Generative Marginalization Models

Sulin Liu, Peter Ramadge, Ryan Adams

Princeton University

Discrete generative modeling

Discrete generative modeling

- **Problem**: generative modeling of discrete objects $\mathbf{x} = [x_1, \cdots, x_D]$, where x_d takes discrete values from $\{1, \cdots, K\}$

Discrete generative modeling

- **Problem**: generative modeling of discrete objects $\mathbf{x} = [x_1, \cdots, x_D]$, where x_d takes discrete values from $\{1, \cdots, K\}$
- **Useful because:**

Discrete generative modeling

- **Problem**: generative modeling of discrete objects $\mathbf{x} = [x_1, \cdots, x_D]$, where x_d takes discrete values from $\{1, \cdots, K\}$
- **Useful because:**
  - LLMs!

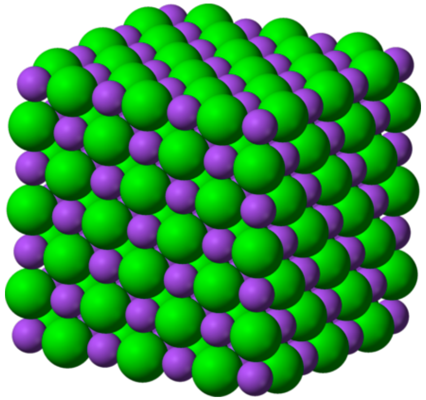
Source: Midjourney. Prompt: LLMs in 2023.

Discrete generative modeling

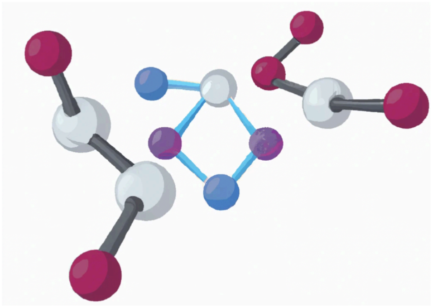
- **Problem**: generative modeling of discrete objects $\mathbf{x} = [x_1, \cdots, x_D]$, where x_d takes discrete values from $\{1, \cdots, K\}$
- **Useful because:**

Discrete generative modeling

- **Problem**: generative modeling of discrete objects $\mathbf{x} = [x_1, \cdots, x_D]$, where x_d takes discrete values from $\{1, \cdots, K\}$
- **Useful because:**
  - Many things in the real world are discrete

