Retrieval-Augmented Layout Transformer for Content-Aware Layout Generation (Supplementary Material)

Daichi Horita[1] Naoto Inoue[2] Kotaro Kikuchi[2] Kota Yamaguchi[2] Kiyoharu Aizawa[1]
[1]The University of Tokyo [2]CyberAgent
{horita,aizawa}@hal.t.u-tokyo.ac.jp
{inoue_naoto,kikuchi_kotaro_xa,yamaguchi_kota}@cyberagent.co.jp

Table of contents:

A. Implementation Details

Architecture details. Our RALF consists of four modules: the image encoder, retrieval augmentation, layout decoder, and optional constraint encoder. Table A provides the number of parameters of these modules.

Image encoder consists of ResNet-50-FPN [9] and the Transformer encoder. We obtain the saliency map by the off-the-shelf saliency detection method [13, 14] and combine them via a pixel-wise maximum operation following the approach in DS-GAN [3].

Retrieval augmentation. We implement the retrieval part using faiss [6]. The layout encoder for retrieved layouts consists of the Transformer encoder and a feed-forward network, which adapts the feature map size of retrieved layouts to the size of the layout decoder. Before training, we pre-train the layout encoder for each dataset and extract features over each training dataset to construct the retrieval database. We note that the parameters of the layout encoder (1.59M) are excluded from the total parameters of RALF since they are set with the retrieval database.

To calculate a cross-attended feature, the image feature acts as the query, and the retrieved layout feature serves as both the key and value. We use multi-head attention [16] as our cross-attention layer. The effectiveness of the cross-attended feature is demonstrated in the comparison of scenarios (B) and (C) in Table 6 in the main paper.

Layout decoder. We employ the Transformer decoder. The configurations of the Transformer layers are as follows: 6 layers, 8 attention heads, 256 embedding dimensions, 1,024 hidden dimensions, and 0.1 dropout rate. The size of bins for the layout tokenizer is set to 128. In the inference phase, for the relationship task, we use a decoding space restriction mechanism [5], which aims to prune the predicted tokens that violate a user-specified constraint.

Module	#Params
Image encoder (ResNet50)	25.02 M
Image encoder (Trans Enc)	4.74 M
Constraint encoder	4.88 M
Retrieval augmentation	1.59 M
Layout decoder	6.59 M
Total	42.82 M

Table A. The number of parameters of each module.

Training details. We implement RALF in PyTorch [12] and train for 50 and 70 epochs with AdamW optimizer [10] for the PKU and CGL datasets, respectively. The training time is about 4 hours and 20 minutes for the PKU dataset and 18 hours for the CGL dataset on a single A100 GPU. We divide the learning rate by 10 after 70% of the total epoch elapsed. We set the batch size, learning rate, weight decay, and gradient norm to 32, 10^{-4}, 10^{-4}, and 10^{-1}, respectively.

Testing details. We generate layouts on three independent trials and report the average of the metrics. We use top-k sampling for all the models that rely on sampling in logit space. We set k and $temperature$ to 5 and 1.0, respectively.

Other baselines. For the training of baseline methods, we follow the original training setting referring to their papers as much as possible. There are some exceptions for a fair comparison. For example, the number of embedding dimensions and hidden dimensions in Transformer is adjusted to roughly match the number of parameters for each model.

