Generative Marginalization Models

Sulin Liu [1] **Peter J. Ramadge** [1] **Ryan P. Adams** [1]

Abstract

We introduce *marginalization models* (MAMs), a new family of generative models for high-dimensional discrete data. They offer scalable and flexible generative modeling by explicitly modeling all induced marginal distributions. Marginalization models enable fast approximation of arbitrary marginal probabilities with a single forward pass of the neural network, which overcomes a major limitation of arbitrary marginal inference models, such as any-order autoregressive models. MAMs also address the scalability bottleneck encountered in training any-order generative models for high-dimensional problems under the context of *energy-based training*, where the goal is to match the learned distribution to a given desired probability (specified by an unnormalized log-probability function such as energy or reward function). We propose scalable methods for learning the marginals, grounded in the concept of "*marginalization self-consistency*". We demonstrate the effectiveness of the proposed model on a variety of discrete data distributions, including images, text, physical systems, and molecules, for *maximum likelihood* and *energy-based training* settings. MAMs achieve orders of magnitude speedup in evaluating the marginal probabilities on both settings. For energy-based training tasks, MAMs enable any-order generative modeling of high-dimensional problems beyond the scale of previous methods. Code is available at github.com/PrincetonLIPS/MaM.

1. Introduction

Deep generative models have enabled remarkable progress across diverse fields, including image generation, audio synthesis, natural language modeling, and scientific discovery. However, there remains a pressing need to better support efficient probabilistic inference for key questions involving marginal probabilities $p(\mathbf{x}_\mathcal{S})$ and conditional probabilities $p(\mathbf{x}_\mathcal{U}|\mathbf{x}_\mathcal{V})$, for appropriate subsets $\mathcal{S}, \mathcal{U}, \mathcal{V}$ of the variables. The ability to directly address such quantities is critical in applications such as outlier or machine-generated content detection [59, 48], masked language modeling [15, 85], image inpainting [86], and constrained protein/molecule design [81, 65]. Furthermore, the capacity to conduct such inferences for arbitrary subsets of variables empowers users to leverage the model according to their specific needs and preferences. For instance, in protein design, scientists may want to manually guide the generation of a protein from a user-defined substructure under a particular path over the relevant variables. This requires the generative model to perform arbitrary marginal inferences.

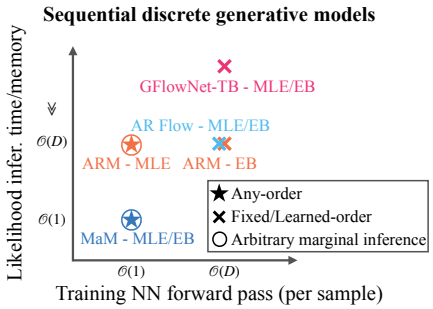


Figure 1. Scalability of sequential discrete generative models. The y-axis unit is # of NN forward passes required.

Towards this end, neural autoregressive models (ARMs) [3, 38] have shown great performance in conditional/marginal inference based on the idea of modeling a high-dimensional joint distribution as a factorization of univariate conditionals using the chain rule of probability. Many efforts have been made to scale up ARMs and enable any-order generative modeling under the setting of maximum likelihood estimation (MLE) [38, 78, 24], and great progress has been made in applications such as masked language modeling [85] and image inpainting [24]. However, marginal likelihood evaluation on a sequence of D variables is limited by $\mathcal{O}(D)$ neural network passes with the most widely-used modern neural network architectures (e.g., Transformers [80] and U-Nets [62]). This scaling makes it difficult to evaluate

[1]Princeton University. Correspondence to: Sulin Liu <sulinl@princeton.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

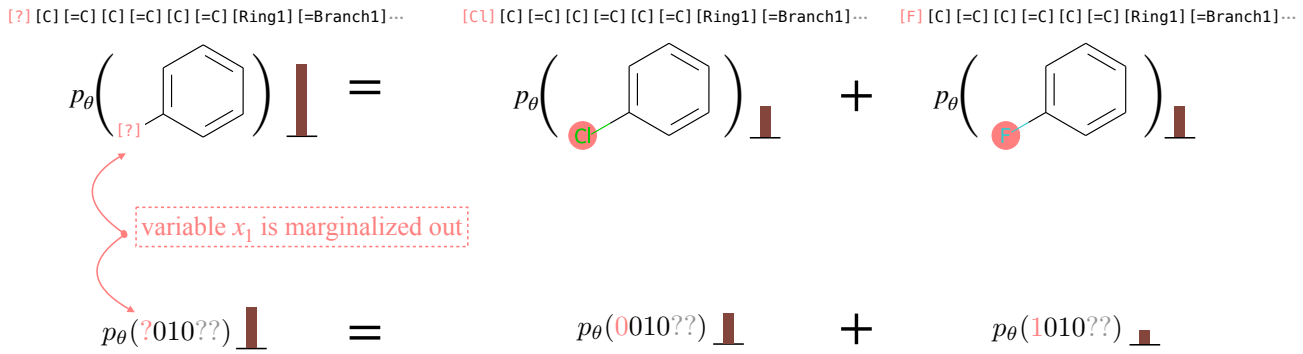


Figure 2. Marginalization models (MAMs) enable estimation of any marginal probability with a neural network θ that learns to "marginalize out" variables. The figure illustrates marginalization of a single variable on bit strings (representing molecules) with two alternatives (versus K in general) for clarity. The bars represent probability masses.

likelihoods on long sequences arising in data such as natural language and proteins. In addition to MLE, the setting of *energy-based training* (EB) has recently received growing interest with its applications in science domains [49, 12, 35]. Instead of empirical data samples, we only have access to an unnormalized (log) probability function (specified by a reward or energy function) that can be evaluated pointwise for the generative model to match. In such settings, ARMs are limited to fixed-order generative modeling and lack scalability in training. The subsampling techniques developed to scale the training of conditionals for MLE are no longer applicable when matching log probabilities in energy-based training (see Section 4.3 for details).

To enhance scalability and flexibility in the generative modeling of discrete data, we propose a new family of generative models, **marginalization models** (MAMs), that directly model the marginal distribution $p(\mathbf{x}_S)$ for any subset of variables $\mathbf{x}_S$ in $\mathbf{x}$. Direct access to marginals has two important advantages: 1) *significantly speeding up inference for any marginal*, and 2) *enabling scalable training of any-order generative models under both MLE and EB settings*.

The unique structure of the model allows it to simultaneously represent the coupled collection of all marginal distributions of a given discrete joint probability mass function. For the model to be valid, it must be consistent with the sum rule of probability, a condition we refer to as "*marginalization self-consistency*" (see Figure 2); learning to enforce this with scalable training objectives is one of the key contributions of this work.

We show that MAMs can be trained under both maximum likelihood and energy-based training settings with scalable learning objectives. We demonstrate the effectiveness of MAMs in both settings on a variety of discrete data distributions, including binary images, text, physical systems, and molecules. We empirically show that MAMs achieve orders of magnitude speedup in marginal likelihood evaluation. For energy-based training, MAMs are able to scale training of any-order generative models to high-dimensional problems that previous methods fail to achieve.

2. Background

We first review two prevalent settings for training a generative model: *maximum likelihood estimation* and *energy-based training*. Then we introduce autoregressive models.

Maximum likelihood (MLE) Given a dataset $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ drawn i.i.d. from a data distribution $p = p_{\text{data}}$, we aim to learn the distribution $p_\theta(\mathbf{x})$ via maximum likelihood estimation:

$$\max_\theta \ \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log p_\theta(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^N \log p_\theta(\mathbf{x}^{(i)}) \quad (1)$$

which is equivalent to minimizing the Kullback-Leibler divergence under the empirical distribution, i.e., minimizing $D_{\text{KL}}(p_{\text{data}}(\mathbf{x})||p_\theta(\mathbf{x}))$. This is the setting that is most commonly used in generation of images (e.g., diffusion models [69, 22, 70]) and language (e.g. GPT [58]) where we can easily draw observed data from the distribution.

Energy-based training (EB) In other cases, data from the distribution are not always available. Instead, we have access to an unnormalized probability distribution $f(\cdot)$ typically specified as $f(\mathbf{x}) = \exp(r(\mathbf{x})/\tau)$ where $r(\mathbf{x})$ is an energy (or reward) function and $\tau > 0$ is a temperature parameter. In this setting, the objective is to match $p_\theta(\mathbf{x})$ to $f(\mathbf{x})/Z$, where Z is the normalization constant of f. This can be done by minimizing the KL divergence [49, 84, 12],

$$\min_\theta \ D_{\text{KL}}\left(p_\theta(\mathbf{x}) \| \frac{f(\mathbf{x})}{Z}\right) \quad (2)$$

The reward function $r(\mathbf{x})$ can be defined either by human preferences or by the physical system from first principles. For example, (a) In aligning large language models, $r(\mathbf{x})$

can represent human preferences [51, 50]; (b) In molecular/material design, it can specify the proximity of a sample's measured or calculated properties to some functional desiderata [2]; and (c) In modeling the thermodynamic equilibrium ensemble of physical systems, it is the (negative) energy function of a given sample [49, 84, 12, 35].

The training objective in Equation (2) can be optimized using a Monte Carlo estimate of the gradient using the REINFORCE algorithm [83]. A generative model θ allows us to efficiently generate samples approximately from the distribution, which would otherwise be much more expensive via running MCMC with the energy function $f(\cdot)$.

Autoregressive models Autoregressive models (ARMs) [3, 38] model a complex high-dimensional distribution $p(\mathbf{x})$ by factorizing it into univariate conditionals using the chain rule:

$$\log p_\phi(\mathbf{x}) = \sum_{d=1}^{D} \log p_\phi\left(x_d \mid \mathbf{x}_{<d}\right), \tag{3}$$

where $\mathbf{x}_{<d} = \{x_1, \ldots, x_{d-1}\}$. ARMs generate examples by sequentially drawing x_1 under $p_\phi(x_1)$, then x_2 under $p_\phi(x_2|x_1)$, and so on. The ARM approach has produced successful discrete-data neural models for natural language, proteins [68, 40, 44], and molecules [66, 19].

Any-order ARMs (AO-ARMs) Uria et al. [78] propose to learn the conditionals of ARMs for arbitrary orderings that include all permutations of $\{1, \ldots, D\}$. Under the MLE setting, the model ϕ is trained by maximizing a lower-bound objective [78, 24] using an expectation under the uniform distribution of orderings. This objective allows scalable training of AO-ARMs with architectures such as the U-Net [62] and Transformers [80], by leveraging efficient parallel evaluation of multiple one-step conditionals for all next-tokens in one forward pass. However, modeling conditionals alone with ARMs results in limitations in both inference and training (more details in Section 4.3):

1. *Test-time marginal likelihood inference*: evaluation of $p_\phi(\mathbf{x})$ or $p_\phi(\mathbf{x}_s)$ requires up to D neural network passes, making it costly for high-dimensional data.

2. *Energy-based training for high-dimensional problems:* the objective in Equation (2) requires evaluating $\log p_\phi(\mathbf{x})$ in full with D network forward passes in order to calculate the difference of $\log p_\phi(\mathbf{x})$ and $f(\mathbf{x})/Z$. Monte Carlo estimate of $\log p_\phi(\mathbf{x})$ no longer works since the objective is matching $\log p$'s instead of maximizing $\log p$ (the MLE case). As a result, this significantly limits ARM's training scalability under the EB setting when D becomes large.

3. Marginalization Models

We propose *marginalization models* (MAMs), a new type of generative model that enables scalable any-order generative modeling on high-dimensional problems as well as efficient marginal evaluation, for both maximum likelihood and energy-based training. The flexibility and scalability of marginalization models are enabled by the explicit modeling of the marginal distribution and scalable training objectives derived from *marginalization self-consistency*.

In this paper, we focus on generative modeling of discrete structures using vectors of discrete variables. The vector representation encompasses various real-world problems with discrete structures, including language sequence modeling, protein design, and molecules with string-based representations (e.g., SMILES [82] or SELFIES [36]). Moreover, vector representations are inherently applicable to any discrete problem, since it is feasible to encode any discrete object into a vector of discrete variables.

Definition Let $p(\mathbf{x})$ be a discrete probability distribution, where $\mathbf{x} = (x_1, \ldots, x_D)$ is a D-dimensional vector and each x_d takes K possible values, i.e. $x_d \in \mathcal{X} \triangleq \{1, \ldots, K\}$.