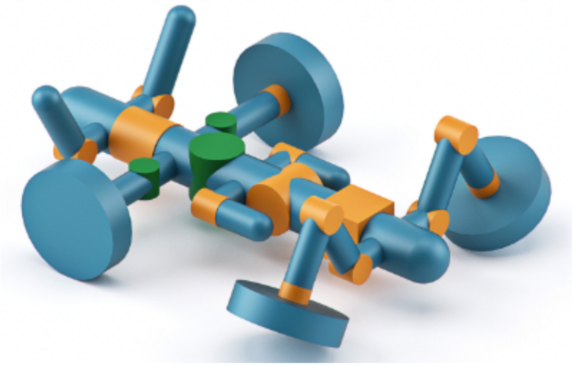


materials



molecules

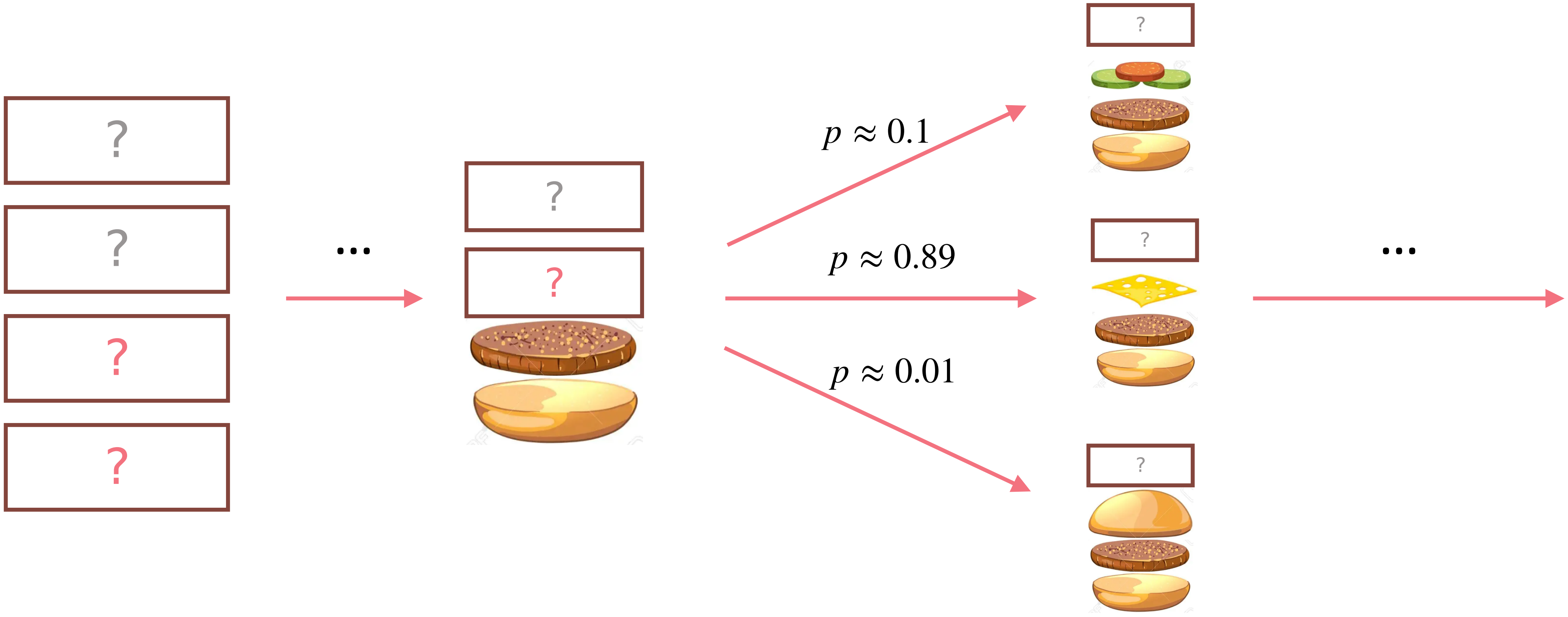
structures



robotic design

Generative models with maximum flexibility

- **Generate from any starting point in any order**



Any-order autoregressive models [Uria et al., 2013; Hoogeboom et al. 2021]

Any-order autoregressive models [Uria et al., 2013; Hoogeboom et al. 2021]

- Given an order σ
  - $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$

Any-order autoregressive models [Uria et al., 2013; Hoogeboom et al. 2021]

- Given an order σ
  - $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$
- Order agnostic lower bound of log likelihood:
  - $\log p_\theta(x) = \log \mathbb{E}_\sigma p_\theta(x \mid \sigma) \geq \mathbb{E}_\sigma \log p_\theta(x \mid \sigma)$

Any-order autoregressive models [Uria et al., 2013; Hoogeboom et al. 2021]

- Given an order σ
  - $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$
- **Order agnostic lower bound** of log likelihood:
  - $\log p_\theta(x) = \log \mathbb{E}_\sigma p_\theta(x \mid \sigma) \geq \boxed{\mathbb{E}_\sigma \log p_\theta(x \mid \sigma)}$

Any-order autoregressive models [Uria et al., 2013; Hoogeboom et al. 2021]

- Given an order σ
  - $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$
- **Order agnostic lower bound** of log likelihood:
  - $\log p_\theta(x) = \log \mathbb{E}_\sigma p_\theta(x \mid \sigma) \geq \boxed{\mathbb{E}_\sigma \log p_\theta(x \mid \sigma)}$

The *dog barks*

Word being predicted — Given words

$P(\text{The dog barks}) = P(\text{dog}) \times P(\text{barks} \mid \text{dog}) \times P(\text{The} \mid \text{dog barks})$

Learning conditionals have a scalability issue

Learning conditionals have a scalability issue

- Likelihood evaluation requires sequential conditionals:
  $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$

Learning conditionals have a scalability issue

- Likelihood evaluation requires sequential conditionals:
  $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$
  - Test time likelihood evaluation is $\mathcal{O}(D)$ 🤦‍♀️