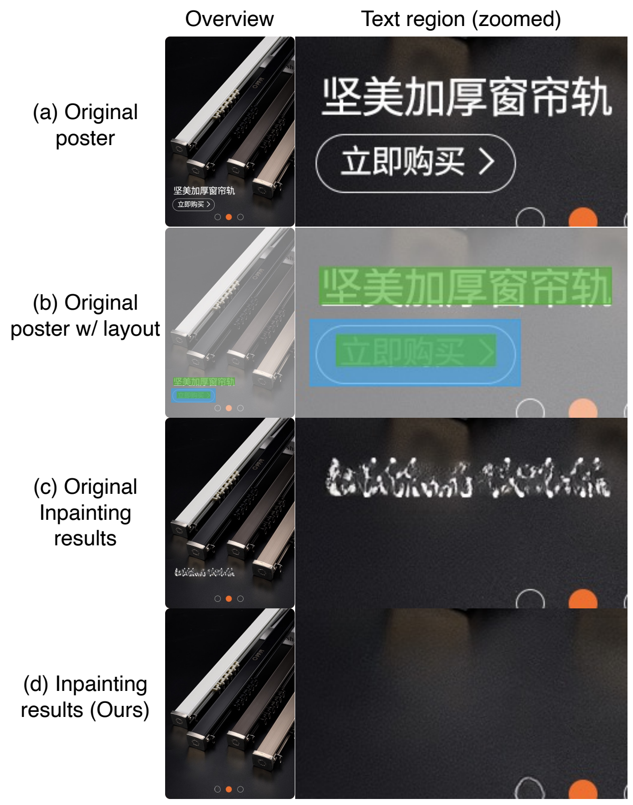
Overview
Text region (zoomed)
(a) Original poster
(b) Original poster w/ layout
(c) Original Inpainting results
(d) Inpainting results (Ours)
坚美加厚窗帘轨
立即购买 >

Figure A. Comparison of inpainting for the dataset preprocessing.

We use ResNet-50-FPN as the image encoder for all of our baseline methods.

B. Dataset Preprocessing

We demonstrate the importance of adequately preprocessing annotated poster images in Fig. A. Layout annotations in existing datasets sometimes exhibit inaccuracies for some underlying factors, including the semi-automatic collection process using object detection models [3] as shown in (a) and (b). The inaccuracy severely harms the image inpainting quality when we fully depend on the annotations, as shown in (c). To cope with the inaccuracy, we slightly dilate the target region for inpainting and get better results with fewer artifacts, as shown in (d). We show more examples in Fig. B. We observe that about 20% of the original inpainted images in PKU contain significant artifacts.

We plot the number of layout elements for each poster in Fig. C. Although we filter out posters with more than 11 layout elements, it only accounts for about 2% of the original dataset.

C. Additional Results

Spatial distribution shift. Figure D shows the visual comparison of canvases and saliency maps between the test and unannotated test split of CGL. We see that the proportion of space occupied by the saliency map is different according to the different values of Mean. As a result, this difference causes the performance degradation in CGL.

Inference speed. Table B compares inference speeds. Compared to Autoreg Baseline, the total inference speed of RALF increases by about 35%. While the latency is produced, our RALF can enhance the quality of generation.

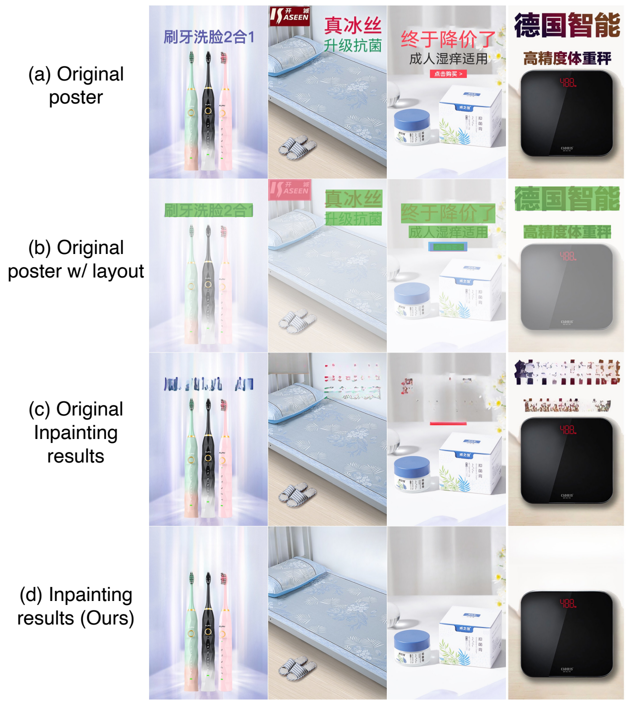
(a) Original poster
(b) Original poster w/ layout
(c) Original Inpainting results
(d) Inpainting results (Ours)
刷牙洗脸2合1
真冰丝
升级抗菌
终于降价了
成人湿痒适用
德国智能
高精度体重秤

Figure B. Comparison of inpainting for the dataset preprocessing.

