A Details of Token Refinement Task

In Sec. 3.2, we conducted preliminary experiments to evaluate the token refinement capability of DDMs. We present the experimental setup and results in more detail.

A.1 Transition Probability Design

We use LayoutDM [21] as the representative of DDMs. The default setting of $\bar{\beta}_{t,K} = (K+1)\bar{\beta}_t$ in LayoutDM is not exactly zero but is sufficiently close. As a baseline, we set $\bar{\beta}_{t,K} = \epsilon$ for any timestep t, where ϵ equals to 10^{-6}. In this setting, the diffusion process primarily induces transitions from regular tokens to `[MASK]`, and transitions between regular tokens rarely occur. Therefore, we can not expect corrections of errors in regular tokens during the generation process. Setting a large value for $\bar{\beta}_{t,K}$ is expected to facilitate transitions between regular tokens during the diffusion process, allowing the corresponding denoising model $p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ to acquire the capability to correct regular tokens. To verify the effect of $\bar{\beta}_{t,K}$ schedule, we consider schedules for $\bar{\beta}_{t,K}$ based on two guidelines. The first involves assigning high $\bar{\beta}_{t,K}$ values later in the diffusion process, while the second involves high $\bar{\beta}_{t,K}$ values earlier. A detailed schedule, including $\bar{\alpha}_t$ and $\bar{\gamma}_t$, is shown in Fig. 12. Here, we adopt a linear scheduling for timesteps.

A.2 Impact of Transition Schedules on FID

In Tab. 5, we report the results of FID for each schedule depicted in Fig. 12. When $\bar{\beta}_{t,K}$ increases from ϵ to 0.05 or 0.1 with the timestep t, the performance is comparable to the baseline for the unconditional generation task; however, we observe degradation in the conditional generation tasks. Conversely, when $\bar{\beta}_{t,K}$ decreases from 0.05 or 0.1 to ϵ, the performance is inferior to the baseline for both unconditional and conditional tasks.

Regarding the degradation in conditional generation tasks, we hypothesize that it stems from the condition gap between training and inference time. Training is conducted in an unconditional manner, where, especially for $\bar{\beta}_{t,K} > \epsilon$, the model learns to restore the original layout while correcting substitutions of regular tokens. On the other hand, in conditional settings, the model is expected to preserve the conditioned regular tokens, leading to the discrepancy between the training and inference phases. When $\bar{\beta}_{t,K} = \epsilon$, substitutions between regular tokens rarely occur, which means that conditioning on regular tokens does not negatively impact the generation process.

Additionally, applying high $\bar{\beta}_{t,K}$ values in the earlier timesteps leads to poor FID in the unconditional setting. This schedule causes rapid replacements of regular tokens, indicated by $\bar{\alpha}_t < 1$, as observed in Fig. 12d and Fig. 12e. The results imply that it is necessary to design a schedule for $\bar{\alpha}_t$ that starts at 1.0 when $t = 0$ and gradually decreases as t increases, reflecting the fundamental concept of the discrete diffusion process.

Table 5: FID scores of LayoutDM for various $\bar{\beta}_{t,K}$ schedules. The best and second-best results are highlighted in **bold** and with underline, respectively.

	FID↓		
$\bar{\beta}_{t,K}$ schedule	Unconditional	C→S+P	C+S→P
$\epsilon \to \epsilon$	6.37	**3.51**	**2.17**
$\epsilon \to 0.05$	**6.22**	4.03	4.38
$\epsilon \to 0.1$	6.29	4.68	5.69
$0.05 \to \epsilon$	7.98	5.21	5.11
$0.1 \to \epsilon$	10.71	8.00	7.96

B More Detailed Experimental Setup

In this section, we describe the experimental setup in detail in addition to the description in Sec. 4.1.

B.1 Datasets

We provide a more detailed explanation of the benchmark datasets used for evaluation, focusing particularly on how the datasets are divided and their respective sample numbers.

- **Rico** [8]: We follow the dataset split in [21], resulting in 35,851 / 2,109 / 4,218 samples for train, validation, and test set.
- **PubLayNet** [50]: We use the dataset split in [21], resulting in 315,757 / 16,619 / 11,142 samples for train, validation, and test splits.
- **Crello** [45]: While the dataset provides various attributes for each element, such as opacity, color, and image data, we only utilize category, position, and size. We use the official splits, which result in 18,714 / 2,316 / 2,331 samples for train, validation, and test set, respectively.

B.2 Implementation Details

Model Architecture. Our Layout-Corrector employs a 4-layer Transformer Encoder with 8 multi-heads. For Token-Critic [29] in Table Tab. 1, we used the same architecture as Layout-Corrector. For LayoutDM [21], VQDiffusion [13], and MaskGIT [5] experiments, we utilized the official implementation of LayoutDM.[3] For LayoutDM* in Sec. 4.4, we used a 12-layer Transformer with 12 multi-heads to obtain the same model size as LayoutDiffusion [47]. Note that we used the same Layout-Corrector architecture with a 4-layer Transformer for LayoutDM*. For LayoutDiffusion [47], we used the official implementation[4].

[3] https://github.com/CyberAgentAILab/layout-dm

[4] https://github.com/microsoft/LayoutGeneration/tree/main/LayoutDiffusion

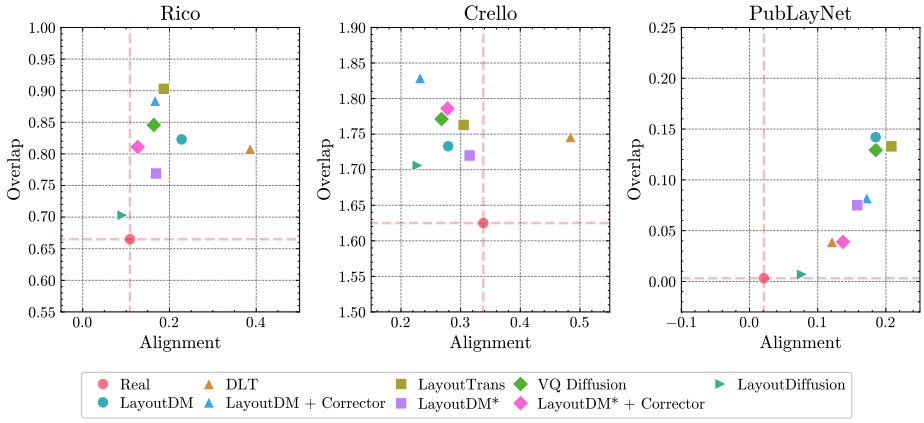


Fig. 11: Alignment and overlap [24] scores across various methods and real data on three datasets. Alignment score is scaled by 100× for visibility.

Training. We employed the shared pre-trained LayoutDM models on Rico and PubLayNet datasets. For other models, including Layout-Corrector, we followed the training configuration of LayoutDM, using AdamW optimizer [25,34] with an initial learning rate of 5.0×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.98)$, and batch size of 64. The number of training epochs varied according to the dataset: 20 for PubLayNet, 50 for Rico, and 75 for Crello. For LayoutDiffusion [47], since the dataset configuration (*i.e.*, the maximum number of elements in a layout) in the official implementation is different from our setting, we trained the model from scratch instead of using the official checkpoints.

C Quantitative Evaluation

In this section, we present additional quantitative evaluation results, including the effectiveness of Layout-Corrector on conditional generation and the results of Alignment and Overlap metrics.

C.1 Effectiveness of Layout-Corrector on Conditional Generation

Tab. 6 shows a comparison of the performance of Token-Critic [29] and Layout-Corrector on conditional generation when applied to three baseline models (*i.e.*, MaskGIT, VQDiffusion, and LayoutDM). Layout-Corrector constantly improves the FID scores of the baseline models.