Marginalization For a subset of indices $\mathcal{S} \subseteq \{1, \ldots, D\}$, let $\mathbf{x}_\mathcal{S}$ and $\mathbf{x}_{\mathcal{S}^c}$ denote the subvectors corresponding to $\mathcal{S}$ and the complement set, $\mathcal{S}^c = \{1, \ldots, D\} \backslash \mathcal{S}$. The marginal of $\mathbf{x}_\mathcal{S}$ is obtained by summing over all values of $\mathbf{x}_{\mathcal{S}^c}$:

$$p(\mathbf{x}_\mathcal{S}) = \sum_{\mathbf{x}_{\mathcal{S}^c}} p(\mathbf{x}_\mathcal{S}, \mathbf{x}_{\mathcal{S}^c}) \tag{4}$$

We refer to (4) as the "*marginalization self-consistency*" that a valid distribution should follow. The goal of a marginalization model θ is to estimate the marginals $p(\mathbf{x}_\mathcal{S})$ for any subset of variables $\mathbf{x}_\mathcal{S}$ as closely as possible. To achieve this, we train a deep neural network that fits $p_\theta(\mathbf{x})$ to a target distribution $p(\mathbf{x})$ while fitting the marginals $p_\theta(\mathbf{x}_\mathcal{S})$ through the marginalization self-consistency principle. In other words, MAM learns to approximately inference the marginals of an arbitrary subset of variables with a single neural net forward pass.[1]

Parameterization A marginalization model parameterized by a neural network θ takes in $\mathbf{x}_\mathcal{S}$ and outputs the marginal log probability $f_\theta(\mathbf{x}_\mathcal{S}) = \log p_\theta(\mathbf{x}_\mathcal{S})$. Note that for different subsets $\mathcal{S}$ and $\mathcal{S}'$, $\mathbf{x}_\mathcal{S}$ and $\mathbf{x}'_\mathcal{S}$ lie in different vector spaces. To unify the vector space that is fed into the NN, we introduce an augmented vector space that additionally includes the "marginalized out" variables $\mathbf{x}_{\mathcal{S}^c}$. By defining a special symbol "△" to denote the missing values of the "marginalized out" variables, the augmented vector representation is D-dimensional and is defined to

[1] Estimating $p(\mathbf{x})$ is a special case of marginal inference where there are no variables to be marginalized.

be: $\mathbf{x}_{\mathcal{S}}^{\text{aug}}(i) = \begin{cases} x_i, & \text{if } i \in \mathcal{S} \\ \triangle, & \text{otherwise} \end{cases}$. Now, the augmented vector representation $\mathbf{x}_{\mathcal{S}}^{\text{aug}}$ of all possible $\mathbf{x}_{\mathcal{S}}$'s has the same dimension D, and for any i-th dimension $\mathbf{x}_{\mathcal{S}}^{\text{aug}}(i) \in \mathcal{X}^{\text{aug}} \triangleq \{1, \cdots, K, \triangle\}$. To given an example, when $D = 4$ and $\mathcal{X} = \{0, 1\}$, for $\mathbf{x}_{\mathcal{S}} = \{x_1, x_3\}$ taking values $x_1 = 0$ and $x_3 = 1$, $\mathbf{x}_{\mathcal{S}}^{\text{aug}} = (0, \triangle, 1, \triangle)$, with the corresponding marginal $p(\mathbf{x}_{\mathcal{S}}^{\text{aug}}) = \sum_{x_2} \sum_{x_4} p(0, x_2, 1, x_4)$. From here onwards we will use $\mathbf{x}_{\mathcal{S}}^{\text{aug}}$ and $\mathbf{x}_{\mathcal{S}}$ interchangeably.

Sampling With the marginalization model, one can sample from the learned distribution by picking an arbitrary order and sampling one variable or multiple variables at a time. In this paper, we focus on the sampling procedure that generates one variable at a time. Sampling multiple variables jointly can also be done in a similar way (see Appendix B.2 for ablation studies). To get the conditionals at each step for generation, we can use the product rule of probability: $p_\theta(x_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) = p_\theta(\mathbf{x}_{\sigma(\leq d)})/p_\theta(\mathbf{x}_{\sigma(<d)})$. However, the above conditional distribution is not exactly valid when the following single-step marginalization consistency in (5) is only approximately enforced,

$$p_\theta(\mathbf{x}_{\sigma(<d)}) \approx \sum_{x_{\sigma(d)}} p_\theta(\mathbf{x}_{\sigma(\leq d)}), \quad \forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D], \tag{5}$$

since the estimated probabilities might not sum up exactly to one. Hence we use following normalized conditional:

$$p_\theta(x_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) = \frac{p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}])}{\sum_{x_{\sigma(d)}} p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}])}. \tag{6}$$

Scalable training of marginalization self-consistency In training, we can impose the marginalization self-consistency by minimizing the *squared error* of the constraints in (5) in log-space. Evaluation of each marginalization constraint in (5) requires K NN forward passes, where K is the number of discrete values x_d can take. This makes mini-batch training challenging to scale when K is large. To address this issue, we augment the marginalization models with learnable conditionals parameterized by another neural network ϕ. The marginalization constraints in (5) can be further decomposed into K parallel marginalization constraints[2].

$$p_\theta(\mathbf{x}_{\sigma(<d)}) p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) \approx p_\theta(\mathbf{x}_{\sigma(\leq d)}), \quad \forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D]. \tag{7}$$

The consistency error for each constraint can be defined correspondingly as follows:

$$\text{ConsistencyError}(\mathbf{x}, \sigma, d) = \left[\log\left(p_\theta(\mathbf{x}_{\sigma(<d)}) p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)})\right) - \log p_\theta(\mathbf{x}_{\sigma(\leq d)})\right]^2.$$

[2]To make sure p_θ is normalized, we can either additionally enforce $p_\theta((\triangle, \cdots, \triangle)) = 1$ or let $Z_\theta = p_\theta((\triangle, \cdots, \triangle))$ be the normalization constant.

Other distances such as KL divergence can also be considered. We choose squared distance for its flexibility in selecting the $q(\mathbf{x})$, allowing us to fit marginals for various use cases with different $q(\mathbf{x})$ at test time. It's also worth noting that training with KL divergence and squared distance are quite similar (see Malkin et al. [47]). The REINFORCE gradient of $D_{\text{KL}}\left(p_\theta(\mathbf{x}_{<\sigma(d)}) p_\phi(x_{\sigma(d)}|\mathbf{x}_{<\sigma(d)}) \parallel p_\theta(\mathbf{x}_{\leq\sigma(d)}).\text{detach}()\right)$ is equivalent to the squared distance loss when q is set to p_θ.

By breaking the original marginalization self-consistency in Equation (4) into highly parallel marginalization self-consistency in Equation (7), we introduce a total of $K^D \times D! \times D \times K$ constraints. Although this increases the number of constraints, it becomes *highly scalable* to train on the marginalization self-consistency via randomly sampling constraints following a specified distribution $q(\mathbf{x})$ and $q(\sigma)$. In our experiments, $q(\sigma)$ is set to the uniform distribution over all orderings $\mathcal{U}(S_D)$ and $q(\mathbf{x})$ is set to the data distribution of interest for marginal inference, such as the empirical data distribution $p_{\text{data}}(\mathbf{x})$ or the generative model's distribution $p_{\theta,\phi}(\mathbf{x})$. We found that a training objective that decomposes into highly parallel terms for sampling is key to effectively fitting marginals with scalability.

4. Training the Marginalization Models

4.1. Maximum Likelihood Estimation Training

In this setting, we train MAMs with the maximum likelihood objective while additionally enforcing the marginalization constraints in Equation (5):

$$\max_{\theta,\phi} \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log p_\theta(\mathbf{x}) \tag{8}$$
$$\text{s.t.} \quad p_\theta(\mathbf{x}_{\sigma(<d)}) p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) \approx p_\theta(\mathbf{x}_{\sigma(\leq d)}), \quad \forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D].$$

Two-stage training A typical way to solve the above optimization problem is to convert the marginalization constraint into another objective and optimize both objectives jointly. However, maximizing $\log p_\theta(\mathbf{x}_{\leq D})$ directly in Equation (8) is unbounded, we empirically found this causes the training to be slow and unstable by over-emphasizing likelihood at the expense of self-consistency. Instead, we identify an theoretically equivalent two-stage optimization formulation that leads to more effective training strategy based on the following observation:

Proposition 1. *Solving the optimization problem in* (8) *is equivalent to the following two-stage optimization procedure, under mild assumptions about the neural networks*

used being universal approximators:

$$\textbf{Stage 1:} \max_{\phi} \mathbb{E}_{\mathbf{x}\sim p_{data}} \mathbb{E}_{\sigma} \sum_{d=1}^{D} \log p_{\phi}\left(x_{\sigma(d)} | \mathbf{x}_{\sigma(<d)}\right)$$

$$\textbf{Stage 2:} \min_{\theta} \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})} \mathbb{E}_{\sigma} \mathbb{E}_{d} \textit{ ConsistencyError}(\mathbf{x}, \sigma, d)$$

where $\sigma \sim \mathcal{U}(S_D)$, $d \sim \mathcal{U}(1, \cdots, D)$ and $q(\mathbf{x})$ is the distribution of interest for marginal likelihood inference.

The first stage can be interpreted as *fitting the conditionals* in the same way as AO-ARMs [78, 24] and the second stage acts as *distilling the marginals* from conditionals through training on *marginalization self-consistency*. The intuition comes from the simple chain rule of probability: we first observe a one-to-one correspondence between the optimal conditionals $\log p_\phi$ and marginals $\log p_\theta$, i.e. $\log p_\theta(\mathbf{x}) = \sum_{d=1}^{D} \log p_\phi\left(x_{\sigma(d)} | \mathbf{x}_{\sigma(<d)}\right)$ for any σ and $\mathbf{x}$. By assuming neural networks are universal approximators, we can split the joint optimization problem into two steps by first finding the optimal conditionals p_ϕ, and then solving for the corresponding optimal marginals p_θ. We provide proof details in Appendix A.1.

There are two main advantages with the reformulated two-stage training. First, the maximum likelihood objective based on conditionals is now bounded and can be optimized in parallel. Secondly, even when compared with joint training with the reformulated conditional-based likelihoods, the decoupled two-stage training leads to improved efficiency, since it avoids wasted compute on fitting marginals on conditionals that are still being actively updated throughout training. Additionally, the two-stage approach eliminates the need to sweep over the hyperparameter that balances the two objectives. In Appendix B.4, we validate this with experiments by comparing two-stage v.s. joint training, both using the reformulated objectives. This aligns with findings in diffusion model distillation [71, 5], where training a standard diffusion model followed by distillation proves easier than training a distilled model from scratch.

4.2. Energy-based Training

In this setting, the two-stage training introduced in Section 4.1 becomes impractical for high-dimensional problems. Stage 1 training (fitting conditionals with $\mathcal{L}_{\text{KL}} = \mathbb{E}_{p_\theta}\left[\sum_{d=1}^{D} \log p_\phi\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right) - \log p(x)\right]$) scales poorly with D as it requires D NN forward passes per datapoint. Therefore, for scalability, we train MAMs by jointly minimizing the KL divergence objective over the marginals and the self-consistency loss term that include both marginals and conditionals:

$$\min_{\theta,\phi} D_{\text{KL}}(p_\theta \,\|\, p) + \lambda\, \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})} \mathbb{E}_{\sigma} \mathbb{E}_{d} \text{ConsistencyError}(\mathbf{x}, \sigma, d),$$

where $\sigma \sim \mathcal{U}(S_D)$, $d \sim \mathcal{U}(1, \cdots, D)$ and $q(\mathbf{x})$ is the distribution of interest for marginal likelihood inference.

Unlike the unbounded likelihood maximization in Section 4.1, matching $\log p_\theta(\mathbf{x})$ with $\log p(\mathbf{x})$ in the KL term does not lead to training instability issues. However, joint training introduces complex dynamics, necessitating careful hyperparameter selection. We find that a wide range of small λ yield best performance. More experiments and discussion are provided in Section B.4.

Scalable training The gradient of the KL divergence term is estimated with the REINFORCE estimator [83]:

$$\begin{aligned} &\nabla_\theta D_{\text{KL}}(p_\theta(\mathbf{x}) || p(\mathbf{x})) \\ =&\mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x}) \left(\log p_\theta(\mathbf{x}) - \log f(\mathbf{x})\right)\right] \\ \approx&\frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log p_\theta(\mathbf{x}^{(i)}) \left(\log p_\theta(\mathbf{x}^{(i)}) - \log f(\mathbf{x}^{(i)})\right) \end{aligned} \tag{9}$$

The consistency-error term can be estimated by randomly sampling data $\mathbf{x}$, ordering σ and step d from the specified distribution.

