LayoutDM: Discrete Diffusion Model for Controllable Layout Generation

Naoto Inoue[1] Kotaro Kikuchi[1] Edgar Simo-Serra[2] Mayu Otani[1] Kota Yamaguchi[1]
[1]CyberAgent, Japan [2]Waseda University, Japan
{inoue_naoto, kikuchi_kotaro_xa}@cyberagent.co.jp ess@waseda.jp
{otani_mayu, yamaguchi_kota}@cyberagent.co.jp

Abstract

Controllable layout generation aims at synthesizing plausible arrangement of element bounding boxes with optional constraints, such as type or position of a specific element. In this work, we try to solve a broad range of layout generation tasks in a single model that is based on discrete state-space diffusion models. Our model, named LayoutDM*, naturally handles the structured layout data in the discrete representation and learns to progressively infer a noiseless layout from the initial input, where we model the layout corruption process by modality-wise discrete diffusion. For conditional generation, we propose to inject layout constraints in the form of masking or logit adjustment during inference. We show in the experiments that our LayoutDM successfully generates high-quality layouts and outperforms both task-specific and task-agnostic baselines on several layout tasks.*[1]

1. Introduction

Graphic layouts play a critical role in visual communication. Automatically creating a visually pleasing layout has tremendous application benefits that range from authoring of printed media [50] to designing application user interface [5], and there has been a growing research interest in the community. The task of layout generation considers the arrangement of elements, where each element has a tuple of attributes, such as category, position, or size, and depending on the task setup, there could be optional control inputs that specify part of the elements or attributes. Due to the structured nature of layout data, it is crucial to consider relationships between elements in a generation. For this reason, current generation approaches either build an autoregressive model [2, 12] or develop a dedicated inference strategy to explicitly consider relationships [21–23].

In this paper, we propose to utilize discrete state-space diffusion models [3, 10, 16] for layout generation tasks. Diffusion models have shown promising performance for various generation tasks, including images and texts [14]. We formulate the diffusion process for layout structure by *modality-wise discrete diffusion*, and train a denoising backbone network to progressively infer the complete layout with or without conditional inputs. To support variable-length layout data, we extend the discrete state-space with a special `PAD` token instead of the typical end-of-sequence token used in autoregressive models. Our model can incorporate complex layout constraints via *logit adjustment*, so that we can refine an existing layout or impose relative size constraints between elements without additional training.

We discuss two key advantages of LayoutDM over existing models for conditional layout generation. Our model avoids the immutable dependency chain issue [22] that hap-

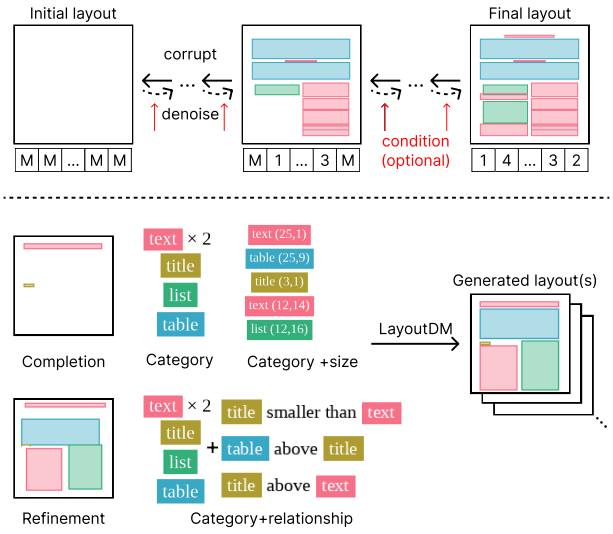


Figure 1. Overview of LayoutDM. **Top**: LayoutDM is trained to gradually generate a complete layout from a blank state in discrete state space. **Bottom**: During sampling, we can steer LayoutDM to perform various conditional generation tasks without additional training or external models.

[1]Please find the code and models at:
https://cyberagentailab.github.io/layout-dm.

pens in autoregressive models [12]. Autoregressive models fail to perform conditional generation when the condition disagrees with the pre-defined generation order of elements and attributes. Unlike non-autoregressive models [22], our model can generate variable-length elements. We empirically show in Sec. 4.5 that naively extending non-autoregressive models by padding results in suboptimal variable length generation while padding combined with our diffusion formulation leads to significant improvement.

We evaluate LayoutDM on various layout generation tasks tackled by previous works [22, 23, 37, 40] using two large-scale datasets, Rico [5] and PubLayNet [50]. LayoutDM outperforms task-agnostic baselines in the majority of cases and shows promising performance compared with task-specific baselines. We further conduct an ablation study to prove the significant impact of our design choices in LayoutDM, including quantization of continuous variables and positional embedding.

We summarize our contributions as follows:

- We formulate the discrete diffusion process for layout generation and propose a modality-wise diffusion and a padding approach to model highly structured layout data.
- We propose to inject complex layout constraints via masking and logit adjustment during the inference, so that our model can solve diverse tasks in a single model.
- We empirically show solid performance for various conditional layout generation tasks on public datasets.

2. Related Work

2.1. Layout Generation

Studies on automatic layout generation have appeared several times in literature [1, 29, 36, 47]. Layout tasks are commonly observed in design applications, including magazine covers, posters, presentation slides, application user interface, or banner advertising [5, 9, 11, 20, 39, 46, 47, 49]. Recent approaches to layout generation consider both unconditional generation [2, 12, 17, 18] and conditional generation in various setups, such as conditional inputs of category or size [21–23, 25], relational constraints [21, 23], element completion [12], and refinement [40]. Some attempt at solving multiple tasks in a single model [22, 37].

BLT [22] points out that the recent autoregressive decoders [2, 12] are not fully capable of considering partial inputs, *i.e*. known elements or attributes, during generation because they have a fixed generation order. BLT addresses the conditional generation by fill-in-the-blank task formulation using a bidirectional Transformer encoder similar to masked language models [6]. However, BLT cannot solve layout completion demonstrated in the decoder-based models because of the requirement of the known number of elements. Our LayoutDM enjoys the best of both worlds and supports a broader range of conditional generation tasks in a single model.

Another layout-specific consideration is the complex user-specified constraints, such as the positional requirements between two boxes (e.g., a header box should be on top of a paragraph box). Earlier approaches [34, 36, 48] propose hand-crafted cost functions representing the violation degree of aesthetic constraints so that those constraints guide the optimization process of layout inference. CLG-LO [21] proposes an aesthetically constrained optimization framework for pre-trained GANs. Our LayoutDM solves such constrained generation tasks on top of the task-agnostic iterative prediction via logit adjustment.

2.2. Discrete Diffusion Models

Diffusion models [42] are generative models characterized by a forward and reverse Markov process. The forward process corrupts the data into a sequence of increasingly noisy variables. The reverse process gradually denoises the variables toward the actual data distribution. Diffusion models are stable to train and achieve faster sampling than autoregressive models by parallel iterative refinement. Recently, many approaches have learned the reverse process by a neural network and show strong empirical performance [7, 14, 44] in continuous state spaces, such as images.

Discrete state spaces are a natural representation of discrete variables, such as text. D3PM [3] extends the pioneering work of Hoogeboom *et al*. [16] to structured categorical corruption processes for diffusion models in discrete state spaces, while maintaining the advantages of diffusion models for continuous state spaces. VQDiffusion [10] develops a corruption approach called mask-and-replace, so as to avoid accumulated prediction errors that are common in models based on iterative prediction. Following the corruption model of VQDiffusion, we carefully design a modality-wise corruption process for layout tasks that involve tokens from disjoint sets of vocabulary per modality.

Several studies consider a conditional input to the inference process of diffusion models. Some approaches alter the reverse diffusion iteration to carefully inject given conditions for free-form image inpainting [31] or image editing by strokes or composition [33]. We extend the discrete state-space diffusion models via hard masking or logit adjustment to support the conditional generation of layouts.

3. LayoutDM

Our LayoutDM builds on discrete-state space diffusion models [3, 10]. We first briefly review the fundamental of discrete diffusion models in Sec. 3.1. Sec. 3.2 explains our approach to layout generation within the diffusion framework while discussing features inherent in layout compared with text. Sec. 3.3 discusses how we extend denoising steps to perform various conditional layout generation by imposing conditions in each step of the reverse process.

3.1. Preliminary: Discrete Diffusion Models

Diffusion models [42] are generative models characterized by a forward and reverse Markov process. While many diffusion models are defined on continuous space with Gaussian corruption, D3PM [3] introduces a general diffusion framework for categorical variables designed primarily for texts. Let $T \in \mathbb{N}$ be a total timestep of the diffusion model, we first explain the forward diffusion process. For a scalar discrete variable with K categories $z_t \in \{1, 2, \dots, K\}$ at timestep $t \in \mathbb{N}$, probabilities that z_{t-1} transits to z_t are defined by using a transition matrix $\boldsymbol{Q}_t \in [0,1]^{K \times K}$, with $[Q_t]_{mn} = q(z_t = m | z_{t-1} = n)$,

$$q(z_t | z_{t-1}) = \boldsymbol{v}(z_t)^\top \mathbf{Q}_t \boldsymbol{v}(z_{t-1}), \tag{1}$$

where $\boldsymbol{v}(z_t) \in \{0,1\}^K$ is a column one-hot vector of z_t. The categorical distribution over z_t given z_{t-1} is computed by a column vector $\mathbf{Q}_t \boldsymbol{v}(z_{t-1}) \in [0,1]^K$. Assuming the Markov property, we can derive $q(z_t | z_0) = \boldsymbol{v}(z_t)^\top \overline{\mathbf{Q}}_t \boldsymbol{v}(z_0)$ where $\overline{\mathbf{Q}}_t = \mathbf{Q}_t \mathbf{Q}_{t-1} \cdots \mathbf{Q}_1$ and:

$$\begin{aligned} q(z_{t-1} | z_t, z_0) &= \frac{q(z_t | z_{t-1}, z_0)\, q(z_{t-1} | z_0)}{q(z_t | z_0)} \\ &= \frac{\left(\boldsymbol{v}(z_t)^\top \mathbf{Q}_t \boldsymbol{v}(z_{t-1})\right) \left(\boldsymbol{v}(z_{t-1})^\top \overline{\mathbf{Q}}_{t-1} \boldsymbol{v}(z_0)\right)}{\boldsymbol{v}(z_t)^\top \overline{\mathbf{Q}}_t \boldsymbol{v}(z_0)}. \end{aligned} \tag{2}$$

Note that due to the Markov property, $q(z_t | z_{t-1}, z_0) = q(z_t | z_{t-1})$. When we consider N-dimensional variables $\boldsymbol{z}_t \in \{1, 2, \dots, K\}^N$, the corruption is applied to each variable z_t independently. In the following, we explain with N-dimensional variables $\boldsymbol{z}_t$.

In contrast to the forward process, the reverse denoising process considers a conditional distribution of $\boldsymbol{z}_{t-1}$ over $\boldsymbol{z}_t$ by a neural network $p_\theta(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t) \in [0,1]^{N \times K}$. $\boldsymbol{z}_{t-1}$ is sampled according to this distribution. Note that the typical implementation is to predict unnormalized log probabilities $\log p_\theta(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t)$ by a stack of bidirectional Transformer encoder blocks. D3PM uses a neural network $\tilde{p}_\theta(\tilde{\boldsymbol{z}}_0 | \boldsymbol{z}_t)$, combines it with the posterior $q(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t, \boldsymbol{z}_0)$, and sums over possible $\tilde{\boldsymbol{z}_0}$ to obtain the following parameterization:

$$p_\theta(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t) \propto \sum_{\tilde{\boldsymbol{z}}_0} q(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t, \tilde{\boldsymbol{z}}_0)\, \tilde{p}_\theta(\tilde{\boldsymbol{z}}_0 | \boldsymbol{z}_t). \tag{3}$$

In addition to the commonly used variational lower bound objective $\mathcal{L}_{\text{vb}}$, D3PM introduces an auxiliary denoising objective. The overall objective is as follows:

$$\mathcal{L}_\lambda = \mathcal{L}_{\text{vb}} + \lambda \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}_t \sim q(\boldsymbol{z}_t | \boldsymbol{z}_0) \\ \boldsymbol{z}_0 \sim q(\boldsymbol{z}_0)}} \left[-\log \tilde{p}_\theta\left(\boldsymbol{z}_0 | \boldsymbol{z}_t\right) \right], \tag{4}$$

where λ is a hyper-parameter to balance the two loss terms.

Although D3PM proposes many variants of $\mathbf{Q}_t$, VQDiffusion [10] offers an improved version of $\mathbf{Q}_t$ called mask-and-replace strategy. They introduce an additional special token `[MASK]` and three probabilities γ_t of replacing the current token with the `[MASK]` token, β_t of replacing the token with other tokens, and α_t of not changing the token. The `[MASK]` token never transitions to other states. The transition matrix $\mathbf{Q}_t \in [0,1]^{(K+1) \times (K+1)}$ is defined by:

$$\mathbf{Q}_t = \begin{bmatrix} \alpha_t + \beta_t & \beta_t & \cdots & \beta_t & 0 \\ \beta_t & \alpha_t + \beta_t & \cdots & \beta_t & 0 \\ \vdots & \vdots & \ddots & \beta_t & 0 \\ \beta_t & \beta_t & \beta_t & \alpha_t + \beta_t & 0 \\ \gamma_t & \gamma_t & \gamma_t & \gamma_t & 1 \end{bmatrix}. \tag{5}$$

$(\alpha_t, \beta_t, \gamma_t)$ is carefully designed so that z_t converges to the `[MASK]` token for sufficiently large t. During testing, we start from $\boldsymbol{z}_T$ filled with `[MASK]` tokens and iteratively sample new set of tokens $\boldsymbol{z}_{t-1}$ from $p_\theta(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t)$.

