reverse diffusion process, we follow [29, 35], and train a model $F_\theta^c(\bar{x}_t, c, \bar{y}_t)$ to directly predict $\bar{x}_0$ from the noise, where $\bar{y}_t$ carries the class information in the diffusion process, as described in the next paragraph. The final loss $\mathcal{L}_{box}$ is a reweighted version of the loss L_t^c, which simplifies the objective and empirically leads to better results:

$$\mathcal{L}_{box} = \mathbb{E}_{\bar{x}_0,\bar{y}_0\sim q(\bar{x}_0,\bar{y}_0|c),t\sim[0,1]}||F_\theta^c(\bar{x}_t, c, \bar{y}_t) - \bar{x}_0||^2$$

Discrete diffusion models. We denote the functions in the discrete process with the superscript d. The discrete diffusion process was defined in [2]. In this setting, we have discrete random variables with K categories, $y_0 \ldots, y_T \in \{1, \ldots, K\}$, where y_0 is drawn from the data distribution, and the forward noising process is defined by the Markov transition matrix:

$$[Q_t]_{i,j} = q^d(y_t = i|y_{t-1} = j)$$

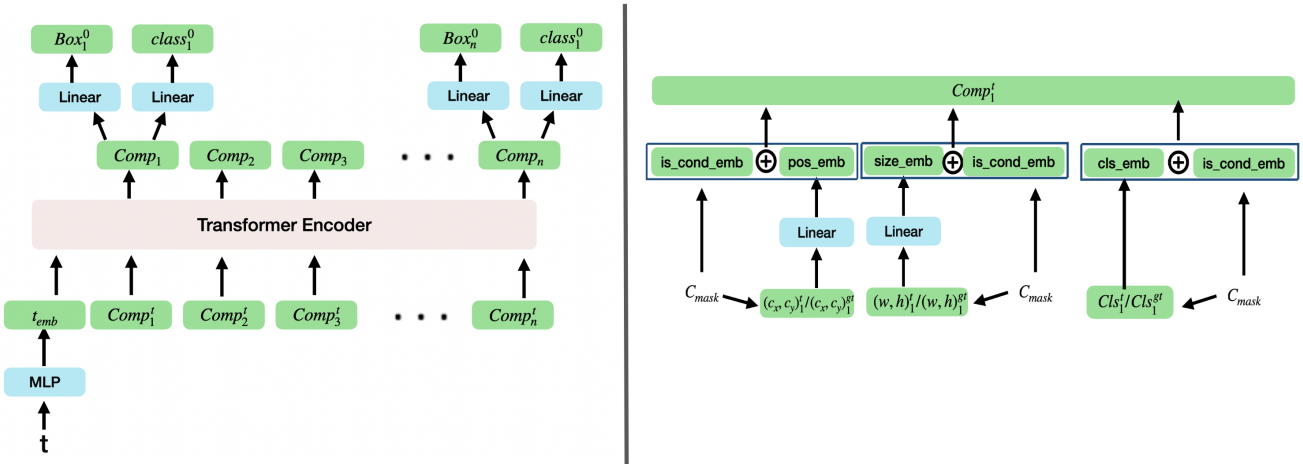

Figure 2: DLT model architecture **Left:** The transformer is fed with an embedding that represents step t of the diffusion process and an embedding for each of the layout components. The output is the cleaned coordinates and classes of components. During inference, the output is noised back to step $t-1$. **Right:** The component embedding consists of a concatenation of position embedding, size embedding and class embedding. Each one of the three sub-embeddings can be either generated from the diffusion process or from the conditioning information. The model is explicitly informed which part of the input is part of the diffusion process by adding a trainable vector to the sub-embedding in case the attribute is part of the conditioning. This condition embedding is separate for each attribute.