C.2 Alignment and Overlap

Fig. 11 shows the relationship between Alignment and Overlap [24] on three datasets. We also show the scores of real data for reference. When compared

Table 6: Performance comparison of baseline models with/without external assessor on conditional generation. *Arch.* represents the architecture of the discrete generative model. Metrics improved by the external module are highlighted in **bold**.

		Rico [8]			Crello [45]			PubLayNet [50]		
Model	Arch.	FID↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑
MaskGIT [5]	Non-AR	30.25	0.759	0.526	31.03	0.821	0.456	16.62	0.498	0.801
+ Token-Critic [29]		**10.93**	0.734	**0.817**	**5.85**	0.759	**0.821**	**8.07**	**0.679**	**0.854**
+ **Corrector (ours)**		**7.78**	**0.814**	**0.795**	**6.53**	**0.843**	**0.789**	**7.86**	**0.503**	**0.937**
VQDiffusion [13]	DDMs	4.01	0.750	0.877	3.98	0.757	0.874	7.57	0.595	0.942
+ Token-Critic [29]		**2.89**	**0.828**	0.836	4.82	**0.802**	0.829	**5.96**	**0.789**	0.827
+ **Corrector (ours)**		**2.53**	**0.790**	**0.878**	**3.63**	**0.791**	0.834	**5.61**	**0.678**	0.932
LayoutDM [21]	DDMs	3.51	0.768	0.899	4.04	0.759	0.876	7.94	0.549	0.939
+ Token-Critic [29]		**3.15**	**0.842**	0.846	4.43	**0.822**	0.816	**6.51**	**0.806**	0.819
+ **Corrector (ours)**		**2.39**	**0.808**	**0.905**	**3.39**	**0.797**	0.855	**5.84**	**0.660**	0.933

(a) C → S + P task

		Rico [8]			Crello [45]			PubLayNet [50]		
Model	Arch.	FID↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑
MaskGIT [5]	Non-AR	8.15	0.821	0.840	9.59	0.822	0.741	5.05	0.584	0.905
+ Token-Critic [29]		**4.51**	0.797	**0.905**	**4.68**	0.771	**0.871**	**3.83**	**0.630**	**0.917**
+ **Corrector (ours)**		**3.61**	**0.825**	**0.894**	**4.26**	**0.826**	**0.842**	**3.97**	**0.607**	**0.934**
VQDiffusion [13]	DDMs	2.37	0.828	0.929	3.89	0.779	0.878	4.05	0.612	0.949
+ Token-Critic [29]		**2.24**	**0.845**	0.926	3.99	**0.787**	**0.881**	**2.58**	**0.724**	0.927
+ **Corrector (ours)**		**2.02**	**0.845**	0.921	**3.46**	**0.813**	0.876	**2.72**	**0.679**	0.935
LayoutDM [21]	DDMs	2.17	0.844	0.928	3.55	0.800	0.885	4.22	0.587	0.941
+ Token-Critic [29]		**2.06**	**0.860**	0.912	3.57	**0.803**	**0.888**	**2.60**	**0.712**	0.925
+ **Corrector (ours)**		**1.91**	**0.856**	0.922	**3.32**	**0.808**	0.882	**2.93**	**0.667**	0.936

(b) C + S → P task

with the baseline of LayoutDM, Layout-Corrector reduces Alignment on three datasets. Regarding Overlap, the score is increased by applying Layout-Corrector on Rico and Crello datasets, while it is reduced on PubLayNet. While those hand-crafted metrics express a quality for intuitive visual appearance, as seen in Appendix E.4, they do not necessarily correlate with improvements in the higher-order generative quality represented by the FID score.

D Qualitative Evaluation

In this section, we present additional qualitative results, including additional visualization of the generation results, visualization of the generation process, fidelity-diversity trade-off of the generation results, and failure cases.

D.1 Additional Results

We report additional qualitative results for three datasets, including Rico, Crello, and PubLaynet. Fig. 13, Fig. 14, and Fig. 15 show the samples of unconditional

generation. Fig. 16, Fig. 17, and Fig. 18 show the samples of C→P+S task. Fig. 19, Fig. 20, and Fig. 21 show the samples of C+S→P task. To demonstrate the diversity, we show eight samples for unconditional generation. For conditional generation, we show four samples for each conditional input.

D.2 Visualization of the Generation Process

We present the layout visualization during the generation process for LayoutDM and its integration with Layout-Corrector. Fig. 22, Fig. 23, Fig. 24 are the results of unconditional generation for the Rico, Crello, and PubLayNet datasets, respectively. At earlier timesteps, such as $t \geq 40$, few elements have been generated, so we focus on visualizing the timesteps from $t = 38$ to 0. The corrector is applied at timesteps $t = \{10, 20, 30\}$, which is the optimal schedule based on the FID score, as discussed in Sec. 4.3. It is important to note that until $t > 30$, both models follow the identical generation process. The results demonstrate that Layout-Corrector effectively eliminates inharmonious elements at the timesteps when the corrector is applied, leading to more consistent results compared to the baseline.

D.3 Fidelity-Diversity Trade-Off

Fig. 25 displays the results from different scheduling scenarios of Layout-Corrector, illustrating layouts generated by LayoutDM [21] with and without the Layout-Corrector under two distinct corrector schedules: $t = \{10, 20, 30\}$ and $t = \{10, 20, \ldots, 90\}$. Layouts generated with Layout-Corrector applied at $t = \{10, 20, 30\}$ demonstrate rich diversity. In contrast, more frequent application of Layout-Corrector at $t = \{10, 20, \ldots, 90\}$ results in a noticeable increase in layouts featuring centrally aligned elements along the horizontal axis, indicating reduced diversity. It is consistent with the observations in Fig. 6, where the more frequent application of Layout-Corrector to LayoutDM enhances fidelity but reduces diversity, highlighting a trade-off between these two aspects.

We showed the histogram of the width attribute across different corrector schedules in Fig. 7. Here, we also report the histogram of the other four attributes (*i.e.*, category, x-center, y-center, and height) in Fig. 26. We observed the same trend as Fig. 7, where the more frequent application of Layout-Corrector amplifies the frequency trends of the original data.

D.4 Typical Failure Cases

While Layout-Corrector can improve the generation quality of baseline models, it is not infallible. Typical failure cases are presented in Fig. 27, where we compare the layouts generated by LayoutDM [21] with and without Layout-Corrector. In Fig. 27a, Layout-Corrector effectively resolves overlap and misalignment in LayoutDM's output, but this produces unnatural blank spaces in the output. This issue arises because, although Layout-Corrector enables DDMs to modify

Table 7: Ablation study on Crello [45] and PubLayNet [50] dataset with unconditional generation.

	$T' = 100$		$T' = 20$	
	FID↓	Align.→	FID↓	Align.→
Layout-Corrector	**4.36**	0.232	**5.11**	0.295
Mask estimation	4.71	0.285	6.22	**0.336**
w/o Self-Atteniton	4.42	0.260	6.11	0.317
Top-K	6.58	**0.300**	5.45	0.296
Correcting at every t	90.24	0.009	48.78	0.038
Real Data	2.32	0.338	2.32	0.338

(a) Crello [45]

	$T' = 100$		$T' = 20$	
	FID↓	Align.→	FID↓	Align.→
Layout-Corrector	11.85	0.172	**15.39**	0.178
Mask estimation	**11.40**	0.167	16.71	0.194
w/o Self-Atteniton	13.49	0.172	20.71	0.286
Top-K	19.96	0.615	23.89	0.443
Correcting at every t	69.21	**0.125**	53.26	**0.092**
Real Data	6.25	0.021	6.25	0.021

(b) PubLayNet [50]

incorrectly generated layouts by resetting tokens with low correctness scores, it does not encourage DDMs to create additional elements, leading to these blank areas. In Fig. 27b, Layout-Corrector fixes an overlap in the bottom-right of LayoutDM's output, yet a new overlap emerges in the top-left of the LayoutDM + Corrector output. This is because Layout-Corrector can not correct tokens generated after its final application.