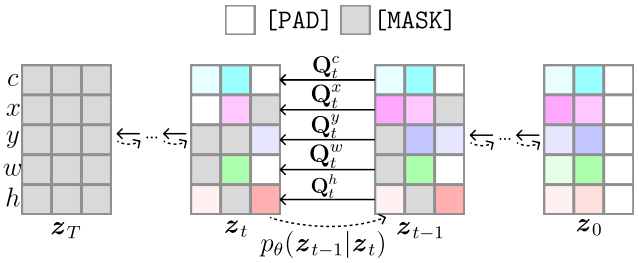


Figure 2. Overview of the corruption and denoising processes in LayoutDM. For simplicity, we use a toy layout consisting of two elements and the model generates three elements at maximum.

3.2. Unconditional Layout Generation

A layout l is a set of elements represented by $l = \{(c_1, \boldsymbol{b}_1), \dots, (c_E, \boldsymbol{b}_E)\}$. $E \in \mathbb{N}$ is the number of elements in the layout. $c_i \in \{1, \dots, C\}$ is categorical information of the i-th element in the layout. $\boldsymbol{b}_i \in [0,1]^4$ is the bounding box of the i-th element in normalized coordinates, where the first two values indicate the center location, and the last two indicate the width and height. Following previous works [2, 12, 22] that regard layout generation as generating a sequence of tokens, we quantize each value in $\boldsymbol{b}_i$ and obtain $[x_i, y_i, w_i, h_i]^\top \in \{1, \dots, B\}^4$, where B is a number of the bins. The layout l is now represented by $l = \{(c_1, x_1, y_1, w_1, h_1), \dots\}$.

In this work, we corrupt a layout in a modality-wise manner in the forward process, and we denoise the corrupted layout while considering all elements and modalities in the reverse process, as we illustrate in Fig. 2. Similarly to D3PM [3], we parameterize p_θ by a Transformer encoder [45], which processes an ordered 1D sequence. To process l by p_θ while avoiding the order dependency issue [22], we randomly shuffle l in element-wise manner and then flatten it to produce $l_{\text{flat}} = (c_1, x_1, y_1, w_1, h_1, c_2, x_2, \dots)$.

Variable length generation Existing diffusion models generate fixed-dimensional data and are not directly applicable to the layout generation because the number of elements in each layout varies. To handle this, we introduce a `[PAD]` token and define a maximum number of elements in the layout as $M \in \mathbb{N}$. Each layout is fixed-dimensional data composed of $5M$ tokens by appending $5(M - E)$ `[PAD]` tokens. `[PAD]` is treated similarly to the ordinary token in VQDiffusion and $\mathbf{Q}_t$'s dimension becomes $(K + 2) \times (K + 2)$.

Modality-wise diffusion Discrete state-space models assume that all the standard tokens are switchable by corruption. However, layout tokens comprise a disjoint set of token groups for each attribute in the element. For example, applying the transition rule Eq. (5) may change a token representing an element's category to another token representing the width. To avoid such invalid switching, we propose to apply disjoint corruption matrices $\mathbf{Q}_t^c, \mathbf{Q}_t^x, \mathbf{Q}_t^y, \mathbf{Q}_t^w, \mathbf{Q}_t^h$ for tokens representing different attributes c, x, y, w, h, as we show in Fig. 2. The size of each matrix is $(C + 2) \times (C + 2)$ for $\mathbf{Q}_t^c$ and otherwise $(B + 2) \times (B + 2)$, where $+2$ is for `[PAD]` and `[MASK]`.

Adaptive Quantization The distribution of the position and size information in layouts is highly imbalanced; e.g., elements tend to be aligned to either left, center, or right. Applying uniform quantization to those quantities as in existing layout generation models [2, 12, 22] results in the loss of information. As a pre-processing, we propose to apply a classical clustering algorithm, such as KMeans [32] on x, y, w, and h independently to obtain balanced position and size tokens for each dataset. We show in Sec. 4.7 how quantization strategy affects the resulting quality.

Decoupled Positional Encoding Previous works apply standard positional encoding to a flattened sequence of layout tokens l_{flat} [2, 12, 22]. We argue that this flattening approach could lose the structure information of the layout and lead to inferior generation performance. In layout, each token has two types of indices: i-th element and j-th attribute. We empirically find that independently applying positional encoding to those indices improves final generation performance, which we study in Sec. 4.7.

3.3. Conditional Generation

We elaborate on solving various conditional layout generation tasks using pre-trained frozen LayoutDM. We inject conditional information in both the initial state $\boldsymbol{z}_T$ and sampled states $\{\boldsymbol{z}_t\}_{t=0}^{T-1}$ during inference but do not modify the denoising network p_θ. The actual implementation of the injection differs by the type of conditions.

Strong Constraints The most typical condition is partially known layout fields. Let $\boldsymbol{z}^{\text{known}} \in \mathbb{Z}^N$ contain the known fields and $\boldsymbol{m} \in \{0, 1\}^N$ be a mask vector denoting the known and unknown field as 1 and 0, respectively. In each timestep t, we sample $\hat{\boldsymbol{z}}_{t-1}$ from $p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ in Eq. (3) and then inject the condition by $\boldsymbol{z}_{t-1} = \boldsymbol{m} \odot \boldsymbol{z}^{\text{known}} + (\mathbf{1} - \boldsymbol{m}) \odot \hat{\boldsymbol{z}}_{t-1}$, where $\mathbf{1}$ denotes a N-dimensional all-ones vector and $\odot$ denotes element-wise product.

Weak Constraints We may impose a weaker constraint during generation, such as an element in the center. We offer a way to impose such constraints in a unified framework without additional training or external neural network models. We propose to adjust the logits to inject weak constraints in log probability space by

$$\log \hat{p}_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) \propto \log p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) + \lambda_\pi \boldsymbol{\pi}, \tag{6}$$

where $\boldsymbol{\pi} \in \mathbb{R}^{N \times K}$ is a prior term that weights the desired outputs, and $\lambda_\pi \in \mathbb{R}$ is a hyper-parameter. The prior term can be defined either hard-coded (Refinement in Sec. 4.5) or through differentiable loss functions (Relationship in Sec. 4.5). Let $\{\mathcal{L}_i\}_{i=1}^L$ be a set of differentiable loss functions given the prediction, the later prior definition can be written by:

$$\boldsymbol{\pi} = -\nabla_{p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)} \sum_{i=1}^{L} \mathcal{L}_i \left(p_\theta \left(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t \right) \right). \tag{7}$$

Although the formulation of Eq. (7) resembles steering diffusion models by gradients from external models [7, 28], our primal focus is incorporating classical hand-crafted energies for aesthetics principles of layout [36] that do not depend on an external model. In practice, we tune the hyper-parameters for imposing weak constraints, such as λ_π. Note that these hyper-parameters are only for inference and are easier to tune than the other training hyper-parameters.

4. Experiment

4.1. Datasets

We use two large-scale datasets for comparison, Rico [5] and PubLayNet [50]. As we mention in Sec. 3.2, an element in a layout for each dataset is described by the five attributes. For preprocessing, we set the maximum number of elements per layout M to 25. If a layout contains more elements, we discard the whole layout.

We provide an overview of each dataset. **Rico** is a dataset of user interface designs for mobile applications containing 25 element categories such as text button, toolbar, and icon. We divide the dataset into 35,851 / 2,109 / 4,218 samples for train, validation, and test splits. **PubLayNet** is a dataset of research papers containing five element categories, such as table, image, and text. We divide the dataset into 315,757 / 16,619 / 11,142 samples for train, validation, and test splits.

4.2. Evaluation Metrics

We employ two primary metrics: FID and Maximum IoU (Max.). These metrics take into account both fidelity and diversity [13], which are two mutually complementary properties widely used in evaluating generative models. FID [13] captures the similarity of generated data to real ones in feature space. We employ an improved feature extraction model for layouts [21] instead of a conventional method [23] to compute FID. Maximum IoU [21] measures the conditional similarity between generated and real layouts. The similarity is measured by computing optimal matching that maximizes average IoU between generated and real layouts that have an identical set of categories. For reference, we show the FID and Maximum IoU computed between the validation and test data as *Real data*.

4.3. Tasks and Baselines

We test LayoutDM on six tasks for evaluation.

Unconditional generates layouts without any conditional input or constraint.

Category→size+position (C→S+P) is a generation task conditioned on the category of each element [22].

Category+size→ position (C+S→P) is conditioned on the category and size of each element.

Completion is conditioned on a small number of elements whose attributes are all known. Given a complete layout, we randomly sample from 0% to 20% of elements.

Refinement is conditioned on a noisy layout in which only geometric information is perturbed [40]. Following RUITE [40], we synthesize the input layout by adding random noise to the size and position of each element. We sample noise from a standard normal distribution with a mean of 0 and a variance of 0.01.

Relationship is conditioned on the category of each element and some relationship constraints between the elements [23]. Following CLG-LO [21], we employ the size and location relationships and randomly sample 10% relationships between elements for the experiment.

The first four tasks handle basic layout fields. We include a few task-agnostic models for comparison using existing controllable layout generation methods or simple adaptation of generative models in the following:

LayoutTrans is a simple autoregressive model [12] trained on a element-level shuffled layout, following [37]. We set a variable generation order to c→w→h→x→y.

MaskGIT* is originally a non-autoregressive model for unconditional fixed-length data generation [4]. We use `[PAD]` to enable variable-length generation.

BLT is a non-autoregressive model with layout-specific decoding strategy [22].

BART is a denoising autoencoder that can solve both comprehension and generation tasks based on Transformer encoder-decoder backbone [24]. We randomly generate a number of `[MASK]` tokens from a uniform distribution between one and the sequence length, and perform masking based on the number.

VQDiffusion* is a diffusion-based model originally for text-to-image generation [10]. We adapt the model for layout using $K = C + 4B + 2$ tokens, including `[PAD]`.

4.4. Implementation Details

We re-implement most of the models since there are few official implementations publicly available except [12, 21, 22][2]. We train all the models on the two datasets with three independent trials and report the average of the results.

LayoutDM follows VQDiffusion for hyper-parameters unless specified, such as configurations for p_θ and the transition matrix parameters *i.e.* α_t and γ_t. We set the loss weight $\lambda = 0.1$ (in Eq. (4)) and the diffusion timesteps $T = 100$. For optimization, we use AdamW [30] with learning rate of 5.0×10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.98$.

Many models, including LayoutDM, use Transformer [45] encoder backbone. We define a shared configuration as follows: 4 layers, 8 attention heads, 512 embedding dimensions, 2048 hidden dimensions, and 0.1 dropout rate. For other models with extra modules, we adjust the number of hidden dimensions to roughly match the number of parameters for a fair comparison. We randomly shuffle elements in the layout to avoid fixed-order generation during training. We search best hyper-parameters to obtain the best FID using the validation set.

4.5. Quantitative Evaluation

C→S+P, C+S→P, Completion In these tasks, we inject conditions by masking. We summarize comparisons in Tab. 1. As task-specific models, we include LayoutVAE [18], NDN-none [23], and LayoutGAN++ [21] for C→S+P. We also adapt these models for C+S→P. LayoutDM outperforms other models except LayoutTrans [12] in completion. The significant performance gap between LayoutDM and VQDiffusion* suggests the contribution of our proposals to go beyond the simple discrete diffusion models discussed in Sec. 3.2. Results in the completion suggest that a combination of padding and diffusion models is the primal key to the generation quality. We find that FID and Maximum IoU are not highly correlated only in the completion task. We conjecture that Maximum IoU may become unstable when categories are also predicted, unlike the C→S+P and C+S→P tasks where categories are given.

Fig. 3 shows the qualitative results of some models, including LayoutDM. We can see that LayoutDM generates

[2]Unfortunately, most datasets have no official train-val-test splits, and previous approaches work on different splits and pre-processing strategies. Furthermore, models for FID computation also vary. Thus, we cannot directly compare our results with the reported figure in the literature.

Table 1. Quantitative comparison in conditional generation given partially known fields. Top two results are highlighted in **bold** and underline, respectively. † indicates the results of BLT trained with `[PAD]` as an additional vocabulary since the original model cannot perform unordered completion in practice.