In our case, we apply the discrete diffusion process on the set of all the classes of layout components $\bar{y}_0 \in \{0, .., K\}^N$. We follow [2, 41] and choose a transition matrix with an absorbing state as the K-th category, denote by [Mask]. In this case for $i < K$:

$$[Q_t]_{i,j} = \begin{cases} \beta & i = j \\ 1-\beta & i = K\ ([Mask]) \\ 0 & else \end{cases}$$

and for $i = K$:

$$[Q_t]_{K,j} = \begin{cases} 1 & j = K \\ 0 & j \neq K \end{cases}$$

In other words, if the class is [Mask], it will remain the same. If not, then with probability β the class remains the same, and with probability $1-\beta$ it will be changed to $[Mask]$. Observe that when $T \to \infty$ we can approximate $\bar{y}_T = [Mask]^N$. In the reverse diffusion process, we train the model $F_\theta^d(\bar{x}_t, c, \bar{y}_t)$ to directly predict the class probabilities of the components $\tilde{P}_\theta(\tilde{y_0}|\bar{y}_t, \bar{x}_t, c)$. The parameterization of the reverse diffusion probability is defined using the forward process

$$p_\theta^d(\bar{y}_{t-1}|\bar{y}_t, \bar{x}_t, c) = \sum_{\tilde{y}_0 \in K^N} q(\bar{y}_{t-1}|\bar{y}_t, \tilde{y}_0) \cdot \tilde{P}_\theta(\tilde{y_0}|\bar{y}_t, \bar{x}_t, c)$$

In this case, the regular cross-entropy loss is a reweighted function of L_t^d (see A.3 in [2]):

$$\mathcal{L}_t = CE(F_\theta^d(\bar{x}_t, c, \bar{y}_t), \bar{y}_0)$$

Following [41], our objective on the class category is

$$\mathcal{L}_{cls} = \mathbb{E}_{\bar{y}_0, \bar{x}_0 \sim q(\bar{y}_0, \bar{x}_0|c), t\sim[0,1]} CE(F_\theta^d(\bar{x}_t, c, \bar{y}_t), \bar{y}_0)$$

Combined diffusion process. Our layout diffusion process is obtained by sampling $(\bar{x}_0, \bar{y}_0)$ from the data distribution. The forward diffusion process is obtained by applying the continuous process q^c and the discrete process q^d independently on $\bar{x}_0, \bar{y}_0$. The model is trained with the combined objective:

$$\mathcal{L}_{model} = \lambda_1 \cdot \mathcal{L}_{box} + \lambda_2 \cdot \mathcal{L}_{cls}$$

The theoretical motivation for the combined loss is explained by parameterizing the reverse process joint distribution as independent over the previous step:

$$p_\theta(\bar{x}_{t-1}, \bar{y}_{t-1}|\bar{x}_t, \bar{y}_t) = p_\theta^c(\bar{x}_{t-1}, |\bar{x}_t, \bar{y}_t) \cdot p_\theta^d(\bar{y}_{t-1}, |\bar{x}_t, \bar{y}_t)$$

Observe that in this decomposition, the probabilities of the continuous part and the discrete part are still dependent on both the continuous coordinates and the discrete classes of the previous diffusion step. With this parametrization, by the additive property of the KL-divergence for independent distributions we have:

$$L_t = L_t^c + L_t^d$$

Therefore, our final loss is a simplified reweight optimization of the variational bound of the joint distribution negative log-likelihood function.