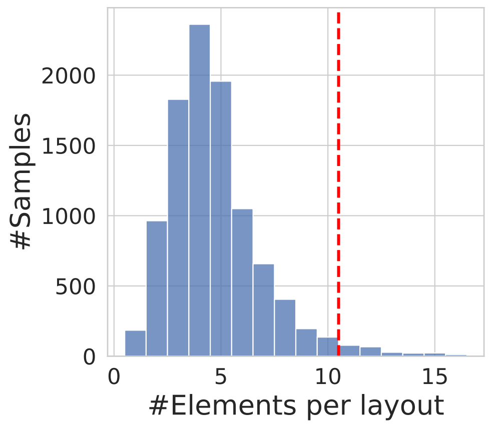

Figure C. Number of elements per layout in the original PKU dataset. A red dashed line indicates the maximum number of elements we use.

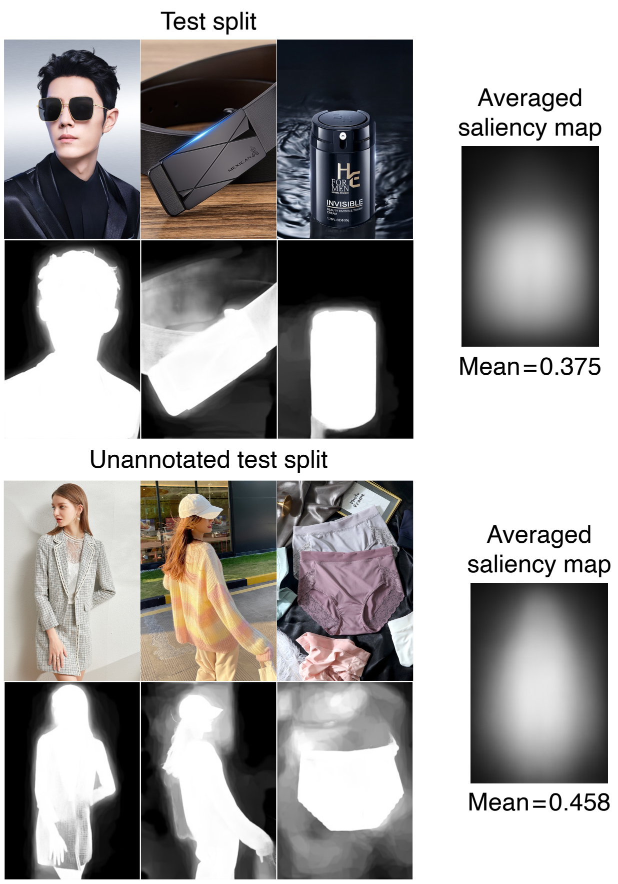

Figure D. Visual comparison of canvases and saliency maps between the test and unannotated test split of the CGL dataset. Canvases are randomly selected from each split. The averaged saliency map is produced by computing the spatial average of all saliency maps of each split. Mean represents the spatial average of all saliency maps of each split.

Impact of a saliency map. We compare scenarios with and without a saliency map in Table C since manually creating an inaccurate saliency map is unreasonable. The result shows that the presence of it has a negligible effect on performance. While we follow previous works [3, 17] to use a saliency map, we might be able to simplify our image encoder.

Comprehensive quantitative comparison. We additionally adopt five metrics.

Graphic metrics. Alignment (Align $\downarrow$) [7, 8] computes how well the elements are aligned with each other. For detailed calculation, please refer to [7, 8]. *Loose underlay effectiveness* ($\mathrm{Und}_L \uparrow$) [3] also calculates the proportion of the total area of valid underlay elements to the total of underlay and non-underlay elements. Note that we define this loose metric as $\mathrm{Und}_L \uparrow$ to distinguish it from the strict underlay effectiveness $\mathrm{Und}_S \uparrow$ introduced in the main manuscript.