E Ablation Study

In this section, we present additional results of the ablation study, including Crello [45] and PubLayNet [50], and architecture of Layout-Corrector, and threshold value θ_{th}. In addition, we compare Layout-Corrector with rule-based post-processing [24].

E.1 Additional Results

Tab. 7 shows the ablation results for Crello and PubLayNet in the unconditional generation task. Please refer to Sec. 4.6 regarding the configurations. As with the results of Rico dataset [8], Layout-Corrector achieves solid performance on both Crello and PubLayNet datasets. For Crello dataset shown in Tab. 7a, we observed that removing the self-attention layer results in a less significant performance drop than in other datasets. We hypothesize that this phenomenon is due to the complex and diverse relationships between elements in Crello, as illustrated by real samples in Fig. 14. When the relationships among elements are complicated, it is challenging for the self-attention layers to capture these relationships, resulting in decreased effectiveness.

E.2 Corrector Architecture

We report the effects of varying the number of Transformer Encoder layers in Layout-Corrector. To investigate this, we trained Layout-Corrector with $\{1, 2, 4, 6\}$

Table 8: The effect of the number of Transformer Encoder layers on Rico test set. The best FID result is highlighted in **bold**.

# of layers	FID↓	# of params [M]	Time/sample [ms]
- (LayoutDM)	6.37	12.4	23.7
1	5.07	+ 4.6	24.6
2	4.94	+ 7.3	24.6
4	**4.79**	+ 12.6	24.6
6	4.93	+ 17.9	24.6

Table 9: The impact of threshold θ_{th} in unconditional generation on three dataset using LayoutDM [21] as DDM. The best result is highlighted in **bold**.

Threshold θ_{th}	Rico			Crello			PubLayNet		
	FID↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑
0.3	5.57	0.763	0.900	5.16	0.775	**0.874**	12.88	0.605	**0.920**
0.4	5.26	0.779	0.897	4.97	0.789	0.861	12.41	0.639	0.918
0.5	5.05	0.787	**0.900**	4.75	0.799	0.861	12.06	0.668	0.914
0.6	4.90	0.794	0.892	4.45	0.806	0.859	**11.78**	0.681	0.911
0.7	**4.79**	0.809	0.892	**4.36**	0.822	0.851	11.85	0.711	0.890
0.8	5.01	0.822	0.876	4.51	0.824	0.834	11.88	0.727	0.887
0.9	5.89	**0.844**	0.858	5.77	**0.849**	0.811	12.60	**0.729**	0.869

layers on the Rico dataset [8] and evaluated FID scores, number of parameters, and inference speed. The application schedule for Layout-Corrector was set to $t = \{10, 20, 30\}$. The results, presented in Tab. 8, indicate that the best FID is achieved with 4 encoder layers. Although the number of parameters increases with the number of layers, the impact on inference speed remains minimal since the corrector is applied just three times.

E.3 Threshold θ_{th}

We compared various threshold values θ_{th} in Tab. 9 on three datasets. The results show that $\theta_{th} = 0.7$ yields the best FID on Rico and Crello, and the second-best on PubLayNet, demonstrating that it performs well across various datasets without tailored calibration. Precision and Recall scores are also presented in the table to provide a more comprehensive analysis. A higher threshold keeps only high-scored tokens, leading to higher fidelity (Precision) at the expense of diversity (Recall). In contrast, a lower threshold allows the inclusion of low-scored tokens, potentially enhancing diversity at the cost of reduced fidelity.

E.4 Effect of post-processing

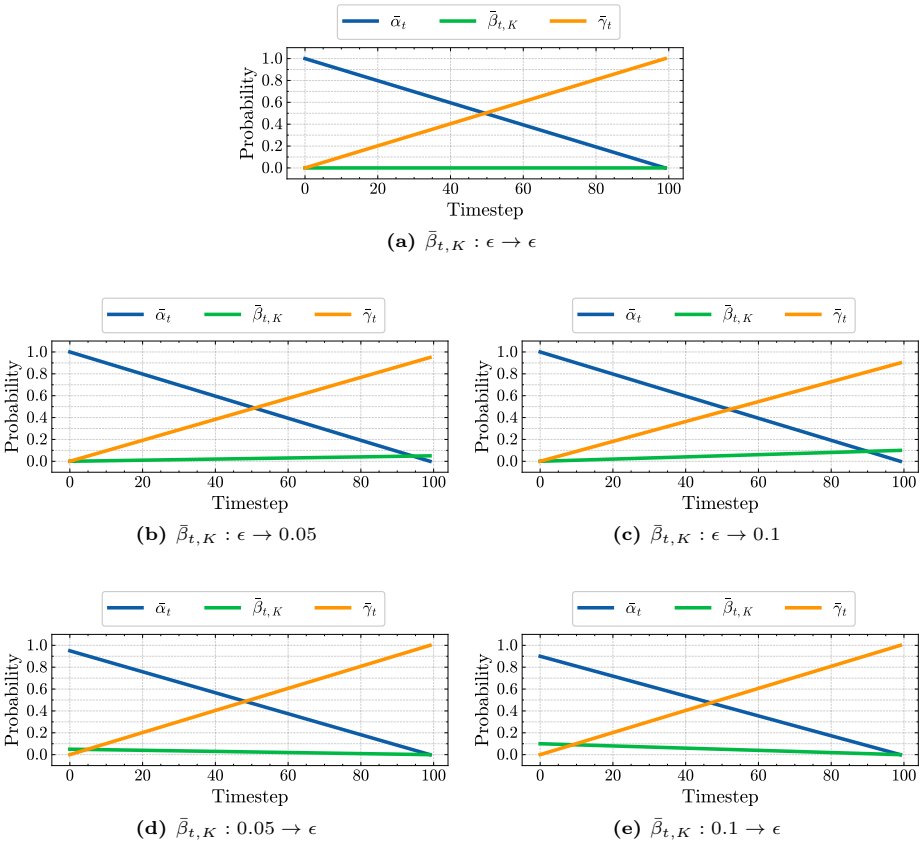
In this section, we investigate the applicability of post-processing to refine layouts. To achieve this, we use the layouts generated by DDMs and refine them using rule-based methods. Following the approach of CLG-LO [24], we apply

Table 10: Performance comparison of baseline models with/without post-processing on the unconditional generation task. Metrics improved by post-processing are highlighted in **bold**.

	Rico [8]			Crello [45]			PubLayNet [50]		
Model	FID↓	Align.→	Overlap→	FID↓	Align.→	Overlap→	FID↓	Align.→	Overlap→
LayoutDM [21]	6.37	0.223	0.841	5.28	0.279	1.733	13.72	0.185	0.142
+ **post-processing**	**6.23**	**0.211**	0.854	**5.20**	0.258	1.738	13.77	**0.16**	**0.052**
LayoutDM + Corrector	4.79	0.167	0.884	4.36	0.232	1.829	11.85	0.172	0.082
+ **post-processing**	4.87	**0.158**	0.897	4.36	0.215	1.834	**10.81**	**0.120**	**0.023**
Real data	1.85	0.109	0.665	2.32	0.338	1.625	6.25	0.021	0.0032

constraint optimization to geometric metrics, including alignment and overlap scores, to minimize these costs while modifying geometric attributes. For datasets characterized by a large overlap, such as Rico and Crello, we adjust the optimization by omitting the overlap term from the objective function and focusing solely on minimizing the alignment.