Dataset	**Publaynet**											
	Conditioned on Category				Category + Size				Uncondtioned			
Model	pIOU	Overlap	Alignment	FID	pIOU	Overlap	Alignment	FID	pIOU	Overlap	Alignment	FID
LT [7]	2.7	7.6	0.41	26.8	7.1	11.7	0.14	22.0	0.62	2.4	0.11	19.3
BLT [16]	0.89	4.4	**0.10**	36.6	1.7	8.1	**0.09**	14.2	0.60	2.7	0.12	69.8
VTN [1]	2.1	6.8	0.29	22.1	5.3	15.3	**0.09**	17.9	0.68	**2.6**	**0.08**	14.5
DLT	**0.67**	**3.8**	0.11	**10.3**	**0.82**	**4.2**	**0.09**	**11.4**	**0.59**	**2.6**	0.11	**13.8**
Dataset	**Rico**											
	Conditioned on Category				Category + Size				Uncondtioned			
Model	pIOU	Overlap	Alignment	FID	pIOU	Overlap	Alignment	FID	pIOU	Overlap	Alignment	FID
LT [7]	25.6	75.2	0.58	14.7	23.8	**69.1**	0.41	8.4	23.2	65.1	0.40	15.2
BLT [16]	30.2	85.1	**0.12**	27.8	24.5	79.3	0.30	10.2	23.0	70.6	0.25	18.7
VTN [1]	25.4	74.2	0.43	14.3	24.1	69.6	0.44	7.1	29.4	72.1	0.26	29.4
DLT	**21.9**	**70.6**	0.18	**9.5**	**17.2**	70.2	**0.28**	**6.3**	**19.3**	**58.4**	**0.21**	**13.9**
Dataset	**Magazine**											
	Conditioned on Category				Category + Size				Uncondtioned			
Model	pIOU	Overlap	Alignment	FID	pIOU	Overlap	Alignment	FID	pIOU	Overlap	Alignment	FID
LT [7]	19.9	71.0	1.5	44.7	21.4	70.2	**1.2**	45.3	21.4	70.0	1.1	42.6
BLT [16]	36.4	133	1.4	49	20.5	56.8	**1.2**	27.3	30.1	134	1.1	52.7
VTN [1]	10.3	38.7	2.4	37.6	9.9	28.8	2.3	29.4	20.1	70.7	**0.9**	62.7
DLT	**5.9**	**16.1**	**1.3**	**26.2**	**6.8**	**19.4**	1.6	**21.7**	**4.8**	**12.1**	1.8	**40.9**

Table 1: Quantitative comparison of layout generation with various constraints setting. The values of Alignment, Overlap, and pIOU are multiplied by 100 for clarity. The DLT method achieve overall the best results among all tested model.

Sampling. In the inference step, we sample $\bar{x}_0 \sim \mathcal{N}(0, I)$, $\bar{y}_0 = [Mask]^N$. In each step t, we apply $F^c_\theta(\bar{x}_t, c, \bar{y}_t)$, $F^d_\theta(\bar{x}_t, c, \bar{y}_t)$, getting $\bar{x}_{pred}$, $\bar{y}_{pred}$ (where we keep class prediction as probability vector). We then apply the noise and sample according to

$$(\bar{x}_{t-1}, \bar{y}_{t-1}) \sim p_\theta(\bar{x}_{t-1}, \bar{y}_{t-1} | \bar{x}_t, \bar{y}_t, c)$$

3.2. Model

Our DLT model is illustrated in Figure 2. Similarly to [35] the backbone of F_θ is a multi-layer Transformer encoder [37]. The input for the transformer network is a set of embedding vectors of the components and an embedding that encodes the process's time-step t. This time embedding is a result of an MLP network that injects the time into the transformer embedding dimension. The transformer encoder outputs an embedding for each component, which is fed to the box head and the class head, both of which are a simple Linear layer. The box head predicts directly the box coordinates, while the class head predicts the class logits.

The component embedding consists of a concatenation of three different embeddings: (1) The position embedding, (2) the size embedding, and (3) the class embedding. The position and size embeddings are implemented by a Linear layer that takes the center and size of the components, respectively. The class embedding is implemented as an embedding layer that holds a separate embedding vector for each class.

3.3. Conditioning Mechanisms

We propose to perform layout editing using a flexible conditioning architecture, that allows the user to choose the specific layout information that is given as part of the conditioning, a capability similar to that in [16]. Following early diffusion image-inpainting works [33, 34, 22], the editing process in [16] is performed only in the inference phase by replacing the known information with the noise in the reverse-diffusion process. In contrast, inspired by the recent advancement in the diffusion image inpainting field [24], we explicitly train the model to perform the layout editing. We perform editing by adding a condition embedding (implemented as a binary embedding layer), that informs the model which part of the input is conditioned and which one is part of the diffusion process. This architecture is illustrated in Figure 2. During training, we condition on a random subset of all components' locations, sizes, and classes. We apply the loss only on the unconditioned information.