Learning conditionals have a scalability issue

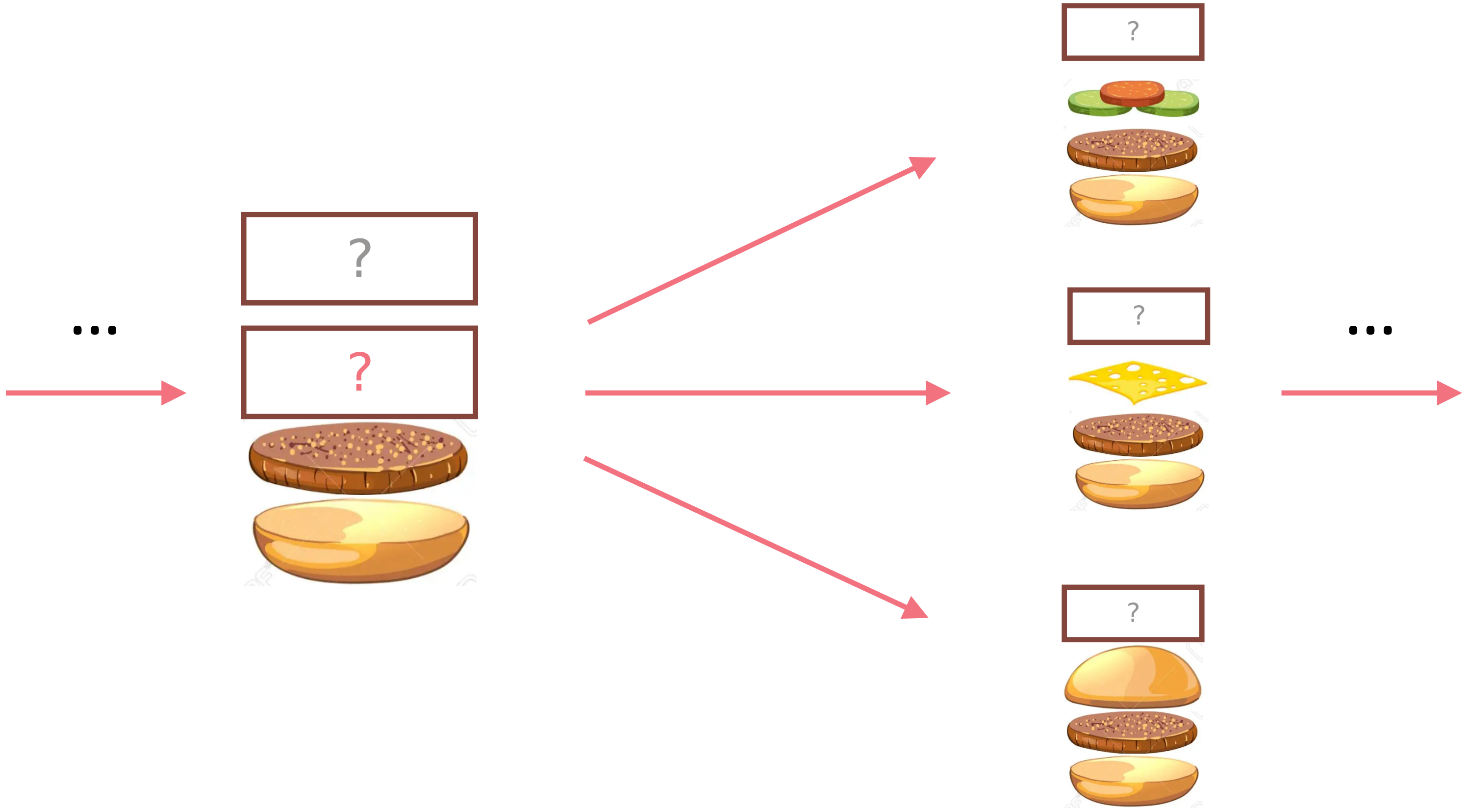
- Likelihood evaluation requires sequential conditionals:
  $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$
  - Test time likelihood evaluation is $\mathcal{O}(D)$ 🤦‍♀️
  - Leads to training non-scalability in energy-based training 🤦‍♀️