Tab. 10 shows the effect of post-processing on LayoutDM and its combination with Layout-Corrector. We observe that post-processing does not significantly affect the FID score, except for LayoutDM + Corrector on PubLayNet, which has lower alignment and overlap scores. We consider that optimization based on geometric constraints is ineffective for layouts with complex structures, such as Rico and Crello. On the other hand, Layout-Corrector outperforms post-processing in terms of FID because it intervenes in the generation process to realize layout correction. This suggests that our learning-based approach is far more effective than simple rule-based optimization.



(a) $\bar{\beta}_{t,K} : \epsilon \rightarrow \epsilon$

(b) $\bar{\beta}_{t,K} : \epsilon \rightarrow 0.05$

(c) $\bar{\beta}_{t,K} : \epsilon \rightarrow 0.1$

(d) $\bar{\beta}_{t,K} : 0.05 \rightarrow \epsilon$

(e) $\bar{\beta}_{t,K} : 0.1 \rightarrow \epsilon$

Fig. 12: Scheduling of transition probabilities for preliminary experiments. Fig. 12a illustrates the baseline schedule used in LayoutDM, where $\bar{\beta}_{t,K}$ is approximately zero at any timestep. Fig. 12b and Fig. 12c demonstrate the schedules that introduce transitions between regular tokens in the later stages of the diffusion process. Conversely, the schedules of Fig. 12d and Fig. 12e promote transitions between them in the early stage of the diffusion process.

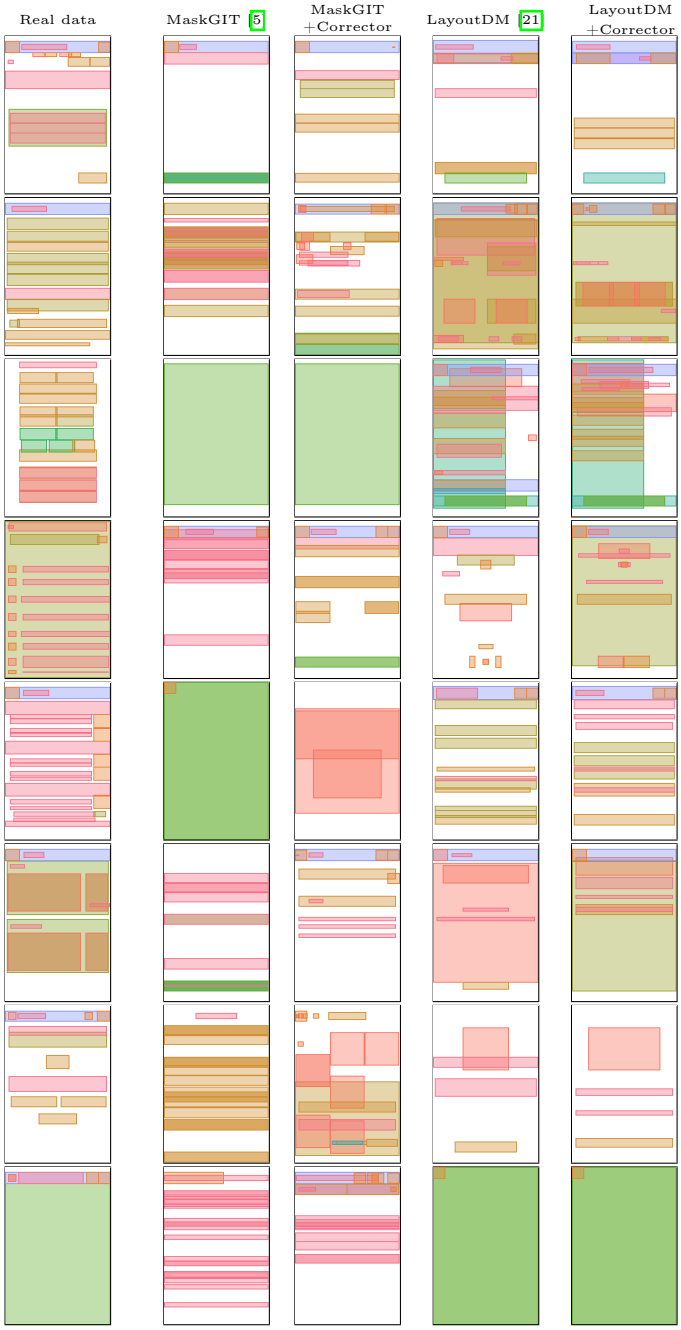


Fig. 13: Comparison of unconditional generation results on Rico, with eight samples from each model to show diversity.

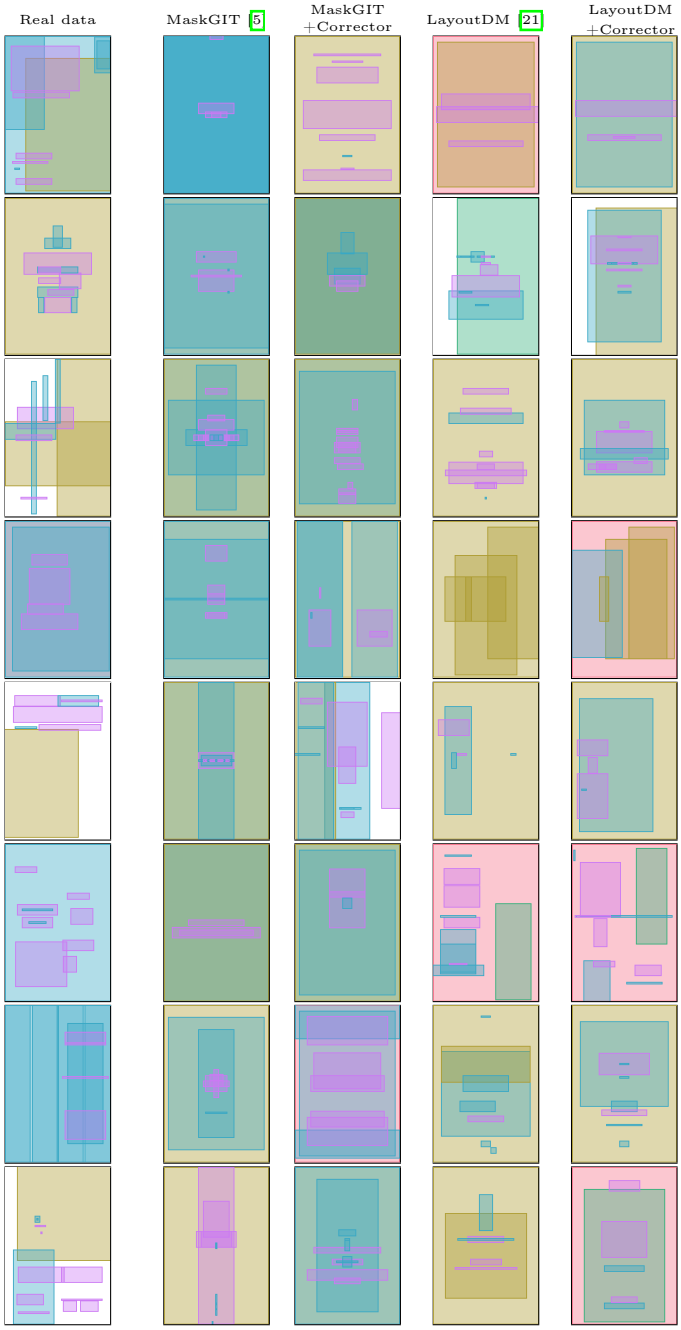


Fig. 14: Comparison of unconditional generation results on Crello, with eight samples from each model to show diversity.

