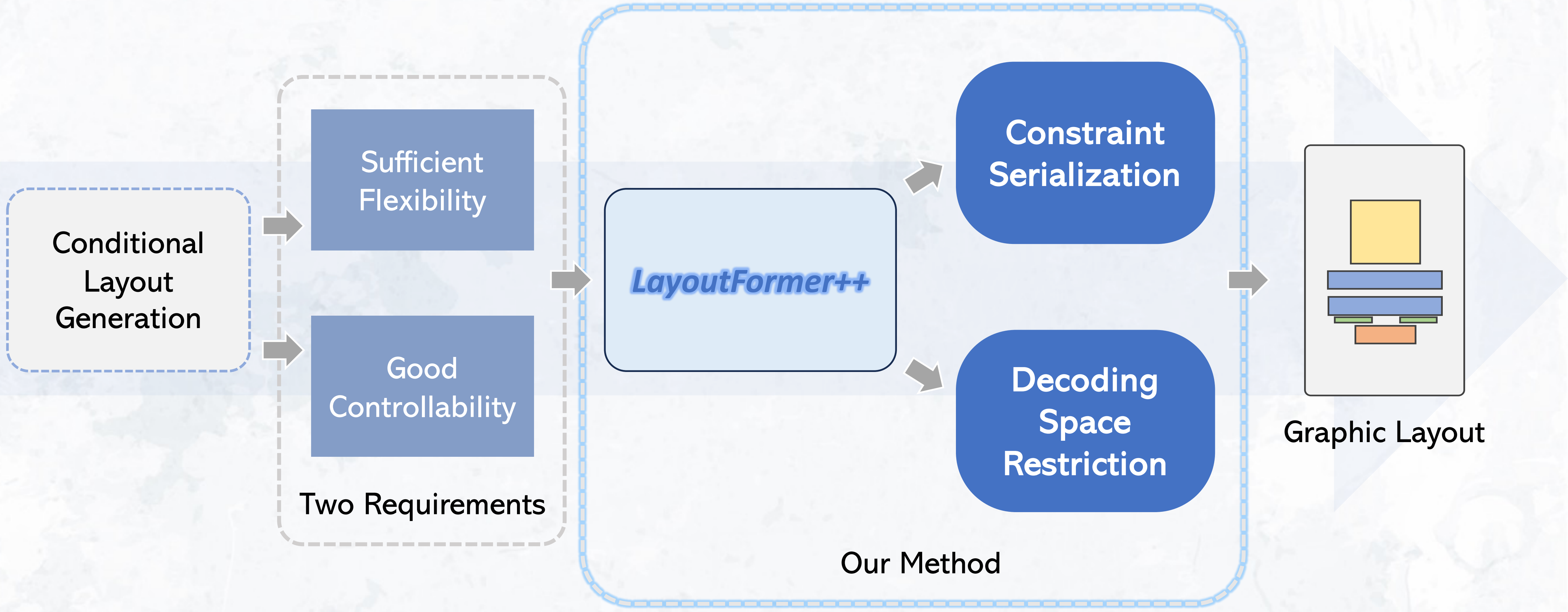
LayoutFormer++:

Conditional Graphic Layout Generation via Constraint Serialization and Decoding Space Restriction

Zhaoyun Jiang, Jiaqi Guo, Shizhao Sun, Huayu Deng, Zhongkai Wu, Vuksan Mijovic, Zijiang James Yang, Jian-Guang Lou, Dongmei Zhang