Efficient sampling with persistent MCMC To efficiently generate approximate samples of p_θ for the REINFORCE estimator, a persistent set of Markov chains are maintained by taking block-wise Gibbs sampling steps following a random ordering using the conditional distribution $p_\phi(\mathbf{x}_{\sigma(d)} | \mathbf{x}_{\sigma(<d)})$ (full algorithm in Appendix A.3), in a similar fashion to persistent contrastive divergence [76]. The samples from the conditional network p_ϕ serve as good approximation to samples from the marginal network p_θ, since they are close to each other when conditionals and marginals are approximately consistent with each other. In experiments, we validate this by observing that the log-probabilities from p_θ and p_ϕ are highly consistent on both random and on-policy samples. In cases when there is a strong discrepancy between p_θ and p_ϕ during training, we can additionally use importance sampling to get an unbiased estimate.

4.3. Addressing Limitations of ARMs

1) **Test-time marginal likelihood inference** Evaluation of a marginal $p_\phi(\mathbf{x}_o)$ with ARMs (or an arbitrary marginal with AO-ARMs) requires applying the conditional p_ϕ up to D times, which is inefficient in time and memory for high-dimensional data. In contrast, MAMs can approximate any arbitrary marginal with just one NN forward pass. This is achieved through explicitly modeling the marginals and training with scalable self-consistency objectives.

2) **EB training for high-dimensional problems** There are two factors that limit the scalability of ARMs for EB training. First, the KL divergence objective in EB training requires evaluating $\log p_\phi(\mathbf{x})$ in full with D network forward passes in order to calculate the difference of $\log p_\phi(\mathbf{x})$ and $f(\mathbf{x})/Z$. One might consider estimating $p_\phi(\mathbf{x})$ with a single-step Monte Carlo estimate $p_\phi(x_{\sigma(d)} | \mathbf{x}_{\sigma(<d)})$, but this leads to high variance of the REINFORCE gradient in Equation (9) due to the product of the score function and

distance terms, which are both of high variance (validated in experiments, see Figure 3). Consequently, training ARMs for energy-based training necessitates a sequence of D conditional evaluations to compute the gradient of the objective function. This constraint leads to an effective batch size of $B \times D$ for batch of B samples, significantly limiting the training scalability of ARMs to high-dimensional problems.

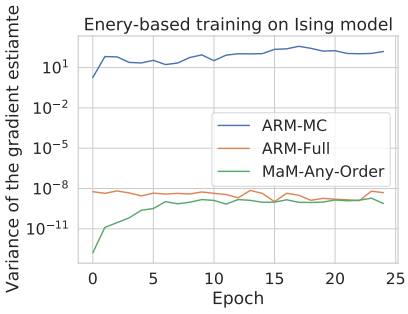


Figure 3. Approximating $\log p_\phi(\mathbf{x})$ with one-step conditional (ARM-MC) results in extremely high gradient variance in energy-based training.

MAMs circumvent the first limiting factor by breaking down the original distribution matching problem into two sub-problems: 1) minimizing the KL divergence between the model's marginal probability estimate $p_\theta(\mathbf{x})$ and the energy function $f(\mathbf{x})$, and 2) ensuring marginals $\log p_\theta$ and conditionals $\log p_\phi$ are self-consistent. The first sub-problem requires evaluating the marginal probability $p_\theta(\mathbf{x})$ with just one network forward pass for each $\mathbf{x}$ sample. The training objective for the second sub-problem is also scalable via simply sampling the highly parallel self-consistency error objectives developed in Equation (7).

The other limiting factor is associated with obtaining Monte Carlo samples for the REINFORCE gradient estimator. Previous methods that use ARMs for energy-based training [84, 12] assume a fixed ordering and require D sequential sampling steps to generate samples, which is slow and costly when the dimension is large. MAM proposes a more cost-effective sampling procedures through the utilization of persistent block-wise Gibbs sampling.

5. Related Work

Autoregressive models Developments in deep learning have greatly advanced the performance of ARMs across different modalities, including images, audio, and text. Any-order (Order-agnostic) ARMs were first introduced in [78] by training with the any-order lower-bound objective for the maximum likelihood setting. Recent work, ARDM [24], demonstrates state-of-the-art performance for any-order discrete modeling of image/text/audio. Germain et al. [20] train an auto-encoder with masking that outputs the sequence of all one-step conditionals for a given ordering, but does not perform as well as methods [79, 85, 24] that predict one-step conditionals under the given masking. Douglas et al. [18] train an AO-ARM as a proposal distribution and uses importance sampling to estimate arbitrary conditional probabilities in a DAG-structured Bayesian network, but with limited experiment validation on a synthetic dataset. Shih et al. [67] utilizes a modified training objective of ARMs for better marginal inference performance but loses any-order generation capability. In the energy-based training setting, ARMs are applied to science problems [12, 84], but suffer in scaling to when D is large. MAMs and ARMs are compared in detail in Section 4.3.

Arbitrary conditional/marginal models For continuous data, VAEAC [29] and ACFlow [39] extend conditional variational encoder and normalizing flow to arbitrary conditional modeling. ACE [73] improves the expressiveness of arbitrary conditional models by directly modeling the energy function, which reduces the constraints on parameterization but increases computation costs due to the need to approximate the normalizing constant. Instead of using neural networks as function approximators, probabilistic circuits (PCs) [9, 13, 56, 8, 54] offer tractable probabilistic models for both conditionals and marginals by building a computation graph with sum and product operations following specific structural constraints. Peharz et al. [54] improved the scalability of PCs by combining arithmetic operations into a single monolithic einsum-operation and automatic differentiation. Further improvements of PCs are achieved through distilling latent variables from pre-trained deep generative models [41, 42] . All methods mentioned above focus on MLE settings. MAMs focus on scalable approximate marginal inference using neural networks as function approximators on both MLE and EB settings.

GFlowNets GFlowNets [2, 4] formulate the problem of generation as matching the probability flow at terminal states to the target normalized density. Compared to ARMs, GFlowNets allow flexible modeling of the generation process by assuming learnable generation paths through a directed acyclic graph (DAG). The advantages of learnable generation paths come with the trade-off of sacrificing the flexibility of any-order generation and exact likelihood evaluation. Under a fixed generation path, GFlowNets reduce to fixed-order ARMs [87]. In Appendix A.4, we further discuss the connections and differences between GFlowNets and AO-ARMs/MAMs. For discrete problems, Zhang et al. [88] train GFlowNets on the squared distance loss with the trajectory balance objective [46]. This is not scalable for large D (for the same reason as ARMs in Section 4.3) and doesn't provide direct access to marginals. In the MLE setting, an energy function is additionally learned from data so that the model can be trained with energy-based training.

6. Experiments

We evaluate marginalization models (MAM) on both MLE and EB settings for discrete problems including images, text, molecules and phyiscal systems. We compare MAMs with baselines that support arbitrary marginal inference[3]: Any-order ARM$^\blacklozenge$ [24], ARM$^\lozenge$ [38], Parallel Any-order ARMs (P-AO-ARM) [24] and Probabilistic Circuit (PC)$^\blacklozenge$ [54]. We also include state-of-the-art generative models on various tasks for comparison: GFlowNet [88], Discrete Flow [77], PixelCNN++ [64], Variational Diffusion Models [33], Sparse Transformers [7, 32] and D3PM [1]. We follow training settings or results from the literature for all baselines. In Appendix B.1, we present additional studies on measuring the marginal self-consistency with a carefully curated synthetic experiment. Neural network architecture and training hyperparameter details are given in Appendix C.

6.1. Maximum Likelihood Estimation Training

We focus on three metrics: test data negative log likelihood (NLL), marginal inference time and marginal inference quality. The later two are only available with baselines that support arbitrary marginal inference. The marginals are evaluated on a randomly sampled mini-batch data of the test set (batch size $= 128$, metrics are averaged over batches). To evaluate marginal estimation quality, the marginal estimates of each model are compared with the marginal estimates of the best-performing model (in terms of NLL). Pearson correlation is reported to measure the quality of marginal likelihoods[4]. (1.0 means a perfect linear correlation with the best model's marginal estimates.) For evaluating NLL, the conditional network and marginal network perform similarly in ablation studies (see Appendix B.2). We use the conditional network for evaluating NLL. The marginal network is used for evaluating marginals.

Image We evaluate MAMs on Binary MNIST [63], CIFAR-10 [37] and ImageNet32 [14, 10]. The image dimension is $1 \times 28 \times 28$ for MNIST and $3 \times 32 \times 32$ for CIFAR-10 and ImageNet32. MAMs achieve competitive NLL on all tasks, equaling the best-performing arbitrary marginal inference models. In terms of marginal inference, MAM produces *high quality* marginal estimates while achieving close to 4 *orders of magnitude speed-up* in computation time. The Pearson correlation coefficients are close to 1.0, which means the marginal estimates are consistent with the best marginal estimates. It can also be interpreted as a measure of marginalization self-consistency, since the the marginals of MAM are evaluated against the same conditionals of AO-ARM and MAM.

Table 1. Pixel modeling on Binary-MNIST

	NLL (bpd) ↓	Pearson ↑	Time (s) ↓
GflowNet [88]	0.189	–	–
AO-ARM$^\blacklozenge$ [24]	0.146	0.99	132.4 ± 0.03
PC (EiNets)$^\blacklozenge$ [54]	0.187	0.75	0.015 ± 0.00
MAM$^\blacklozenge$	0.146	0.99	0.018 ± 0.00

Table 2. Pixel modeling on CIFAR-10

	NLL (bpd) ↓	Pearson ↑	Time (s) ↓
D3PM [1]	3.44	–	–
PixelCNN++ [64]	2.99	–	–
VDM [33]	2.49	–	–
Sparse Transformer [7, 32]	2.56	–	–
PC (LVD-PG)$^\blacklozenge$ [42]	3.87	–	–
AO-ARM$^\blacklozenge$ (800 epochs)	2.88	0.99	2401 ± 1
MAM$^\blacklozenge$ (800 epochs)	2.88	0.98	0.495 ± 0.00

Molecule We evaluate MAM on the molecular generation benchmark MOSES [55] refined from the ZINC database [72]. The generated molecules from MAM and AO-ARM are comparable to standard state-of-the-art molecular generative models, such as CharRNN [66], JTN-VAE [30], and LatentGAN [57] (see Tables 10 and 11), with added controllability and flexibility in any-order generation. MAM supports much much faster marginal inference, which is useful for domain scientists to easily reason about the likelihood of (sub)structures. Generated molecules and property histogram plots of are available in Appendix C.4.

Text We train a character-level generative model on Text8 [45], which consists of 100M characters from Wikipedia in chunks of 250 character. MAM achieves significant speed-up in marginal inference while maintaining comparable performance as an arbitrary marginal inference model. The test NLL is close to a Transformer model that is trained to only model one ordering (from left to right).

6.2. Energy-based Training

In the existing literature, only ARM with fixed variable order has been used for this training setting (for example in Wu et al. [84], Damewood et al. [12]). We additionally implement two more baselines: ARM-MC that uses single-step conditional as a Monte Carlo estimate to $\log p_\phi$ and GFlowNet [46]. The effective batch size for ARM and GFlowNet is $B \times D$ for batch of B data samples (due to reasons mentioned in Section 4.3), and $B \times 1$ for ARM-MC and MAM . MAM and ARM use the REINFORCE gradient estimator with baseline. GFlowNet is trained on per-sample gradient of squared distance [88]. Note that MAM is an any-

[3]We use $\blacklozenge$ to denote that the model supports arbitrary marginal inference. $\lozenge$ is used for ARMs with fixed ordering since they only partially support arbitrary marginal inference.

[4]When measuring AO-ARM against itself, two random orderings are measured against each other.