Task	Category → Size+Position				Category+Size → Position				Completion			
Dataset	Rico		PubLayNet		Rico		PubLayNet		Rico		PubLayNet	
Model	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑	FID ↓	Max. ↑
Task-specific models												
LayoutVAE [18]	33.3	0.249	26.0	0.316	30.6	0.283	27.5	0.315	-	-	-	-
NDN-none [23]	28.4	0.158	61.1	0.162	62.8	0.219	69.4	0.222	-	-	-	-
LayoutGAN++ [21]	6.84	0.267	24.0	0.263	6.22	0.348	9.94	0.342	-	-	-	-
Task-agnostic models												
LayoutTrans [12]	5.57	0.223	14.1	0.272	3.73	0.323	16.9	0.320	**3.71**	0.537	8.36	0.451
MaskGIT* [4]	26.1	0.262	17.2	0.319	8.05	0.320	5.86	0.380	33.5	0.533	19.7	**0.484**
BLT [22]	17.4	0.202	72.1	0.215	4.48	0.340	5.10	**0.387**	117†	0.471†	131†	0.345†
BART [24]	3.97	0.253	9.36	**0.320**	3.18	0.334	5.88	0.375	8.87	0.527	9.58	0.446
VQDiffusion* [10]	4.34	0.252	10.3	0.319	3.21	0.331	7.13	0.374	11.0	0.541	11.1	0.373
LayoutDM	**3.55**	**0.277**	**7.95**	0.310	**2.22**	**0.392**	**4.25**	0.381	9.00	**0.576**	**7.65**	0.377
Real data	1.85	0.691	6.25	0.438	1.85	0.691	6.25	0.438	1.85	0.691	6.25	0.438

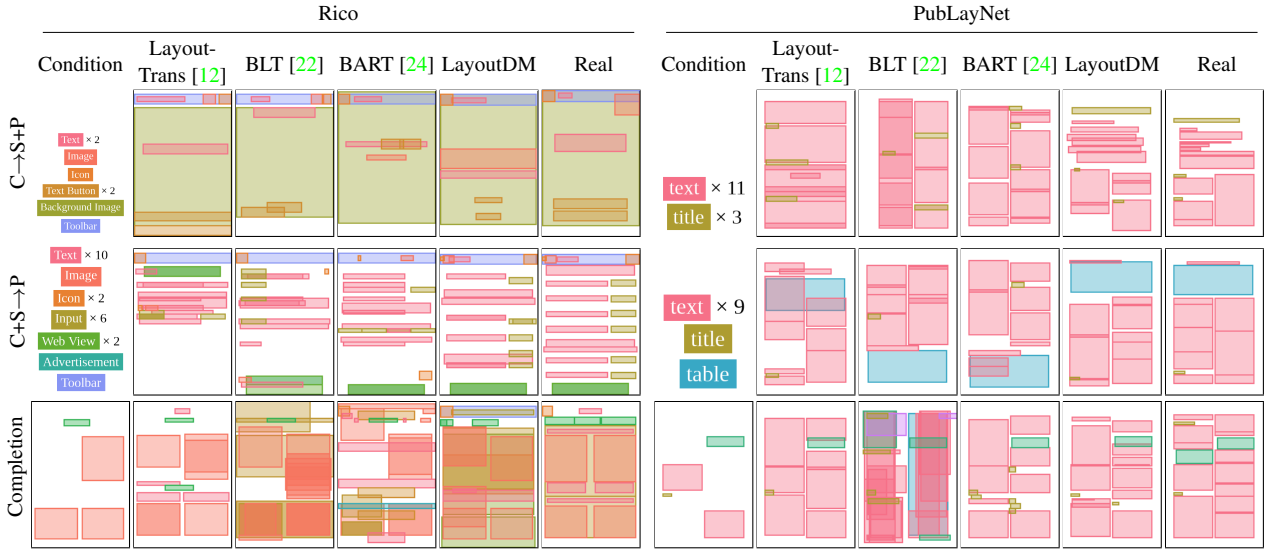


Figure 3. Comparison in conditional generation given partially known fields.

high-quality layouts with fewer layout aesthetics violations, such as misalignment and overlap, given diverse conditions.

Unconditional Generation Tab. 2 summarizes the results of unconditional generation. Unconditional layout generation methods often assume fixed order for element generation *e.g.* top-to-bottom rather than random order for better generation quality by constraining the prediction. For reference, we additionally report the results of LayoutTrans [12] trained on the fixed element order (LayoutTrans-fixed). Although we design LayoutDM's primarily for conditional generation, LayoutDM achieves the best FID under random element order setting. We conjecture that BLT's poor performance is due to train-test mask distribution inconsistency caused by their hierarchical masking strategy for training. BLT masks a randomly sampled number of fields from a single semantic group *i.e.* category, position, or size. However, decoding starts with all masked tokens in inference. The alignment metric of Real data stays at 0.109 in Rico. Too small alignment values of LayoutTrans and MaskGIT can be a signal of producing trivial outputs in Rico.

Table 2. Quantitative comparison in unconditional generation. Top two results are highlighted in **bold** and underline, respectively.

Model	Dataset	Rico		PubLayNet	
		FID ↓	Align. ↓	FID ↓	Align. ↓
LayoutTrans-fixed [12]		**6.47**	0.133	17.1	**0.084**
LayoutTrans [12]		7.63	0.068	**13.9**	0.127
MaskGIT* [4]		52.1	**0.015**	27.1	0.101
BLT [22]		88.2	1.030	116	0.153
BART [24]		11.9	0.090	16.6	0.116
VQDiffusion* [10]		7.46	0.178	15.4	0.193
LayoutDM		6.65	0.162	**13.9**	0.195
Real data		1.85	0.109	6.25	0.0214

Table 3. Quantitative comparison in the refinement task. Top two results are highlighted in **bold** and underline, respectively.

Model	Dataset	Rico			PubLayNet		
		FID ↓	Max. ↑	Sim ↑	FID ↓	Max. ↑	Sim ↑
Task-specific models							
RUITE [40]		3.23	**0.421**	**0.221**	**6.39**	**0.415**	**0.174**
Task-agnostic models							
Noisy input		134	0.213	0.177	130	0.242	0.147
LayoutDM		**2.77**	0.370	0.205	6.75	0.352	0.149
w/o logit adj.		3.55	0.277	0.168	7.95	0.310	0.127
Real data		1.85	0.691	0.260	6.25	0.438	0.216

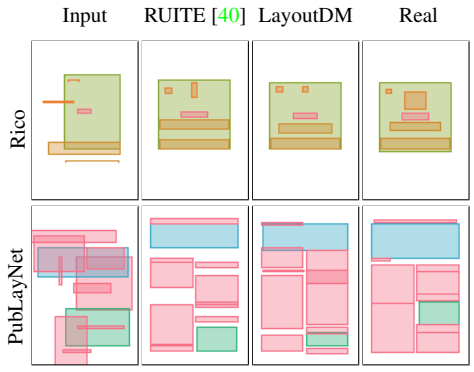


Figure 4. Qualitative comparison in the refinement task.

Refinement Our LayoutDM performs this task with a combination of the strong constraints of element categories, *i.e.*, setting $\boldsymbol{z}^{\text{known}} = \{(c_1, \texttt{[MASK]}, \ldots, \texttt{[MASK]}), \ldots\}$, and the weak constraints that geometric outputs appear near noisy inputs. As an example of the weak constraint, we describe a constraint that imposes the x-coordinate estimate of i-th element close to the noisy continuous observation $\hat{x}_i$. We denote a sliced vector of the prior term $\boldsymbol{\pi}$ in Eq. (6) that corresponds to the x-coordinate of i-th element as $\boldsymbol{\pi}_x^i \in \mathbb{R}^K$

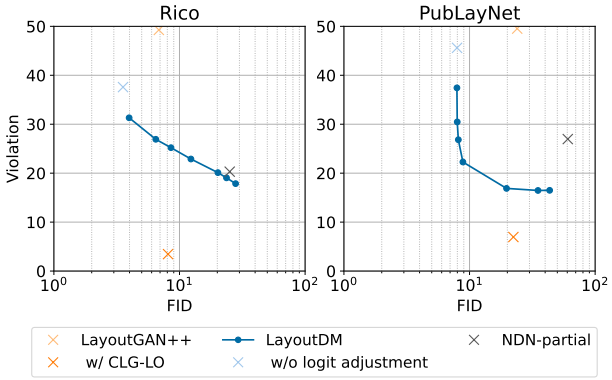


Figure 5. Quality-violation trade-off in the relationship task. Lower scores indicate better performance for both metrics.

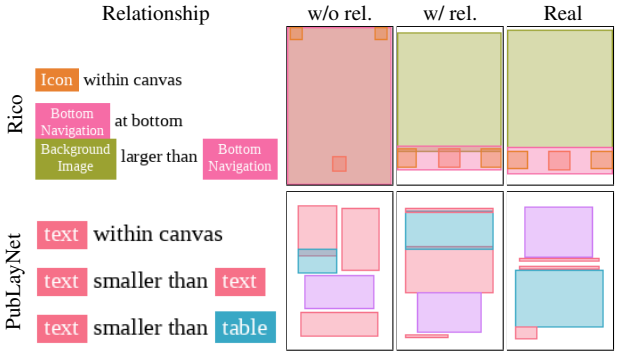


Figure 6. Qualitative comparison in the relationship task.

and define by:

$$\left[\boldsymbol{\pi}_x^i\right]_j = \begin{cases} 1 & \text{if } |\text{loc}(j) - \hat{x}_i| < m \text{ and } j \in X \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where m is a hyper-parameter indicating a margin, X is a set of indices denoting tokens for x in the vocabularies, and $\text{loc}(j)$ is a function that returns the centroid value of j-th token in the vocabularies. We define similar constraints for the other geometric variables and elements.

We summarize the performance in Tab. 3. We additionally report DocSim [38] (Sim) to measure the similarity of a predicted and its corresponding ground truth layout. Imposing noisy geometric fields as a weak prior significantly improves the masking-only model and makes the performance much closer to RUITE [40], which is a denoising model not applicable to other layout tasks. We compare some results in Fig. 4. Both LayoutDM and RUITE successfully recover complete layouts from non-trivially noisy layouts.

Relationship We use Eq. (7) to incorporate the relational constraints during the sampling step of LayoutDM. We follow [21] to employ the loss functions penalizing size and

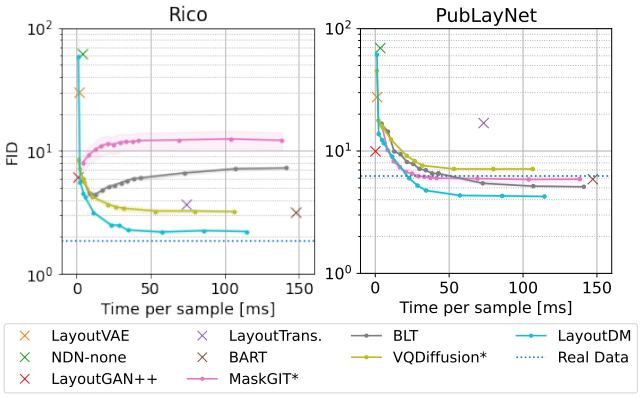


Figure 7. Speed-quality trade-off of different models for C+S→P.

Table 4. Ablation study results on layout-specific modification in unconditional generation of Rico [5] dataset.

	FID ↓	Align. ↓
LayoutDM	**6.65**	0.162
w/o modality-wise diff.	7.32	**0.156**
w/o decoupled pos. enc.	6.78	0.227
w/ uniform-quantization	7.58	0.256
w/ percentile-quantization	9.79	0.232
Real Data	1.85	0.109

location relationships between elements that do not match user specifications. We define the loss functions for continuous bounding boxes, and we have to convert the predicted discrete bounding boxes to continuous ones in a differentiable manner. Given estimated probabilities of discrete x-coordinates $p(x)$, for example, we compute the continuous x-coordinate $\bar{x}$ by $\bar{x} = \sum_{n \in X} p(x = n) \operatorname{loc}(n)$. Similar conversion applies to the other attributes. Empirically, we find that applying the logit adjustment multiple times (three times in our experiments) to each diffusion step moderately improves performance.

We compare LayoutDM with two task-specific approaches: NDN-partial [23] and CLG-LO based on LayoutGAN++ [21]. We show the results in Fig. 5. We additionally report constraint violation error rates [21]. LayoutDM can control the strength of the logit adjustment as in Eq. (6) and produces an FID-violation trade-off curve. LayoutDM is comparable to NDN-partial in Rico and outperforms NDN-partial by a large margin in PubLayNet. Although LayoutDM is inferior to CLG-LO in both datasets, note that the average runtime of CLG-LO is 4.0s, which is much slower than 0.5s in LayoutDM. We show some results of LayoutDM in Fig. 6.

4.6. Speed-Quality Trade-off

Runtime is also essential for a controllable generation. We show a speed-quality trade-off curve for C+S→P as shown in Fig. 7. The Transformer encoder-only models, such as LayoutDM and BLT, can achieve fast generation at the sacrifice of quality. We employ fast-sampling strategy employed in discrete diffusion models [3] for LayoutDM by $p_\theta(\boldsymbol{z}_{t-\Delta}|\boldsymbol{z}_t) \propto \sum_{\tilde{\boldsymbol{z}}_0} q(\boldsymbol{z}_{t-\Delta}, \boldsymbol{z}_t|\tilde{\boldsymbol{z}}_0)\tilde{p}_\theta(\tilde{\boldsymbol{z}}_0|\boldsymbol{z}_t)$, where $\Delta \in \mathbb{N}$ indicates a step size for generation in $\frac{T}{\Delta}$ steps. Despite being a task-agnostic model, LayoutDM achieves the best quality-speed trade-off except for task-specific LayoutGAN++ [21] that runs under 10ms.