4. Experiments

In this section, we demonstrate the effectiveness of our DLT model, by evaluating our proposed method on 3 diverse layout benchmarks. Our proposed model is able to outperform state-of-the-art layout generation models on

layout synthesis and editing tasks with respect to various metrics that examine realism, alignment, and overlapping.

4.1. Datasets

We evaluate our model on three popular public layout generation datasets that represent different graphic design tasks. *RICO* [4, 21] provides user interface designs for mobile applications. It contains 91K layouts with 27 different component types. *PubLayNet* [40] contains 330K samples of scientific documents. The layout components consist of five categories: text, title, figure, list, and table. *Magazine* [38] contains 4K magazine pages with components from six categories (texts, images, headlines, over-image texts, over-image headlines, and backgrounds). We follow the evaluation protocol of previous works [17, 18, 15], and exclude layouts with non-top 13 category components in the RICO dataset. We also exclude layouts with more than 10 components in both the RICO and PubLayNet datasets. Following [15], we use 85% of the RICO dataset for training, 5% for validation, and 10% for testing. In PubLayNet and Magazine, we used the official data split.

4.2. Evaluation metrics

We use five common metrics in the literature to evaluate the quality of the generated layout: *Fréchet inception distance* (FID) [8], which measures the distance between the generated layout distribution and the real layout. Following [17], we compute the score by training a network to classify between real layouts and layouts with additional injected noise (for this, we used the intermediate features of the network). *Overlap* [18] measures the total overlapping area between any two components. *Perceptual IOU* (pIOU) [16] computes the overlap of each component divided by the union area of all objects. *Alignment score* [17] measure the alignments between pair of components, where the assumption is that in graphic design every component has to be aligned to at least one of the other components either by the center or by one of the sides. We used *DocSim* [25] to measure the similarity between the generated layouts and the conditioned layouts.

4.3. Conditioning setting

Following [16], we tested three different conditioning layout generation scenarios: *Category* conditioning, where only the component categories are provided and the model has to predict their size and location in the layout; *Category + Size* conditioning, where the model is given also the size of the components and only needs to predict the position; and *Unconditioned* generation, where no information is provided to the model.

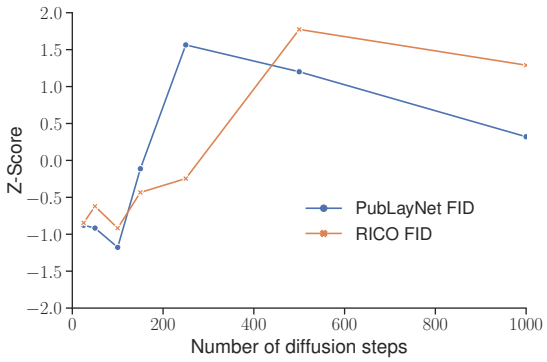

Figure 3: Z-score of the FID metric across a varying number of diffusion steps. Best performances were achieved with 100 diffusion steps.

4.4. Compared methods

We compare our proposed method to state-of-the-art layout generation models. LayoutTransformer (LT) [7] is based on an autoregressive transformer architecture. For the implementation, we used nucleus sampling [12] with $p = 0.9$, which significantly improves the diversity of the results. In order to support the category + size conditioning, we changed the order of components tokens to (w, h, x, y) instead of (x, y, w, h), and verified that this change does not affect the model's performance. For the BLT [16] implementation, we used top-k sampling (k = 5) only in the unconditioned setting as suggested by [16]. We also compare ourselves to the Variational Transformer Network (VTN) [1], a conditional VAE method.