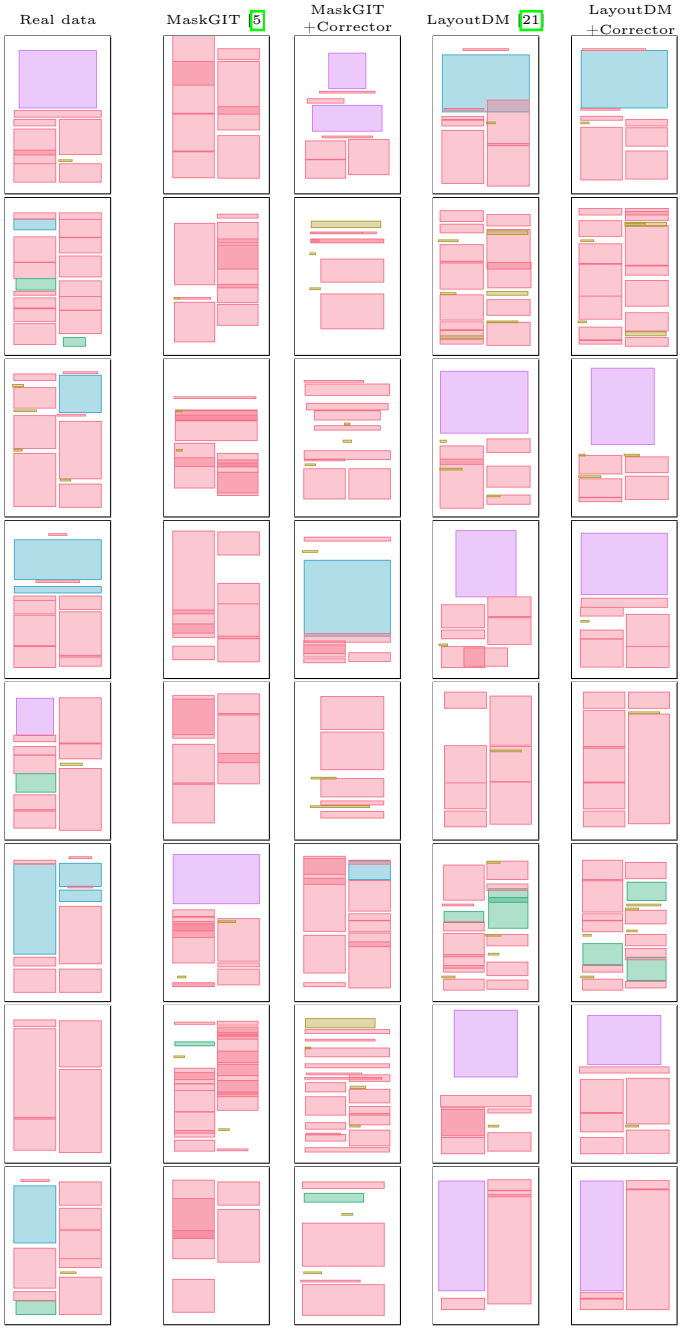


Fig. 15: Comparison of unconditional generation results on PubLayNet, with eight samples from each model to show diversity.

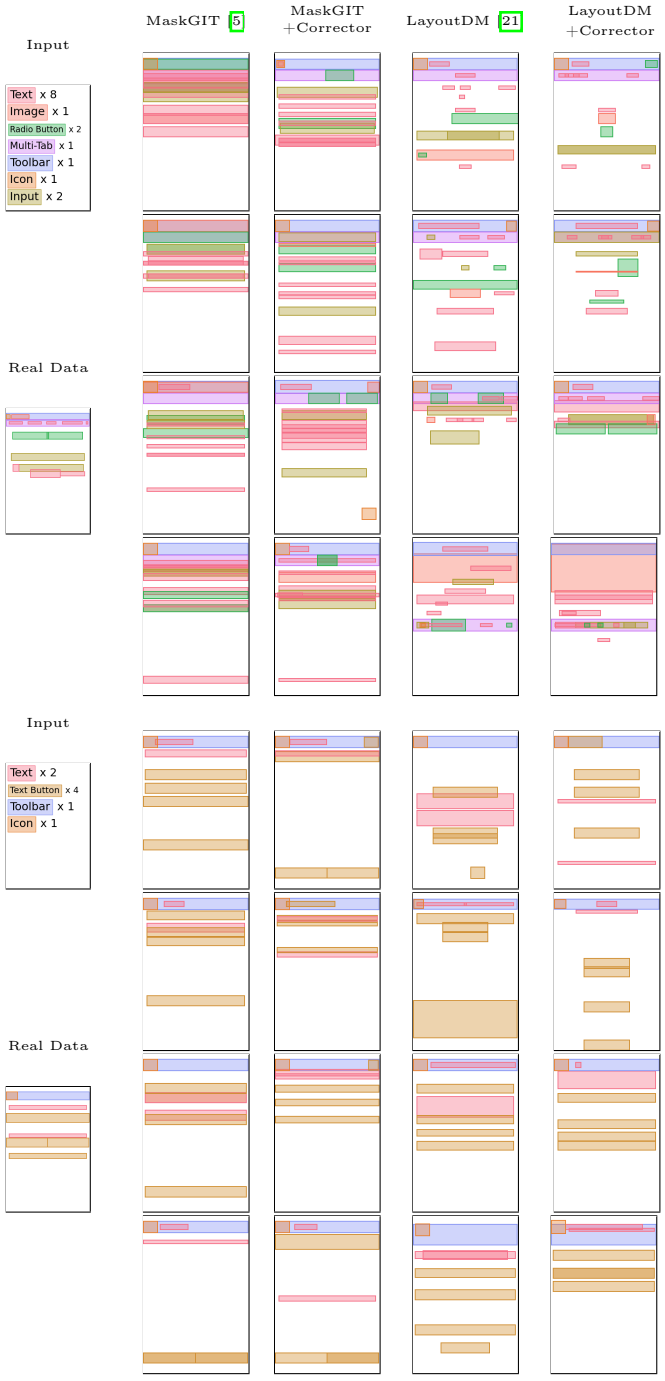


Fig. 16: Comparison of conditional generation results for C→S+P on Rico, with four samples per condition input from each model to show diversity.