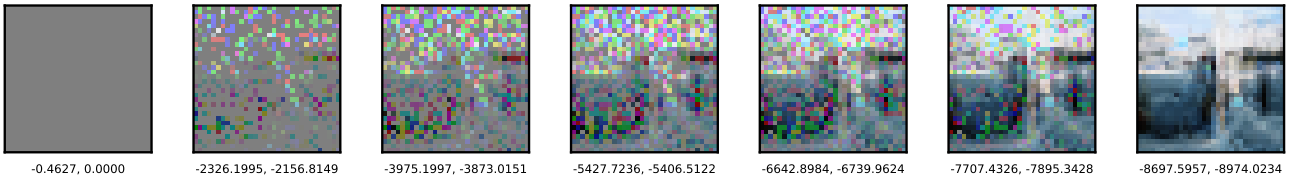


Figure 4. An example of the marginal estimates of an ImageNet32 image along the generation trajectory using a random ordering. The numbers in the captions show that the learned (log) marginals (left) v.s. learned (log) conditionals (right) are approximately self-consistent.

Table 3. Pixel modeling on ImageNet32

	NLL (bpd) ↓	Pearson ↑	Time (s) ↓
Image Transformer [53]	3.77	–	–
VDM [33]	3.72	–	–
PC (LVD-PG)♦ [42]	4.06	–	–
AO-ARM♦ (16 epochs)	3.60	0.99	4995 ± 1
MAM♦ (16 epochs)	3.60	0.98	1.243 ± 0.00

Table 4. Character modeling on text8

	NLL (bpc) ↓	Pearson ↑	Time (s) ↓
D3PM [1]	1.47	–	–
Discrete Flow [77]	1.23	–	–
Transformer [80]	1.35	–	–
AO-ARM♦ (3000 epochs)	1.48	0.987	41.40 ± 0.01
MAM♦ (3000 epochs)	1.48	0.945	0.005 ± 0.00

order generative model, which is a more difficult learning task than ARM that uses fixed ordering and GFlowNet that uses learned ordering.

Table 5. Energy-based modeling of Ising model ($D = 100$)

	NLL (bpd) ↓	KL div. ↓	Time (s) ↓
ARM-One-Order◇ [12]	0.79	-78.63	5.3±0.1e-01
ARM-MC-One-Order◇	24.84	-18.01	5.3±0.1e-01
GFlowNet [88]	0.78	-78.17	–
MAM-Any-Order♦	0.80	-77.77	3.7±0.1e-04

Table 6. Energy-based modeling of Ising model ($D = 900$)

	NLL (bpd) ↓	KL div. ↓	Time (s) ↓
ARM-One-Order◇ [12]	– Out of GPU memory –		
Random Samples	1.00	-623.9	–
MAM-Any-Order♦	0.83	-685.8	3.7±0.1e-04

Physical systems Ising models [28] model interacting spins on a square lattice and are widely studied in mathematics and physics (see MacKay [43]). The spins of the D sites are represented by a D-dimensional binary vector $\mathbf{x}$, whose distribution $p^*(\mathbf{x}) \propto \exp\left(-E_{\mathbf{J}}(\mathbf{x})\right)$ is determined by the energy function $E_{\mathbf{J}}(\mathbf{x}) \triangleq -\mathbf{x}^\top \mathbf{J}\mathbf{x} - \boldsymbol{\theta}^\top \mathbf{x}$, with $\mathbf{J}$

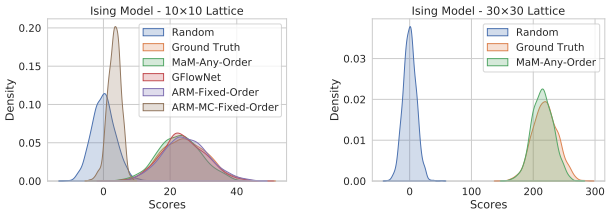


Figure 5. Ising model. Left: $D = 10 \times 10$. Right: $D = 30 \times 30$

being the adjacency matrix. We compare MAM with ARM, ARM-MC, and GFlowNet on a 10×10 ($D = 100$) and a larger 30×30 ($D = 900$) Ising model where ARMs and GFlowNets fail to scale. We generate 2000 ground truth samples following Grathwohl et al. [21] and we measure test negative log-likelihood on those samples. We also measure $D_{\mathrm{KL}}(p_\theta(\mathbf{x})||p^*)$ by sampling from the learned model and evaluating $\sum_{i=1}^{M}(\log p_\theta(\mathbf{x}_i) - \log f^*(\mathbf{x}_i))$. Figure 5 contains KDE plots of $-E_{\mathbf{J}}(\mathbf{x})$ for the generated samples. We validate the analysis in Section 4.3, the ARM-MC gradient has high variance which leads to non-convergence or mode collapse. MAM achieves significant speedup in marginal inference and is the only model that supports any-order generative modeling. The performance in terms of KL divergence and likelihood are only slightly worse than models with fixed/learned order, which is expected since any-order modeling is harder than fixed-order modeling, and MAM is solving a more complicated task of jointly learning conditionals and marginals. On a 30×30 ($D = 900$) Ising model, MAM achieves a bpd of 0.835 while ARM and GFlowNet fail to fit in the GPU memory (see Figure 5 and Table 6).

Molecular generation with target property In this task, we are interested in training generative models towards a specific target property of interest $g(x)$, such as lipophilicity (logP), synthetic accessibility (SA), etc. We define the distribution of molecules to follow $p^*(x) \propto \exp(-(g(x) - g^*)^2/\tau)$, where g^* is the target value of the property and τ is a temperature parameter. We train ARM and MAM for lipophilicity of target values 4.0 and -4.0, both with $\tau = 1.0$ and $\tau = 0.1$. Both models are trained for 4000 iterations with batch size 512. Results are shown in Figure 6 (additional results in Appendix C).

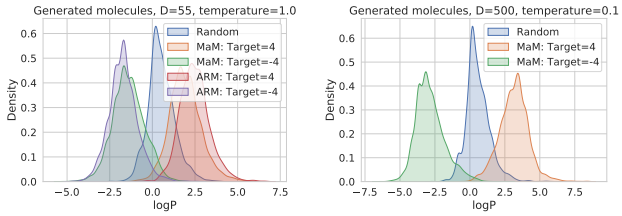


Figure 6. EB molecule targeted generation. Left: 55d. Right: 500d

Findings are consistent with the Ising model experiments. Again, MAM performs just marginally below ARM. However, only MAM supports any-order modeling and scales to high-dimensional problems. Figure 6 (right) shows molecular generation with MAM for $D = 500$.

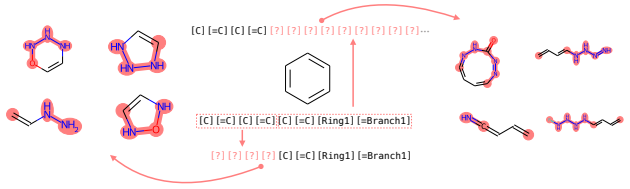


Figure 7. Illustration of conditional design of molecules towards low lipophilicity from a user-defined substructure in a given order. Left: Impainting the left 4 characters. Right: Impainting the right 4-20 characters. Shaded regions denote the impainted structures.

6.3. Comparison with Parallel AO-ARMs

Inference of AO-ARMs can be parallelized with fewer steps using dynamic programming at cost of minimal log-likelihood degradation, which make it a strong baseline for accelerated inference as shown in Hoogeboom et al. [24]. In Figure 8, we compare the quality of MAM against P-AO-ARM (PARM) with varying number of sampling steps T. MAM is consistently faster and produces better-correlated marginal estimates than PARM. PARM's effectiveness varies across datasets. Text and molecule data require more steps of PARM for accurate estimation due to their sequential dependencies. Interestingly, ImageNet32 needs much fewer PARM steps for correlated log-likelihoods (despite values being quite off), suggesting easier parallelization of sampling/inference once some pixels are filled.

6.4. Out-of-distribution Robustness

The marginal estimates from MAM are not perfectly-normalized but only approximate log-likelihood values. Hence we test the how useful and robust those approximate marginals are in real-world use cases which are often out-of-distribution with various degrees.

We tested MAM's marginal estimates on generated "synthetic" data from masked MNIST (Appendix C.3.3) and Text8 examples (Appendix C.5). MAM $\log p$ estimates maintain a high correlation with actual log-likelihoods on data that are on-manifold but slightly out-of-distribution. In Appendix C.5.1, we also tested the model's generalizability for length extrapolation on Text8. The model is trained on $D = 250$ and tested on sequences with $D = 300$ from the same dataset. MAM's predicted $\log p$ marginals generalize gracefully to longer sequences. The quality matches Parallel AO-ARM with $10 \sim 20$ steps while using significantly less time ($2000\times$). Finally, in Appendix C.4,we tested the model on a more challenging task: using MAM model's marginal likelihood estimates trained on Molecular Sets (a general chemical space of drug-like compounds) to distinguish between two focused chemical spaces (tyrosine kinase inhibitors and organic photodiodes) that are not seen during training. We created 1000 pairs consisting of one of each using datasets from Subramanian et al. [74], controlling for other factors like SMILES length and chemical space to increase difficulty. MAM marginals correctly identified the drug molecule 74% of the time (v.s. 79% for AO-ARM), with 90% alignment on marginal estimates with AO-ARM.

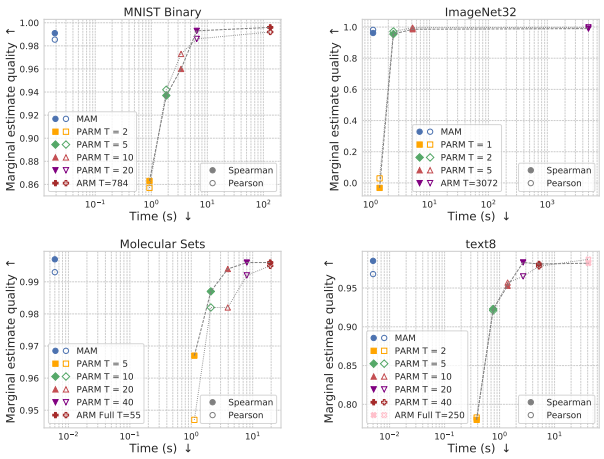


Figure 8. Comparison with Parallel AO-ARMs.

7. Conclusion

In conclusion, marginalization models are a novel family of generative models for high-dimensional discrete data that offer scalable and flexible generative modeling. These models explicitly model all induced marginal distributions, allowing for fast evaluation of arbitrary marginal probabilities with a single neural net forward pass. MAMs also support scalable training objectives for any-order generative modeling, which previous methods struggle to achieve under the energy-based training context. Potential future work includes designing novel neural network architectures that automatically satisfy the marginalization self-consistency.

Acknowledgments

We thank members of the Princeton Laboratory for Intelligent Probabilistic Systems and anonymous reviewers for valuable discussions and feedback. We thank Andrew Novick and Eric Toberer for valuable discussions on energy-based training in scientific applications. We thank Akshay Subramanian and Soojung Yang for valuable discussions on designing the drug-or-photodiode out-of-distribution evaluation task. This work was partially supported by NSF grants IIS-2007278 and OAC-2118201.

Impact Statement

As a deep learning model, MAM has the risk of low robustness on data from unseen domain or manifold. In practice, one should not blindly apply it to data that is far away from the training data distribution and expect the marginal likelihood estimate can be trusted. For the same reason, MAM will also be susceptible to adversarial attacks just as other commonly deep learning models.

MaM enables training of a new type of generative model. Access to fast marginal likelihood is helpful for many downstream tasks such as outlier detection, protein/molecule design or screening. By allowing the training of order-agnostic discrete generative models scalable for distribution matching, it enhances the flexibility and controllability of generation towards a target distribution. This also poses the potential risk of deliberate misuse, leading to the generation of content/designs/materials that could cause harm to individuals.