THU-AM-184

Highlights



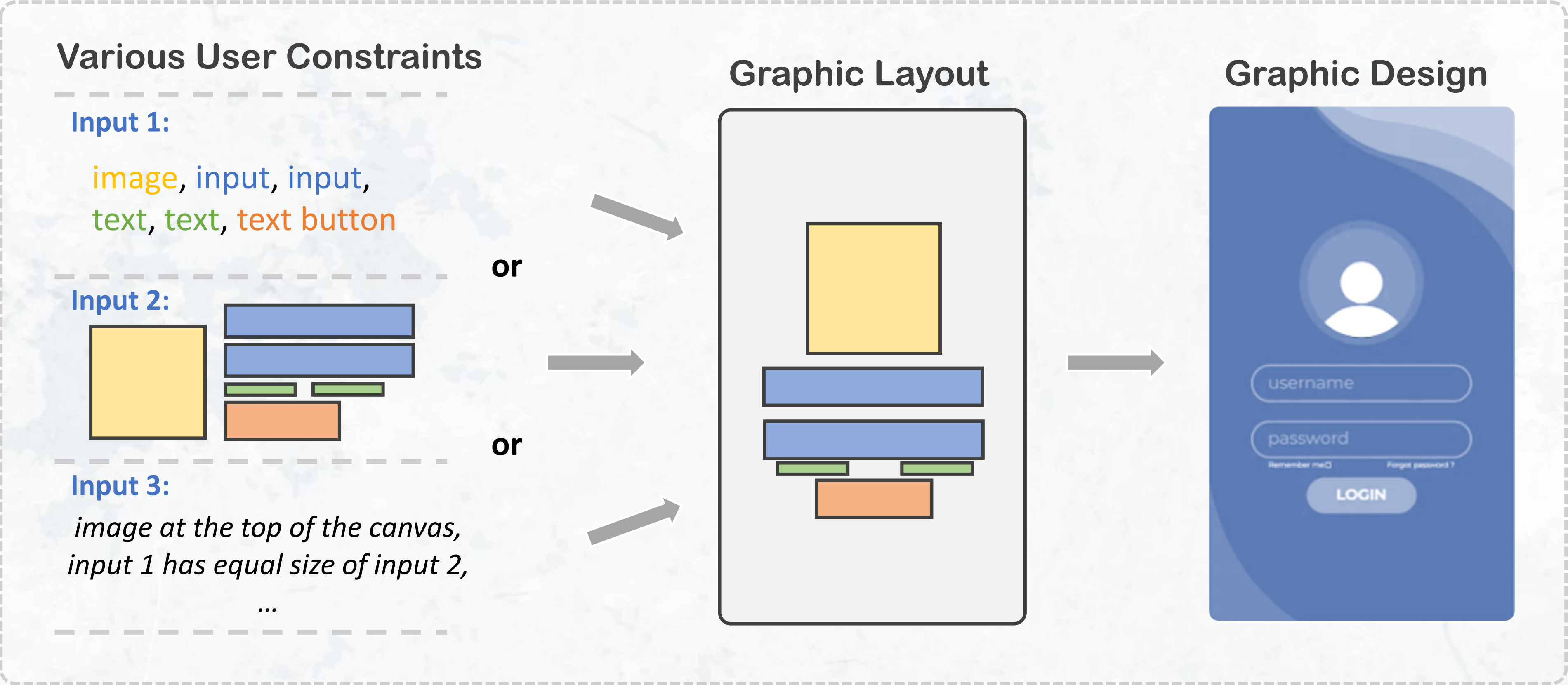
Conditional Layout Generation
Sufficient Flexibility
Good Controllability
Two Requirements
LayoutFormer++
Constraint Serialization
Decoding Space Restriction
Our Method
Graphic Layout

01

What is the conditional graphic layout generation?

Conditional Graphic Layout Generation

Take various user constraints as input and generate layouts as output:



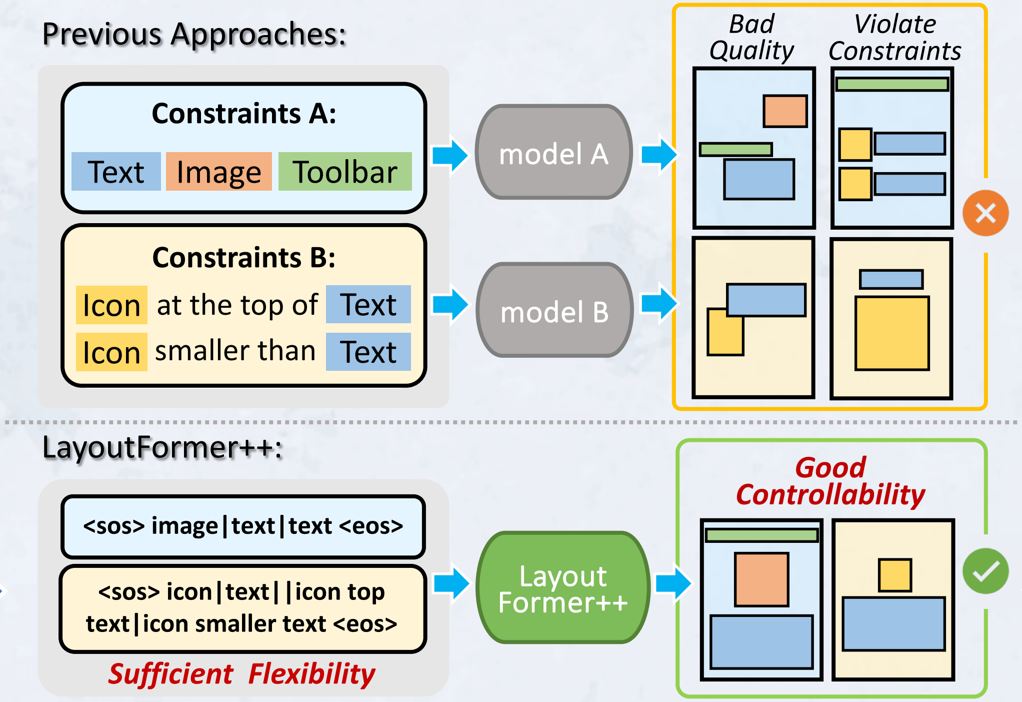
Two Critical Requirements

But existing work…

simply focus on tackling a single task without considering whether they can be applied to other tasks.

Sufficient Flexibility

The model should be able to handle diverse user constraints.



But existing work…

have no satisfactory methods to ensure good controllability.

Good Controllability

The model should generate layouts conforming to user constraints as many as possible without harming quality.

02

How do we achieve
these two requirements?

Model Overview

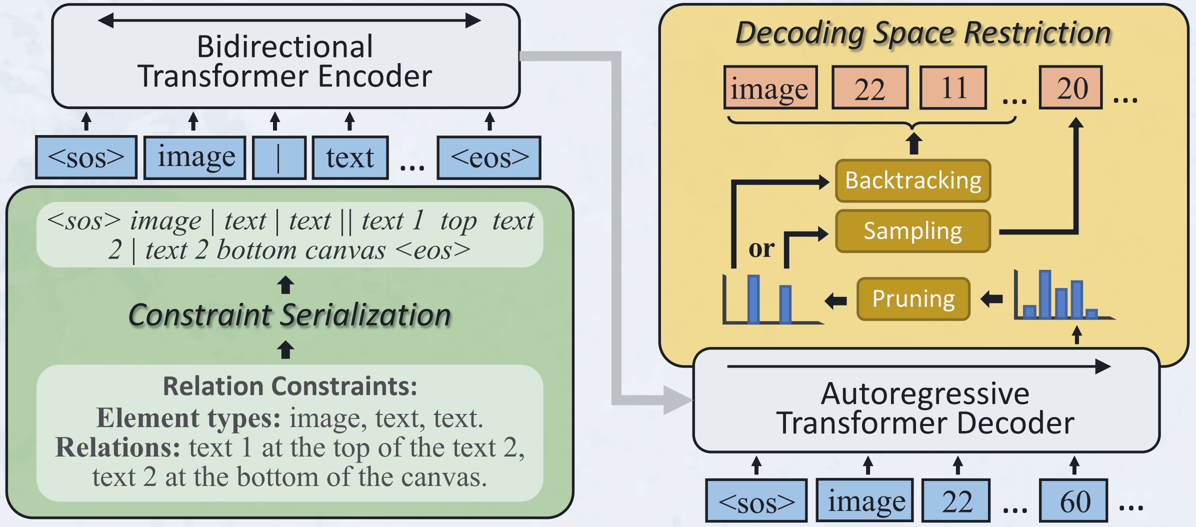
We propose a unified model called *LayoutFormer++.*

Decoding Space Restriction

To ensure the constraint satisfaction with high generation quality.

Constraint Serialization

To support the different scenarios of conditional layout generation



Constraint Serialization Scheme - I

Serializing Layout

- Each element can be described by 5 tokens:
  the type c, left and top coordinate x and y, width w and height h.

- Following the state-of-the-art approaches, we represent a layout by concatenating all the elements' tokens in a sequence:

$$L = \{\langle sos\rangle c_1 x_1 y_1 w_1 h_1 \ldots c_N x_N y_N w_N h_N \, \langle eos\rangle\}$$

Constraint Serialization Scheme - II

Serializing Constraints

There are two critical questions in serializing constraints:

1. How to represent each constraint in a sequence format?
2. How to combine different constraints into a complete sequence?

