LayoutDiffusion: Improving Graphic Layout Generation by Discrete Diffusion Probabilistic Models

ICCV23 PARIS

Junyi Zhang[1], Jiaqi Guo[2], Shizhao Sun[2], Jian-Guang Lou[2], Dongmei Zhang[2]

1 Shanghai Jiao Tong University 2 Microsoft Research Asia

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Microsoft

Code & Model

Abstract & Key Takeaways

Creating graphic layouts is a fundamental step in graphic designs. Prior methods on layout generation either grapple with biased context using autoregressive approaches or struggle to synthesis plausible layout in a single pass. In this study, we introduce **LayoutDiffusion**, an iterative non-autoregressive approach for layout generation. <u>Here are the central takeaways:</u>

1) We aim to design a **semantically consistent mild corruption process** for layout data, but it is challenging due to its **heterogeneous nature** (both discrete and continuous data).
2) To tackle the challenge, we summarize three critical factors for achieving a mild forward process for the layout, i.e.**, legality, coordinate proximity and type disruption**.
3) Based on these factors, we develop **a block-wise transition matrix** coupled with a **piece-wise linear noise schedule**.
4) By our design of forward process, we enable certain **conditional layout generation** tasks **without additional training**.
5) **Comprehensive experiments and user studies** reveal that LayoutDiffusion considerably surpasses existing techniques across all tasks (**notably achieving an 8x better FID on RICO Un-Gen**), even without re-training for conditional tasks.

Background on Graphic Layout Generation

Graphic Layout

A layout is made up of a set of elements, where each element can be described by its type c, bounding box with left l, top t, right r, and bottom b coordinates.

(a). Graphic designs and layouts

By discretizing the coordinates, we can model layout as a sequence:

$$\mathbf{x} = \{\langle \text{sos} \rangle c_1 l_1 t_1 r_1 b_1 \| \ldots \| c_N l_N t_N r_N b_N \langle \text{eos} \rangle\}$$

Unconditional & Conditional Layout Generation Tasks

Examples of (1). Unconditional Generation, (2)Generation conditioned on element types, and (3). Refinement.

(b). Typical subtasks of graphic layout generation
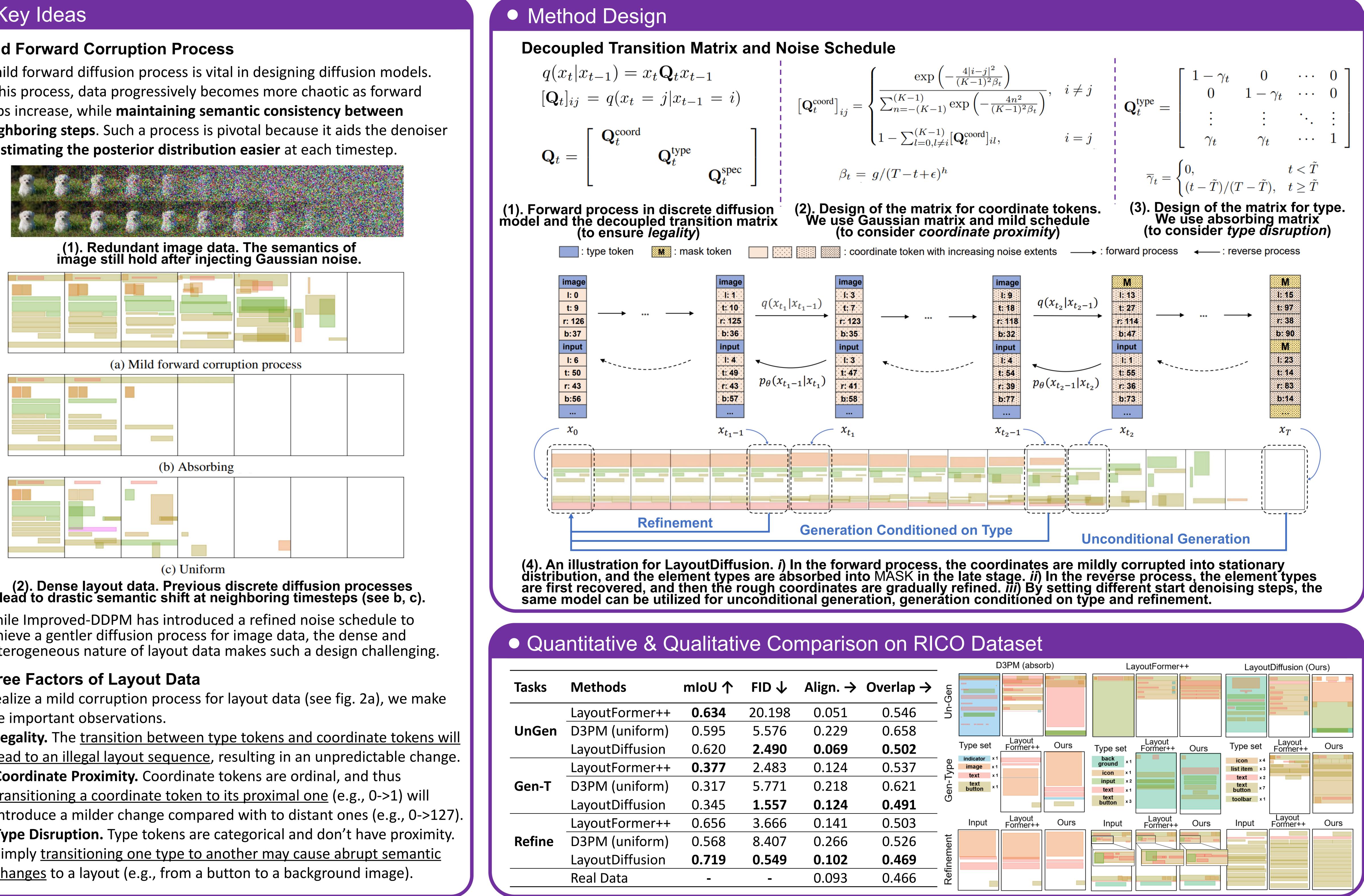
Key Ideas