4.5. Implementation details

The transformer encoder was trained with 4 layers, 8 attention heads and a latent dimension of 512. For the component location, we used the same pre-process as in [3]. The diffusion model was trained with a cosine noise schedule [23]. For the discrete process, we used the hyper-parameter $\beta = 0.15$. We set the number of discrete diffusion steps to $T/10$ (where T is the number of the continuous steps). During training and inference, we synced between the discrete and the continuous process by taking $Q^{t//10}$, where $0 \leq t \leq T$ is the continuous step. For the loss hyper-parameters, we used $\lambda_1 = 5$ and $\lambda_2 = 1$.

During training, for each sample, we uniformly sample between the three different conditioning settings: Category, Category + size, and unconditioned. We mask the loss on the outputs which correspond to the conditioned inputs.

4.6. Number of diffusion steps

The performance of the model over different numbers of diffusion steps can be seen in Figure 3. It is important to

Docsim metric	Conditined on Category			Category + Size		
Model	Publaynet	RICO	Magazine	Publaynet	RICO	Magazine
LT [7]	9.3±0.17	15.9±0.52	10.8±0.62	18.5±0.19	26.1±0.71	15.5±0.79
BLT [16]	14.4±0.21	16.4±0.21	12.4±0.70	20.7±0.13	25.8±0.56	19.5±0.95
VTN [1]	9.8±0.23	18.5±0.46	8.5±0.65	19.7±0.21	28.1±0.67	7.8±0.83
DLT	**16.2**±0.23	**20.4**±0.42	**13.5**±0.72	**24.3**±0.17	**28.8**±0.48	**22.7**±0.81

Table 2: Docsim metric results on the tested datasets in the two conditioning scenarios, results are multiplied by 100 for clarity. Our method outperforms all tested models both when conditioning on category and when conditioning on category+size.

Setting	Publaynet Category+Size			
Model	pIOU	Overlap	Alignment	FID
Edit only in inference	5.8	16.6	0.19	14.1
Without condition embedding	1.1	5.3	0.17	4.3
DLT	**0.7**	**4.5**	**0.13**	**2.9**

Table 3: Quantitative comparison of different conditioning approaches on the Publaynet dataset. Conditioning is done on all the component sizes and categories + half of the component locations. The values of Alignment, Overlap, and pIOU are multiplied by 100 for clarity.

note that the DLT inference time is a function of the number of diffusion steps and is not dependent on the number of components, unlike the other models we tested. Given the results in Figure 3, we choose $T = 100$ diffusion steps for all our tested experiments. This means that our model inference time is ~ 2 times slower on the RICO and Publaynet datasets. However, our model is ~ 1.65 times faster on the Magazine dataset.

4.7. Results

Each model was tested with four trials (on the test set). We report the averaged results of the metrics in Table 1, and the standard deviation results in Appendix A. While the BLT model performs better than the LT model in the conditioning scenarios, and the LT model performs better in the unconditioned setting, our model outperforms the tested methods by statistically significant margins in both scenarios with respect to all tested metrics. The only exception is the alignment score where in some of the cases the BLT model performs better. This is due to the small number of bins (32) in the discretization of the bounding box dimensions. It is important to note that this discretization limits the resolution of the components in the generated layouts.

Results of the Docsim measurement between the generated layouts and the conditioning layout can be seen in Table 2. Our model outperforms baseline models in the conditioning scenarios also with respect to this metric. Furthermore, Figure 4 provides a qualitative comparison of the generated layouts which highlights the common failure cases. For more results, see Appendix A.

4.8. Key components Analysis

In order to investigate the effectiveness of our proposed method, we conduct an ablation study on the key components of the model.

Conditioning Mechanism. We compare our conditioning mechanism, given in Section 3.3, to two alternatives: (1) Performing editing only during inference, and (2) Removing the condition embedding. In (1), we train the model only according to the unconditioned setting (see Section 4.5). At the inference step, we override the noise in the reverse diffusion process with the conditioned information. It is important to note that this approach (at the discrete case) is equivalent to the conditioning approach in [16]. In option (2), we still sample during training from the three conditioning settings as described in Section 4.5, with a slight modification that we condition also on half of the components (chosen randomly). In this option, we remove the condition embedding. This means that the model does not have explicit information if the input is part of the diffusion process or the real conditioning input. The DLT model was also trained with the same sampling as in option (2).