References

[1] Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. (pages 7, 8, and 17)

[2] Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021. (pages 3, 6, and 17)

[3] Bengio, S. and Bengio, Y. Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3): 550–557, 2000. (pages 1 and 3)

[4] Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023. (pages 6, 16, and 17)

[5] Berthelot, D., Autef, A., Lin, J., Yap, D. A., Zhai, S., Hu, S., Zheng, D., Talbott, W., and Gu, E. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023. (page 5)

[6] Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015. (page 21)

[7] Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. (page 7)

[8] Choi, Y., Vergari, A., and Van den Broeck, G. Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA. URL: http://starai. cs. ucla. edu/papers/ProbCirc20. pdf*, 2020. (pages 6 and 17)

[9] Chow, C. and Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968. (page 6)

[10] Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. (pages 7 and 22)

[11] Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. (page 15)

[12] Damewood, J., Schwalbe-Koda, D., and Gómez-Bombarelli, R. Sampling lattices in semi-grand canonical ensemble with autoregressive machine learning. *npj Computational Materials*, 8(1):61, 2022. (pages 2, 3, 6, 7, and 8)

[13] Darwiche, A. A differential approach to inference in Bayesian networks. *Journal of the ACM (JACM)*, 50 (3):280–305, 2003. (page 6)

[14] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. (pages 7 and 22)

[15] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. (page 1)

[16] Dinh, L., Krueger, D., and Bengio, Y. NICE: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. (page 17)

[17] Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016. (page 17)

[18] Douglas, L., Zarov, I., Gourgoulias, K., Lucas, C., Hart, C., Baker, A., Sahani, M., Perov, Y., and Johri, S. A universal marginalizer for amortized inference in generative models. *Advances in Approximate Bayesian Inference, NIPS 2017 Workshop*, 2017. (page 6)

[19] Flam-Shepherd, D., Zhu, K., and Aspuru-Guzik, A. Language models can learn complex molecular distributions. *Nature Communications*, 13(1):3293, 2022. (page 3)

[20] Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pp. 881–889. PMLR, 2015. (page 6)

[21] Grathwohl, W., Swersky, K., Hashemi, M., Duvenaud, D., and Maddison, C. Oops I took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, pp. 3831–3841. PMLR, 2021. (pages 8 and 22)

[22] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. (page 2)

[23] Hoogeboom, E., Peters, J., Van Den Berg, R., and Welling, M. Integer discrete flows and lossless compression. *Advances in Neural Information Processing Systems*, 32, 2019. (page 17)

[24] Hoogeboom, E., Gritsenko, A. A., Bastings, J., Poole, B., Berg, R. v. d., and Salimans, T. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021. (pages 1, 3, 5, 6, 7, 9, 15, 17, 18, 22, 23, and 24)

[25] Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021. (page 17)

[26] Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. (page 15)

[27] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. (page 15)

[28] Ising, E. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift fur Physik*, 31(1):253–258, February 1925. doi: 10.1007/BF02980577. (page 8)

[29] Ivanov, O., Figurnov, M., and Vetrov, D. Variational autoencoder with arbitrary conditioning. *arXiv preprint arXiv:1806.02382*, 2018. (page 6)

[30] Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018. (page 7)

[31] Johnson, D. D., Austin, J., Berg, R. v. d., and Tarlow, D. Beyond in-place corruption: Insertion and deletion in denoising probabilistic models. *arXiv preprint arXiv:2107.07675*, 2021. (page 17)

[32] Jun, H., Child, R., Chen, M., Schulman, J., Ramesh, A., Radford, A., and Sutskever, I. Distribution augmentation for generative modeling. In *International Conference on Machine Learning*, pp. 5006–5019. PMLR, 2020. (page 7)

[33] Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. (pages 7 and 8)

[34] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 29, 2016. (page 17)

[35] Köhler, J., Invernizzi, M., De Haan, P., and Noé, F. Rigid body flows for sampling molecular crystal structures. *arXiv preprint arXiv:2301.11355*, 2023. (pages 2 and 3)

[36] Krenn, M., Häse, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1 (4):045024, 2020. (pages 3 and 22)

[37] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009. (pages 7 and 22)

[38] Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 29–37. JMLR Workshop and Conference Proceedings, 2011. (pages 1, 3, and 7)

[39] Li, Y., Akbar, S., and Oliva, J. Acflow: Flow models for arbitrary conditional likelihoods. In *International Conference on Machine Learning*, pp. 5831–5841. PMLR, 2020. (page 6)

[40] Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379 (6637):1123–1130, 2023. (page 3)

[41] Liu, A., Zhang, H., and Broeck, G. V. d. Scaling up probabilistic circuits by latent variable distillation. *arXiv preprint arXiv:2210.04398*, 2022. (pages 6 and 17)

[42] Liu, X., Liu, A., Van den Broeck, G., and Liang, Y. Understanding the distillation process from deep generative models to tractable probabilistic circuits. In *International Conference on Machine Learning*, pp. 21825–21838. PMLR, 2023. (pages 6, 7, 8, and 17)

[43] MacKay, D. J. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003. (page 8)

[44] Madani, A., Krause, B., Greene, E. R., Subramanian, S., Mohr, B. P., Holton, J. M., Olmos Jr, J. L., Xiong, C., Sun, Z. Z., Socher, R., et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, pp. 1–8, 2023. (page 3)

[45] Mahoney, M. Large text compression benchmark, 2011. (page 7)

[46] Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022. (pages 6, 7, and 17)

[47] Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*, 2022. (pages 4 and 17)

[48] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305*, 2023. (page 1)

[49] Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), 2019. (pages 2 and 3)

[50] OpenAI. ChatGPT, 2023. URL https://openai.com. (page 3)

[51] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022. (page 3)

[52] Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, 30, 2017. (page 17)

[53] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. Image transformer. In *International conference on machine learning*, pp. 4055–4064. PMLR, 2018. (page 8)

[54] Peharz, R., Lang, S., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Van den Broeck, G., Kersting, K., and Ghahramani, Z. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference on Machine Learning*, pp. 7563–7574. PMLR, 2020. (pages 6, 7, and 17)

[55] Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11:565644, 2020. (pages 7 and 26)

[56] Poon, H. and Domingos, P. M. Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2011. (page 6)

[57] Prykhodko, O., Johansson, S. V., Kotsias, P.-C., Arús-Pous, J., Bjerrum, E. J., Engkvist, O., and Chen, H. A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics*, 11(1):1–13, 2019. (page 7)

[58] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. (page 2)

[59] Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J., and Lakshminarayanan, B. Likelihood ratios for out-of-distribution detection. *Advances in neural information processing systems*, 32, 2019. (page 1)

[60] Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538. PMLR, 2015. (page 17)

[61] Rippel, O. and Adams, R. P. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013. (page 17)

[62] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015. (pages 1 and 3)

[63] Salakhutdinov, R. and Murray, I. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 872–879, 2008. (pages 7 and 21)

[64] Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017. (page 7)

[65] Schneuing, A., Du, Y., Harris, C., Jamasb, A., Igashov, I., Du, W., Blundell, T., Lió, P., Gomes, C., Welling, M., et al. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*, 2022. (page 1)

[66] Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1):120–131, 2018. (pages 3 and 7)

[67] Shih, A., Sadigh, D., and Ermon, S. Training and inference on any-order autoregressive models the right way. *arXiv preprint arXiv:2205.13554*, 2022. (pages 6 and 18)

[68] Shin, J.-E., Riesselman, A. J., Kollasch, A. W., McMahon, C., Simon, E., Sander, C., Manglik, A., Kruse, A. C., and Marks, D. S. Protein design and variant prediction using autoregressive generative models. *Nature Communications*, 12(1):2403, 2021. (page 3)

[69] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015. (pages 2 and 17)

[70] Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. (page 2)

[71] Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. (page 5)

[72] Sterling, T. and Irwin, J. J. ZINC 15–ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015. (page 7)

[73] Strauss, R. and Oliva, J. B. Arbitrary conditional distributions with energy. *Advances in Neural Information Processing Systems*, 34:752–763, 2021. (page 6)

[74] Subramanian, A., Greenman, K. P., Gervaix, A., Yang, T., and Gómez-Bombarelli, R. Automated patent extraction powers generative modeling in focused chemical spaces. *Digital Discovery*, 2(4):1006–1015, 2023. (pages 9 and 26)

[75] Tabak, E. G. and Turner, C. V. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013. (page 17)

[76] Tieleman, T. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1064–1071, 2008. (page 5)

[77] Tran, D., Vafa, K., Agrawal, K., Dinh, L., and Poole, B. Discrete flows: Invertible generative models of discrete data. *Advances in Neural Information Processing Systems*, 32, 2019. (pages 7, 8, and 17)

[78] Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475. PMLR, 2014. (pages 1, 3, 5, 6, and 15)

[79] Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016. (page 6)

[80] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. (pages 1, 3, and 8)

[81] Wang, J., Lisanza, S., Juergens, D., Tischer, D., Watson, J. L., Castro, K. M., Ragotte, R., Saragovi, A., Milles, L. F., Baek, M., et al. Scaffolding protein functional sites using deep learning. *Science*, 377(6604): 387–394, 2022. (page 1)

[82] Weininger, D., Weininger, A., and Weininger, J. L. SMILES. 2. algorithm for generation of unique SMILES notation. *Journal of Chemical Information and Computer Sciences*, 29(2):97–101, 1989. (pages 3 and 22)

[83] Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992. (pages 3 and 5)

[84] Wu, D., Wang, L., and Zhang, P. Solving statistical mechanics using variational autoregressive networks. *Physical review letters*, 122(8):080602, 2019. (pages 2, 3, 6, and 7)

[85] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32, 2019. (pages 1 and 6)

[86] Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., and Do, M. N. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5485–5493, 2017. (page 1)

[87] Zhang, D., Chen, R. T., Malkin, N., and Bengio, Y. Unifying generative models with gflownets. *arXiv preprint arXiv:2209.02606*, 2022. (pages 6 and 17)

[88] Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pp. 26412–26428. PMLR, 2022. (pages 6, 7, 8, and 17)

A. Additional Technical Details

A.1. Proof of Proposition 1

Proof. From the single-step marginalization self-consistency in (7), we have

$$\log p_\theta(\mathbf{x}) = \sum\nolimits_{d=1}^{D} \log p_\phi\left(x_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}\right),\ \forall \mathbf{x}, \sigma.$$

Therefore we can rewrite the optimization in (8) as:

$$\begin{aligned} \max_{\phi} \quad & \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)} \sum\nolimits_{d=1}^{D} \log p_\phi\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right) \\ \text{s.t.} \quad & p_\theta(\mathbf{x}_{\sigma(<d)})p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) = p_\theta(\mathbf{x}_{\sigma(\le d)}),\ \forall \sigma \in S_D, \mathbf{x} \in \{1,\cdots,K\}^D, d \in [1:D]. \end{aligned} \tag{10}$$

Let p^* be the optimal probability distribution that maximizes the likelihood on training data, and from the chain rule we have:

$$p^* = \arg\max_{p} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log p(\mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)} \sum\nolimits_{d=1}^{D} \log p\left(x_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}\right)$$

Then p^* is also the optimal solution to (10) the marginalization constraints are automatically satisfied by p^* since it is a valid distribution. From the universal approximation theorem [27, 26, 11], we can use separate neural networks to model p_θ (marginals) and p_ϕ (conditionals), and obtain optimal solution to (10) with θ^* and ϕ^* that approximates p^* arbitrarily well.

Specifically, if θ^* and ϕ^* satisfy the following three conditions below, they are the optimal solution to (10):

$$p_{\phi^*}\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right) = p^*\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right), \quad \forall\, \mathbf{x}, \sigma \tag{11}$$

$$p_{\theta^*}(\mathbf{x}_s) = p^*(\mathbf{x}_s)Z_{\theta^*}, \quad \forall\, \mathbf{x}, s \subseteq \{1,\cdots,D\} \tag{12}$$

$$p_{\theta^*}(\mathbf{x}_{\sigma(<d)})p_{\phi^*}(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) = p_{\theta^*}(\mathbf{x}_{\sigma(\le d)}),\ \forall \sigma \in S_D, \mathbf{x} \in \{1,\cdots,K\}^D, d \in [1:D] \tag{13}$$

where Z_{θ^*} is the normalization constant of p_{θ^*} and is equal to $p_{\theta^*}((\triangle,\cdots,\triangle))$. It is easy to see from the definition of conditional probabilities that satisfying any two of the optimal conditions leads to the third one.

To obtain the optimal ϕ^*, it suffices to solve the following optimization problem:

$$\textbf{Stage 1:} \quad \max_{\phi}\ \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)} \sum\nolimits_{d=1}^{D} \log p_\phi\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right)$$

because $p^* = \arg\max_p \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)} \sum_{d=1}^{D} \log p^*\left(x_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}\right)$ due to chain rule. Solving Stage 1 is equivalent to finding ϕ^* that satisfies condition (11). Then we can obtain the optimal θ^* by solving for condition (13) given the optimal conditionals ϕ^*:

$$\textbf{Stage 2:} \quad \min_{\theta}\ \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)}\mathbb{E}_{d\sim\mathcal{U}(1,\cdots,D)} \left(\log[p_\theta(\mathbf{x}_{\sigma(<d)})p_{\phi^*}(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)})] - \log p_\theta(\mathbf{x}_{\sigma(\le d)})\right)^2$$

□

A.2. Expected Lower bound of Log-Likelihood

Here we present the expected lower bound objective used for training AO-ARMs under maximum likelihood setting, which was first proposed by Uria et al. [78]. Hoogeboom et al. [24] provided the expected lower bound perspective.