	CGL-GAN	LayoutDM†	Autoreg Baseline	RALF			
				DreamSim	Retrieval	Network	Total
Time [s]	0.012	0.495	0.225	0.022	0.031	0.252	0.305

Table B. Inference time comparison on the PKU dataset. RALF consists of three components – feature extraction (DreamSim), layout retrieval (Retrieval), and layout generation (Network). The total inference time (Total) is the sum of these individual components.

Method	Saliency map	Occ $\downarrow$	Rea $\downarrow$	Und $\uparrow$	Ove $\downarrow$	FID $\downarrow$
Autoreg Baseline		0.132	0.0169	0.45	0.021	11.78
Autoreg Baseline	✓	0.134	0.0165	0.44	0.018	13.51
RALF		0.122	0.0129	0.90	0.007	3.97
RALF	✓	0.119	0.0129	0.92	0.008	3.45

Table C. Quantitative results without and with a saliency map.

Density (Den $\uparrow$) and *Coverage* (Cov $\uparrow$) [11] compute fidelity and diversity aspects of the generated layouts against ground-truth layouts. Please refer to [11] for more details.

Content metrics. Salient consistency ($\mathrm{R_{shm}} \downarrow$) [17] computes the Euclidean distance between the output logits of the canvases with or without layout regions masked using a pre-trained VGG16 [15].

Tables D and E present the quantitative result on the annotated test split without user constraints on the PKU and CGL datasets, respectively. RALF notably improves Density and Coverage metrics, indicating that RALF can generate better layouts in terms of both fidelity and diversity. RALF does not achieve the best score regarding $\mathrm{R_{shm}}$ and Alignment. However, these metrics may not be very reliable since the best scores for these metrics largely deviate from the scores for Real-Data, unlike other metrics.

Retrieval augmentation for baseline method. Table F shows the results of retrieval augmentation for CGL-GAN and LayoutDM†. Even for constrained generation tasks, retrieval augmentation achieves a better quality of generation for other generators on almost all metrics.

Impact on changing #Dim in layout decoder. Table G provides the results of RALF and Autoreg Baseline while changing the number of parameters in the layout decoder. We modify the number of features (#Dim) and hidden dim to four times the number of #Dim. RALF's performance peaks when #Dim is 256. Autoreg Baseline's performance improves as #Dim increases, but the model with #Dim=768 still clearly underperforms RALF with #Dim=256. Thus, retrieval augmentation enables us to use a relatively compact network. This result aligns with the trend observed in other domains, such as image generation [1]. We conjecture that slight performance degradation as we increase #Dim over 256 in RALF is caused by overfitting as we watch loss curves for training and validation.

Visual comparison on constrained generation. Figures E and F provide the qualitative comparisons of constrained generation for the PKU and CGL datasets, respectively. The results demonstrate that our RALF successfully generates well-fitted, non-overlapping, and rational layouts even in constrained generation tasks.

References

[1] Andreas Blattmann, Robin Rombach, Kaan Oktay, and Björn Ommer. Retrieval-Augmented Diffusion Models. In *NeurIPS*, 2022. 3

[2] Yunning Cao, Ye Ma, Min Zhou, Chuanbin Liu, Hongtao Xie, Tiezheng Ge, and Yuning Jiang. Geometry Aligned Variational Transformer for Image-conditioned Layout Generation. In *ACM MM*, 2022. 5

[3] Hsiao Yuan Hsu, Xiangteng He, Yuxin Peng, Hao Kong, and Qing Zhang. PosterLayout: A New Benchmark and Approach for Content-Aware Visual-Textual Presentation Layout. In *CVPR*, 2023. 1, 2, 3, 5

[4] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. In *CVPR*, 2023. 5

[5] Z. Jiang, J. Guo, S. Sun, H. Deng, Z. Wu, V. Mijovic, Z. Yang, J. Lou, and D. Zhang. LayoutFormer++: Conditional Graphic Layout Generation via Constraint Serialization and Decoding Space Restriction. In *CVPR*, 2023. 1

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 1

[7] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained Graphic Layout Generation via Latent Optimization. In *ACM MM*, 2021. 3

[8] 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), 2021. 3

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017. 1

[10] Ilya Loshchilov and Frank Hutter. Fixing Weight Decay Regularization in Adam. In *ICLR*, 2019. 1

