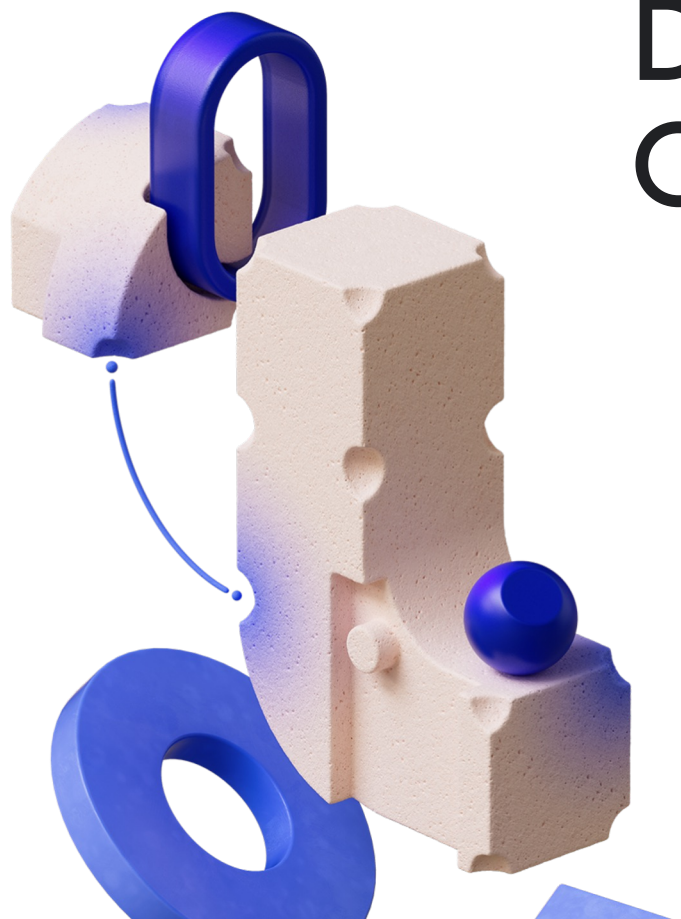




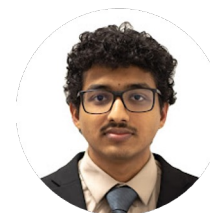
Diffusion Models for Constrained Generation



Dheeraj
Nagaraj



Karthikeyan
Shanmugam



Gautham
Govind Anil



Sachin
Yadav

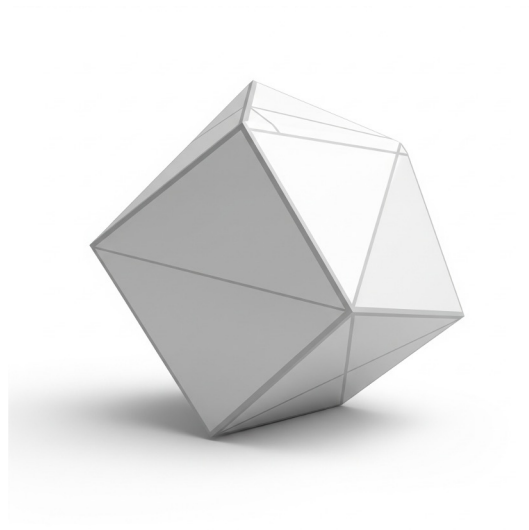


Generative Modeling

- Dataset D of n samples.
- Generate more samples from the same distribution.
- We are interested in the individual samples themselves rather than statistics as in classical MCMC.
- **Autoregressive Models:** Generates text word-by-word by learning to generate text from a large corpus.
- **Diffusion Models:** Generate images by denoising 'white noise'.
- Samples could be discrete, continuous or mixed-mode.

Generation with Explicit Constraints

- Classical Case. Constraint set specified explicitly.
- Design algorithms to generate random variables constrained to this set.



Sample from a convex body.

4	3	5	2	6	9	7	8	1
6	8	2	5	7	1	4	9	3
1	9	7	8	3	4	5	6	2
8	2	6	1	9	5	3	4	7
3	7	4	6	8	2	9	1	5
9	5	1	7	4	3	6	2	8
5	1	9	3	2	6	8	7	4
2	4	8	9	5	7	1	3	6
7	6	3	4	1	8	2	5	9

Sample the solution to a SuDoKu



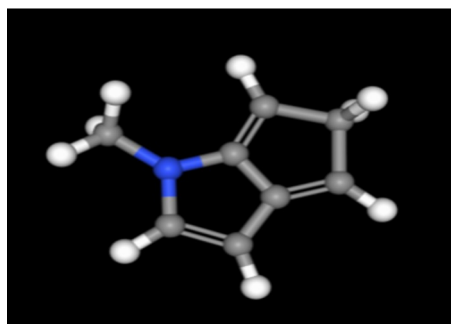
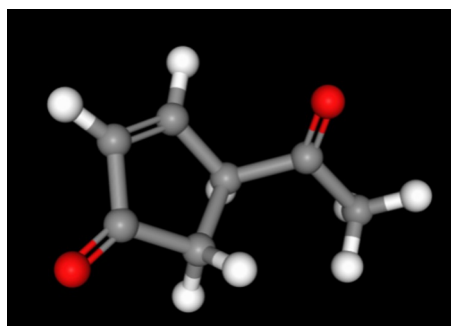
Implicit Constraints/ Unspecified Constraints

- Lots of data available, we know it is constrained.
- We don't know how to describe the constraints explicitly.
- Might be impossible/ implausible to do so algorithmically.