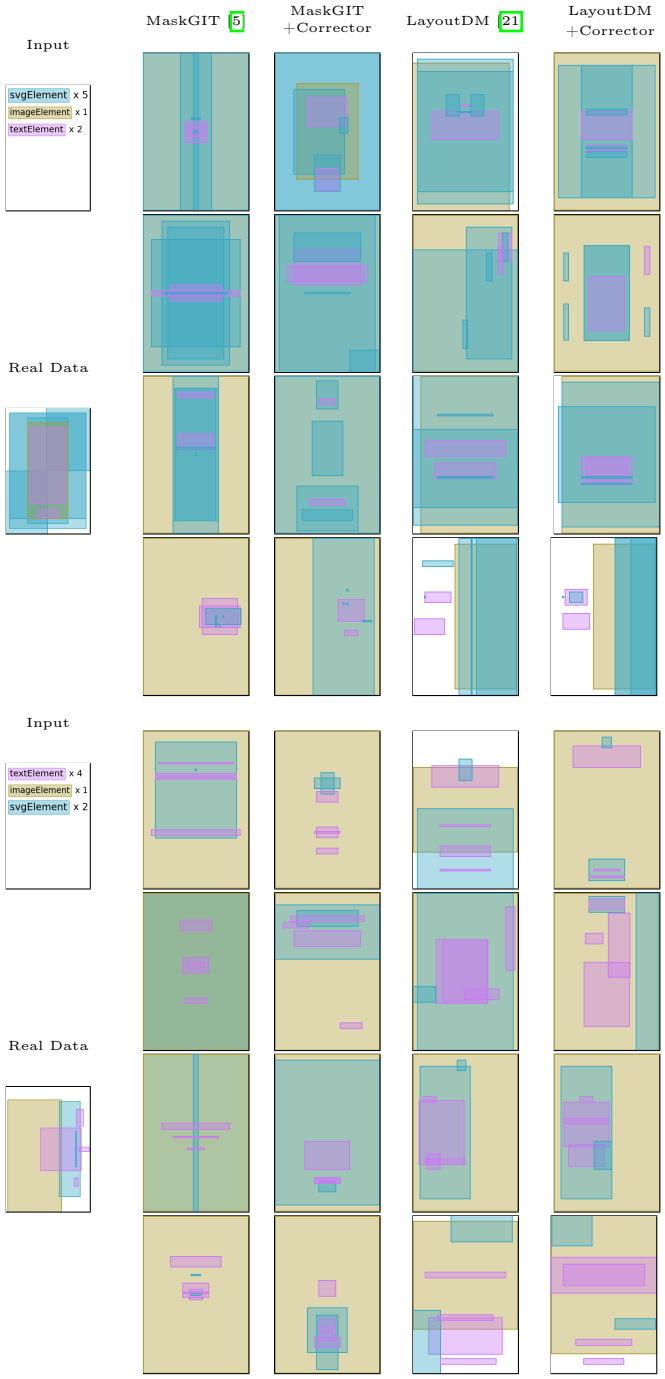


Fig. 17: Comparison of conditional generation results for C→S+P on Crello, with four samples per condition input from each model to show diversity.

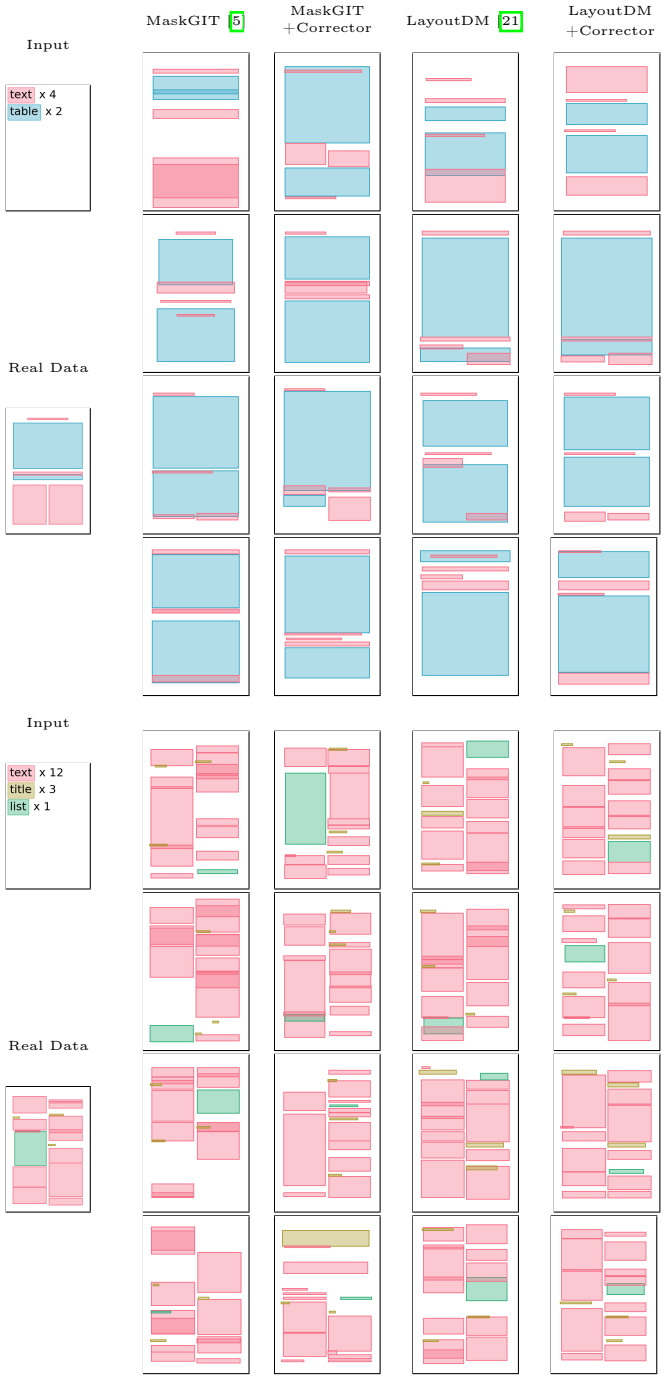


Fig. 18: Comparison of conditional generation results for C→S+P on PubLayNet, with four samples per condition input from each model to show diversity.

Fig. 19: Comparison of conditional generation results for C+S→P on Rico, with four samples per condition input from each model to show diversity.

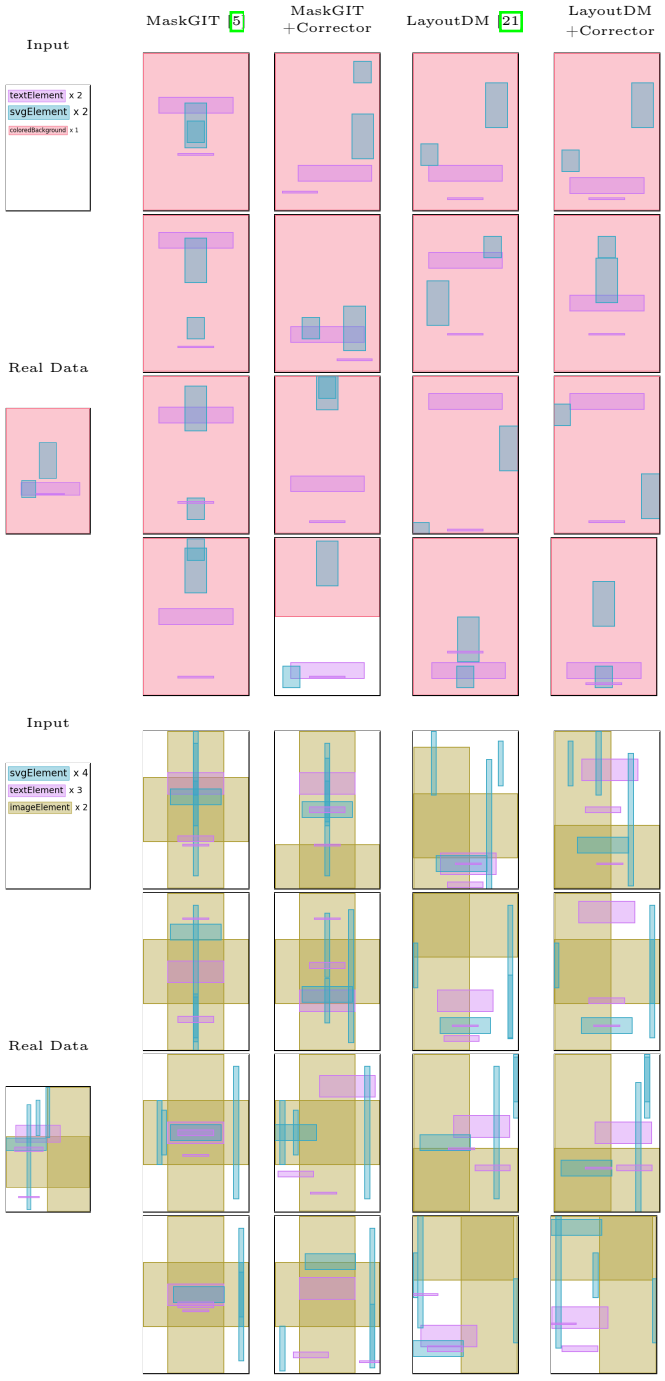


Fig. 20: Comparison of conditional generation results for C+S→P on Crello, with four samples per condition input from each model to show diversity.