[11] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable Fidelity and Diversity Metrics for Generative Models. In *ICML*, 2020. 3

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. 1

[13] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. BASNet: Boundary-Aware Salient Object Detection. In *CVPR*, 2019. 1

[14] Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. Highly Accurate Dichotomous Image Segmentation. In *ECCV*, 2022. 1

[15] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015. 3

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017. 1

[17] Min Zhou, Chenchen Xu, Ye Ma, Tiezheng Ge, Yuning Jiang, and Weiwei Xu. Composition-aware Graphic Layout GAN for Visual-textual Presentation Designs. In *IJCAI*, 2022. 3, 5

Method	PKU									
	Content			Graphic						
	Occ ↓	Rea ↓	$\mathrm{R_{shm}}$ ↓	Align ↓	$\mathrm{Und_L}$ ↑	$\mathrm{Und_S}$ ↑	Ove ↓	Den↑	Cov↑	FID↓
Real-Data	0.112	0.0102	13.94	0.00379	0.99	0.99	0.0009	0.95	0.95	1.58
Top1-Retrieval	0.212	0.0218	16.33	0.00371	0.99	0.99	0.002	1.07	0.97	1.43
CGL-GAN [17]	0.138	0.0164	14.32	0.00311	0.81	0.41	0.074	0.70	0.68	34.51
DS-GAN [3]	0.142	0.0169	14.95	0.00347	0.89	0.63	0.027	1.10	0.82	11.80
ICVT [2]	0.146	0.0185	13.92	0.00228	0.63	0.49	0.318	0.35	0.40	39.13
LayoutDM† [4]	0.150	0.0192	**13.06**	0.00298	0.64	0.41	0.190	0.74	0.59	27.09
Autoreg Baseline	0.134	0.0164	14.43	**0.00192**	0.79	0.43	0.019	1.13	0.79	13.59
RALF (Ours)	**0.119**	**0.0129**	14.11	0.00267	**0.98**	**0.92**	**0.008**	**1.25**	**0.97**	**3.45**

Table D. Unconstrained generation results on the PKU test split.

Method	CGL									
	Content			Graphic						
	Occ ↓	Rea ↓	$\mathrm{R_{shm}}$ ↓	Align ↓	$\mathrm{Und_L}$ ↑	$\mathrm{Und_S}$ ↑	Ove ↓	Den↑	Cov↑	FID↓
Real-Data	0.125	0.0170	14.33	0.00240	0.99	0.98	0.0002	0.93	1.00	0.79
Top1-Retrieval	0.214	0.0266	16.02	0.00254	0.99	0.99	0.0005	1.01	0.90	0.93
CGL-GAN [17]	0.157	0.0237	14.12	0.00320	0.67	0.29	0.161	0.31	0.28	66.75
DS-GAN [3]	0.141	0.0229	14.85	0.00257	0.71	0.45	0.057	0.64	0.40	41.57
ICVT [2]	**0.124**	0.0205	**13.40**	0.00319	0.55	0.42	0.310	0.16	0.22	65.34
LayoutDM† [4]	0.127	0.0192	14.15	0.00242	0.92	0.82	0.020	0.87	0.93	2.36
Autoreg Baseline	0.125	0.0190	14.22	**0.00234**	0.97	0.92	0.011	1.05	0.91	2.89
RALF (Ours)	0.125	**0.0180**	14.26	0.00236	**0.99**	**0.98**	**0.004**	**1.09**	**0.96**	**1.32**

Table E. Unconstrained generation results on the CGL test split.