A comparison of the methods on the Publaynet dataset, conditioning on the Category+Size of all the components and on all the information of half of the components, is given in Table 3. As seen, performing conditioning only during inference significantly degraded the model's performance, whereas adding the condition embedding further improves the results. This information (which component is part of the conditioning) allows the model to resolve the conditioning ambiguity and align the generated components according to the conditioned components. For more details, see Appendix B.

Joint Continues-Discrete diffusion process. In order to evaluate the effectiveness of our proposed joint continues-discrete diffusion process, we compare it to running two sequential diffusion processes independently. In the first compared model, we first run a discrete diffusion process on the category. During training, we replace the unconditioned setting described in Section 4.5 by conditioning on the box location and size where we set the location and size of all boxes to be zero. In this way, the model is trained on each sample either to reverse the discrete process or the continuous process (but not on both of them). At inference, we

Figure 4: Qualitative comparison of generated layouts between all tested methods on PubLayNet dataset with the three tested conditioning settings.

first run the discrete diffusion process on the component categories (with the location and size equal to zero), and then given the predicted categories we run the continuous reverse diffusion process conditioning on the reverse discrete process results. In the second compared method, we first run the continuous reverse diffusion process and then run the discrete reverse process. In training, this is done by replacing the unconditioned setting with two settings: (1) Running the continuous diffusion process on the component location + size conditioned on all categories equal to the mask token. (2) Running the discrete diffusion process conditioned on the components' locations + sizes. Observe that also in this method for each sample we train the model to reverse either the discrete or the continuous diffusion process. At inference, we first run the reverse continuous process conditioned on all categories equal to the mask token to predict the component location. Then we run the reverse discrete process to predict the component category conditioned on the box location.

We provide results for the unconditioned setting over the Publaynet dataset in Table 4. As can be seen, our proposed joint continuous-discrete diffusion process outperforms the sequential diffusion process. It is also important to note that the inference time of the joint model is twice faster.

Setting	Publaynet Unconditioned			
Model	pIOU	Overlap	Alignment	FID
Class before boxes	0.59	2.1	0.11	50.8
Class after boxes	0.53	2.3	0.14	20.5
DLT (Joint process)	**0.23**	**1.8**	**0.11**	**20.1**

Table 4: Quantitative comparison of different approaches to combine the discrete and the continuous diffusion processes on the Publaynet dataset in the unconditioned setting. The values of Alignment, Overlap, and pIOU are multiplied by 100 for clarity.

5. Conclusions and future work

In this work, we present a new method for generating synthetic layouts given partial information on the layout structure. We introduce the DLT model, a Diffusion layout transformer that performs a joint continuous-discrete diffusion process on the dimensions and categories of layout components. The model offers flexible conditioning on the input by using a condition embedding that explicitly specifies on which part of the components the diffusion process is applied. Experimental results on three public datasets over three different conditioning settings demonstrate the effectiveness and flexibility of our proposed method, and that it outperforms existing methods. We further investigate the effectiveness of our model's key components. By comparing different alternatives, we show the added value both of the joint continuous-discrete process and the conditioning mechanism. A limitation of our current work is that the generation process is done without considering the content of the components. This information is very important for resolving many ambiguities in the generation process and can lead to significant improvements in the results. Since the embedding inputs for the DLT transformer backbone represent a layout component, it is natural to extend our solution by encoding more information about the component content in the component embedding. We intend to further explore this direction in future work.