Implicit Constraints/ Unspecified Constraints

● Molecule Generation:

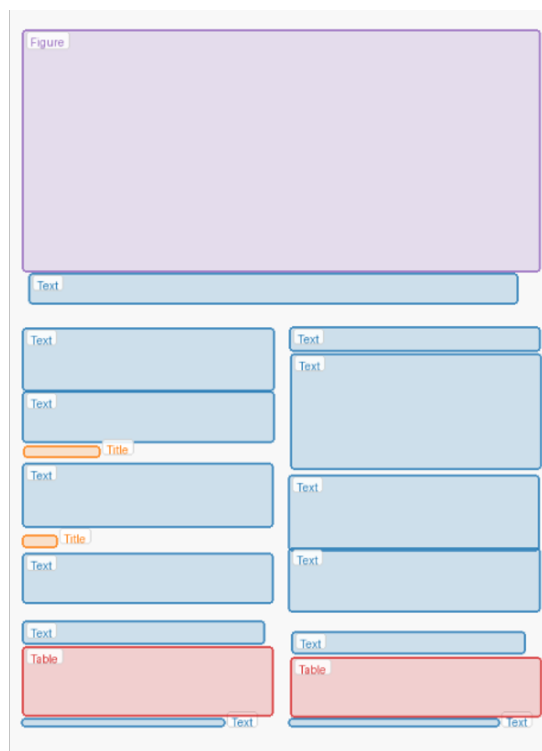
■ C (Carbon) ■ H (Hydrogen) ■ O (Oxygen) ■ N (Nitrogen)



- A dataset with 100k molecule structures.
- What is a valid molecule?
- Valencies are satisfied, bond angles and lengths are plausible.
- Is that sufficient? Or Necessary?
- Harder Question: When do proteins bind to a substrate? Can we describe it mathematically? (no idea)

Implicit Constraints/ Unspecified Constraints

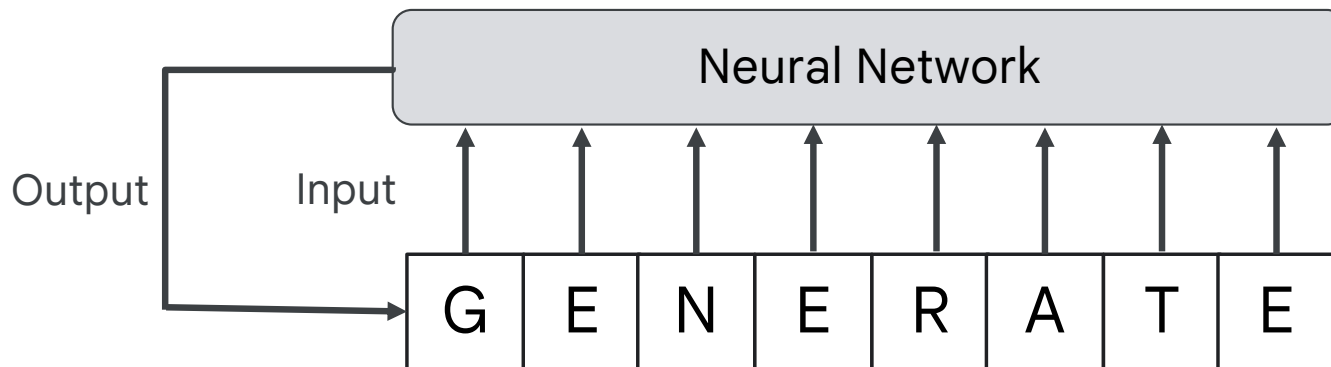
- Layout Generation:



- A Layout with text boxes, image boxes, figure boxes etc.
- Constraint: no overlap
- What makes it appealing?

How do Autoregressive Models Behave?

Generate joint distributions coordinate by coordinate (or token by token)





How do Autoregressive Models Behave?

4	3	5	2	6	9	7	8	1
6	8	2	5	7	1	4	9	3
1	9	7	8	3	4	5	6	2
8	2	6	1	9	5	3	4	7
3	7	4	6	8	2	9	1	5
9	5	1	7	4	3	6	2	8
5	1	9	3	2	6	8	7	4
2	4	8	9	5	7	1	3	6
7	6	3	4	1	8	2	5	9

How AR models solve it:

4	3	5	2	6	9	7	8	1
6	8	2	5	7	1		9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

- Is this how humans naturally solve it?
- Is this the 'best' way?
- Should we move beyond autoregression?

Continuous Diffusion Models



$$dX_t = -X_t dt + \sqrt{2} dB_t$$

destructive process



constructive process

$$dX_t = X_t dt + 2 \nabla \log p_{T-t}(X_t) dt + \sqrt{2} dB_t$$

- Learn the score function $\nabla \log p_t()$ via Tweedie's formula
- Sample new images from pure noise!

Continuous Diffusion Models



constructive process

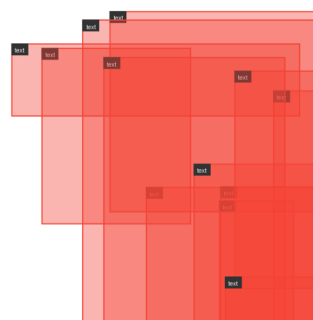
$$dX_t = X_t dt + 2\nabla \log p_{T-t}(X_t) dt + \sqrt{2} dB_t$$

- All the tokens (such as pixels) are generated simultaneously instead of one-by-one as in autoregressive models.
- Can be sped up by coarsening the numerical discretization parameter (sampling steps can be varied between ~1000 to ~25)

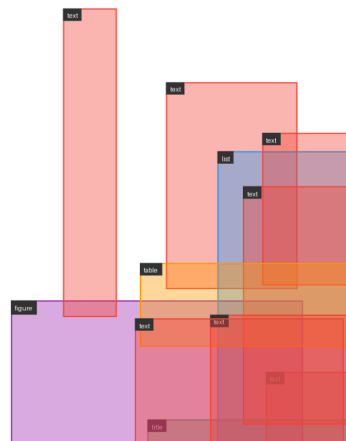
Discrete and Mixed Mode Diffusions?