4.7. Ablation Study

We investigate whether techniques in Sec. 3.2 improve the performance. First, we evaluate a choice of quantization methods for the geometric fields of elements. Instead of KMeans, we compute centroids for the quantization by:

- Uniform: This is dataset-agnostic quantization, which is popular in previous works [2, 12, 22]. Following [12], we choose $\{0.0, \frac{1}{B}, \ldots, \frac{B-1}{B}\}$ and $\{\frac{1}{B}, \ldots, \frac{B-1}{B}, 1.0\}$ for the position and size, respectively.
- Percentile: we sort the data into equally sized groups and obtain average values for each group as the centroids. This is dataset-specific quantization similar to KMeans.

We show the result at the bottom of Tab. 4. We additionally report the Alignment metric (Align.) used in [21] since the choice of the quantization affects the alignment between elements. Compared to Linear and Percentile, KMeans quantization significantly improves both FID and Alignment. We confirm that our modality-wise diffusion and decoupled positional encoding both moderately improve the performance, as we show at the top half of Tab. 4.

5. Discussion

LayoutDM is based on diffusion models for discrete state-space. Using continuous state-space as in latent diffusion models [41] would be interesting. Extension of LayoutDM to handle various layout properties such as color [19] and image/text [46] is also appealing.

We believe our proposed logit adjustment can incorporate more attributes. Attribute-conditional LayoutGAN [26] considers area, aspect ratio, and reading order of elements for fine-grained control. Since these attributes can be easily converted to the size and location relationship constraints, incorporating them with our LayoutDM is not very difficult.

Potential negative impact Our model might be used to automatically generate the basic structure of fake websites or mobile applications, which could lead to scams or the spreading of misinformation.

References

[1] Maneesh Agrawala, Wilmot Li, and Floraine Berthouzoz. Design principles for visual communication. *Communications of the ACM*, 54(4), 2011. 2

[2] Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *CVPR*, 2021. 1, 2, 3, 4, 8

[3] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021. 1, 2, 3, 8

[4] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *CVPR*, 2022. 5, 6, 7, 11, 12, 13, 16, 17, 18, 19, 20, 21

[5] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *UIST*, 2017. 1, 2, 4, 8, 12

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 2

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 2, 4

[8] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 11

[9] Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. DOC2PPT: Automatic presentation slides generation from scientific documents. In *AAAI*, 2022. 2

[10] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *CVPR*, 2022. 1, 2, 3, 5, 6, 7, 11, 12, 13, 16, 17, 18, 19, 20, 21

[11] Shunan Guo, Zhuochen Jin, Fuling Sun, Jingwen Li, Zhaorui Li, Yang Shi, and Nan Cao. Vinci: an intelligent graphic design system for generating advertising posters. In *CHI*, 2021. 2

[12] Kamal Gupta, Alessandro Achille, Justin Lazarow, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layout-Transformer: Layout generation and completion with self-attention. In *ICCV*, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 16, 17, 18, 19, 20, 21

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 5

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1, 2, 12

[15] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*, 2019. 11

[16] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. In *NeurIPS*, 2021. 1, 2

[17] Zhaoyun Jiang, Shizhao Sun, Jihua Zhu, Jian-Guang Lou, and Dongmei Zhang. Coarse-to-fine generative modeling for graphic layouts. In *AAAI*, 2022. 2

[18] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. LayoutVAE: Stochastic scene layout generation from a label set. In *CVPR*, 2019. 2, 5, 6, 12

[19] Kotaro Kikuchi, Naoto Inoue, Mayu Otani, Edgar Simo-Serra, and Kota Yamaguchi. Generative colorization of structured mobile web pages. In *WACV*, 2023. 8

[20] Kotaro Kikuchi, Mayu Otani, Kota Yamaguchi, and Edgar Simo-Serra. Modeling visual containment for web page layout optimization. *Computer Graphics Forum*, 40(7), 2021. 2

[21] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *ACM MM*, 2021. 1, 2, 5, 6, 7, 8, 11, 12, 13

[22] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. BLT: Bidirectional layout transformer for controllable layout generation. In *ECCV*, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 16, 17, 18, 19, 20, 21

[23] 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. In *ECCV*, 2020. 1, 2, 5, 6, 8, 12, 13

[24] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, 2020. 5, 6, 7, 12, 13, 16, 17, 18, 19, 20, 21

[25] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. LayoutGAN: Generating graphic layouts with wireframe discriminators. In *ICLR*, 2019. 2, 13

[26] Jianan Li, Jimei Yang, Jianming Zhang, Chang Liu, Christina Wang, and Tingfa Xu. Attribute-conditioned layout gan for automatic graphic design. *IEEE TVCG*, 27(10), 2020. 8, 11, 13

[27] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. Diffusion-LM improves controllable text generation. In *NeurIPS*, 2022. 11, 12

[28] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *ECCV*, 2022. 4

[29] Simon Lok and Steven Feiner. A survey of automated layout techniques for information presentations. In *SmartGraphics*, 2001. 2

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[31] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022. 2

[32] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, 1967. 4

[33] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided

image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 2

[34] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM TOG*, 30(4), 2011. 2

[35] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *ICML*, 2020. 13

[36] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning layouts for single-pagegraphic designs. *IEEE TVCG*, 20(8), 2014. 2, 4

[37] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. ATISS: Autoregressive transformers for indoor scene synthesis. In *NeurIPS*, 2021. 2, 5

[38] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. READ: Recursive autoencoders for document layout generation. In *CVPRW*, 2020. 7

[39] 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. *IEEE TVCG*, 27(2), 2020. 2

[40] Soliha Rahman, Vinoth Pandian Sermuga Pandian, and Matthias Jarke. RUITE: Refining ui layout aesthetics using transformer encoder. In *26th International Conference on Intelligent User Interfaces-Companion*, 2021. 2, 5, 7, 12, 22

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 8

[42] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 2, 3

[43] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 12

[44] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *NeurIPS*, 2020. 2

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 3, 5

[46] Kota Yamaguchi. CanvasVAE: Learning to generate vector graphics documents. In *ICCV*, 2021. 2, 8

[47] Xuyong Yang, Tao Mei, Ying-Qing Xu, Yong Rui, and Shipeng Li. Automatic generation of visual-textual presentation layout. *TOMM*, 12(2), 2016. 2

[48] Lap Fai Yu, Sai Kit Yeung, Chi Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. Make it home: automatic optimization of furniture arrangement. *ACM TOG*, 30(4), 2011. 2

[49] Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson WH Lau. Content-aware generative modeling of graphic design layouts. *ACM TOG*, 38(4), 2019. 2

[50] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. PubLayNet: largest dataset ever for document layout analysis. In *ICDAR*, 2019. 1, 2, 4

A. Implementation Details

A.1. Baseline

We explain more details on task-agnostic layout generation baselines using masking, especially when the original model is not designed for layout generation. We mostly describe unconditional generation cases, but partial layout fields can be easily injected by hard masking.

BART: BART is a denoising autoencoder and was originally designed for learning a sequence-to-sequence model for text generation. Text is usually represented as a 1D sequence of discrete tokens. Since we also handle the shuffled layout as a 1D sequence of discrete tokens during training, a BART-like model may be another solid baseline. To build a task-agnostic layout generation model, we apply random masking similar to the noise pattern of MaskGIT [4], instead of text-specific noises, such as span-level masking.

MaskGIT*: MaskGIT [4] is originally built for unconditional image generation. Following recent two-stage approaches for efficient image modeling, such as VQ-GAN [8], MaskGIT first generates a small number of discrete tokens and subsequently decodes those tokens into a continuous high-dimensional image by a pre-trained neural decoder. We consider the first generation part of MaskGIT to be another baseline. We use `[PAD]` to enable variable-length generation. For a masking schedule during decoding, *i.e*. fraction of the tokens masked in each iteration, we employ a cosine schedule as in MaskGIT.

VQDiffusion*: VQDiffusion [10] is a discrete diffusion-based model designed for text-to-image generation. To adapt VQDiffusion for conditional layout generation with minimal modification, we (i) remove the text conditioning branch in the reverse process, (ii) replace the image tokens with layout tokens, and (iii) add `[PAD]` token to enable variable-length generation. As described in the main manuscript, there are three major differences between VQDiffusion* and our proposed LayoutDM: modality-wise diffusion, decoupled positional encoding, and adaptive quantization.

We adjust the number of parameters for each model to have about 12M parameters for a fair comparison. We show the exact numbers in Tab. 5.

A.2. Relationship Guidance

Similarly to the main manuscript, let us denote the predicted coordinates of an i-th element $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i) \in [0, 1]^4$. We follow [21] to define the loss for penalizing size and location relationships between elements that do not match user specifications. For example, if we want to make the j-th element larger than the i-th element, the loss is defined by:

$$g_{lg}(i, j) = \max\left((1 + \gamma)\,\hat{w}_i\hat{h}_i - \hat{w}_j\hat{h}_j, 0\right), \quad (9)$$

where γ is a tolerance parameter, which is empirically set to 0.1. If we want to make the j-th element above the i-th element, the loss is defined by:

$$g_{ab}(i, j) = \max\left(\left(\hat{y}_j + \frac{\hat{h}_j}{2}\right) - \left(\hat{y}_i - \frac{\hat{h}_i}{2}\right), 0\right), \quad (10)$$

which compares the bottom of the j-th element and the top of the i-th element. Please refer to the code for losses for the rest of the relationships.

Although it is not experimentally demonstrated, we believe that it is also possible to incorporate area, aspect ratio, and reading order constraints used in Attribute-conditioned GAN [26].

- Area: given a target area of the element $a_i \in \mathbb{R}$, we use $|a_i - \hat{h}_i\hat{w}_i|$ as a loss.
- Aspect ratio: Given a target aspect ratio $r_i \in \mathbb{R}$, we use $|r_i - \frac{\hat{h}_i}{\hat{w}_i}|$ as a loss.
- Reading order: we follow [22] and define that the reading order solely depends on the distance between the left-top of the canvas and each element. We first compute the distance by $\hat{d}_i = \sqrt{(\hat{x}_i - \frac{\hat{w}_i}{2})^2 + (\hat{y}_i - \frac{\hat{h}_i}{2})^2}$. We can use $\max(\hat{d}_i - \hat{d}_j, 0)$ as a loss to make the i-th element come before the j-th element in the reading order.

A.3. Hyper-parameters

We search for the best hyper-parameters using a validation set. During sampling from $p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ for all the tasks, we search for p used in nucleus (or top-p) sampling [15] out of $\{0.90, 0.95, 0.99, 1.0\}$. We train the models for 50 and 20 epochs in Rico and PubLayNet, respectively.

We attempt a grid search for additional hyper-parameters in the refinement task. The ranges of possible values are the following: the distance margin m in $\{0.1, 0.2\}$ and the weighting term λ_π in $\{1.0, 2.0, 3.0, 4.0, 5.0\}$.

A.4. Evaluation

In unconditional generation, the model generates 1,000 samples from the random seed. In conditional generation, the test set of each dataset is used to make a partial input for conditional generation and the model generates one sample per each data in the test set.

B. Additional Results

B.1. Ablation Study

State space Continuous state space diffusion models have gained much attention compared to discrete state space models. Recently, Li *et al*. [27] propose DiffusionLM that adapts the continuous models to handle discrete text generation. DiffusionLM introduces an embedding and rounding step to bridge the continuous and discrete state spaces. We

	Rico	PubLayNet
LayoutVAE [18] (C→S+P)	13.2	13.0
NDN-none [23] (C→S+P)	21.8	21.8
LayoutGAN++ [21] (C→S+P)	12.9	12.9
LayoutVAE [18] (C+S→P)	14.7	14.5
NDN-none [23] (C+S→P)	14.8	14.8
LayoutGAN++ [21] (C+S→P)	13.0	13.0
LayoutTrans [12]	12.7	12.7
LayoutTrans-fixed [12]	12.7	12.7
MaskGIT* [4]	12.7	12.7
BLT [22]	12.7	12.7
RUITE [40]	12.7	12.7
BART [24]	12.8	12.8
VQDiffusion* [10]	12.4	12.4
LayoutDM	12.4	12.4

Table 5. The number of parameters [M] used for each model.

	State	#steps	Sampler	FID ↓
LayoutDM	dis.	100	-	**6.65**
VQDiffusion* [10]	dis.	100	-	<u>7.46</u>
DiffusionLM [27]	con.	100	DDIM	34.5
	con.	100	DDPM	24.8
	con.	1000	DDIM	33.8
	con.	1000	DDPM	22.8

Table 6. Ablation study results on the choice of state spaces: discrete (dis.) and continuous (con.), in the unconditional generation task of Rico [5] dataset. Top two results are highlighted in **bold** and <u>underline</u>, respectively.

train DiffusionLM (with 12.6M parameters) and show the results in Tab. 6. We show the results of DiffusionLM with embedding dimensions $d = 16$ because it works best out of $\{16, 64, 128\}$ in Rico [5] dataset. Although We tried different samplers (DDPM [14] and DDIM [43]) and training timesteps, DiffusionLM is still far behind the discrete state space models in layout generation as shown in Tab. 6.