Given an ordering σ,

$$\log p(\mathbf{x} \mid \sigma) = \sum\nolimits_{d=1}^{D} \log p\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right). \tag{14}$$

By taking the expectation over all orderings σ, we can derive a lower bound on the log-likelihood via Jensen's inequality.

$$\begin{aligned} \log p_\phi(\mathbf{x}) = \log \mathbb{E}_\sigma\, p_\phi(\mathbf{x} \mid \sigma) &\overset{\text{Jensen's inequality}}{\ge} \mathbb{E}_\sigma \sum\nolimits_{d=1}^{D} \log p_\phi\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right) \\ &= \mathbb{E}_{\sigma\sim\mathcal{U}(S_D)}\, D\, \mathbb{E}_{d\sim\mathcal{U}(1,\ldots,D)} \log p_\phi\left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right) \\ &= D\, \mathbb{E}_d\, \mathbb{E}_\sigma\, \frac{1}{D-d+1} \sum\nolimits_{j\in\sigma(\ge d)} \log p_\phi\left(x_j \mid \mathbf{x}_{\sigma(<d)}\right), \end{aligned} \tag{15}$$

where $\sigma \sim \mathcal{U}(S_D)$, $d \sim \mathcal{U}(1,\ldots,D)$ and $\mathbf{x}_{\sigma(<d)} = \{x_{\sigma(1)},\ldots,x_{\sigma(d-1)}\}$. $\mathcal{U}(S)$ denotes the uniform distribution over a finite set S and $\sigma(d)$ denotes the d-th element in the ordering.

A.3. Algorithms

We present the algorithms for training MAM for maximum likelihood and energy-based training settings in Algorithm 1 and Algorithm 2.

Algorithm 1 MLE training of MAMs

Input: Data $\mathcal{D}_{\text{train}}$, $q(\mathbf{x})$, network θ and ϕ
Stage 1: Train ϕ with Equation (15) used in AO-ARM
for minibatch $\mathbf{x} \sim \mathcal{D}_{\text{train}}$ **do**
  Sample $\sigma \sim \mathcal{U}(S_D)$, $d \sim \mathcal{U}(1, \cdots, D)$
  $\mathcal{L} \leftarrow \frac{D}{D-d+1} \sum_{j \in \sigma(\geq d)} \log p_\phi \left(x_j | \mathbf{x}_{\sigma(<d)}\right)$
  Update ϕ with gradient of $\mathcal{L}$
end for
Stage 2: Train θ to distill the marginals from optimized conditionals ϕ
for minibatch $\mathbf{x} \sim q(\mathbf{x})$ **do**
  Sample $\sigma \sim \mathcal{U}(S_D)$, $d \sim \mathcal{U}(1, \cdots, D)$
  $\mathcal{L} \leftarrow$ squared error of the inconsistencies in Equation (7)
  Update θ with gradient of $\mathcal{L}$
end for

Algorithm 2 Energy-based training of MAMs

Input: $q(\mathbf{x})$, network θ and ϕ, Gibbs sampling block size M
Joint training of ϕ and θ:
for j in $\{1, \cdots, N\}$ **do**
  Sample $\sigma \sim \mathcal{U}(S_D)$
  Update $\mathbf{x} \sim p_\phi(\mathbf{x}_{\sigma(\leq M)} | \mathbf{x}_{\sigma(>M)})$
  ▷ Persistent block Gibbs sampling
  Sample $\tilde{\mathbf{x}} \sim q(\mathbf{x})$
  Sample $\tilde{d} \sim \mathcal{U}(1, \cdots, D)$, $\tilde{\sigma} \sim \mathcal{U}(S_D)$
  $\mathcal{L}_{\text{penalty}} \leftarrow$ squared error of Equation (7), for $\tilde{d}$ and $\tilde{\sigma}$ with $\tilde{\mathbf{x}}$
  $\nabla_{\theta,\phi} D_{\text{KL}} \leftarrow$ REINFORCE est. with $\mathbf{x}$
  $\nabla_{\theta,\phi} \leftarrow \nabla_{\theta,\phi} D_{\text{KL}} + \lambda \nabla_{\theta,\phi} \mathcal{L}_{\text{penalty}}$
  Update θ and ϕ with gradient
end for

A.4. Connections between MAMs and GFlowNets

In this section, we identify an interesting connection between generative marginalization models and GFlowNets. The two type of models are designed with different motivations. GFlowNets are motivated by learning a policy to generate according to an energy function and MAMs are motivated from any-order generation through learning to perform marginalization. However, under certain conditions, there exists an interesting connection between generative marginalization models and GFlowNets. In particular, the marginalization self-consistency condition derived from the definition of marginals in Equation (4) has an equivalence to the "detailed balance" constraint in GFlowNet under the following specific conditions.

Observation 1. *When the directed acyclic graph (DAG) used for generation in GFlowNet is specified by the following conditions, there is an equivalence between the marginalization self-consistency condition in Equation (7) for* MAM *and the detailed balance constraint proposed for GFlowNet [4]. In particular, the $p_\theta(x_{\sigma(d)} | \mathbf{x}_{\sigma(<d)})$ in* MAM *is equivalent to the forward policy $P_F(\mathbf{s}_{d+1} \mid \mathbf{s}_d)$ in GFlowNet, and the marginals $p_\theta(x_{\sigma(d)})$ are equal to the flows $F(\mathbf{s}_d)$ up to a normalizing constant.*

- *DAG Condition: The DAG used for generation in GFlowNet is defined by the given tree-like structure: a sequence $\mathbf{x}$ is generated by incrementally adding one variable at each step, following a uniformly random ordering σ i.e. $\sigma \sim \mathcal{U}(S_D)$. At step d, the state along the generation trajectory is defined to be $\mathbf{s}_d = \mathbf{x}_{\sigma(\leq d)}$.*

- *Backward Policy Condition: At step $D-d$, the backward policy under the DAG is fixed by removing (un-assigning) the value of the $d+1$-th element under ordering σ , i.e. $P_B(\mathbf{s}_d \mid \mathbf{s}_{d+1}; \sigma) = \mathbb{1}_{\{\mathbf{s}_d = \mathbf{x}_{\sigma(\leq d)}\}}$. Or equivalently, the backward policy removes (un-assigns) one of the existing variables at random, i.e. $P_B(\mathbf{s}_d \mid \mathbf{s}_{d+1}) = {}^1/_{d+1} \mathbb{1}_{\{\mathbf{s}_d \subset \mathbf{s}_{d+1}\}}$.*

Intuitively, this is straight forward to understand, since GFlowNet generates a discrete object autoregressively. The model was proposed to enhance the flexibility of generative modeling by allowing for a learned ordering, as compared with autoregressive models (see [88] Sec. 5 for a discussion). When the generation ordering is fixed, it is reduced to autoregressive models with fixed ordering, which is discussed in [87]. Observation 1 presented above for any-order ARMs can be seen as a extended result of the connection between GFlowNets and fixed-order ARMs.

We have seen the interesting connection of GFlowNets with ARMs (and MAMs). Next, we discuss the differences between GFlowNets and MAMs.

Remark 1. *The detailed balance constraint was proposed only as a theoretical result in Bengio et al. [4]. In actual experiments, GFlowNets are trained using either flow matching [2] or trajectory balance [46, 88].*

Zhang et al. [88] is the most relevant GFlowNet work that targets the discrete problem setting. Training is done via minimizing the squared distance loss with trajectory balance objective. For the MLE training, it proposes to additionally learn an energy function from data so that the trajectory balance objetive can still be applied. In particular, MAM is different from GFlowNet in Zhang et al. [88] in three main aspects.

- First of all, MAMs target any-order generation and direct access to marginals, where as GFlowNets aim for flexibility in learning generation paths and does not offer exact likelihood or direct access to marginals under learnable generation paths. When the generation path is fixed to follow a ordering or random ordering, they are reduced to ARMs or any-order ARMs, which allow for exact likelihood. However, training with the trajectory balance objective does not offer direct access to marginals (just like how ARMs do not offer direct access to marginals but only conditionals).
- Second, training under MLE setting is signiticantly different: GFlowNets learn an additional learned energy function to reduce MLE training back to energy-based training, while MAMs directly maximizes the expected lower bound on the log-likelihood under the marginalization self-consistent constraint.
- Lastly, the training objective is different under energy-based training. GFlowNets are trained on squared distance under the expectation to be specified to be either on-policy, off-policy, or a mixture of both. MAMs are trained on KL divergence where the expectation is defined to be on-policy. It is possible though to train MAMs with squared distance and recently Malkin et al. [47] have shown the equivalence of the gradient of KL divergence and the on-policy expectation of the per-sample gradient of squared distance (which is the gradient actually used for training GFlowNets).

A.5. Additional literature on discrete generative models

Discrete diffusion models Discrete diffusion models learn to denoise from a latent base distribution into the data distribution. Sohl-Dickstein et al. [69] first proposed diffusion for binary data and was extended in Hoogeboom et al. [25] for categorical data and both works adds uniform noise in the diffusion process. A wider range of transition distributions was proposed in D3PM [1] and insert-and-delete diffusion processes have been explored in Johnson et al. [31]. Hoogeboom et al. [24] explored the connection between ARMs and diffusion models with absorbing diffusion and showed that OA-ARDMs are equivalent to absorbing diffusion models in infinite time limit, but achieves better performance with a smaller number of steps.

Discrete normalizing flow Normalizing flows transform a latent base distribution into the data distribution by applying a sequence of invertible transformations [61, 75, 16, 69, 60, 17, 34, 52]. They have been extended to discrete data [77, 23] with carefully designed discrete variable transformations. Their performance is competitive on character-level text modeling, but they do not allow any-order modeling and could be limited to discrete data with small number of categories due to the use of a straight-through gradient estimators.

Discussion of neural generative models and Probabilistic circuits Probabilistic circuits [8, 54, 41, 42] is a powerful modeling approach exhibiting fast and exact marginalization though the design of the model's structure and operations. In contrast, neural generative models are highly expressive, allowing them to perform powerful approximate inference. Despite not having the exact marginalization property, the neural network approach has the advantage of much greater flexibility in

modeling the complex distributions found in practical applications [67, 24]. Hence, a trade-off currently exists between exact marginalization and approximate marginalization with a more expressive network. Our work falls in the neural generative models category, but directly approximates marginals. Direct modeling of marginals opens opportunities for more flexible sampling, as shown in Appendix B.2, and more scalable approximate marginal inference and training under EB settings.

B. Ablation Studies

B.1. Testing marginal self-consistency

The *marginal self-consistency* in MAMs is enforced through optimizing the scalable training objective. Here we empirically examine how well they are enforced in practice. First we look at *checkerboard*, a synthetic problem often used for testing clustering algorithms. More recently it has been used for testing and visualizing both continuous and discrete generative models. We define a discrete input space by discretizing the continuous coordinates of points in 2D. To be more concrete, the origin range $[-4, 4]$ of each dimension is converted into a 16-bit string following the standard way of converting float to string. The target unnormalized probability $p(\mathbf{x})$ is set to 1 for points within dark squares and $1e-10$ within light squares (since it is infeasible to set it to $\ln(0) = -\infty$ for a NN to learn, and in practice $1e-10$ is negligible compared to 1). We trained a 5-layer MLP with hidden node size 2048 and residual connections on this problem on both MLE and EBM settings and $q(\mathbf{x})$ is set to be a balanced mixture of ground truth data and samples from p_θ for MLE or uniform random for EBM:

$$\min_\theta -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} p_\theta(\mathbf{x}) + \lambda\, \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\mathbb{E}_\sigma \mathbb{E}_d \Big(\log \textstyle\sum_{x_{\sigma(d)}} p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}]) - \log p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}])\Big)^2.$$

$$\min_\theta D_{\text{KL}}(p_\theta(\mathbf{x}) \,\|\, p(\mathbf{x})) + \lambda\, \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\mathbb{E}_\sigma \mathbb{E}_d \Big(\log \textstyle\sum_{x_{\sigma(d)}} p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}]) - \log p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}])\Big)^2.$$

For this problem, only a marginal network θ is trained to predict the $\log p$ of any marginals. Upon training to convergence, the generative models perform on par or better than state of the art discrete generative models and achieve a 20.68 test NLL. See Figure 9 for a comparison of ground truth and learned PMF heatmap. It can be seen the PMF are approximated quite accurately. We investigate how well the marginal self-consistency are enforced, by looking at the marginal estimates of MAMs trained with $\lambda = 1e2$ and $\lambda = 1e4$. We evaluate marginals over the first dimension ($0-16$ bits) by fixing the second dimension ($17-32$ bits) to 1.0 (bit string $= 0001111111111111$). We plot marginals by marginalizing out bit $3-16$ (i.e. $(x_1, x_2, \cdots)$) and bit $5-16$ (i.e. $(x_1, x_2, x_3, x_4, \cdots)$). In Figure 12, when $\lambda = 1e4$, the self-consistency are more strictly enforced, leading to matched marginals. Notice that there is some tiny residue PMF at the light squares due to the $1e-10$ approximation applied to points with 0 probability, but they are negligible compared to the significant probability masses. After normalizing the marginals over all possibilities, the marginals are almost exactly matched. In Figure 13, when $\lambda = 1e2$, the self-consistency are more loosely enforced as compared to $\lambda = 1e4$. But it is notable that they are only shifted by a constant as compared to the ground truth marginals. This means although marignal self-consisteny is not strictly enforced when $\lambda = 1e2$, softly enforcing it leads to shifted but consistent estimates of marginals, as the NN learns to generalize and predict symmetric probabilities for symmetric regions. Using the constant-shifted marginals to sample will result in the same distribution with the ground truth, because the normalized MAM marginals match the ground truth almost exactly. This is observed in the samples generated under $\lambda = 1e2$ in Figure 9 and consistent normalized marginals in Figure 13.