- Real world data can be discrete (such as language) or mixed mode (such as layouts, images+text)
- Can we write down diffusion models for discrete data and discrete-continuous data? Why is this interesting?

Time Step: 0



Time Step: 0



Discrete Diffusion Models (Masked)

- Continuous Noising Processes $\longleftrightarrow$ Continuous Diffusion Models
- Discrete Noising Processes $\longleftrightarrow$ Discrete Diffusion Models

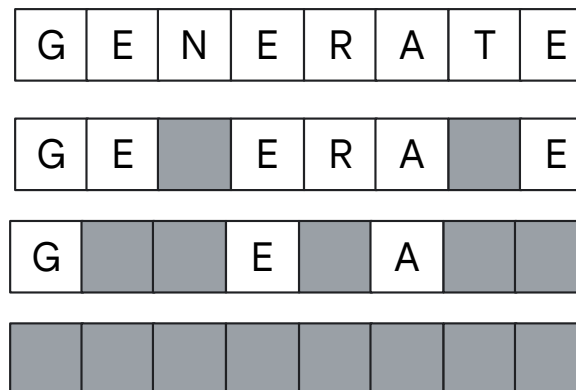
First work: D3PM by Austin et al, lots of variants and follow up work

Example 1: Masked Diffusion, how to noise?

Step 1: Flip each to mask with probability p .

Step 2: Flip each to mask with probability p .

Step 3: Flip each to mask with probability p .



Discrete Diffusion Models: Masked

Noising:

Step 0	G	E	N	E	R	A	T	E
Step 1	G	E		E	R	A		E
Step 2	G			E		A		
Step 3								

Denoising:

Step 3: Start with Full Noise

--	--	--	--	--	--	--	--	--

Step 2: Predict original, apply Bayes rule

		N					T	E
--	--	---	--	--	--	--	---	---

Step 1: Predict original, apply Bayes rule

	E	N		R	A	T	E	
--	---	---	--	---	---	---	---	--

Step 0: Predict original, apply Bayes rule

G	E	N	E	R	A	T	E	
---	---	---	---	---	---	---	---	--

Autoregressive Models

G								
G	E							
G	E	N						
G	E	N	E					
G	E	N	E	R				
G	E	N	E	R	A			
G	E	N	E	R	A	T		
G	E	N	E	R	A	T	E	



Discrete Diffusion Models (Multinomial)

Example 2: Multinomial Noise

Step 1: Flip to independently sampled with probability p

G	E	N	E	R	A	T	E
---	---	---	---	---	---	---	---

G	E	K	E	R	A	H	E
---	---	---	---	---	---	---	---

Step 2: Flip to independently sampled with probability p

G	L	G	E	S	A	H	B
---	---	---	---	---	---	---	---

Step 3: Flip to independently sampled with probability p

U	P	G	E	A	I	H	E
---	---	---	---	---	---	---	---

(Grey spaces are not real, for illustration purposes only)



Discrete Diffusion Models: Multinomial

Denoising:

Step 4: Start with Full Noise

U	P	G	E	A	I	H	E
---	---	---	---	---	---	---	---

Step 3: Predict original, apply Bayes rule

U	P	N	E	A	I	L	E
---	---	---	---	---	---	---	---

Step 2: Predict original, apply Bayes rule

G	P	N	E	A	A	L	E
---	---	---	---	---	---	---	---

Step 1: Predict original, apply Bayes rule

G	E	N	E	A	A	T	E
---	---	---	---	---	---	---	---

Step 0: Predict original, apply Bayes rule

G	E	N	E	R	A	T	E
---	---	---	---	---	---	---	---

Multiple flips at the same token possible, allows backtracking!

That is, have rough drafts, and refine by correcting the mistakes.



Comparison of Some Key Points

	Autoregressive	Masked Diffusion	Multinomial Diffusion
Generation Order	Causal (left to right)	Arbitrary	Arbitrary
Noise vs Signal	Known	Known	Unknown
Tokens per call	1	1 or Many	1 or Many
Backtracking	No	No	Yes
Inpainting	No	Yes	Yes

(Algorithmic changes can allow more features for autoregressive or masked diffusion models.)

How to Learn Discrete Diffusion?

- Most proposals for learning hinge on predicting the original unnoised sequence given a noisy sequence:

NN(

	E	E	
--	---	---	--

 , step 2) predicts

D	E	E	P
---	---	---	---

NN(

Z	E	E	O
---	---	---	---

 , step 2) predicts

D	E	E	P
---	---	---	---

- Austin et al (2021): Variational Lower Bound + Cross Entropy.
- Meng et al (2022): Ratio Matching
- Varma et al (2024): Binary classification

What are they good for (potentially)?

Flips tokens one by one to maintain accuracy. Detrimental to speed.

- **Planning:** With highly dependent tokens as in planning problems (3SAT, molecule generation), diffusion models can maintain global constraints.
- **Speed:** Can parallelly produce multiple tokens, giving a speedup over autoregressive models.

Hard to model joint distribution on a large product space. Assumes independent flips, detrimental to planning.