Constraint Serialization Scheme - III

Serializing Constraints

- *Constraint Representation*
  - take the constraint "put an image on top of a button" as an example.
  - build the vocabulary for elements and relationships, such as $\{\boldsymbol{image_1, \ldots, image_{K_1}, \ldots, button_1, \ldots, button_{Kt}}\}$ and $\{\boldsymbol{top, \ldots, small}\}$.
  - then concatenate the tokens of element and relationships into a sequence:

$$\boldsymbol{image1\ \ top\ \ button1}$$

- *Constraint Combination*
  - Concatenate the token sequences of different constraints in a fix order.

Constraint Serialization Scheme - IV

Serialization examples

<table>
<tr><th colspan="2">Input Sequence</th></tr>
<tr><td>*Gen-T*: <sos> image | text | text <eos></td><td>*Gen-TS*: <sos> image 36 36 | text 60 20 | text 60 20 <eos></td></tr>
<tr><td>*Gen-R*: <sos> image | text | text || text 1 top text 2 | text 2 bottom canvas <eos></td><td>*Refinement*: <sos> image 20 13 35 34 | text 11 59 61 21 | text 9 87 63 19 <eos></td></tr>
<tr><td>*Completion*: <sos> image 20 13 35 34 <eos></td><td>*UGen*: <sos> <eos></td></tr>
<tr><th colspan="2">Output Sequence</th></tr>
<tr><td colspan="2"><sos> image 22 11 36 36 | text 10 58 60 20 | text 10 89 60 20 <eos></td></tr>
</table>

Decoding Space Restriction Strategy - I

During inference, we introduce

- **Constraint Pruning Module**
- **Probability Pruning Module**
- **Backtracking Mechanism**

to ensure the constraint satisfaction.

Algorithm 1: Decoding Space Restriction

Input: Encoder hidden state M; User constraints S.
Output: Layout sequence O.

Initialize step index t, the back time for each step B and the predicted sequence O.
while $(O[-1] \neq \text{EOS})$ ***and*** $(t < maxLen)$ **do**
    $P \leftarrow \text{Decoder}(O, M)[t]$;
    $P' \leftarrow \text{ConstraintPruning}(P, S)$;
    $P' \leftarrow \text{ProbabilityPruning}(P', \theta)$;
    if $(P'$ is $\varnothing)$ ***and*** $(B[t] < maxBack)$ **then**
        $t' \leftarrow \text{Backtracking}(P, S, t)$;
        $B[t] \leftarrow B[t] + 1$;
        $O \leftarrow O[:t]$;
        $t \leftarrow t'$;
    else
        $o \leftarrow \text{Sampling}(P')$;
        $O \leftarrow O \cup o$;
        $t \leftarrow t + 1$;
    end
end

Decoding Space Restriction Strategy - II

‣ Constraint Pruning Module

- In each decoding step t, the decoder predicts the probabilities P of the possible values for current attribute.

- The constraint pruning module prunes the value in P which may violate the related constraints.

Algorithm 1: Decoding Space Restriction

Input: Encoder hidden state M; User constraints S.
Output: Layout sequence O.

Initialize step index t, the back time for each step B and the predicted sequence O.
while $(O[-1] \neq \text{EOS})$ ***and*** $(t < maxLen)$ **do**
  $P \leftarrow \text{Decoder}(O, M)[t]$;
  $P' \leftarrow \text{ConstraintPruning}(P, S)$;
  $P' \leftarrow \text{ProbabilityPruning}(P', \theta)$;
  if $(P'$ is $\varnothing)$ ***and*** $(B[t] < maxBack)$ **then**
    $t' \leftarrow \text{Backtracking}(P, S, t)$;
    $B[t] \leftarrow B[t] + 1$;
    $O \leftarrow O[:t]$;
    $t \leftarrow t'$;
  else
    $o \leftarrow \text{Sampling}(P')$;
    $O \leftarrow O \cup o$;
    $t \leftarrow t + 1$;
  end
end

Decoding Space Restriction Strategy - III

- **Probability Pruning Module**
  - It checks each value's probability in P'. The probabilities that are lower than the predetermined threshold θ will be pruned by setting as 0.