B.2. Sampling with marginals v.s. conditionals

The trained marginalization model comes with two networks. The conditional network ϕ estimates any-order conditionals $p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)})$, and the marginal network θ estimates arbitrary marginals $p_\theta(\mathbf{x}_{\sigma(\leq d)})$. When MAM is used for sampling, either network can be used. With the conditional network ϕ, samples can be drawn autoregressively one variable at each step. Or the marginals can be used to draw variables using the normalized conditional:

$$p_\theta(\mathbf{x}_{s_i}|\mathbf{x}_{s(<i)}) = \frac{p_\theta([\mathbf{x}_{s_i}, \mathbf{x}_{s(<i)}])}{\sum_{\mathbf{x}_{s_i}} p_\theta([\mathbf{x}_{s_i}, \mathbf{x}_{s(<i)}])}.$$

where $\mathbf{x}_{s_i}$ is the next block of variables (can be multiple) to sample at step i and $\mathbf{x}_{s(<i)}$ are the previously sampled variables. We show with experiments that the marginals are also effective to be used for sampling and they provide extra flexibility

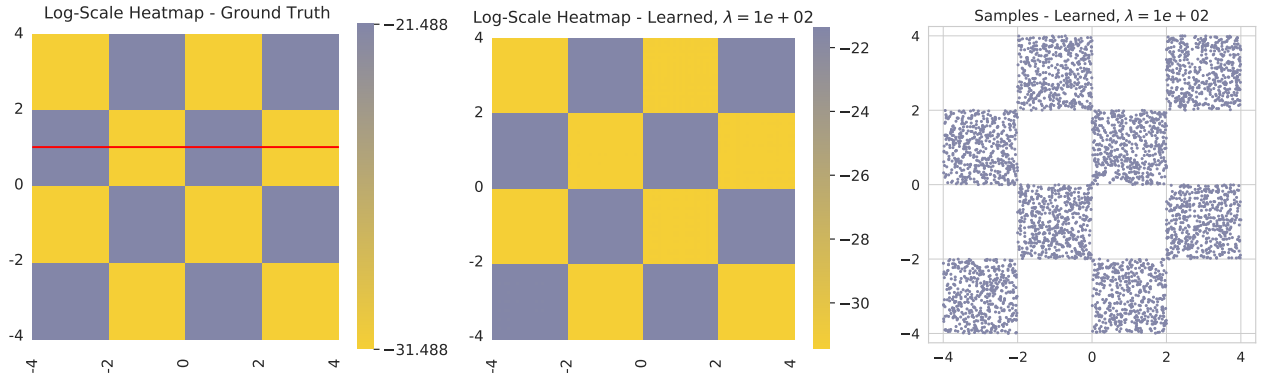


Figure 9. PMF heat map under EB training. The learned PMF and ground truth PMF are consistent to each other relatively well. The MSE on log p (or p) of dark pixels is 0.0033 (or $7.67e-20$) and the MSE on light pixels is 0.0076 (or $3.73e-30$). We are evaluating marginals along the red line: i.e. fixing $(\mathbf{x}_{17}, \cdots, \mathbf{x}_{32}) = (0, 0, 0, 1, 1, 1, \cdots, 1)$, which correspond to 1 in floating number for y-axis, and perform marginalization over $(\mathbf{x}_1, \cdots, \mathbf{x}_{16})$. $(0, 0, \cdots)$ corresponds to $[0, 2]$. $(0, 1, \cdots)$ corresponds to $[2, 4]$. $(1, 0, \cdots)$ corresponds to $[-2, 0]$. $(1, 1, \cdots)$ corresponds to $[-4, -2]$.

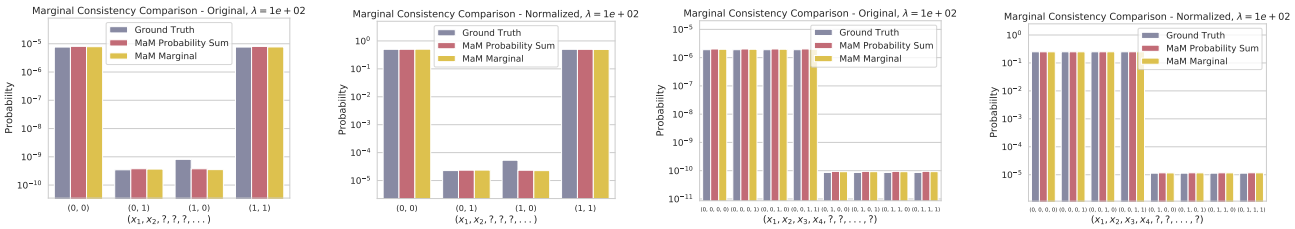


Figure 10. Marginal consistency $\lambda = 1e2$ under EB training. Ground truth: summing over ground truth PMF. MAM Probability Sum: summing over learned PMF from MAM. MAM Marginal: direct estimate with MAM. The small discrepancy in $p(1, 0, ?, \cdots, ?)$ is due to the corner case of $(1, 0, 0, 0, \cdots, 0)$ be assigned to a positive value due to numerical errors in float conversion.

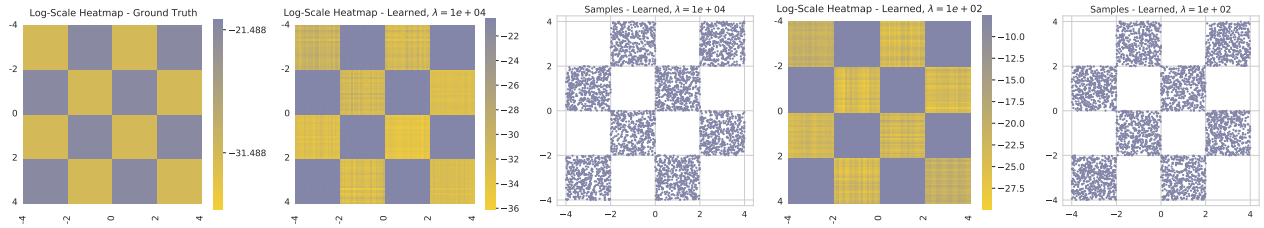


Figure 11. PMF heat map under MLE training. The learned PMF and ground truth PMF are consistent to each other relatively well. The MSE on log p (or p) of dark pixels is 0.533 (or $2.5e-19$) and the MSE on light pixels is 2.5 (or $3e-28$).

in the sampling procedure. We test sampling with different block sizes using the marginals with random orderings and compare them to sampling with conditionals in Figure 14. The samples generated are of similar quality. And those different sampling procedures exhibit similar likelihood on test data. However, sampling with large block size enables to trade compute memory for less time spent (due to fewer steps) in generation inference, which we find it interesting to explore for future work. Compared with the conditional network, the marginal network allows sampling in arbitrary block variable size and ordering. This illustrates the potential utility of MAMs in flexible generation with tractable likelihood.

B.3. Choice of q in sampling the marginalization self-consistency for training

In simple examples such as the synthetic checkerboard problem, it does not really matter, we have tried p_{data} or p_θ or random, or a mixture of them. All work fairy well given that the problem is relatively easy.

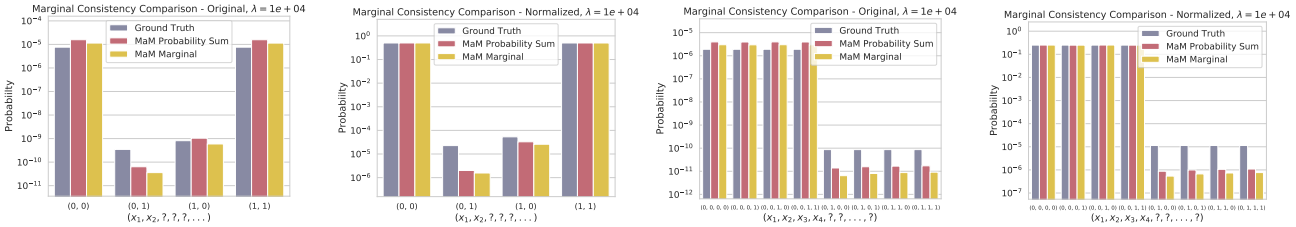


Figure 12. Marginal consistency $\lambda = 1e4$ with MLE training. Ground truth: summing over ground truth PMF. MAM Probability Sum: summing over learned PMF from MAM. MAM Marginal: direct estimate with MAM. Note that p for $(0, 1)$ and $(1, 0)$ should be in principle close to zero, but are non-zero due to float-to-int converting numerical errors.

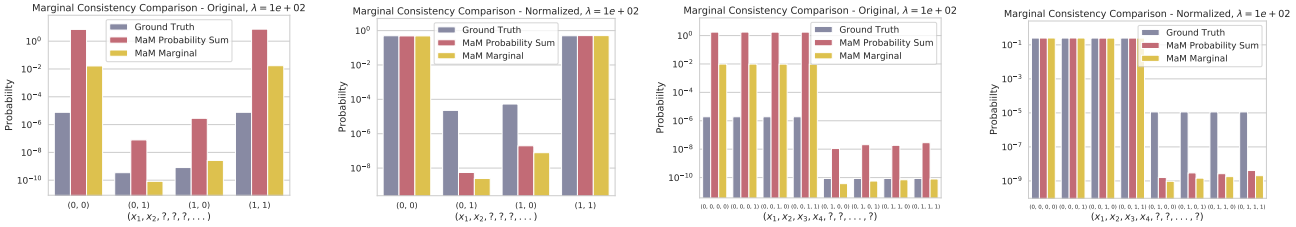
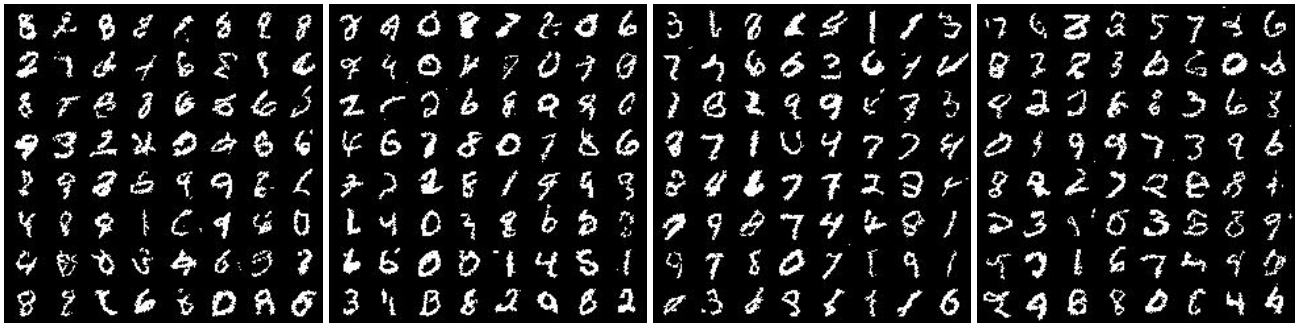
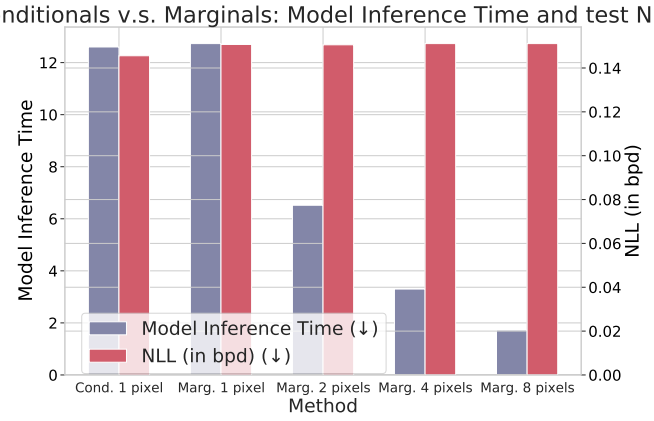


Figure 13. Marginal consistency $\lambda = 1e2$ with MLE training.