References

[1] Diego Martín Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. *CoRR*, abs/2104.02416, 2021. 1, 3, 5, 6, 7

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *CoRR*, abs/2107.03006, 2021. 2, 3, 4

[3] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. *arXiv preprint arXiv:2211.09788*, 2022. 6

[4] 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. 1, 6

[5] Xinhan Di and Pengqian Yu. Multi-agent reinforcement learning of 3d furniture layout simulation in indoor graphics scenes. *CoRR*, abs/2102.09137, 2021. 1

[6] Mengxi Guo, Dangqing Huang, and Xiaodong Xie. The layout generation algorithm of graphic design based on transformer-cvae. *CoRR*, abs/2110.06794, 2021. 1

[7] Kamal Gupta, Alessandro Achille, Justin Lazarow, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layout generation and completion with self-attention. *CoRR*, abs/2006.14615, 2020. 1, 2, 5, 6, 7

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. 6

[9] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey A. Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *CoRR*, abs/2210.02303, 2022. 3

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. 3

[11] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022. 2

[12] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019. 6

[13] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. *CoRR*, abs/2301.12661, 2023. 3

[14] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. *CoRR*, abs/1907.10719, 2019. 1, 2

[15] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. *CoRR*, abs/2108.00871, 2021. 2, 6

[16] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. BLT: bidirectional layout transformer for controllable layout generation. *CoRR*, abs/2112.05112, 2021. 1, 2, 3, 5, 6, 7

[17] Hsin-Ying Lee, Weilong Yang, Lu Jiang, Madison Le, Irfan Essa, Haifeng Gong, and Ming-Hsuan Yang. Neural design network: Graphic layout generation with constraints. *CoRR*, abs/1912.09421, 2019. 1, 3, 6

[18] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators. *CoRR*, abs/1901.06767, 2019. 1, 2, 6

[19] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. *CoRR*, abs/2211.10440, 2022. 3

[20] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo P. Mandic, Wenwu Wang, and Mark D. Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *CoRR*, abs/2301.12503, 2023. 3

[21] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. Learning design semantics for mobile apps. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 569–579, New York, NY, USA, 2018. ACM. 6

[22] Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. *CoRR*, abs/2108.01073, 2021. 5

[23] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *CoRR*, abs/2102.09672, 2021. 6

[24] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. *CoRR*, abs/2112.10741, 2021. 2, 3, 5

[25] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. READ: recursive autoencoders for document layout generation. *CoRR*, abs/1909.00302, 2019. 6

[26] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022. 3

[27] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *CoRR*, abs/2209.14988, 2022. 3

[28] Chunyao Qian, Shizhao Sun, Weiwei Cui, Jian-Guang Lou, Haidong Zhang, and Dongmei Zhang. Retrieve-then-adapt: Example-based automatic generation for proportion-related infographics. *CoRR*, abs/2008.01177, 2020. 1

[29] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125, 2022. 2, 3

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021. 3

[31] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *CoRR*, abs/2205.11487, 2022. 3

[32] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. *CoRR*, abs/2209.14792, 2022. 3

[33] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. 5

[34] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *CoRR*, abs/1907.05600, 2019. 5

[35] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Amit H Bermano, and Daniel Cohen-Or. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022. 3, 5

[36] Kerem Turgutlu, Sanat Sharma, and Jayant Kumar. Layoutbert: Masked language layout model for object insertion. *CoRR*, abs/2205.00347, 2022. 1, 2

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 5

[38] Ying Cao Xinru Zheng, Xiaotian Qiao and Rynson W.H. Lau. Content-aware generative modeling of graphic design layouts. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2019)*, 38, 2019. 1, 6

[39] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *CoRR*, abs/2207.09983, 2022. 2, 3

[40] Xu Zhong, Jianbin Tang, and Antonio Jimeno-Yepes. Publaynet: largest dataset ever for document layout analysis. *CoRR*, abs/1908.07836, 2019. 1, 6

[41] Zixin Zhu, Yixuan Wei, Jianfeng Wang, Zhe Gan, Zheng Zhang, Le Wang, Gang Hua, Lijuan Wang, Zicheng Liu, and Han Hu. Exploring discrete diffusion models for image captioning. *CoRR*, abs/2211.11694, 2022. 2, 3, 4