Algorithm 1: Decoding Space Restriction

Input: Encoder hidden state M; User constraints S.
Output: Layout sequence O.

Initialize step index t, the back time for each step B and the predicted sequence O.
while $(O[-1] \neq \text{EOS})$ ***and*** $(t < maxLen)$ **do**
    $P \leftarrow \text{Decoder}(O, M)[t]$;
    $P' \leftarrow \text{ConstraintPruning}(P, S)$;
    $P' \leftarrow \text{ProbabilityPruning}(P', \theta)$;
    if $(P'$ is $\varnothing)$ ***and*** $(B[t] < maxBack)$ **then**
        $t' \leftarrow \text{Backtracking}(P, S, t)$;
        $B[t] \leftarrow B[t] + 1$;
        $O \leftarrow O[:t]$;
        $t \leftarrow t'$;
    else
        $o \leftarrow \text{Sampling}(P')$;
        $O \leftarrow O \cup o$;
        $t \leftarrow t + 1$;
    end
end

Decoding Space Restriction Strategy - IV

- **Backtracking Mechanism**

- When the probabilities are all pruned, the backtracking mechanism checks why the P is pruned as empty and decide which step t' to backtrack the decoding process to.

- For example, the constraint $s = \{w_i \leq w_j\}$ restricts the feasible values of w_i by w_j. In this case, the step of w_j is chosen as the backtracking step.

Algorithm 1: Decoding Space Restriction

Input: Encoder hidden state M; User constraints S.
Output: Layout sequence O.

Initialize step index t, the back time for each step B and the predicted sequence O.
while $(O[-1] \neq \text{EOS})$ ***and*** $(t < maxLen)$ **do**
    $P \leftarrow \text{Decoder}(O, M)[t]$;
    $P' \leftarrow \text{ConstraintPruning}(P, S)$;
    $P' \leftarrow \text{ProbabilityPruning}(P', \theta)$;
    if $(P'$ is $\varnothing)$ ***and*** $(B[t] < maxBack)$ **then**
        $t' \leftarrow \text{Backtracking}(P, S, t)$;
        $B[t] \leftarrow B[t] + 1$;
        $O \leftarrow O[:t]$;
        $t \leftarrow t'$;
    else
        $o \leftarrow \text{Sampling}(P')$;
        $O \leftarrow O \cup o$;
        $t \leftarrow t + 1$;
    end
end