Task	Method	Retrieval	PKU					CGL				
			Content		Graphic			Content		Graphic		
			Occ ↓	Rea ↓	Und ↑	Ove ↓	FID↓	Occ ↓	Rea ↓	Und ↑	Ove ↓	FID↓
Unconstraint	Real Data		0.112	0.0102	0.99	0.0009	1.58	0.125	0.0170	0.98	0.0002	0.79
	Top-1 Retrieval		0.212	0.0218	0.99	0.002	1.43	0.214	0.0266	0.99	0.0005	0.93
	CGL-GAN		0.138	0.0164	0.41	0.074	34.51	0.157	0.0237	0.29	0.161	66.75
	CGL-GAN	✓	0.144	0.0164	0.63	0.039	13.28	0.172	0.0245	0.42	0.157	60.67
	LayoutDM†		0.150	0.0192	0.41	0.190	27.09	0.127	0.0192	0.82	0.020	2.36
	LayoutDM†	✓	0.123	0.0144	0.51	0.091	10.03	0.126	0.0187	0.85	0.019	1.97
$\mathrm{C} \rightarrow \mathrm{S} + \mathrm{P}$	CGL-GAN		0.132	0.0158	0.48	0.038	11.47	0.140	0.0213	0.65	0.047	23.93
	CGL-GAN	✓	0.140	0.0153	0.66	0.030	10.23	0.138	0.0202	0.82	0.021	10.01
	LayoutDM†		0.152	0.0201	0.46	0.172	20.50	0.127	0.0192	0.79	0.026	3.39
	LayoutDM†	✓	0.121	0.0141	0.55	0.088	9.02	0.127	0.0189	0.81	0.026	3.36
$\mathrm{C} + \mathrm{S} \rightarrow \mathrm{P}$	CGL-GAN		0.129	0.0155	0.48	0.043	9.11	0.129	0.0202	0.75	0.027	6.96
	CGL-GAN	✓	0.146	0.0178	0.57	0.036	7.74	0.135	0.0207	0.78	0.020	6.01
	LayoutDM†		0.143	0.0185	0.45	0.122	24.90	0.127	0.0190	0.82	0.021	2.18
	LayoutDM†	✓	0.123	0.0144	0.59	0.071	10.68	0.127	0.0188	0.83	0.020	1.77
Completion	CGL-GAN		0.146	0.0175	0.42	0.076	27.18	0.174	0.0231	0.21	0.182	78.44
	CGL-GAN	✓	0.146	0.0169	0.71	0.039	12.46	0.155	0.0230	0.46	0.102	48.82
	LayoutDM†		0.135	0.0175	0.35	0.134	21.70	0.127	0.0192	0.76	0.020	3.19
	LayoutDM†	✓	0.120	0.0143	0.45	0.071	12.96	0.126	0.0189	0.79	0.018	2.55
Refinement	CGL-GAN		0.122	0.0141	0.39	0.090	6.40	0.124	0.0182	0.86	0.024	1.20
	CGL-GAN	✓	0.129	0.0157	0.37	0.072	4.91	0.133	0.0194	0.85	0.013	1.56
	LayoutDM†		0.115	0.0121	0.57	0.008	2.86	0.127	0.0188	0.75	0.018	1.98
	LayoutDM†	✓	0.115	0.0121	0.57	0.007	2.91	0.126	0.0186	0.76	0.019	1.79

Table F. Retrieval augmentation for CGL-GAN and LayoutDM† on the PKU and CGL test split for unconstrained and constrained generation.

Method	#Dim	#ParamsDec	PKU					CGL				
			Content		Graphic			Content		Graphic		
			Occ ↓	Rea ↓	Und ↑	Ove ↓	FID↓	Occ ↓	Rea ↓	Und ↑	Ove ↓	FID↓
Autoreg Baseline	128	2.55M	0.146	0.0184	0.41	0.030	18.86	0.127	0.0196	0.86	0.013	3.60
RALF			0.123	0.0141	0.71	0.007	4.14	0.125	0.0180	0.97	0.005	1.27
Autoreg Baseline◇	256	6.59M	0.134	0.0165	0.44	0.018	13.51	0.125	0.0190	0.92	0.011	2.90
RALF◇			0.119	0.0129	0.92	0.008	3.45	0.125	0.0180	0.98	0.004	1.31
Autoreg Baseline	512	19.46M	0.128	0.0150	0.57	0.011	10.85	0.122	0.0184	0.95	0.009	2.74
RALF			0.122	0.0131	0.94	0.010	3.61	0.128	0.0182	0.97	0.004	1.72
Autoreg Baseline	768	38.82M	0.122	0.0150	0.70	0.012	8.46	0.124	0.0183	0.95	0.008	2.26
RALF			0.126	0.0131	0.93	0.008	3.19	0.131	0.0187	0.97	0.004	1.72

Table G. Qualitative result of varying network parameters on unconstrained generation metrics on the PKU and CGL test split. We modify the number of features (#Dim) in the input of cross-attention layers and the sequence to the decoder layer. #ParamsDec indicates the number of parameters of the layout decoder. ◇ represents the setting of our experiments in the main manuscript.