Refinement The logit adjustment proposed in the main manuscript has some choices for injecting positional prior. Without loss of generality, we describe a constraint that imposes the x-coordinate estimate of i-th element close to the noisy continuous observation $\hat{x}_i$. We denote a sliced vector of the prior term $\pi(\boldsymbol{z}_{t-1})$ that corresponds to the x-coordinate of i-th element as $\boldsymbol{\pi}_x^i \in \mathbb{R}^K$.

- Gaussian: j-th token is more likely to be sampled when $\mathrm{loc}(j)$ is closer to $\hat{x}_i$. The prior is defined by:

$$\left[\boldsymbol{\pi}_x^i\right]_j = \begin{cases} (\mathrm{loc}\,(j) - \hat{x}_i)^2 & \text{if } |\mathrm{loc}(j) - \hat{x}_i| < m \text{ and } j \in X \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

  The ranges of possible values are similar to our method used in the main manuscript (Default).
- Negation: j-th token is never sampled when $\mathrm{loc}(j)$ is far away from $\hat{x}_i$. The prior is defined by:

$$\left[\boldsymbol{\pi}_x^i\right]_j = \begin{cases} 0 & \text{if } |\mathrm{loc}(j) - \hat{x}_i| < m \text{ and } j \in X \\ -\infty & \text{otherwise.} \end{cases} \tag{12}$$

  The ranges of possible values are the following: the distance margin m in $\{0.2, 0.4, 0.6, 0.7, 0.8, 0.9\}$.

We show the quantitative evaluation results in Tab. 7. We can see that Default outperforms other possible choices by a large margin.

	FID ↓	Max. ↑	Sim ↑
Default	**2.77**	**0.370**	**0.205**
Gaussian	5.82	<u>0.330</u>	<u>0.188</u>
Negation	<u>3.78</u>	0.276	0.169

Table 7. Ablation study results on the choice of logit adjustment methods in the refinement task. Top two results are highlighted in **bold** and <u>underline</u>, respectively.

B.2. Speed-Quality Trade-off

We show more speed-quality trade-off curves in Fig. 8. We perform generation with a batch size of 64 and report the average runtime to generate a single layout for all the models. Lightly colored regions around the line plots, such as the one in BLT for C→S+P in Rico represent the standard deviation of three trials for each model, though the deviations are too small to see in most cases.

B.3. More Results

We show more results compared with task-specific baselines in C→S+P (Fig. 9), C+S→P (Fig. 11), unconditional generation (Fig. 13), the refinement task (Fig. 15) for PubLayNet. Typical failure cases are frequent overlap between elements (often in BLT), unnecessarily broad blank space (often in LayoutTrans.), and lack of diversity. We show more results in C→S+P (Fig. 10), C+S→P (Fig. 12), unconditional generation (Fig. 14), the refinement task (Fig. 16) for Rico. Rico is more difficult to generate since the number of categories is large and elements are less aligned compared to PubLayNet.

B.4. Diversity-Fidelity Trade-off

We introduce density and coverage metrics by [35] to analyze the results from a different viewpoint. Density measures fidelity; *i.e.*, how closely generated samples resemble real ones. Coverage measures diversity; *i.e.*, whether generated samples cover the full variability of the real samples. We plot the diversity and fidelity of iterative refinement-based models in Fig. 17 as we increase the number of timesteps for the iterative prediction. Discrete diffusion-based models usually have higher coverage scores and lower density scores. We conjecture that the coverage difference comes from the inference decoding strategy. BLT [22] and MaskGIT* [4] fix high-confident predictions and re-initialize lower-confident fields by `[MASK]` for the next step that leads to higher fidelity. In contrast, discrete diffusion-based models *randomly* corrupt the predictions and result in higher diversity.

B.5. Alignment and Overlap

We additionally show the metrics reported in many previous works: Alignment and Overlap. Note that these metrics only capture the fidelity of generated layouts. There are a few variants for both Alignment [21, 23, 25, 26] and Overlap [21, 25, 26]. We employ the definition in [21]. We scale the values of Alignment by $100\times$ for visibility. For reference, we show Alignment and Overlap computed in a validation set as *Real data*. The lowest score in Alignment or Overlap does not always mean the best performance for a model, but a model closest to *Real data* is the best model. We show the result in Fig. 18. In the fixed-length generation *i.e.* C$\rightarrow$S+P and C+S$\rightarrow$P, LayoutDM performs almost comparably to VQDiffusion* [10] and BART [24], and better than the other models. In the variable-length generation *i.e.* the completion task and unconditional generation, autoregressive models, such as BART [24] and LayoutTrans. [12], are moderately better than LayoutDM. This result is reasonable since these models predict the fields one by one. Diffusion-based models, such as LayoutDM and VQDiffusion*, are better than BLT and MaskGIT*. We believe this is because diffusion models avoid the error accumulation in iterative prediction according to [10].

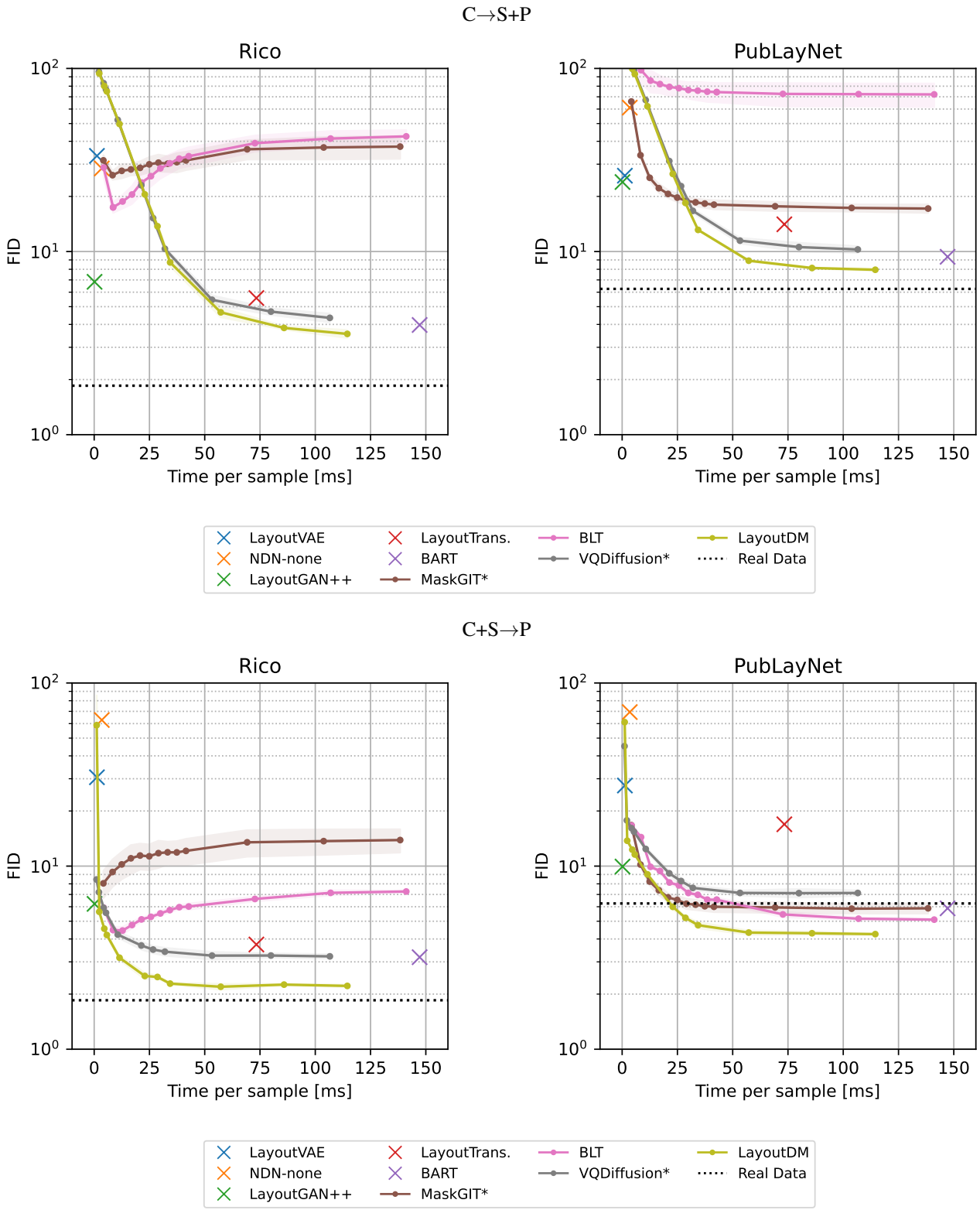
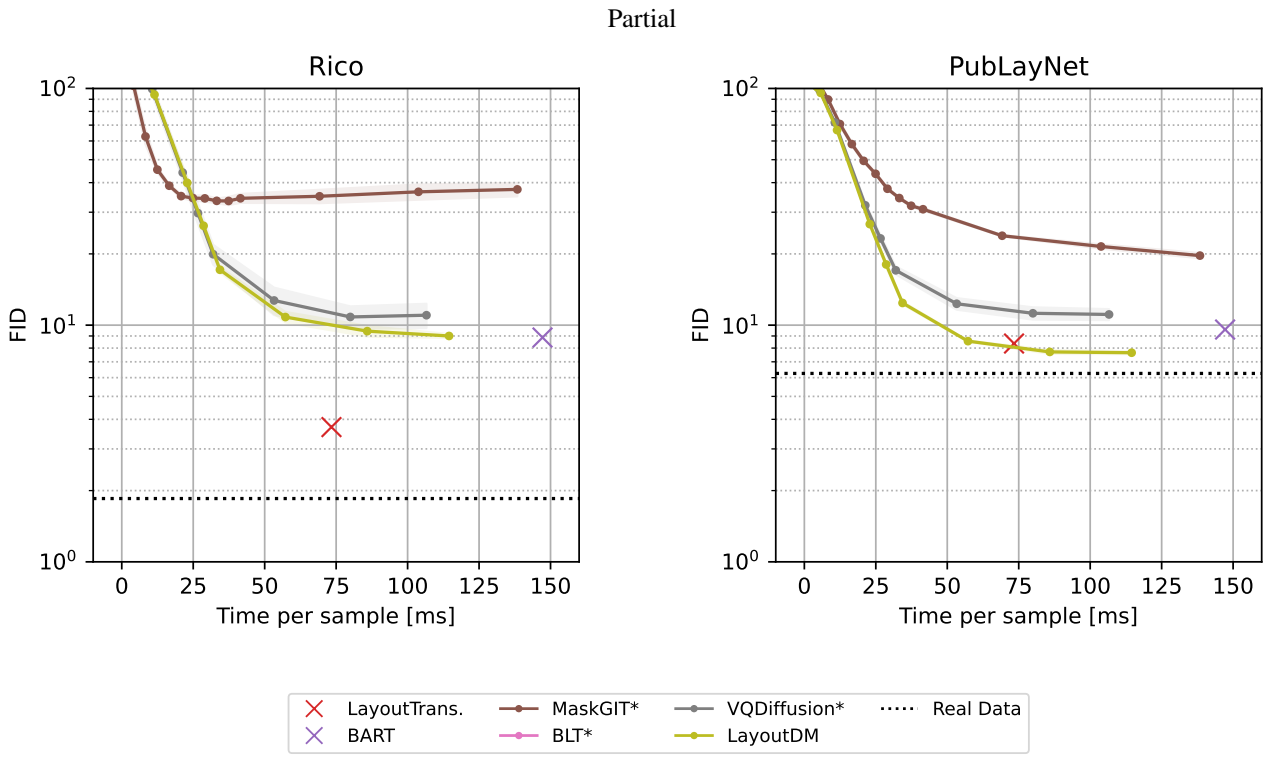


Figure 8. Speed-quality trade-off of different models.