Learning conditionals have a scalability issue

- Likelihood evaluation requires sequential conditionals:
  $\log p_\theta(x) = \log p_\theta(x_{\sigma(1)}) + \log p_\theta(x_{\sigma(2)} \mid x_{\sigma(1)}) + \log p_\theta(x_{\sigma(3)} \mid x_{\sigma(1)}, x_{\sigma(2)}) + \cdots$
  - Test time likelihood evaluation is $\mathcal{O}(D)$ 🤦‍♀️
  - Leads to training non-scalability in energy-based training 🤦‍♀️
    - $\min_\theta \; D_{\mathbf{KL}}(p_\theta(x) \parallel \frac{f(x)}{Z})$

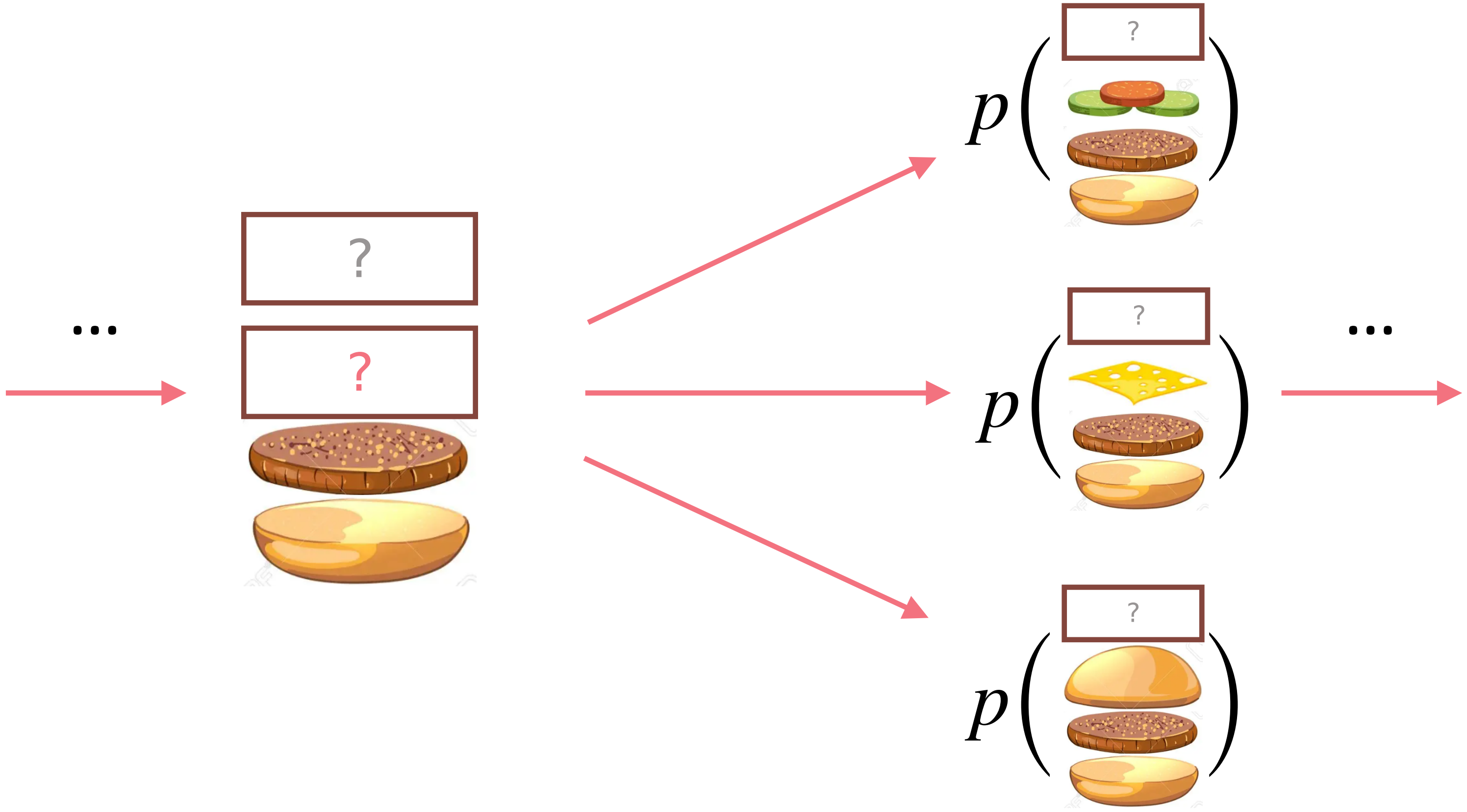
The dream of discrete generative modeling