Mild Forward Corruption Process

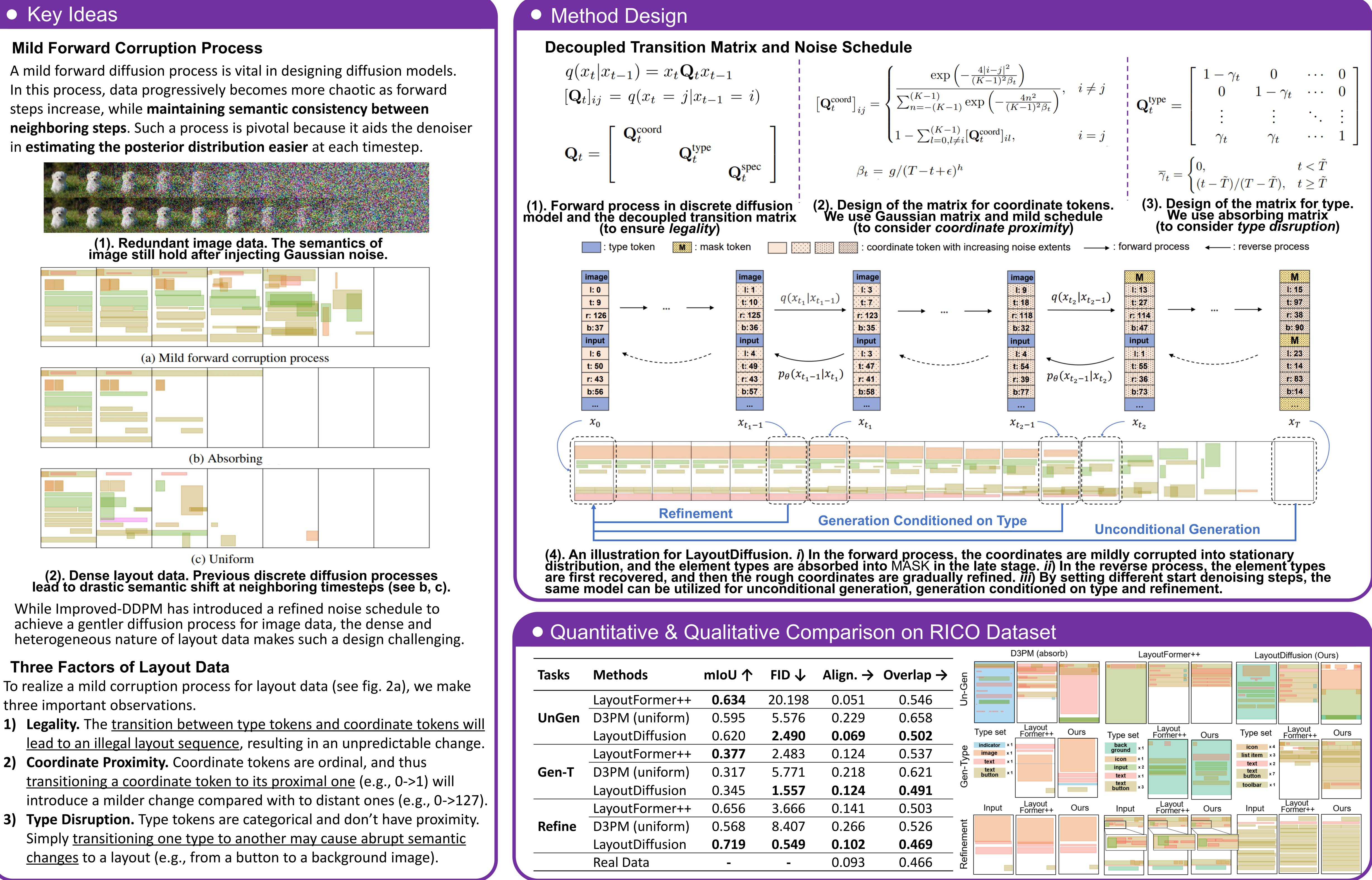A mild forward diffusion process is vital in designing diffusion models. In this process, data progressively becomes more chaotic as forward steps increase, while **maintaining semantic consistency between neighboring steps**. Such a process is pivotal because it aids the denoiser in **estimating the posterior distribution easier** at each timestep.