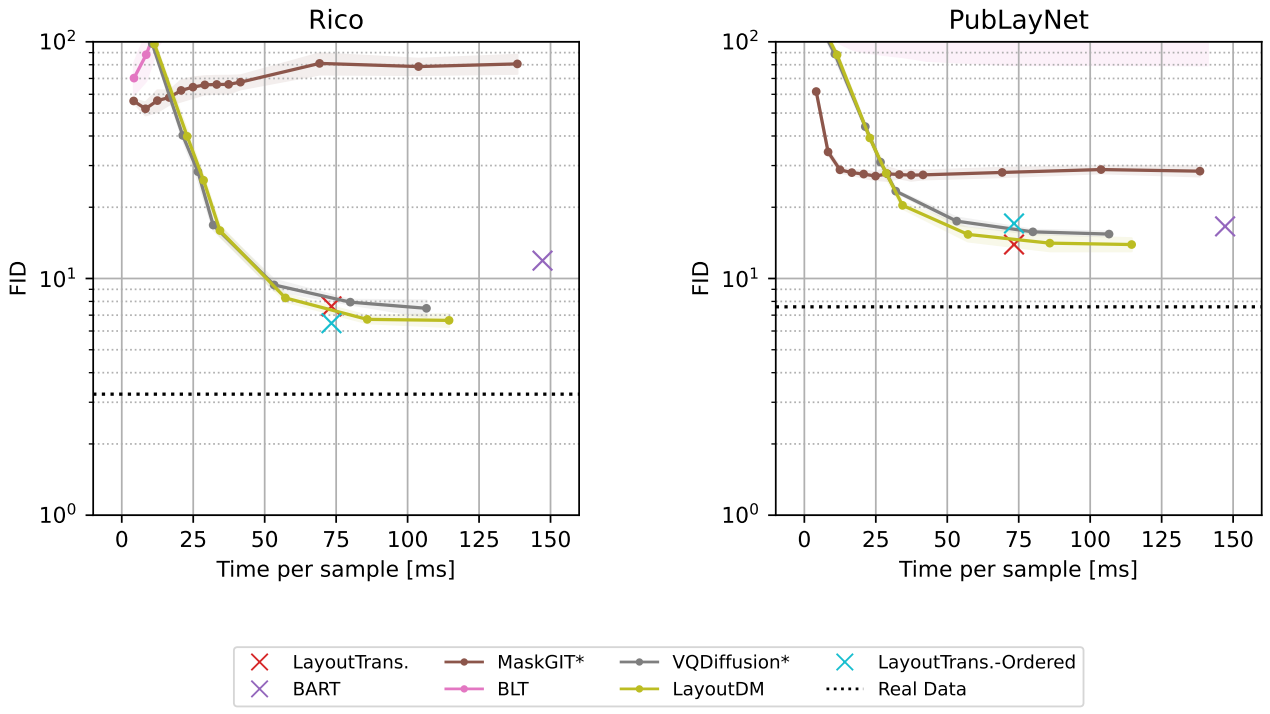


Figure 8. (cont.) Speed-quality trade-off of different models.

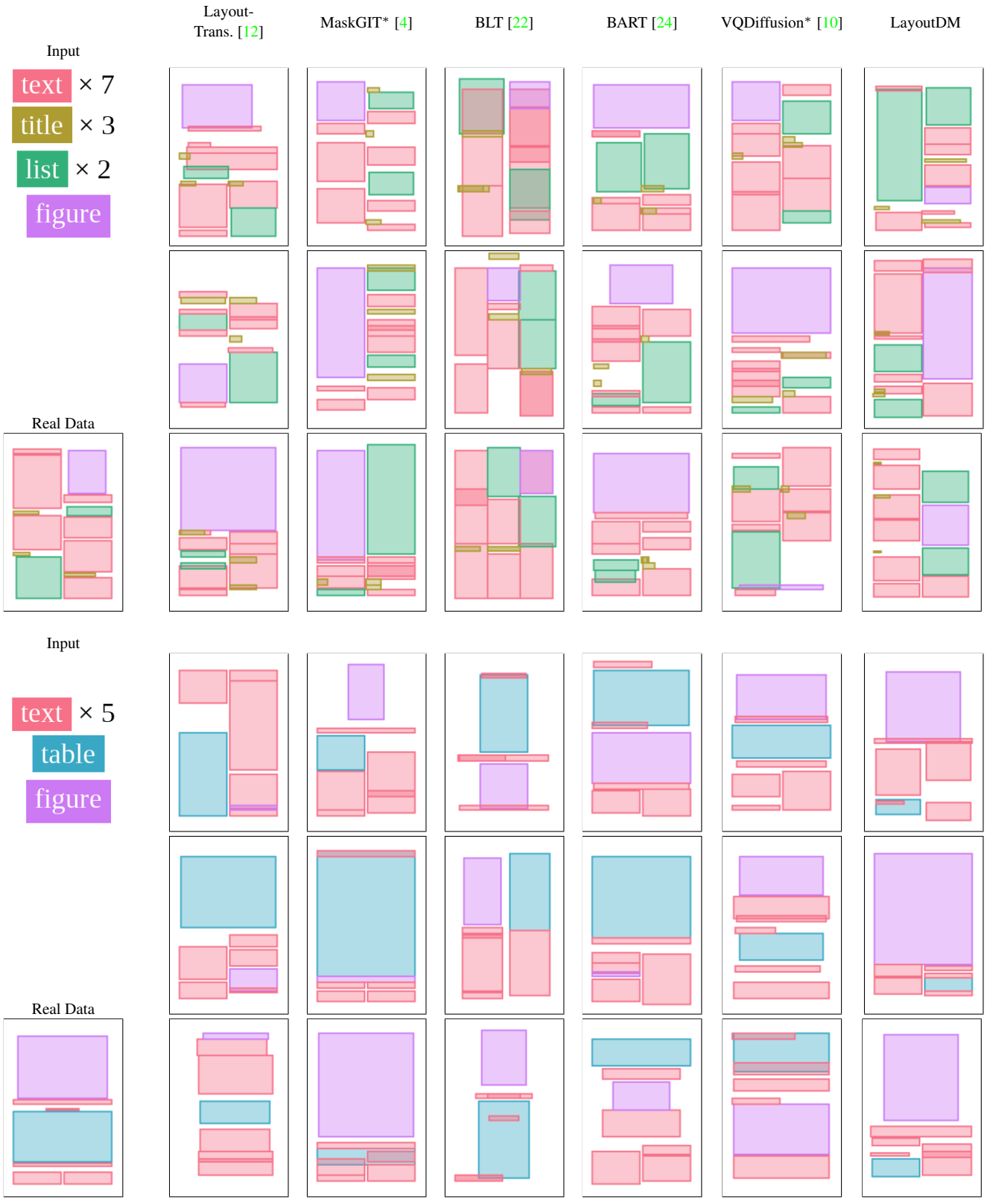


Figure 9. Comparison of conditional generation in C→S+P for PubLayNet. We obtain three samples from each model to demonstrate the diversity.

Figure 10. Comparison of conditional generation in C→S+P for Rico. We obtain three samples from each model to demonstrate the diversity.

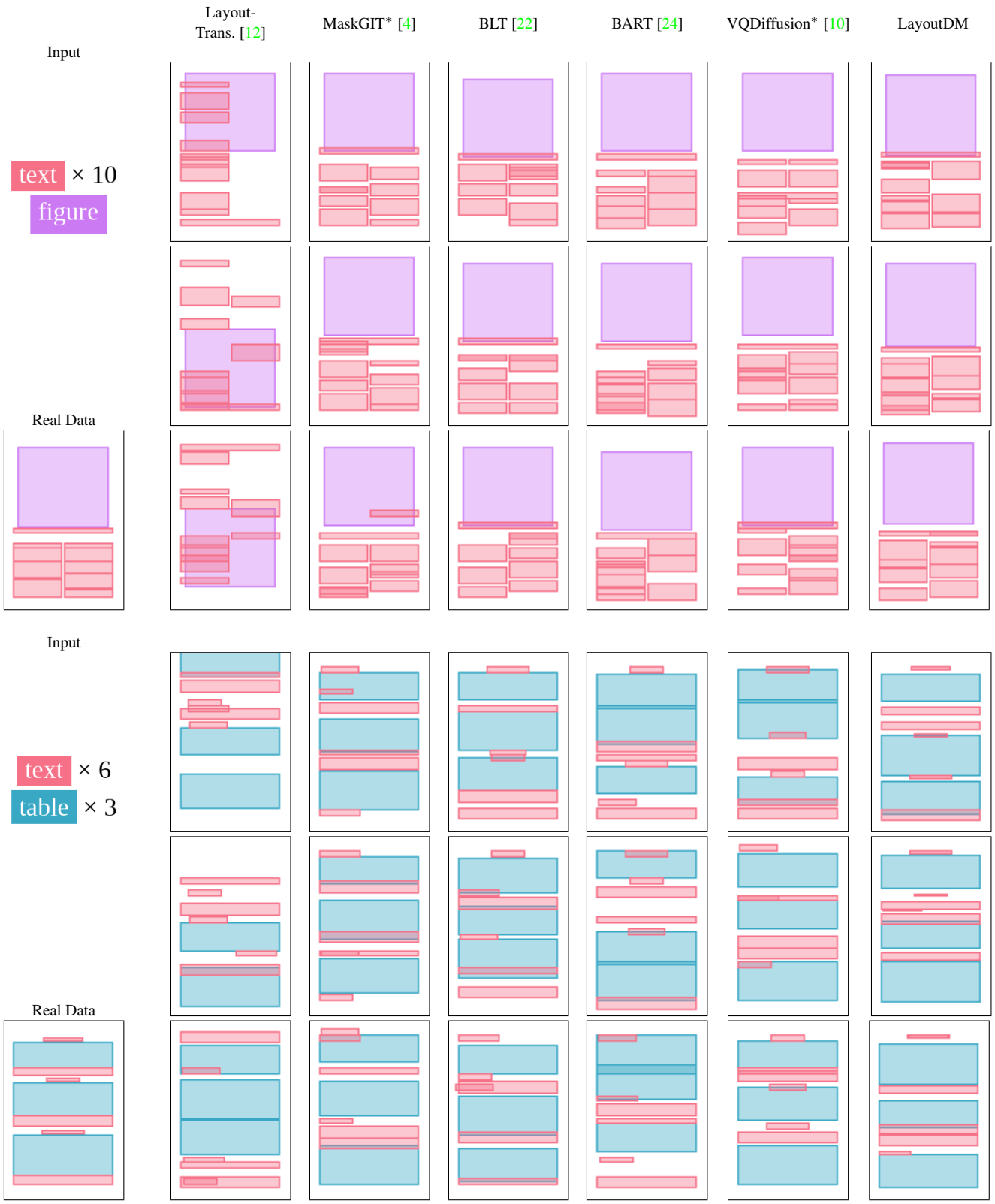


Figure 11. Comparison of conditional generation in C+S→P for PubLayNet. We obtain three samples from each model to demonstrate the diversity. Note that the size condition of each element is not shown for limited space. Please refer to Real Data for the size.

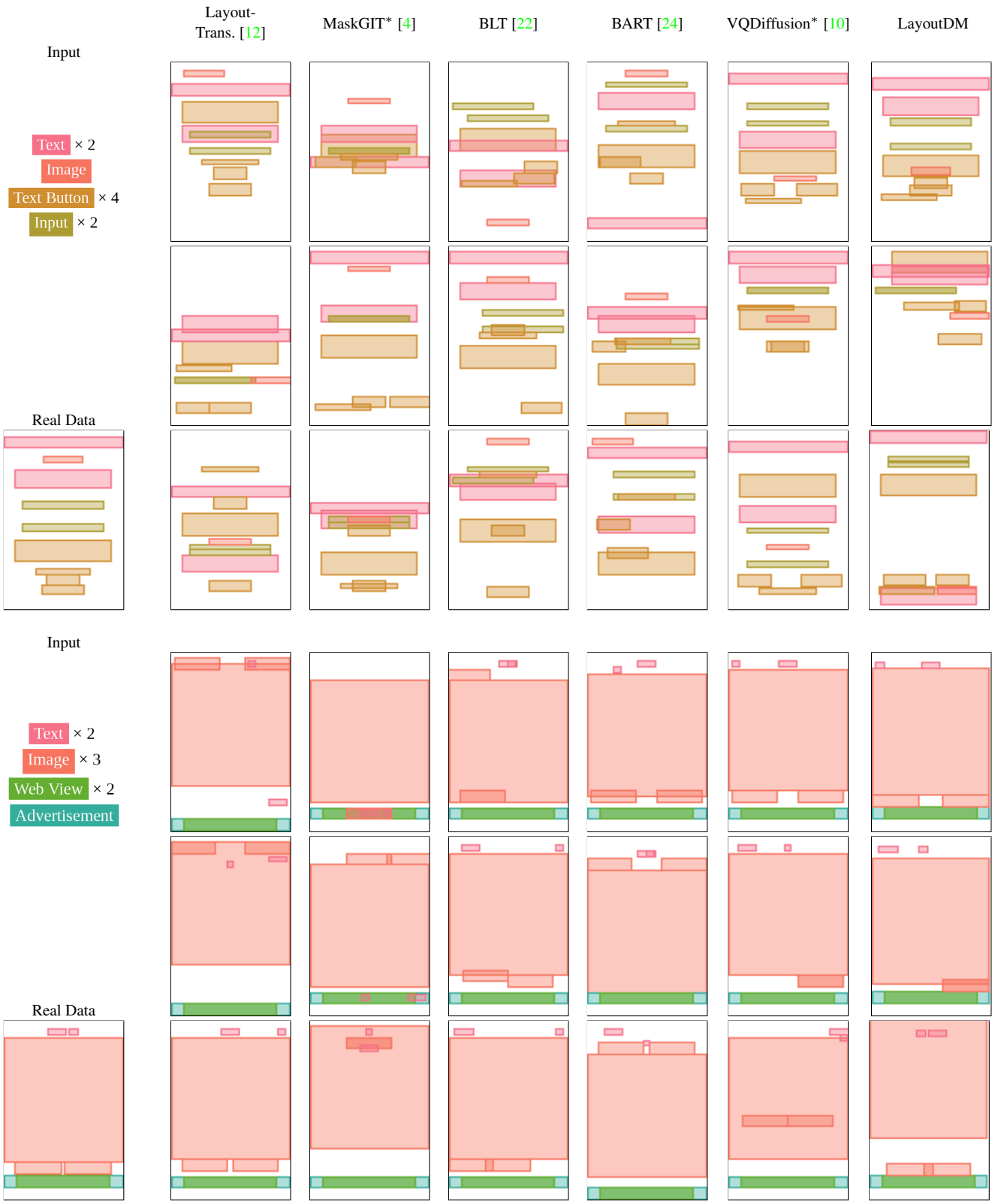


Figure 12. Comparison of conditional generation in C+S→P for Rico. We obtain three samples from each model to demonstrate the diversity. Note that the size condition of each element is not shown for limited space. Please refer to Real Data for the size.