- We can do anything we like if we have access to the marginals
  - Any-order generation



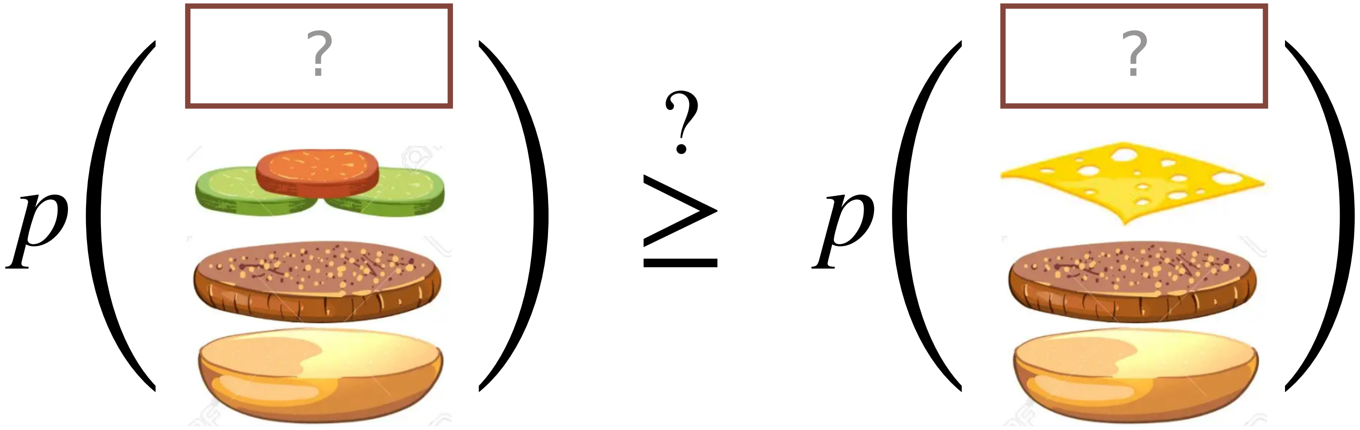
The dream of discrete generative modeling

- We can do anything we like if we have access to the marginals
  - Any-order generation



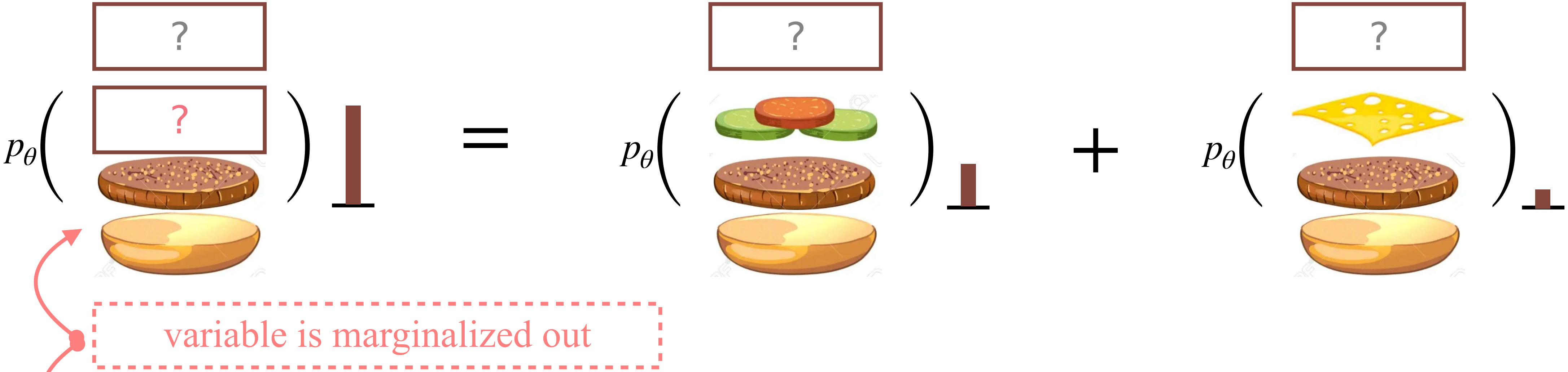
The dream of discrete generative modeling

- **We can do anything we like if we have access to the marginals**
  - Comparing likelihoods



How do we learn the marginals?