(1). Redundant image data. The semantics of image still hold after injecting Gaussian noise.

(a) Mild forward corruption process

(b) Absorbing

(c) Uniform

(2). Dense layout data. Previous discrete diffusion processes lead to drastic semantic shift at neighboring timesteps (see b, c).

While Improved-DDPM has introduced a refined noise schedule to achieve a gentler diffusion process for image data, the dense and heterogeneous nature of layout data makes such a design challenging.

Three Factors of Layout Data

To realize a mild corruption process for layout data (see fig. 2a), we make three important observations.

1) **Legality.** The <u>transition between type tokens and coordinate tokens will lead to an illegal layout sequence</u>, resulting in an unpredictable change.
2) **Coordinate Proximity.** Coordinate tokens are ordinal, and thus <u>transitioning a coordinate token to its proximal one</u> (e.g., 0->1) will introduce a milder change compared with to distant ones (e.g., 0->127).
3) **Type Disruption.** Type tokens are categorical and don't have proximity. Simply <u>transitioning one type to another may cause abrupt semantic changes</u> to a layout (e.g., from a button to a background image).

Method Design

Decoupled Transition Matrix and Noise Schedule

$$q(x_t|x_{t-1}) = x_t \mathbf{Q}_t x_{t-1}$$
$$[\mathbf{Q}_t]_{ij} = q(x_t = j | x_{t-1} = i)$$

$$\mathbf{Q}_t = \begin{bmatrix} \mathbf{Q}_t^{\text{coord}} & & \\ & \mathbf{Q}_t^{\text{type}} & \\ & & \mathbf{Q}_t^{\text{spec}} \end{bmatrix}$$

$$[\mathbf{Q}_t^{\text{coord}}]_{ij} = \begin{cases} \dfrac{\exp\left(-\frac{4|i-j|^2}{(K-1)^2\beta_t}\right)}{\sum_{n=-(K-1)}^{(K-1)} \exp\left(-\frac{4n^2}{(K-1)^2\beta_t}\right)}, & i \neq j \\ 1 - \sum_{l=0, l\neq i}^{(K-1)} [\mathbf{Q}_t^{\text{coord}}]_{il}, & i = j \end{cases}$$

$$\mathbf{Q}_t^{\text{type}} = \begin{bmatrix} 1-\gamma_t & 0 & \cdots & 0 \\ 0 & 1-\gamma_t & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_t & \gamma_t & \cdots & 1 \end{bmatrix}$$

$$\beta_t = g/(T-t+\epsilon)^h$$

$$\bar{\gamma}_t = \begin{cases} 0, & t < \tilde{T} \\ (t-\tilde{T})/(T-\tilde{T}), & t \geq \tilde{T} \end{cases}$$

(1). Forward process in discrete diffusion model and the decoupled transition matrix (to ensure *legality*)

(2). Design of the matrix for coordinate tokens. We use Gaussian matrix and mild schedule (to consider *coordinate proximity*)

(3). Design of the matrix for type. We use absorbing matrix (to consider *type disruption*)

■ : type token   M : mask token   ▦ : coordinate token with increasing noise extents   → : forward process   ⤸ : reverse process

(4). An illustration for LayoutDiffusion. *i*) In the forward process, the coordinates are mildly corrupted into stationary distribution, and the element types are absorbed into MASK in the late stage. *ii*) In the reverse process, the element types are first recovered, and then the rough coordinates are gradually refined. *iii*) By setting different start denoising steps, the same model can be utilized for unconditional generation, generation conditioned on type and refinement.

Quantitative & Qualitative Comparison on RICO Dataset

Tasks	Methods	mIoU ↑	FID ↓	Align. →	Overlap →
UnGen	LayoutFormer++	**0.634**	20.198	0.051	0.546
	D3PM (uniform)	0.595	5.576	0.229	0.658
	LayoutDiffusion	0.620	**2.490**	0.069	0.502
Gen-T	LayoutFormer++	**0.377**	2.483	0.124	0.537
	D3PM (uniform)	0.317	5.771	0.218	0.621
	LayoutDiffusion	0.345	**1.557**	**0.124**	**0.491**
Refine	LayoutFormer++	0.656	3.666	0.141	0.503
	D3PM (uniform)	0.568	8.407	0.266	0.526
	LayoutDiffusion	**0.719**	**0.549**	**0.102**	**0.469**
	Real Data	-	-	0.093	0.466