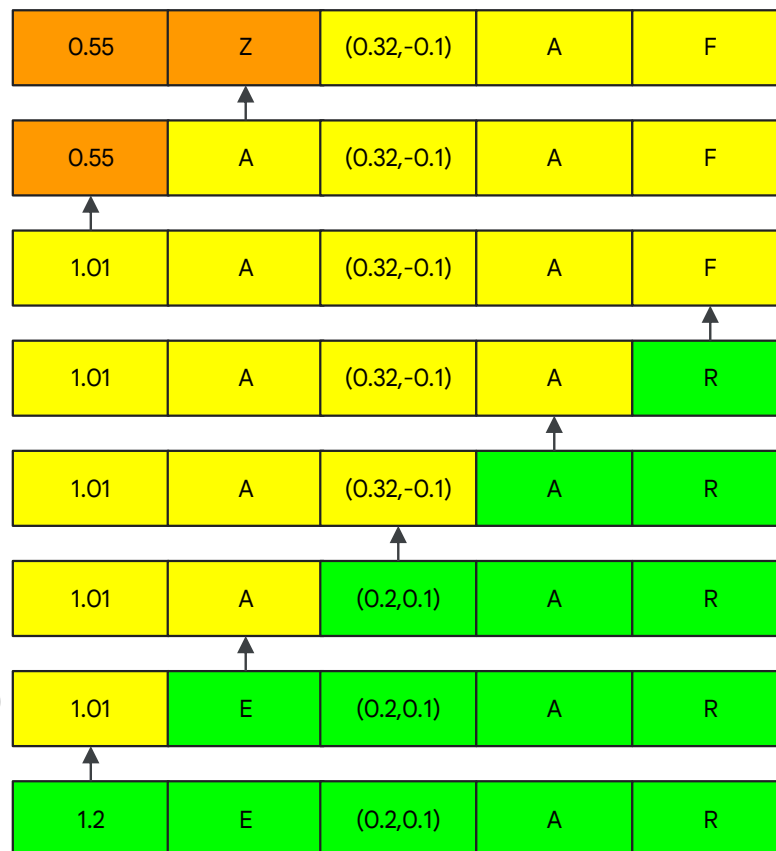


Performance for Constrained Generation

- We introduce Interleaved Gibbs Diffusion (IGD); framework for generating discrete-continuous data
- Does not assume factorizability of reverse process; can capture strong dependencies needed for constrained generation
- Supports conditional generation; also proposes an inference-time strategy for improving generations
- SoTA performance on Layout Generation, Molecule Generation and Random 3-SAT

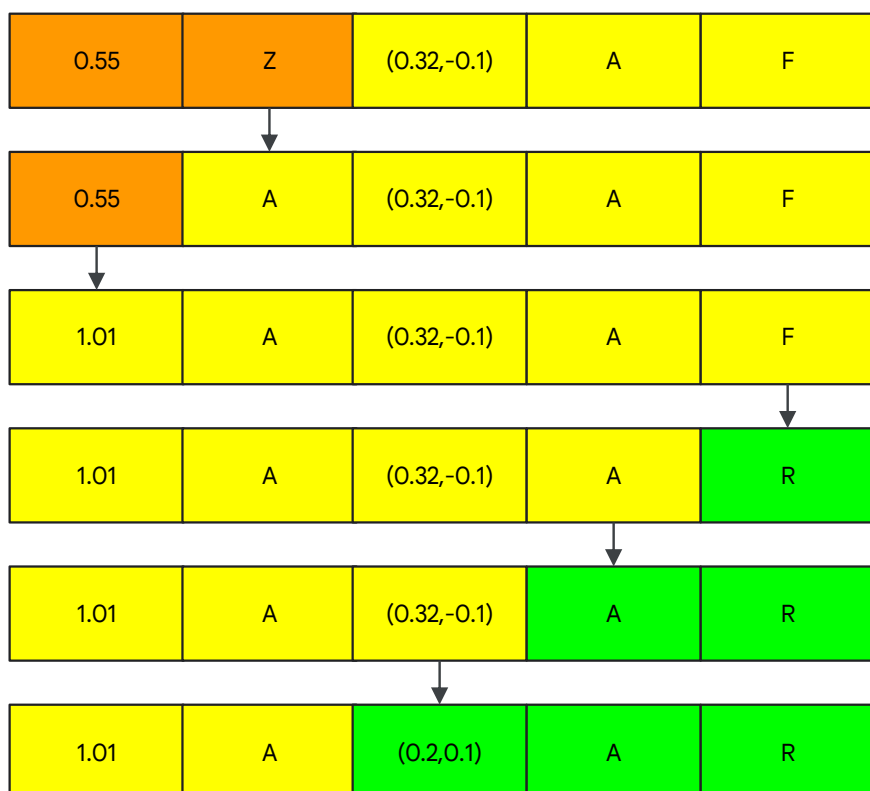
Forward Process

Round robin order

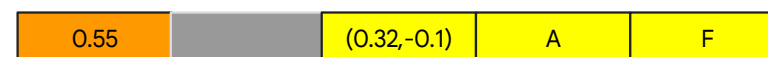


- Time independent, round robin Gibbs type Markov Chain.
- Converges to stationarity after $O(\log n)$ passes.
- Noises each coordinate group one by one.

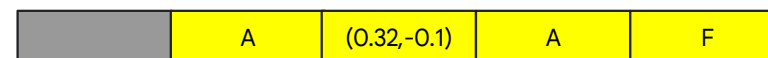
Reverse Process



Discrete denoise: Sample partially noised Position 2, given:



Cts denoise: Sample partially noised Position 1, given:



- Reverse order as forward process
- Time dependent Markov Chain
- Continuous denoiser can be implemented via standard diffusion model framework
- How to design the discrete sampler?