- **By enforcing marginalization self-consistency:**



$$p_\theta(?010??) = p_\theta(0010??) + p_\theta(1010??)$$

How do we learn the marginals?

How do we learn the marginals?

- Marginalization self-consistency:

$$p_\theta(\mathbf{x}_{\sigma(<d)}) = \sum_{x_{\sigma(d)}} p_\theta(\mathbf{x}_{\sigma(\leq d)}),$$

$$\forall \sigma \in S_D, x_i \in \{1, \cdots, K\}, d \in \{1, \cdots, D\}$$

How do we learn the marginals?

- Marginalization self-consistency:

$$p_\theta(\mathbf{x}_{\sigma(<d)}) = \sum_{x_{\sigma(d)}} p_\theta(\mathbf{x}_{\sigma(\le d)}),$$

$$\forall \sigma \in S_D, x_i \in \{1,\cdots,K\}, d \in \{1,\cdots,D\}$$

- When K is large, split into parallel self-consistency constraints:

$$p_\theta(\mathbf{x}_{\sigma(<d)})p_\phi(\mathbf{x}_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}) = p_\theta(\mathbf{x}_{\sigma(\le d)}),$$

$$\forall \sigma \in S_D, \mathbf{x} \in \{1,\cdots,K\}^D, d \in \{1,\cdots,D\}$$

How do we learn the marginals? — Maximum likelihood training

How do we learn the marginals? — Maximum likelihood training

$$\max_{\theta,\phi} \; \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log p_\theta(\mathbf{x})$$

How do we learn the marginals? — Maximum likelihood training

$$\max_{\theta,\phi} \; \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log p_\theta(\mathbf{x})$$

s.t. marginalization self-consistency constraints

How do we learn the marginals? — Maximum likelihood training

$$\max_{\theta,\phi} \ \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log p_\theta(\mathbf{x})$$

s.t. marginalization self-consistency constraints

- **(Theoretically justified) two-stage training:**
  - Stage 1: Learn the conditionals ϕ — maximizing log-likelihood lower-bound
  - Stage 2: Distill the marginals θ — minimizing marginalization self-consistency errors for the

How do we learn the marginals? — Energy-based training (reverse KL)

How do we learn the marginals? — Energy-based training (reverse KL)

$$\min_{\theta,\phi} \; D_{\mathrm{KL}}\left(p_\theta \,\|\, \frac{f}{Z}\right)$$

How do we learn the marginals? — Energy-based training (reverse KL)

$$\min_{\theta,\phi} \; D_{\mathrm{KL}}(p_\theta \,\|\, \frac{f}{Z})$$

s.t. marginalization self-consistency constraints

How do we learn the marginals? — Energy-based training (reverse KL)

$$\min_{\theta,\phi} \; D_{\mathrm{KL}}(p_\theta \parallel \frac{f}{Z})$$

$$\text{s.t.} \quad \text{marginalization self-consistency constraints}$$

- Penalized objective:
  - $D_{\mathrm{KL}}(p_\theta \parallel p) + \lambda \, \text{Self-consistency Penalty}(p_{\theta,\phi})$

How do we learn the marginals? — Energy-based training (reverse KL)

$$\min_{\theta,\phi} \; D_{\mathrm{KL}}(p_\theta \,\|\, \frac{f}{Z})$$

$$\text{s.t.} \quad \text{marginalization self-consistency constraints}$$

- Penalized objective:
  - $D_{\mathrm{KL}}(p_\theta \,\|\, p) + \lambda$ Self-consistency Penalty$(p_{\theta,\phi})$

- Scalable Training
  - KL divergence: **REINFORCE + Persistent block-Gibbs sampling**
  - Penalty: randomly sampling the self-consistency constraints