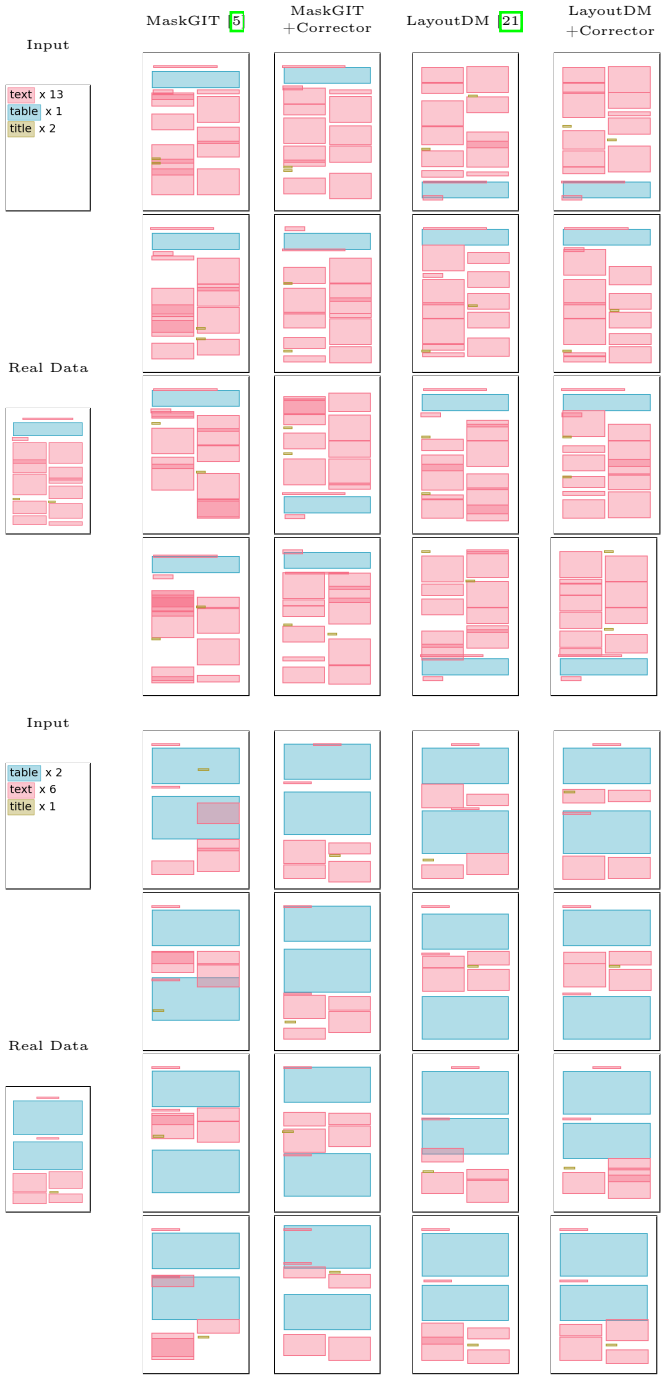


Fig. 21: Comparison of conditional generation results for C+S→P on PubLayNet, with four samples per condition input from each model to show diversity.

(a) Example1: LayoutDM

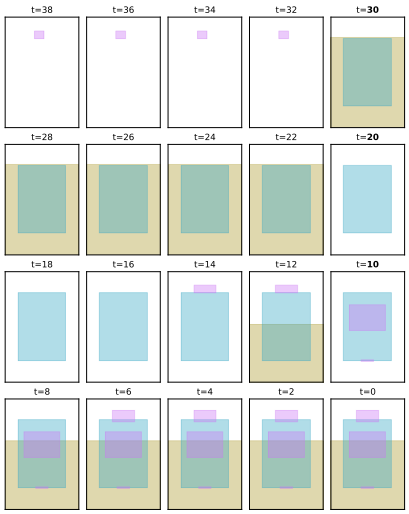
(b) Example1: LayoutDM + Corrector

(c) Example2: LayoutDM

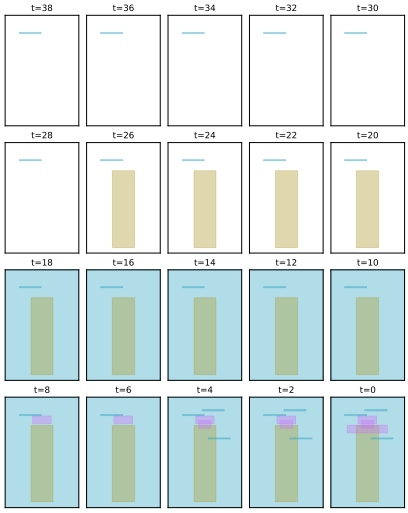
(d) Example2: LayoutDM + Corrector

Fig. 22: Comparison of unconditional generation process for Rico. Left: the results of LayoutDM. Right: the results of LayoutDM in conjunction with Layout-Corrector. The timestep is denoted at the top of each layout visualization, and the timesteps when the corrector is applied are highlighted by **bold** in Fig. 22b and Fig. 22d.

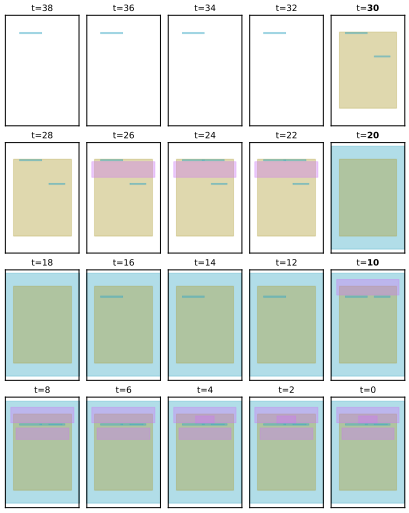
(a) Example1: LayoutDM



(b) Example1: LayoutDM + Corrector



(c) Example2: LayoutDM



(d) Example2: LayoutDM + Corrector

Fig. 23: Comparison of unconditional generation process for Crello. Left: the results of LayoutDM. Right: the results of LayoutDM in conjunction with Layout-Corrector. The timestep is denoted at the top of each layout visualization, and the timesteps when the corrector is applied are highlighted by **bold** in Fig. 23b and Fig. 23d.

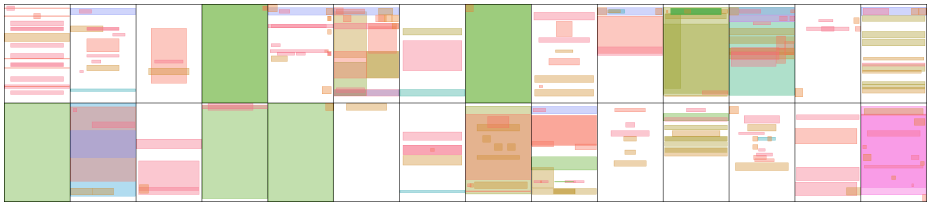
(a) Example1: LayoutDM

(b) Example1: LayoutDM + Corrector

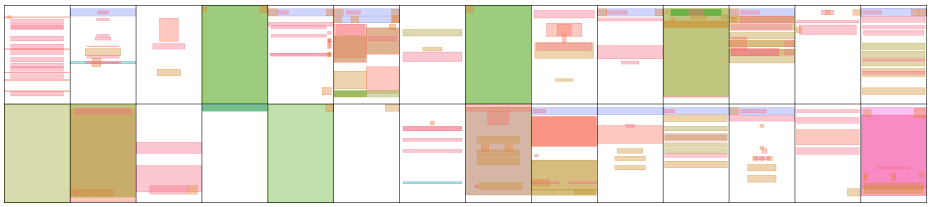
(c) Example2: LayoutDM

(d) Example2: LayoutDM + Corrector

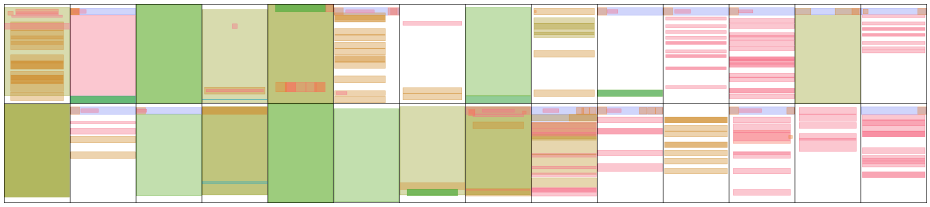
Fig. 24: Comparison of unconditional generation process for PubLayNet. Left: the results of LayoutDM. Right: the results of LayoutDM in conjunction with Layout-Corrector. The timestep is denoted at the top of each layout visualization, and the timesteps when the corrector is applied are highlighted by **bold** in Fig. 24b and Fig. 24d.