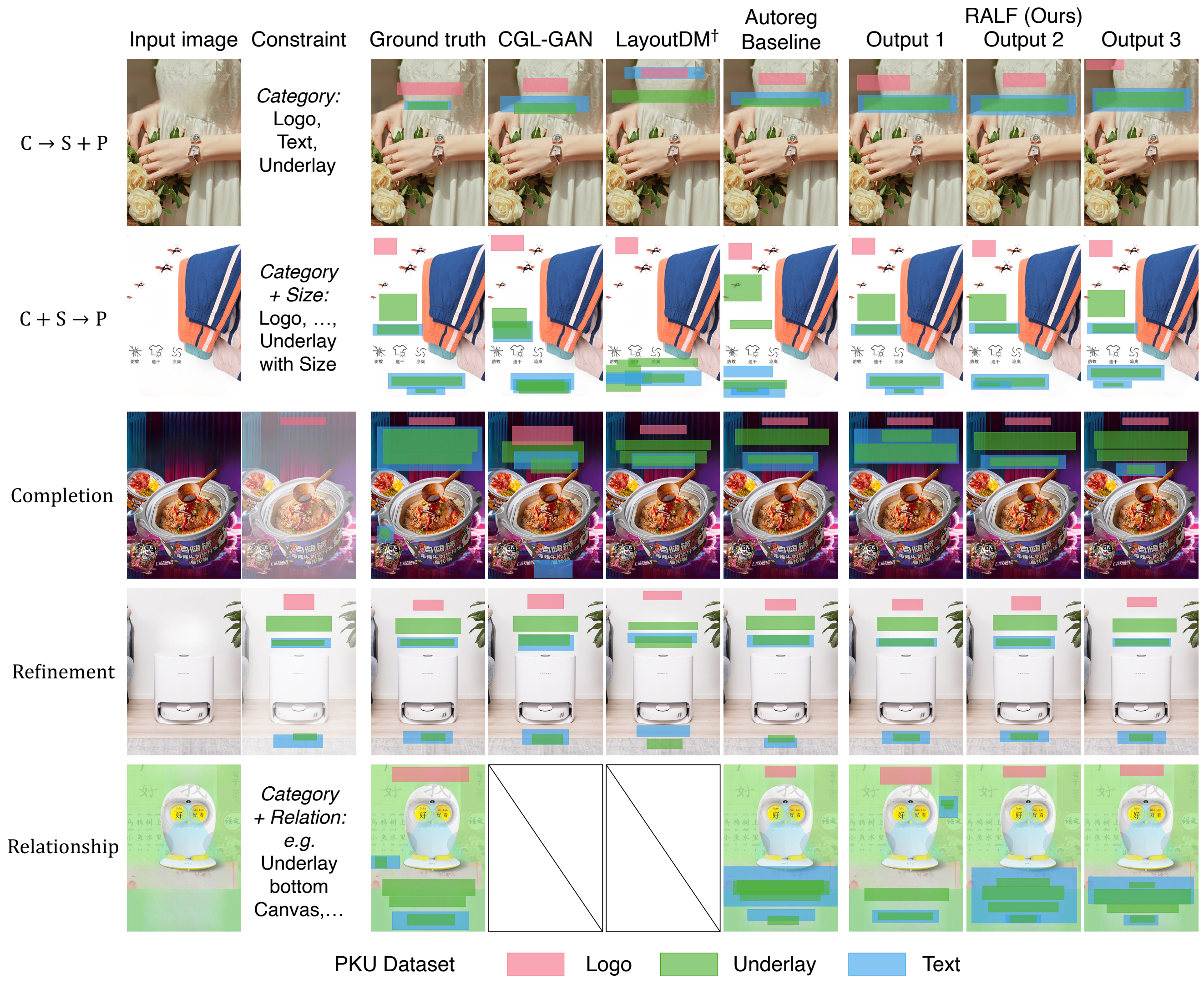

Figure E. Visual comparison of constrained generation with baselines on the PKU annotated test split.