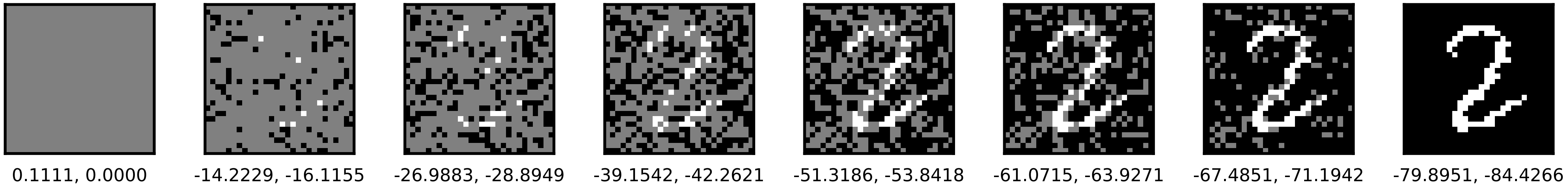
Results — Maximum likelihood training

Results — Maximum likelihood training

- **Marginals are distillable**
  - We show this for binary-MNIST, text8, molecules

Results — Maximum likelihood training

- **Marginals are distillable**
  - We show this for binary-MNIST, text8, molecules



Results — Maximum likelihood training

- **Marginals are distillable**
  - We show this for binary-MNIST, text8, molecules

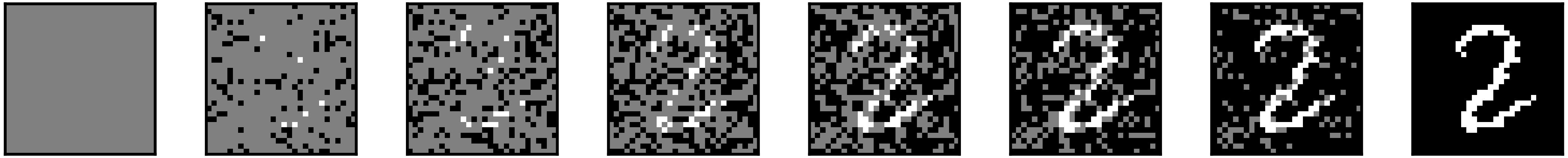


Table 1: Performance Comparison on Binary-MNIST

Model	**NLL (bpd)** ↓	**Spearman's** ↑	**Pearson** ↑	**LL inference time (s)** ↓
AO-ARM-E-U-Net	**0.148**	1.0	1.0	661.98 ± 0.49
AO-ARM-S-U-Net	0.149	**0.996**	**0.993**	132.40 ± 0.03
MaM-U-Net	0.149	0.992	**0.993**	$\mathbf{0.018 \pm 0.00}$
GflowNet-MLP	0.189	–	–	–