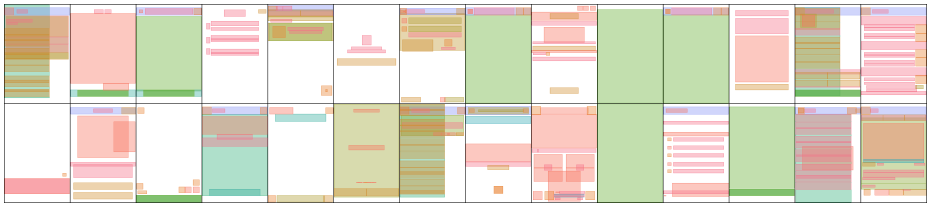
(a) LayoutDM (FID = 6.38, Precision = 0.750)



(b) LayoutDM + Corrector $t = \{10, 20, 30\}$ (FID = 4.79, Precision = 0.811)

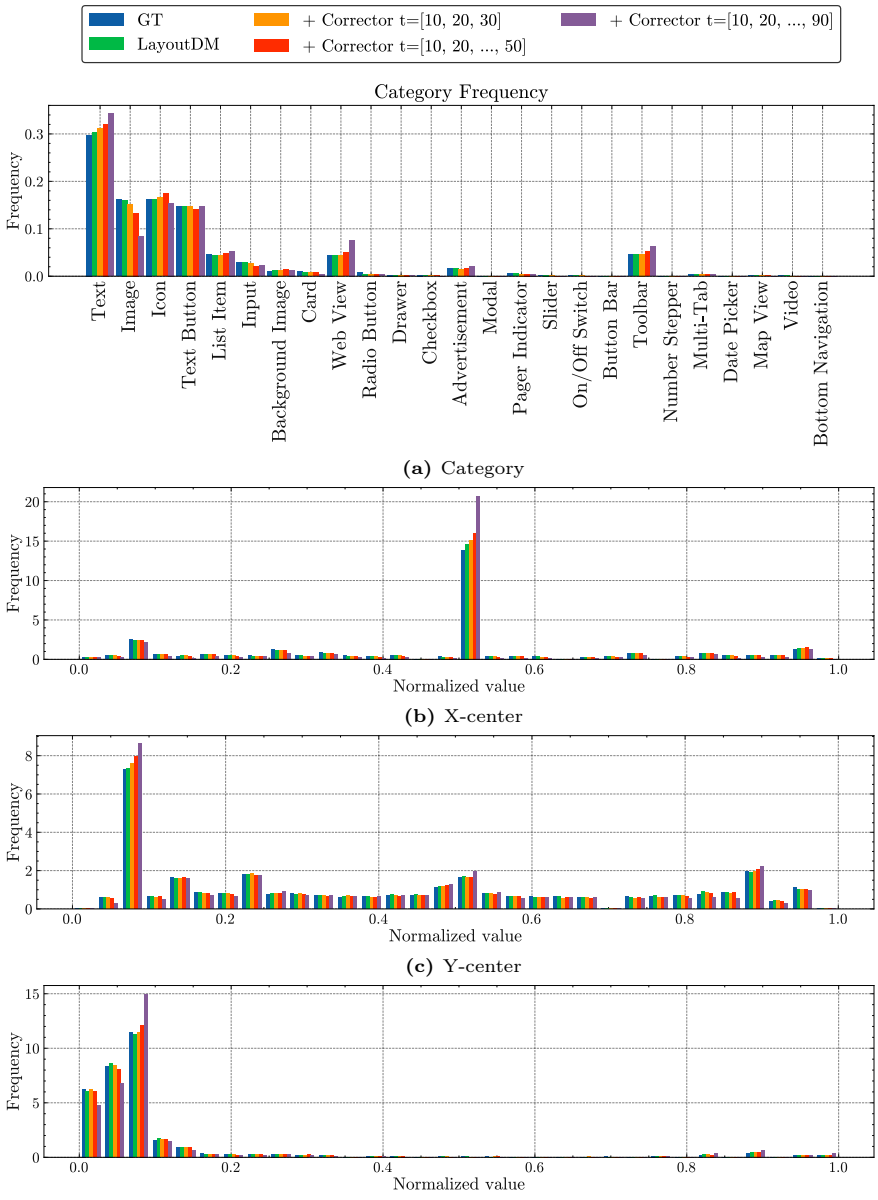


(c) LayoutDM + Corrector $t = \{10, 20, 30, \ldots, 90\}$ (FID = 19.90, Precision = 0.914)



(d) Real data

Fig. 25: Visualization of unconditional generation on the Rico dataset. This figure displays outputs from LayoutDM and LayoutDM + Layout-Corrector under two distinct corrector scheduling scenarios. Fig. 25b illustrates the results of our default schedule ($t = \{10, 20, 30\}$), which produces high-quality and diverse layouts. In contrast, Fig. 25c shows that increasing the frequency of Layout-Corrector application leads to layouts with more elements centered along the horizontal axis, indicating reduced diversity.



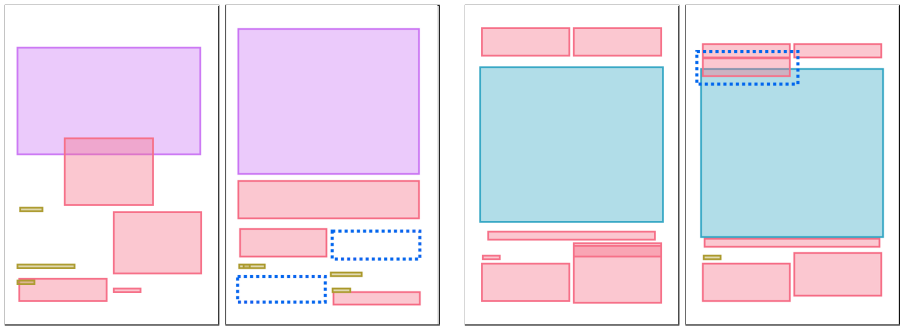
(a) Category

(b) X-center

(c) Y-center

(d) Height

Fig. 26: Histogram of the category (Fig. 26a), X-center (Fig. 26b), Y-center (Fig. 26c), and height (Fig. 26d) of elements on the Rico dataset on different corrector schedules.



(a) Left: LayoutDM. Right: LayoutDM + Corrector

(b) Left: LayoutDM. Right: LayoutDM + Corrector

Fig. 27: Typical failure cases on the unconditional task on PubLayNet dataset. We show the outputs from LayoutDM with and without Layout-Corrector. In Fig. 27a, although Layout-Corrector resolves overlapping elements found in the LayoutDM output, it leads to unnatural blank spaces in the LayoutDM + Corrector output. In Fig. 27b, while Layout-Corrector rectifies an overlap in the bottom-right of the LayoutDM output, a new overlap appears in the top-left in the LayoutDM + Corrector output.