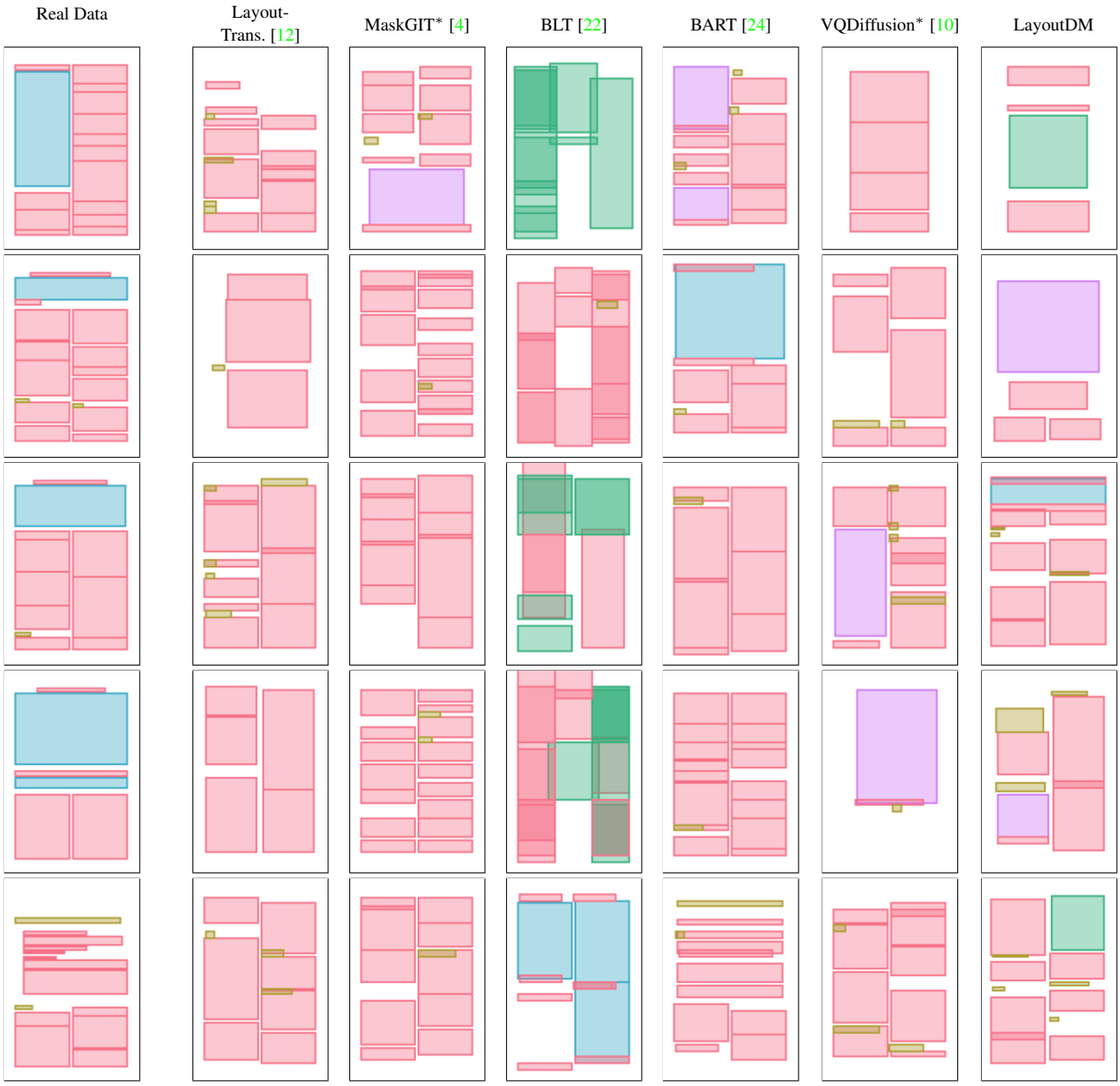


Figure 13. Comparison of unconditional generation for PubLayNet. We obtain five samples from each model to demonstrate the diversity.

Figure 14. Comparison of unconditional generation for Rico. We obtain five samples from each model to demonstrate the diversity.

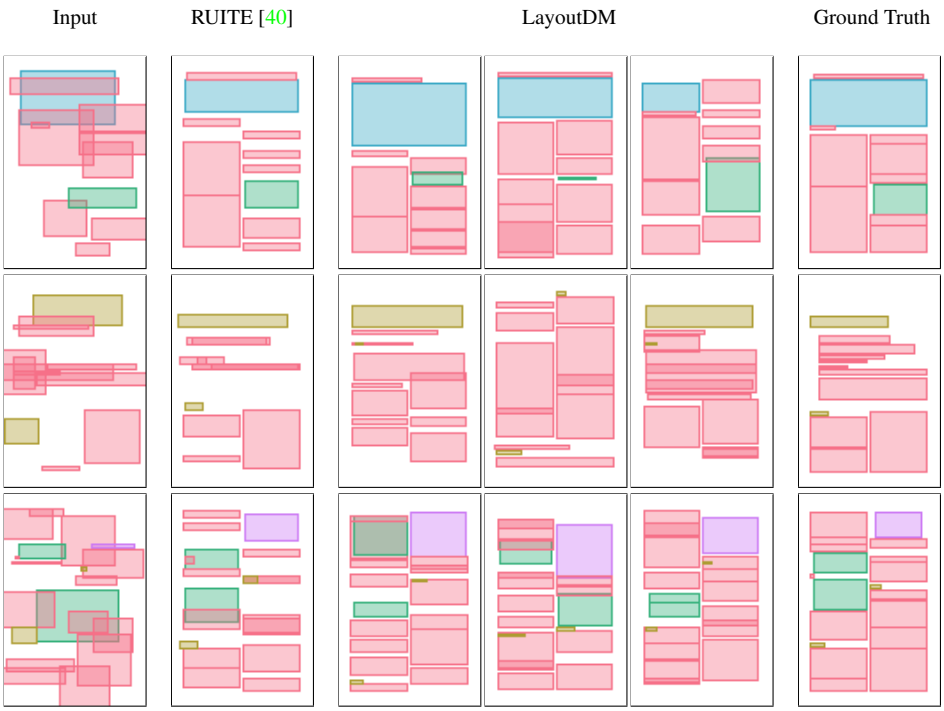


Figure 15. Comparison of the refinement task for PubLayNet. We obtain three samples from LayouytDM to demonstrate the diversity.

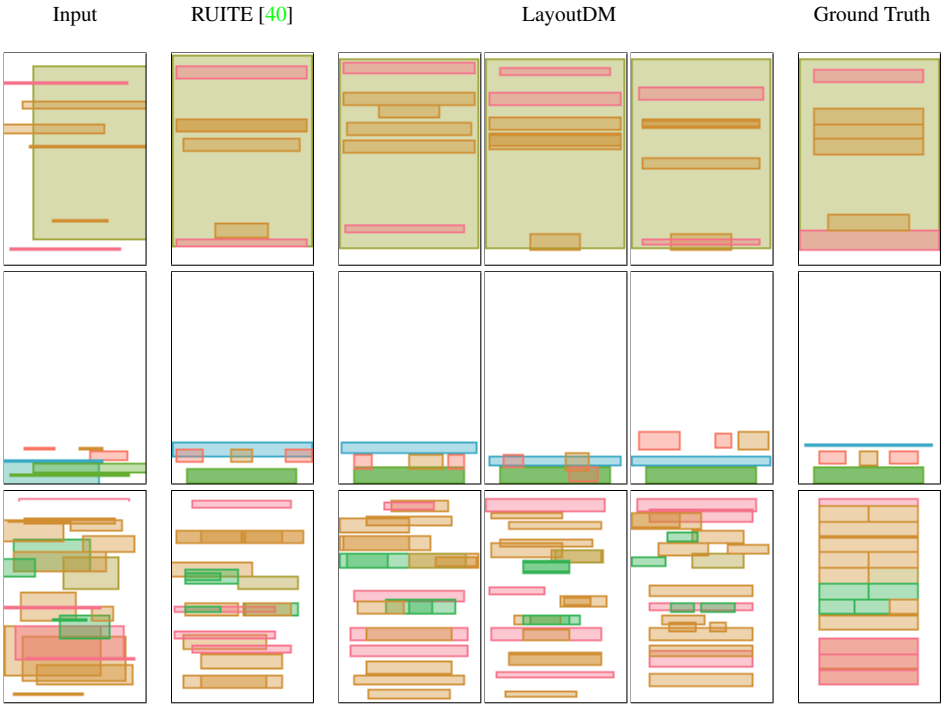


Figure 16. Comparison of the refinement task for Rico. We obtain three samples from LayouytDM to demonstrate the diversity.

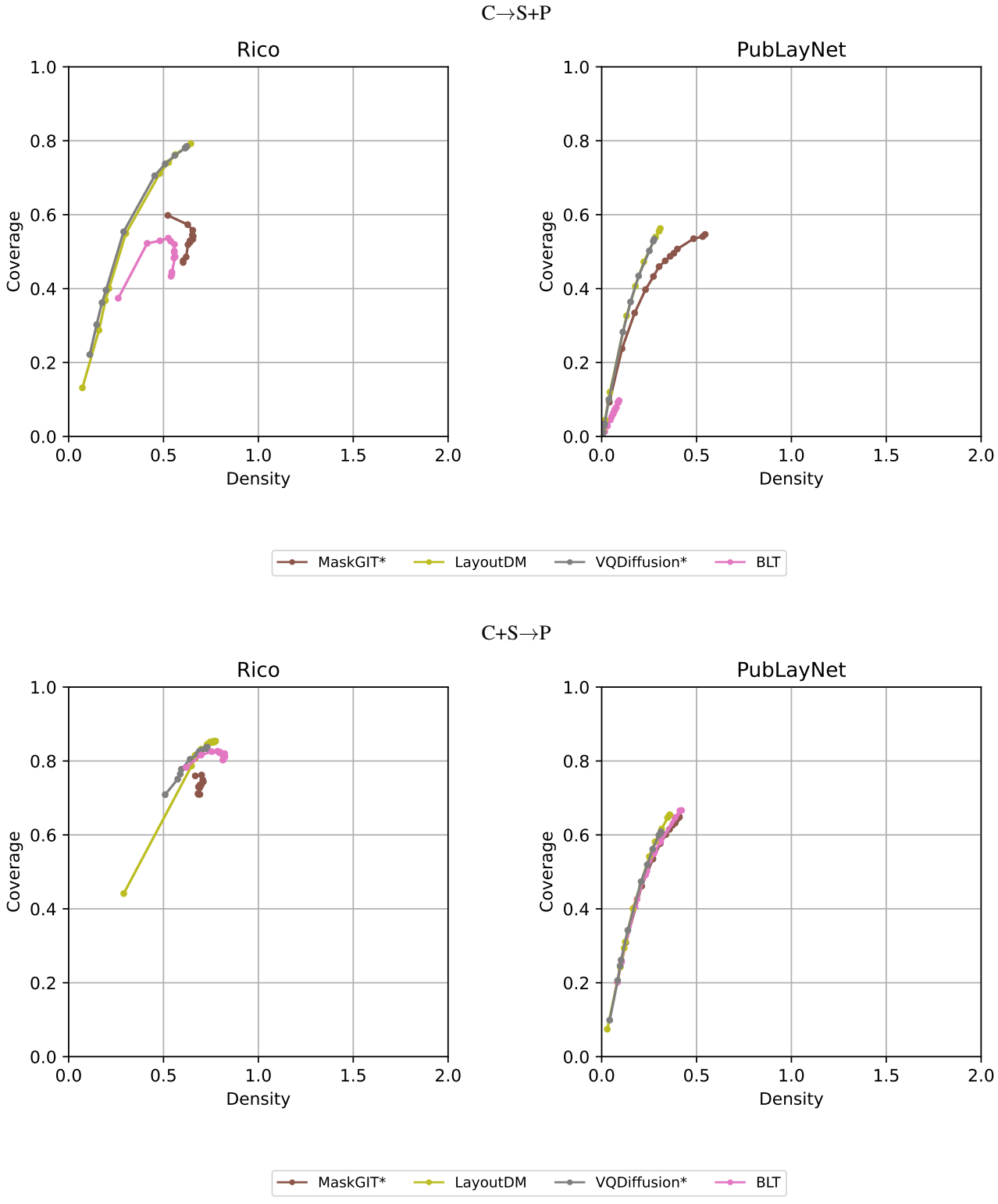


Figure 17. Density-coverage trade-off of different models.