Reverse Process - Interleaved Denoising

```

input  $\hat{s}^T \sim P_T$ , discrete denoiser DiscDen, continuous
      denoiser ContDen, noise positions  $\{i_t\}$ 
output  $\hat{s}^0 \sim \pi$ 
for  $t \in [T, T - 1, \dots, 1]$  do
     $\hat{s}_{-i_t}^{(t-1)} = \hat{s}_{-i_t}^{(t)}$ 
    if  $\hat{s}_{i_t}^{(t)}$  is discrete then
         $\hat{s}_{i_t}^{(t-1)} = \text{DiscDen}(\hat{s}^{(t)}, i_t, t)$ 
    else
         $\hat{s}_{i_t}^{(t-1)} = \text{ContDen}(\hat{s}^{(t)}, i_t, t)$ 
    end if
end for
  
```

Reverse Process

$\text{DiscDen}(\hat{s}^{(t)}, i_t, t)$ ideally samples from $\mathbb{P}(s_{i_t}^{(t)} = \cdot | s_{-i_t}^{(t+1)} = \hat{s}_{-i_t}^{(t+1)})$

$\text{ContDen}(\hat{s}_{i_t}^{(t)}, i_t, t)$ ideally samples from $\mathbb{P}(s_{i_t}^{(t)} = \cdot | s^{(t+1)} = \hat{s}^{(t+1)})$

- If the denoisers are ideal, we recover the target distribution **π exactly**
- Such a guarantee does not exist if we assume factorizability
- Essentially, we sacrifice speed for an exact reversal

Learning the Denoisers

Lemma 4.1. *Let $s \in \mathcal{S}^L$. Then, for $x \in \mathcal{X}$ and discrete $s_{i_t}^{(t)}$, we can write $\mathbb{P} \left(s_{i_t}^{(t)} = x | s_{-i_t}^{(t+1)} = s_{-i_t} \right)$ as :*

$$\frac{\mathbb{P}(z_t = x)}{\mathbb{P}(z_t = \phi)} \left(\frac{1}{\mathbb{P} \left(z_t = x | s_{-i_t}^{(t+1)} = s_{-i_t}, s_{i_t}^{(t+1)} = x \right)} - 1 \right)$$

where $(s^{(0)}, \dots, s^{(T)},)$ is obtained from forward process.

Learning the Denoisers

Lemma 4.2. *Under the considered forward process where noising occurs independently, we have:*

$$\nabla_{s_{i_t}^{(t,k+1)}} \log q(s_{i_t}^{(t,k+1)} | s_{-i_t}^{(t,k+1)}) = -\frac{1}{\sqrt{1 - \bar{\alpha}}} \mathbb{E} [\epsilon | s^{(t,k+1)}]$$

Score = MMSE estimation of cumulative noise added from time 0



Training Algorithm

Algorithm 2: Model Training

Input: D , f_θ , FwdPrCs, opt, T , $\{i_t\}_{t=0}^{T-1}$, $\{\Pi_t\}$, $\{\beta_j\}$

Output: Trained θ

For each training iteration do:

1. $s^{(0)} \sim D$

2. Sample $t \in \{0, 1, \dots, T-1\}$

3. **if** $s_{i_t}^{(0)}$ is discrete then

$$(s^{(t)}, z_t, s^{(t+1)}) = \text{FwdPrCs}(s^{(0)}, t, \{i_\tau\}_{\tau=0}^t, \{\Pi_\tau\}_{\tau=0}^t)$$

$$L = \text{BCE}\left(f_\theta(s_{i_t}^{(t+1)}), z_t\right)$$

4. **else**

$$(s^{(t)}, \epsilon) = \text{FwdPrCs}(s^{(0)}, t, \{i_\tau\}_{\tau=0}^t, \{\beta_j\})$$

$$L = \|\epsilon - f_\theta(s_{i_t}^{(t)})\|_2^2$$

5. $\theta \leftarrow \text{opt.update}(\theta, L)$

Conditional generation

A binary mask indicates which elements are fixed during forward and reverse process and they are handled appropriately.

ReDeNoise

After completing reverse process, the generated samples are re-noised to some fixed level and then denoised for potential quality improvements.

A first Example: 3 SAT

- Pick $n = 9$, draw ~100k random 3 SAT instances at the hardness threshold, and solve them via standard branch and bound algorithm.
- Train the IGD sampler to sample a satisfying variable assignment given an unseen clause
- We do not telling it that we are trying to solve 3 SAT. Can it learn to satisfy the constraints on its own?

Time 1/53

Clause Satisfaction

x2	¬x1	¬x3	¬x2	x1	¬x1	x4	¬x2	x1
¬x6	¬x7	¬x6	¬x7	x4	¬x5	¬x8	¬x5	x2
x3	x8	¬x7	x5	x5	x5	¬x8	x6	¬x7
¬x2	¬x4	x1	x2	x1	¬x4	¬x1	¬x1	x1
¬x4	x5	¬x3	x9	x8	¬x4	x2	¬x4	x5
¬x1	¬x1	¬x3	x5	¬x2	x3	¬x7	x3	¬x1
x6	¬x4	x6	¬x7	x6	x4	¬x9	¬x5	x2
x6	x3	¬x2	¬x4	¬x2	x5	x7	x2	x4
¬x5	¬x5	¬x9	¬x8	x3	x6	x7	x8	x3
x2	x2	x3	¬x4	¬x5	¬x4	x1	¬x5	x7
x1	x6	x5	x8	¬x7	¬x8	¬x5	x2	x1
¬x7	¬x9	x6	¬x9	¬x8	x9	¬x4	¬x5	x9