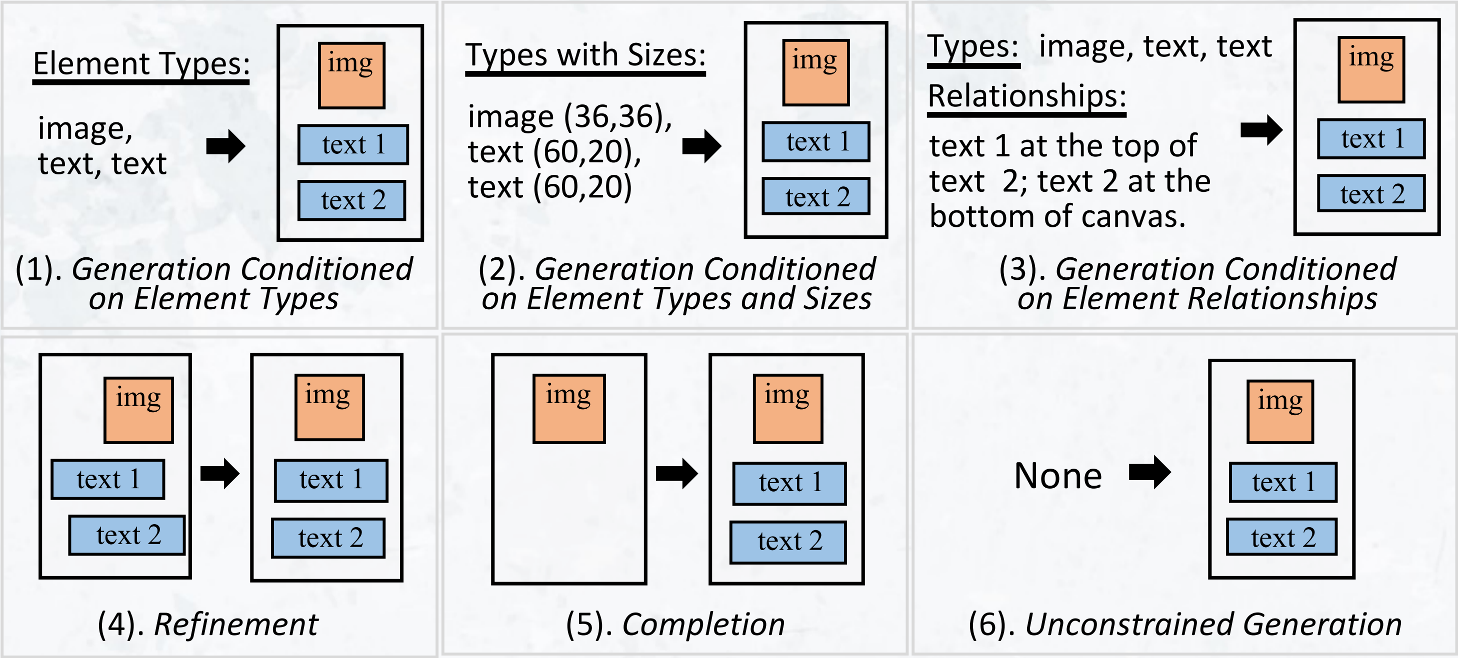
03

Experimental Results

Experiment Setups

Tasks and Baselines

We compare with state-of-the-art approaches on
6 typical graphic layout generation tasks:

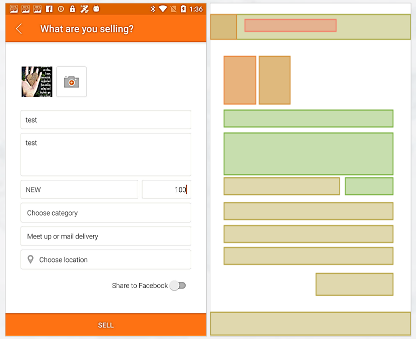


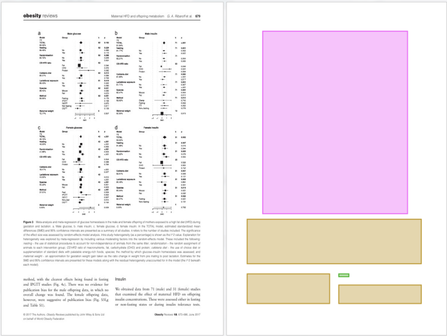
Experiment Setups

Datasets

RICO

PubLayNet





Evaluation Metrics

For Generation Quality:

mIoU, Alignment, Overlap, FID

For Constraint Satisfaction:

Constraint Violation Rate

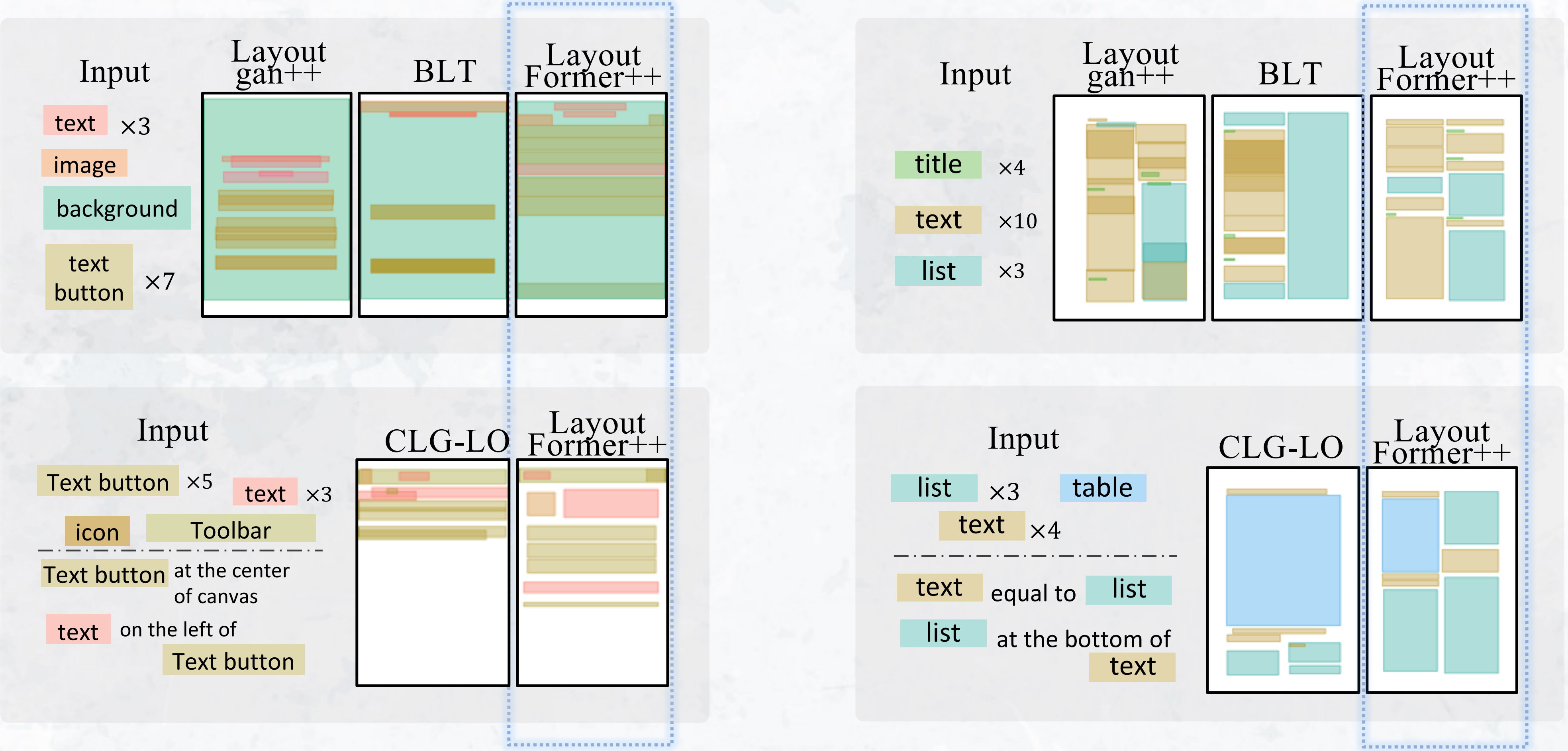
Evaluations on Sufficient Flexibility - I

Quantitative Comparison

Tasks	Methods	RICO				PubLayNet			
		mIoU ↑	FID ↓	Align. ↓	Overlap ↓	mIoU ↑	FID ↓	Align. ↓	Overlap ↓
Gen-T	NDN-none	0.35	13.76	0.56	0.55	0.31	35.67	0.35	0.17
	LayoutGAN++	0.298	5.954	0.261	0.620	0.297	14.875	0.124	0.148
	BLT	0.216	25.633	0.150	0.983	0.140	38.684	0.036	0.196
	LayoutFormer++	**0.432**	**1.096**	0.230	**0.530**	**0.348**	**8.411**	**0.020**	**0.008**
Gen-TS	BLT	0.604	0.951	0.181	0.660	0.428	7.914	0.021	0.419
	LayoutFormer++	**0.620**	**0.757**	0.202	**0.542**	**0.471**	**0.720**	0.024	**0.037**
Gen-R	NDN	0.36	-	0.56	-	0.31	-	0.36	-
	CLG-LO	0.286	8.898	0.311	0.615	0.277	19.738	0.123	0.200
	LayoutFormer++	**0.424**	**5.972**	0.332	**0.537**	**0.353**	**4.954**	**0.025**	**0.076**
Refinement	RUITE	0.811	0.107	0.133	0.483	0.781	0.061	0.029	0.020.
	LayoutFormer++	**0.816**	**0.032**	**0.123**	0.489	**0.785**	0.086	**0.024**	**0.006**
Completion	LayoutTransformer	0.363	6.679	0.194	0.478	0.077	14.769	0.019	0.0013
	LayoutFormer++	**0.732**	**4.574**	**0.077**	0.487	**0.471**	**10.251**	0.020	0.0022
UGen	LayoutTransformer	0.439	22.884	0.052	0.471	0.062	36.304	0.031	0.0009
	VTN	0.686	76.064	0.461	0.694	0.210	103.373	0.205	0.211
	Coarse2Fine	0.360	46.483	0.128	0.676	0.361	50.854	0.221	0.142
	LayoutFormer++	**0.742**	**19.688**	**0.047**	0.547	**0.417**	46.522	**0.029**	**0.0009**

Evaluations on Sufficient Flexibility - II

Qualitative Comparison



Evaluations on Good Controllability - I

We first compare *LayoutFormer++* with some approaches which pay attention to the constraint satisfaction.