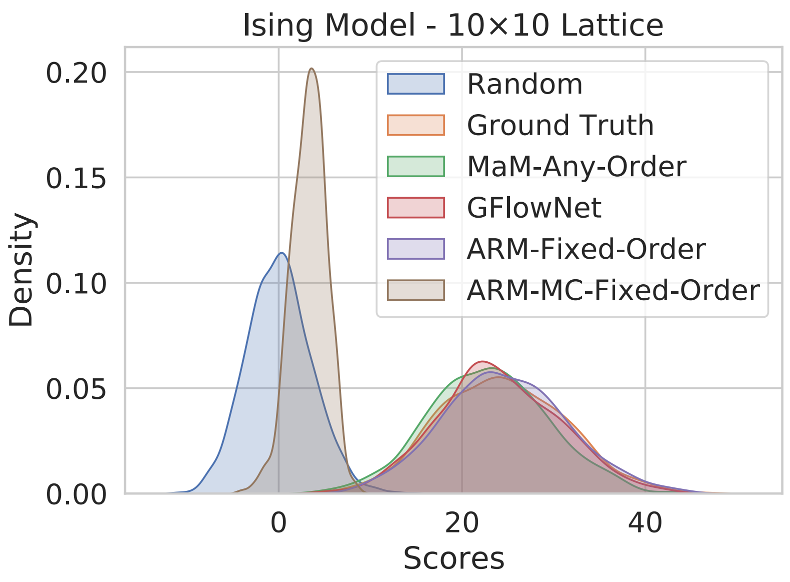
Results — Energy-based training

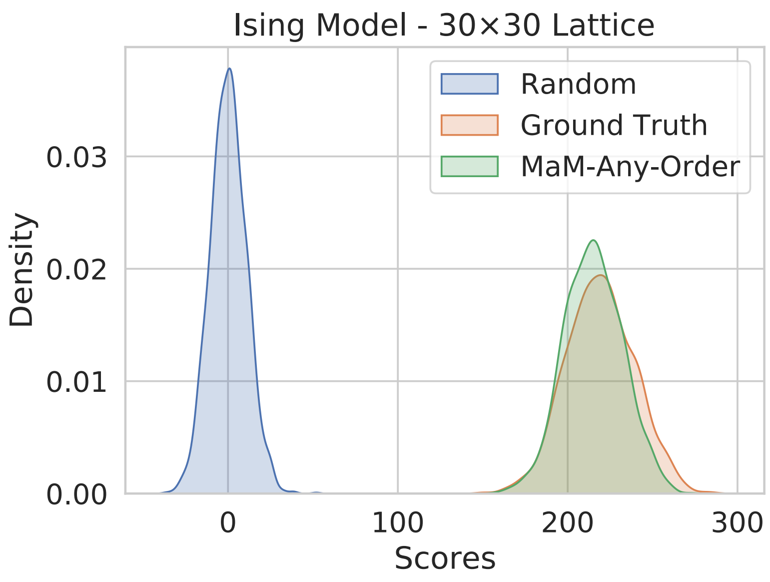
Results — Energy-based training

- **Modeling marginals make training scalable**
  - Ising model, molecule generation

Results — Energy-based training

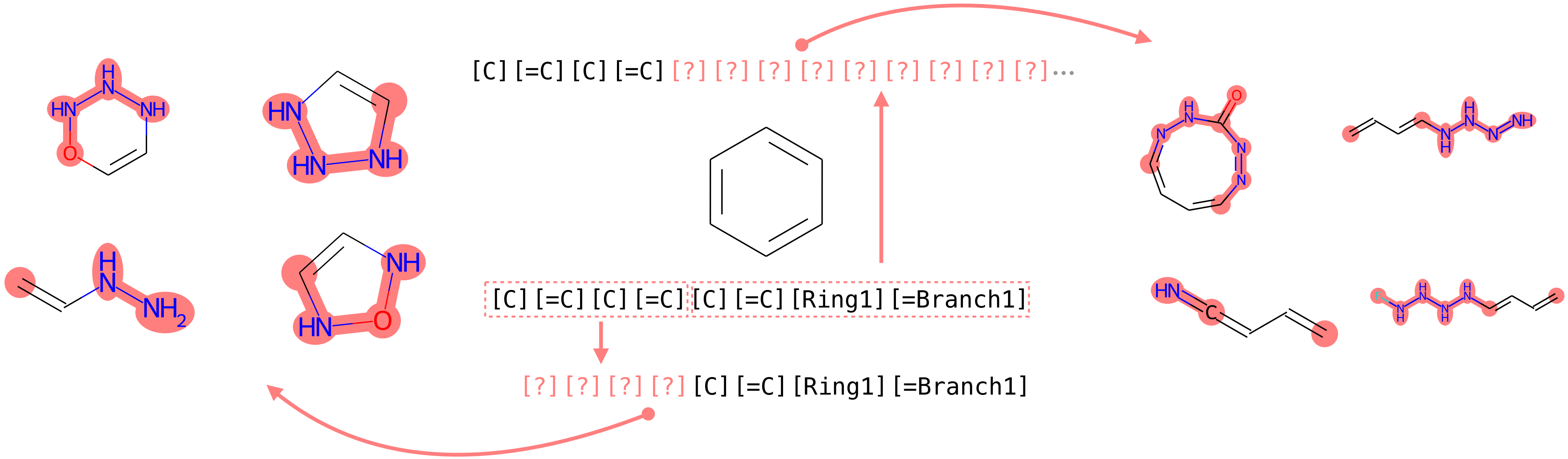
- **Modeling marginals make training scalable**
  - Ising model, molecule generation





Results — Energy-based training

- **Modeling marginals make training scalable**
    - Ising model, molecule generation



Conditionally generate molecules towards low lipophilicity from user-defined substructures.

Conclusions

Conclusions

- **Marginals are learnable/distillable**

Conclusions

- **Marginals are learnable/distillable**

- **Marginals —> scalable energy-based autoregressive modeling**

Thank you!

arxiv and code coming soon..