Current Variable Assignment

1	0	0	0	0	1	1	0	0
x1	x2	x3	x4	x5	x6	x7	x8	x9

Time 1/53

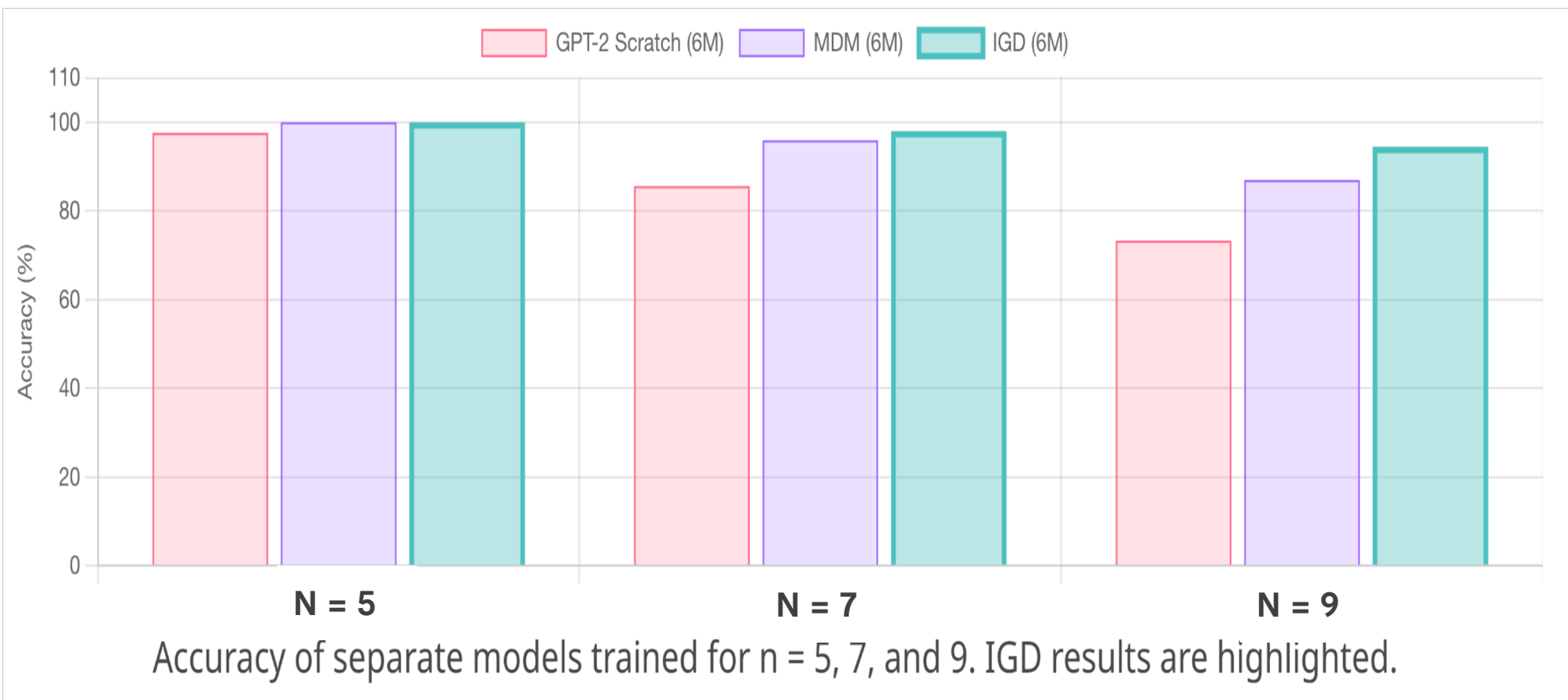
Clause Satisfaction

x3	¬x1	¬x3	¬x5	¬x4	¬x4	¬x4	x4	¬x2
x5	¬x6	x9	¬x8	¬x7	x7	¬x8	¬x6	¬x4
¬x7	x9	x5	¬x9	¬x8	x8	x9	¬x5	¬x6
¬x2	x3	¬x4	x3	x2	¬x3	x2	¬x3	¬x4
x7	¬x5	x5	¬x6	¬x4	¬x7	x6	¬x4	x8
x8	x9	x6	x7	x8	x8	¬x9	¬x5	x5
x1	x5	¬x2	x7	x4	¬x2	¬x3	x5	¬x1
¬x2	x6	¬x8	x7	x5	¬x4	¬x7	¬x7	¬x2
¬x9	¬x8	¬x7	x7	x7	¬x7	¬x9	¬x5	x8
x2	¬x5	x3	x3	x4	x6	¬x1	¬x2	x4
x5	x8	x4	x4	x6	x8	¬x3	¬x5	x5
x8	¬x9	¬x7	x7	x7	¬x7	¬x9	¬x7	x8
¬x5	¬x2	¬x2	¬x5	x3	x4	¬x5	x4	¬x2
x6	x5	x6	x7	x7	¬x5	x8	x5	x4
x8	x8	x6	¬x8	x8	x9	x9	¬x7	x8
¬x1	¬x3	x5	¬x3	x2	x1	¬x2	x4	x7
¬x6	¬x4	¬x6	¬x6	x3	x5	¬x8	x1	x5
¬x7	x7	¬x7	¬x6	x8	x6	x2	x6	x9

Current Variable Assignment

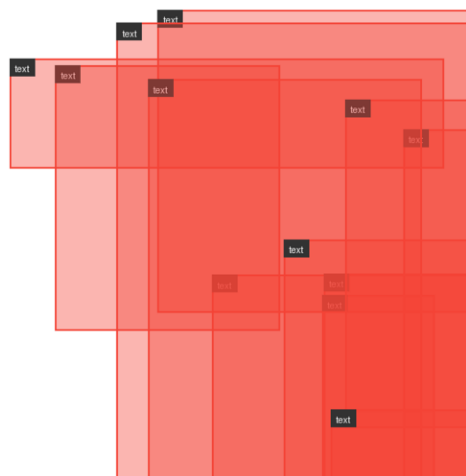
0	0	1	1	1	1	1	1	1
x1	x2	x3	x4	x5	x6	x7	x8	x9