Tasks	Method		RICO mIoU ↑	RICO FID ↓	RICO Align. ↓	RICO Overlap ↓	RICO Vio. % ↓	PubLayNet mIoU ↑	PubLayNet FID ↓	PubLayNet Align. ↓	PubLayNet Overlap ↓	PubLayNet Vio. % ↓
Gen-T	LayoutGAN++		0.298	5.954	0.261	0.620	0.	0.297	14.875	0.124	0.148	0.
	Layout Former++	Full	**0.432**	**1.096**	**0.230**	**0.530**	**0.**	**0.348**	**8.411**	**0.020**	**0.008**	**0.**
		- Back	0.431	1.320	0.272	0.550	0.	0.345	9.367	0.020	0.009	0.
		- Back&Prune	0.439	1.392	0.206	0.545	5.5	0.345	9.373	0.020	0.009	0.05
Gen-TS	BLT		0.604	0.951	0.181	0.660	0.	0.428	7.914	0.021	0.419	0.
	Layout Former++	Full	**0.620**	**0.757**	0.202	**0.542**	**0.**	**0.471**	**0.720**	0.024	**0.037**	**0.**
		- Back	0.613	0.782	0.206	0.543	0.	0.464	0.903	0.026	0.044	0.
		- Back&Prune	0.613	0.801	0.206	0.545	≈0.	0.464	0.903	0.026	0.044	≈0.
Gen-R	CLG-LO		0.286	8.898	0.311	0.615	3.66	0.277	19.738	0.123	0.200	6.66
	Layout Former++	Full	**0.424**	**5.972**	0.332	**0.537**	11.84	**0.353**	**4.954**	**0.025**	**0.076**	**3.9**
		- Back	0.419	8.604	0.284	0.544	12.75	0.352	5.152	0.023	0.075	5.70
		- Back&Prune	0.458	5.126	0.221	0.546	33.04	0.358	4.620	0.022	0.030	16.09

Evaluations on Good Controllability - II

Then we compare LayoutFormer++ framework with:

-Back: LayoutFormer++ without backtracking mechanism.

-Back&Prune: LayoutFormer++ without both pruning modules and the backtracking mechanism.

			RICO					PubLayNet				
Tasks	Method		mIoU ↑	FID ↓	Align. ↓	Overlap ↓	Vio. % ↓	mIoU ↑	FID ↓	Align. ↓	Overlap ↓	Vio. % ↓
	LayoutGAN++		0.298	5.954	0.261	0.620	0.	0.297	14.875	0.124	0.148	0.
Gen-T	Layout Former++	Full	**0.432**	**1.096**	**0.230**	**0.530**	**0.**	**0.348**	**8.411**	**0.020**	**0.008**	**0.**
		- Back	0.431	1.320	0.272	0.550	0.	0.345	9.367	0.020	0.009	0.
		- Back&Prune	0.439	1.392	0.206	0.545	5.5	0.345	9.373	0.020	0.009	0.05
	BLT		0.604	0.951	0.181	0.660	0.	0.428	7.914	0.021	0.419	0.
Gen-TS	Layout Former++	Full	**0.620**	**0.757**	0.202	**0.542**	**0.**	**0.471**	**0.720**	0.024	**0.037**	**0.**
		- Back	0.613	0.782	0.206	0.543	0.	0.464	0.903	0.026	0.044	0.
		- Back&Prune	0.613	0.801	0.206	0.545	≈0.	0.464	0.903	0.026	0.044	≈0.
	CLG-LO		0.286	8.898	0.311	0.615	3.66	0.277	19.738	0.123	0.200	6.66
Gen-R	Layout Former++	Full	**0.424**	**5.972**	0.332	**0.537**	11.84	**0.353**	**4.954**	**0.025**	**0.076**	**3.9**
		- Back	0.419	8.604	0.284	0.544	12.75	0.352	5.152	0.023	0.075	5.70
		- Back&Prune	0.458	5.126	0.221	0.546	33.04	0.358	4.620	0.022	0.030	16.09

THANKS