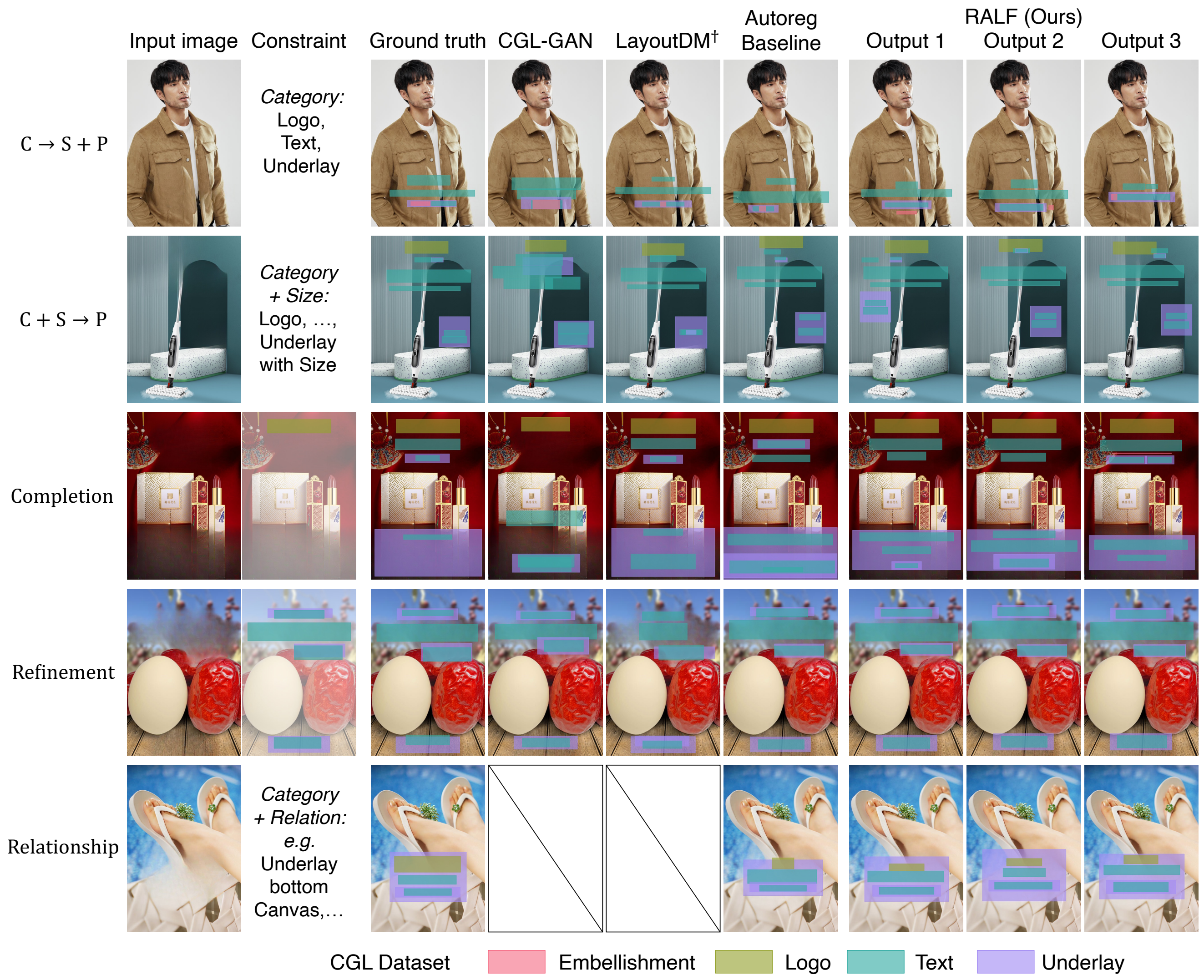

Figure F. Visual comparison of constrained generation with baselines on the CGL annotated test split.