3 SAT: Results

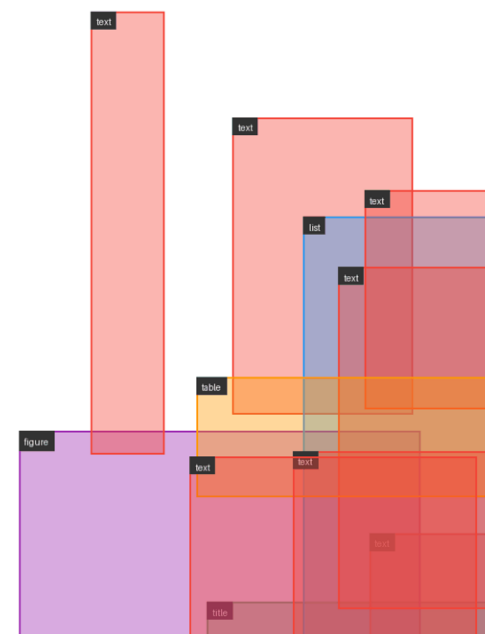


Layout Generation: Results

Time Step: 0

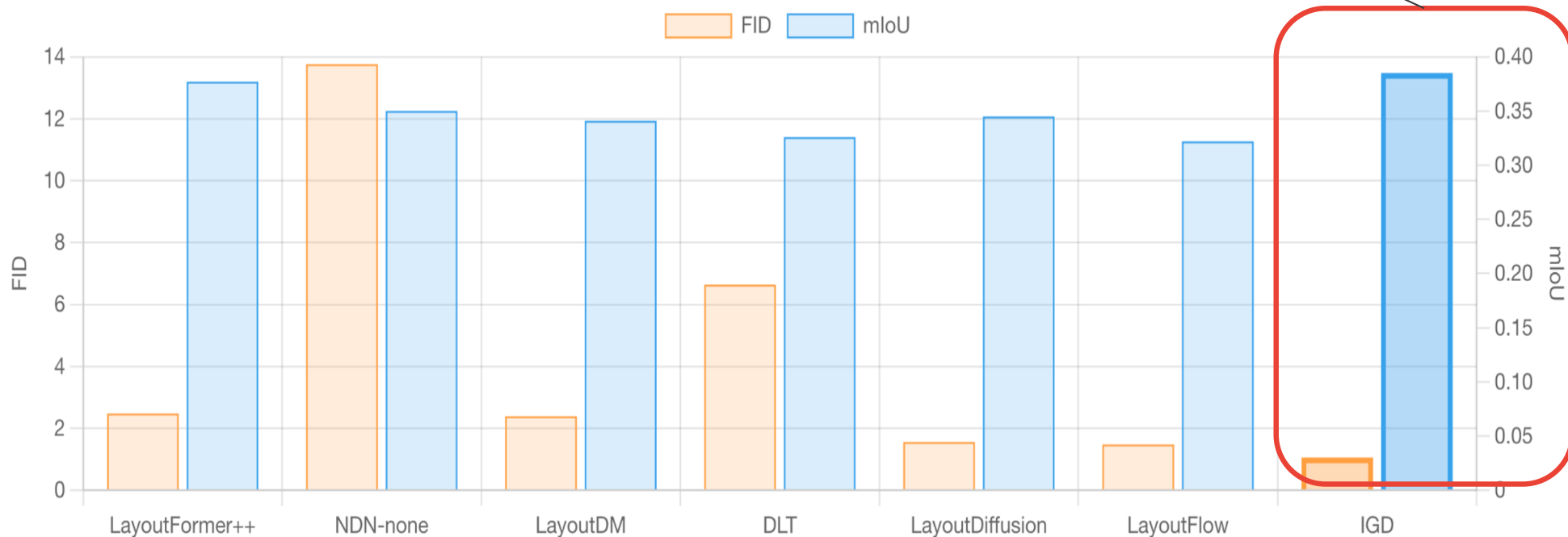


Time Step: 0



Google DeepMind

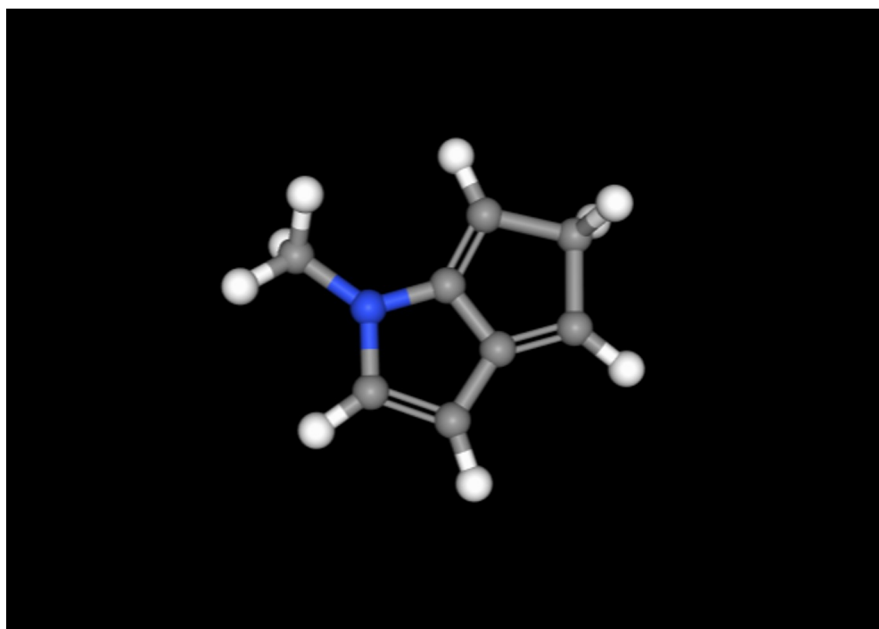
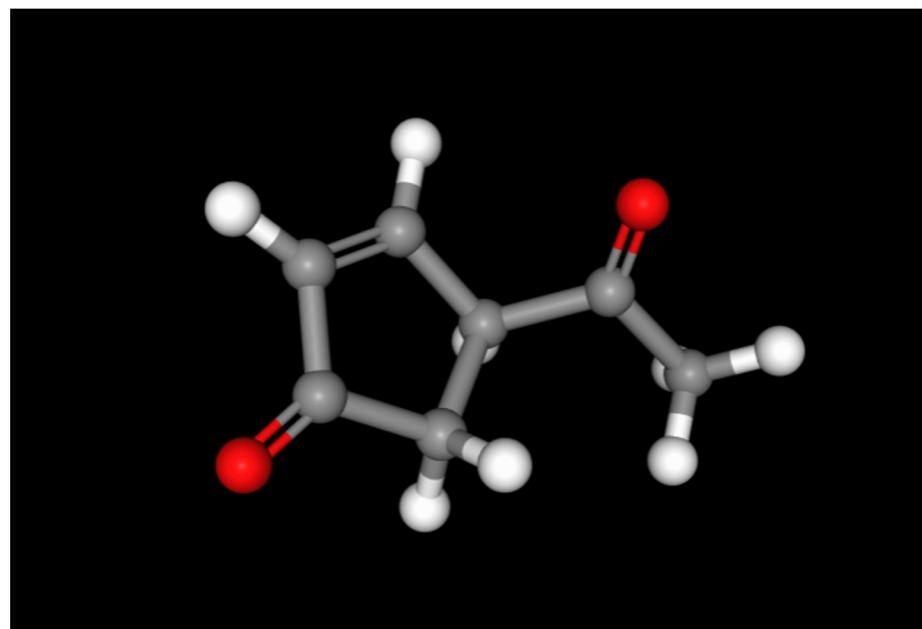
Layout Generation: Results

Ours!

Results on the RICO dataset for category conditioned layout generation. Lower FID is better; higher mIoU is better.