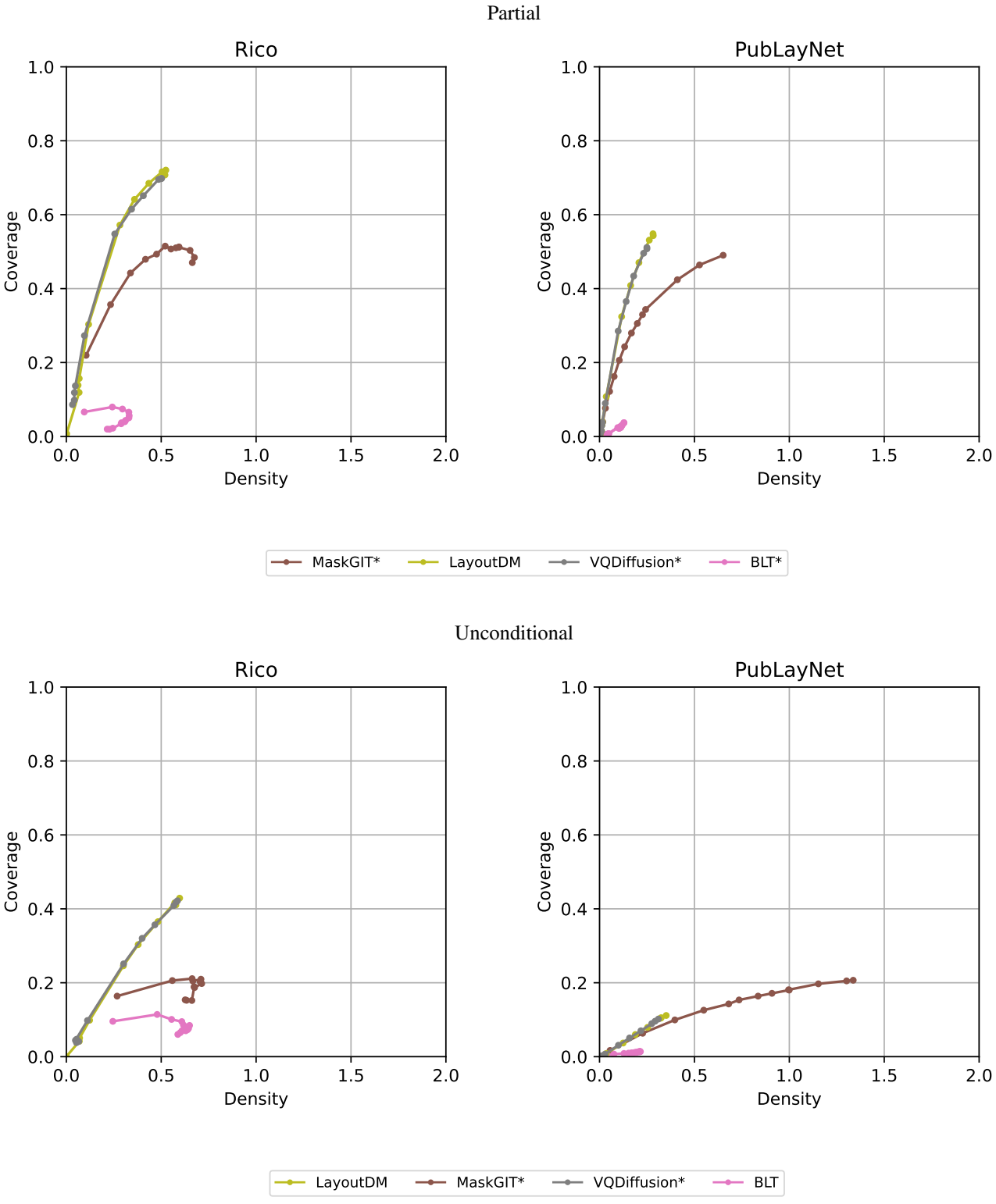


Figure 17. (cont.) Density-coverage trade-off of different models.

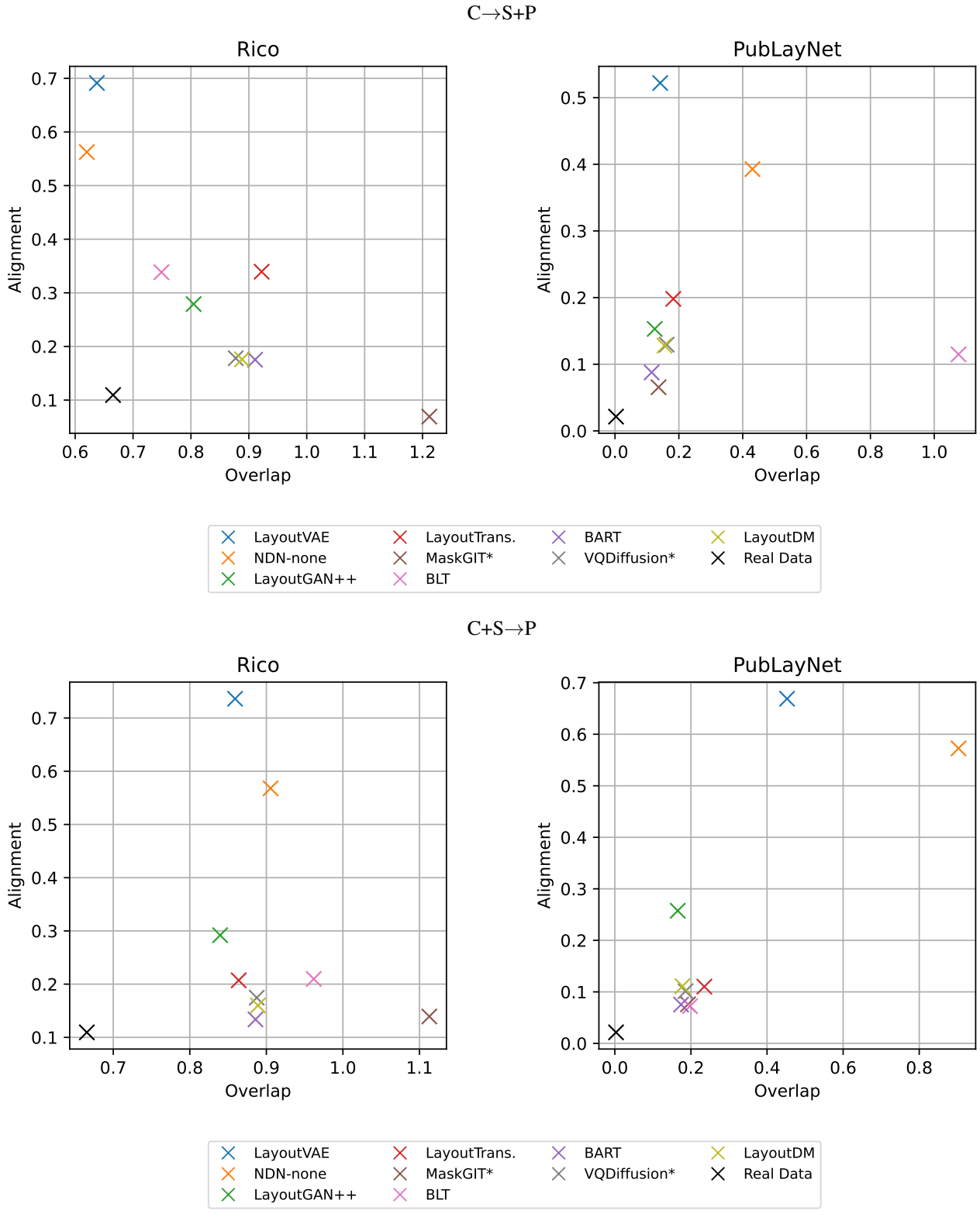
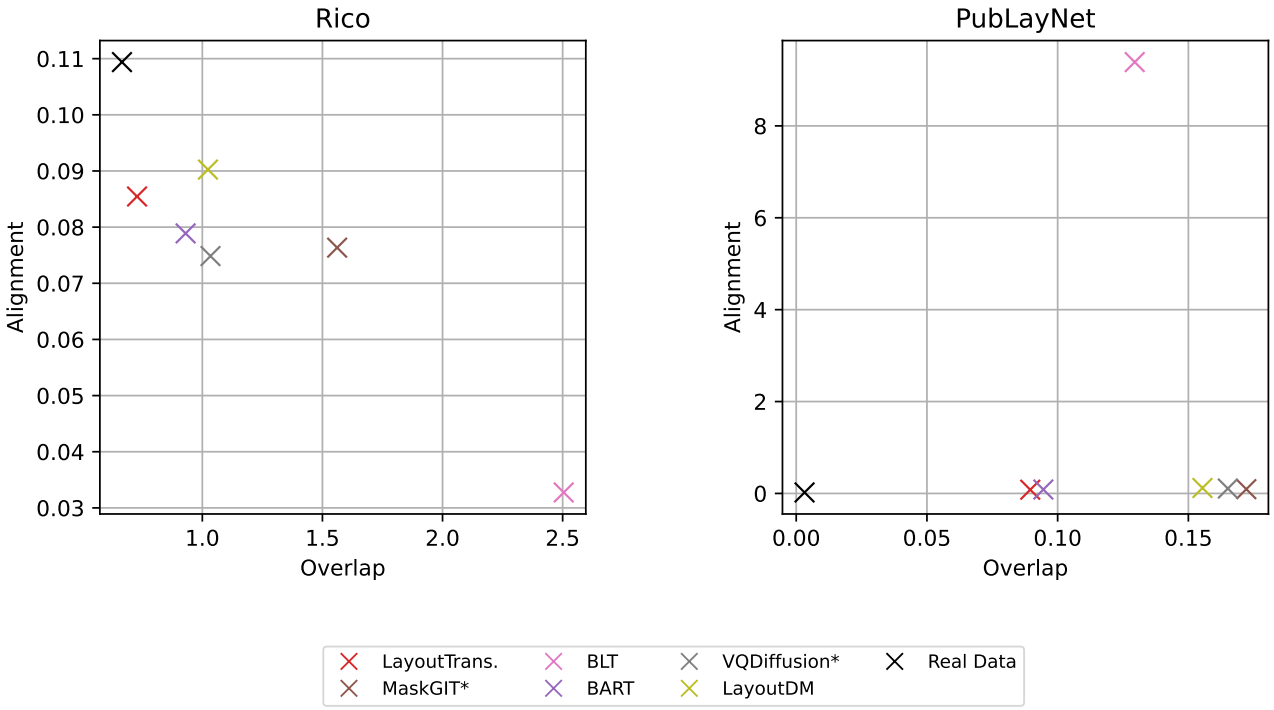


Figure 18. Alignment and overlap of different models.

Partial



Unconditional

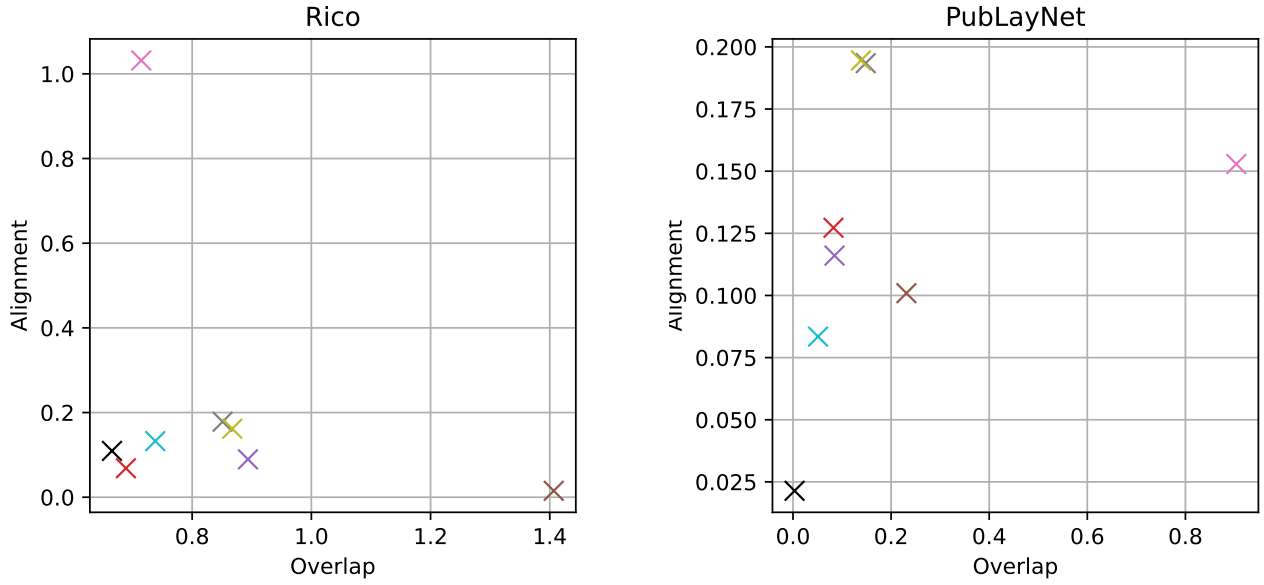


Figure 18. (cont.) Alignment and overlap of different models.