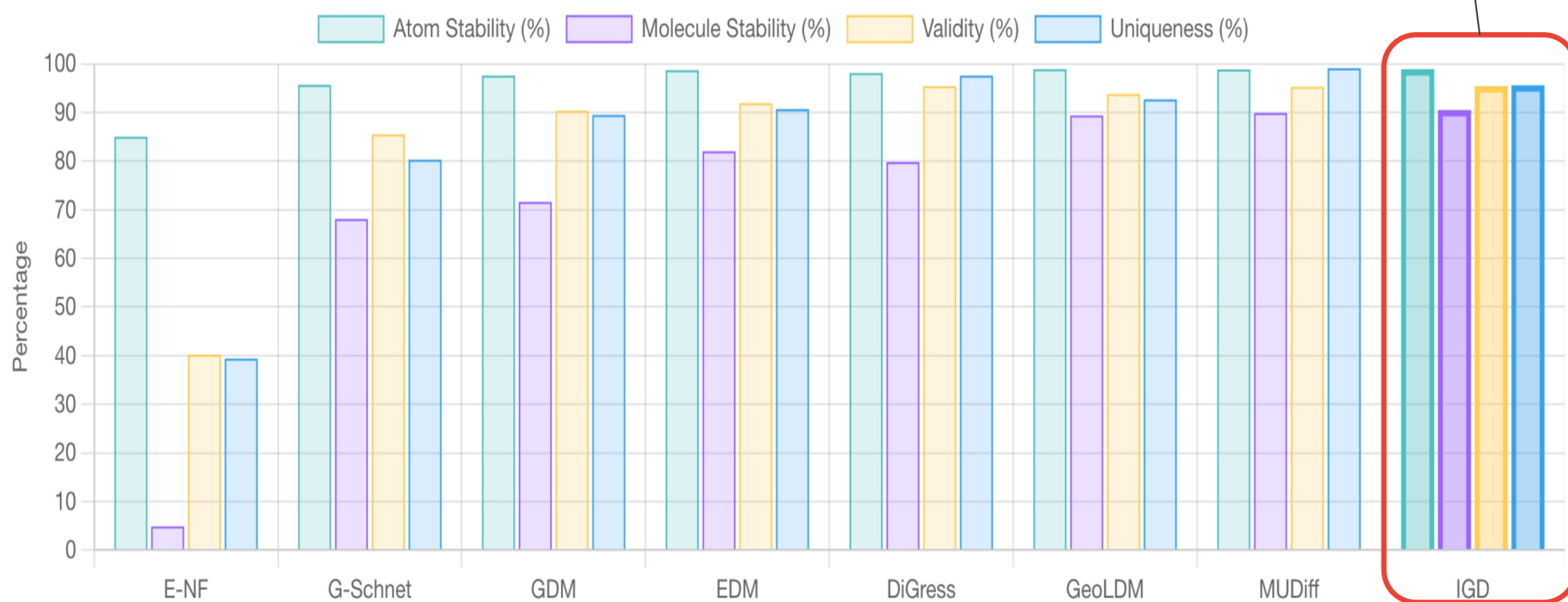
Google DeepMind

Molecule Generation: Results



■ C (Carbon) ■ H (Hydrogen) ■ O (Oxygen) ■ N (Nitrogen)

Molecule Generation: Results



Performance on the QM9 benchmark. Metrics are reported in percentages.



Thank you.

Google DeepMind

Interleaved Gibbs Diffusion: <https://arxiv.org/abs/2502.13450>

Glauber Generative Model: <https://arxiv.org/abs/2405.17035>