(a) Marg.: 1 pixel per step **(b)** Marg.: 2 pixels per step **(c)** Marg.: 4 pixels per step **(d)** Marg.: 8 pixels per step

(e) Cond.: 1 pixel per step



(f) Inference time and NLL comparison

Figure 14. MAM sampling using marginal network (a-d) with different number of variables at each step v.s. sampling using conditional network (e) with 1 variable at eacth step. (f) compares NLL under different sampling procedures and the model inference time.

In real-world data problems, it boils down to what the marginal will be used for at test time. Uniform distribution over x will be a bad choice if there is a data manifold we care about. If it will be used for generation, for example in the MNIST Binary

example in Appendix B.2, q is set to a mixture of p_θ and p_{data}. If it will be used for mariginal inference on the data manifold, p_{data} will be enough. We all know the NN is not robust on data it hasn't trained on, and so are the marginal networks, they will not give correct estimates if we evaluate on arbitrary datapoint off the manifold or policy.

B.4. Two-Stage v.s. Joint Training

On MNIST maximum likelihood training, we compare two-stage training and joint training in Figure 15. Both training uses the decomposed conditionals for the log likelihood objective, otherwise joint training will lead to inflated log likelihoods. We observe that two-stage training converges faster than joint training (20 epochs v.s. 80 epochs) and needs less GPU memory since it only requires gradient of one model instead of two models. For joint training, it is observed that smaller λ is preferred for fast convergence and better performance while large λ hurts the model's inference performance.

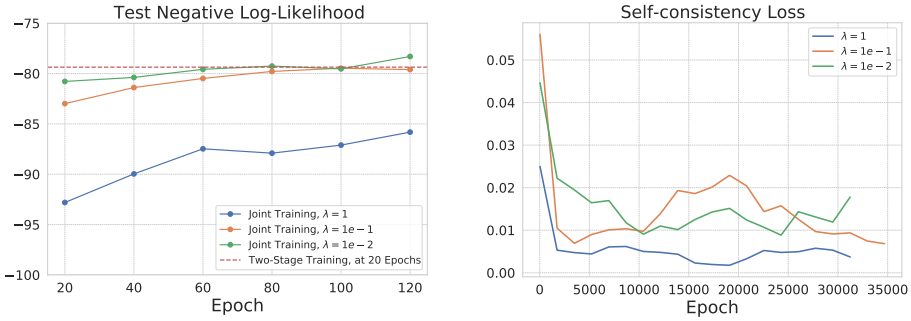


Figure 15. Two-stage training v.s. joint training on MNIST-Binary maximum likelihood training. λ is the penalty hyperparameter of self-consistency error term.

For joint training under energy-based training setting, we empirically test out how λ affects models performance. We find that a wide range of small λ leads to best results. See Figure 16 for training dynamics of different λ values. Our hypothesis is that $\mathcal{L}_{\text{KL}}$ is easier to fit than $\mathcal{L}_{\text{SC}}$, since it only involves fitting one term instead of many constraints. When λ is relatively small, $\mathcal{L}_{\text{KL}}$ is closely fitted first, then training objective is left with $\lambda\mathcal{L}_{\text{SC}}$. Since optimization with Adam is scale-invariant, the training converges to similar solutions. When λ is too large, $\mathcal{L}_{\text{SC}}$ is first fitted very close to 0, but this restricts the flexibility of the conditionals and marginals to fit $\mathcal{L}_{\text{KL}}$ well, hence hurting its generative performance.

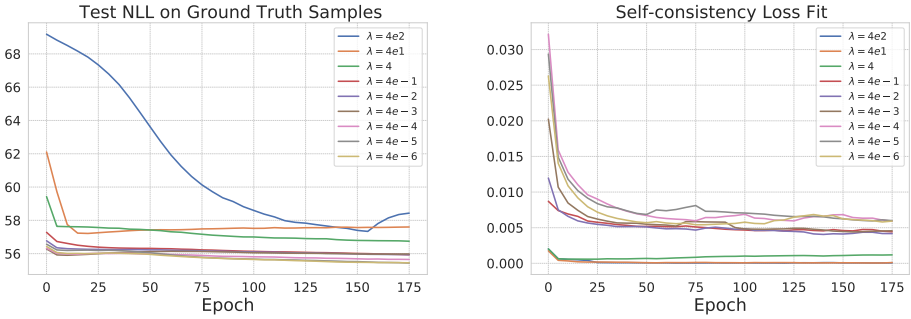


Figure 16. Two-stage training v.s. joint training on Ising Model 10×10 energy-based training. λ is the penalty hyperparameter of self-consistency error term.

C. Additional Experiments Details

C.1. Dataset details

Binary MNIST Binary MNIST is a dataset introduced in [63] that stochastically set each pixel to "1" or "0" in proportion to its pixel intensity. We use the training and test split of [63] provided in https://github.com/yburda/iwae/tree/master [6].

Table 7. Length Extrapolation on Text8

Method	Spearman ↑	Pearson ↑	Time (s) ↓
MAM	0.947	0.944	0.006
P-AO-ARM $T = 2$	0.859	0.854	2.223
P-AO-ARM $T = 5$	0.923	0.931	5.683
P-AO-ARM $T = 10$	0.927	0.931	11.63
P-AO-ARM $T = 20$	0.957	0.970	23.28
AO-ARM $T = 300$	0.969	0.966	349.7

CIFAR-10 The CIFAR-10 dataset [37] comprises 60,000 32x32 color images across 10 classes, split into 50,000 training and 10,000 test images. It's used for image recognition and classification tasks in machine learning.

ImageNet32 ImageNet32 [14, 10] is a downsampled variant of the ImageNet dataset, resized to 32x32 pixels. It maintains the diversity of the original with over 14 million images across thousands of categories, but in a lower resolution for computational efficiency.

Molecular Sets The molecules in MOSES are represented either in SMILES [82] or SELFIES [36] strings. We construct a vocabulary (including a stop token) from all molecules and use discrete valued strings to represent molecules. It is worth noting that MAM can also be applied for modeling molecules at a coarse-grained level with predefined blocks, which we leave for future work.

The test set used for evaluating likelihood estimate quality is constructed in a similar manner to Binary MNIST, by drawing sets of random samples from the test dataset.

text8 In this dataset, we use a vocabulary of size 27 to represent the letter alphabet with an extra value to represent spaces.

The test set of datasets used for evaluating likelihood estimate quality is constructed in a similar manner to Binary MNIST, each set is generated by randomly masking out portions of a test text sequence (by 50, 100, 150, 200 tokens) and generating samples.

Ising model The Ising model is defined on a 2D cyclic lattice. The $\mathbf{J}$ matrix is defined to be $\sigma \mathbf{A}_N$, where σ is a scalar and $\mathbf{A}_N$ is the adjacency matrix of a $N \times N$ grid. Positive σ encourages neighboring sites to have the same spins and negative σ encourages them to have opposite spins. The bias term $\boldsymbol{\theta}$ places a bias towards positive or negative spins. In our experiments, we set σ to 0.1 and $\boldsymbol{\theta}$ to $\mathbf{1}$ scaled by 0.2. Since we only have access to the unnormalized probability, we generate 2000 samples following [21] using Gibbs sampling with $1,000,000$ steps for 10×10 and 30×30 lattice sizes. Those data serve as ground-truth samples from the Ising model for evaluating the test log-likelihood.

Molecular generation with target property During training, we need to optimize on the loss objective on samples generated from the neural network model. However, if the model generates SMILES strings, not all strings correspond to a valid molecule, which makes training at the start challenging when most generated SMILES strings are invalid molecules. Therefore, we use SELFIES string representation as it is a 100% robust in that every SELFIES string corresponds to a valid molecule and every molecule can be represented by SELFIES.

C.2. Training details

Binary MNIST, CIFAR10, ImageNet32

- Pixel values are converted to scalar values as input. "0", "1" for Binary MNIST, "$0-255$" for CIFAR-10 and ImageNet. "$\triangle$" takes the value 0. For each pixel, there is an additional mask indicating if it is a "$\triangle$".
- U-Net with 4 ResNet Blocks for MNIST, 32 ResNet Blocks for CIFAR-10 sand ImageNet, interleaved with attention layers for both AO-ARM and MAM. MAM uses two separate neural networks for learning marginals ϕ and conditionals θ. Input resolution is $1 \times 28 \times 28$ or $3 \times 32 \times 32$ with 256 channels used.
- The mask is concatenated to the input. $3/4$ of the channels are used to encode input. The remaining $1/4$ channels encode the mask cardinality (see [24] for details).

- MAM first learns the conditionals ϕ and then learns the marginals θ by finetuning on the downsampling blocks and an additional MLP with 2 hidden layers of dimension 4096. We observe it is necessary to distill the marginals by not only finetuning on the additional MLP but also on the downsampling blocks to get a good fitting of the marginal probability, which shows marginal network and conditional network rely on different features to make the final prediction.
- Batch size is 128 for MNIST and 32 for CIFAR-10 and ImageNet. Adam is used with learning rate 0.0001. Gradient clipping is set to 100. Both AO-ARM and MAM conditionals are trained for 100 epochs on MNIST, 800 epochs on CIFAR-10, 16 epochs on ImageNet. MAM marginals are finetuned from the trained conditionals for 25 epochs on MNIST, 25 epochs on CIFAR-10 and 3 epochs on ImageNet.

  The effectiveness of the proposed two-stage training is validated during experiments. Distilling marginals from conditionals are much faster and easier than learning conditionals and marginals jointly from scratch. And distilling marginals require much fewer epochs than fitting the conditionals.

MOSES and text8

- Transformer with 12 layers, 768 dimensions, 12 heads, 3072 MLP hidden layer dimensions for both AO-ARM and MAM. Two separate networks are used for MAM.
- SMILES or SELFIES string representation and "$\triangle$" are first converted into one-hot encodings as input to the Transformer.
- MAM first learns the conditionals ϕ and then learns the marginals θ by finetuning on the MLP of the Transformer.
- Batch size is 512 for MOSES and 256 for text8.
- AdamW is used with learning rate 0.0005, betas $0.9/0.99$, weight decay 0.001. Gradient clipping is set to 0.25. Both AO-ARM and MAM conditionals are trained for 1000 epochs for text8 and 200 epochs for MOSES. The MAM marginals are finetuned from the trained conditionals for 200 epochs.

Ising model and molecule generation with target property

- Ising model input are of $\{0, 1, \triangle\}$ values and are one-encoded as input to the neural network. The same is done for molecule SELFIES strings.
- MLP with residual layers, 3 hidden layers, feature dimension is 2048 for Ising model. 6 hidden layers, feature dimension 4096 for molecule target generation.
- Adam is used with learning rate of 0.0001. Batch size is 512 and 4096 for molecule target generation. ARM, GFlowNet and MAM are trained with $19,800$ steps for the Ising model. ARM and MAM are trained with $3,000$ steps for molecule target generation.
- Separate networks are used for conditionals and marginals of MAM. They are trained jointly with penalty parameter λ set to 4.

Compute

- All models are trained on a single NVIDIA A100. The evaluation time is tested on an NVIDIA GTX 1080Ti.

C.3. Additional results on Images

C.3.1. CIFAR-10

We train MaMs conditionals for 800 epochs and then further train 25 epochs to fit the marginals. MAM achieves a test NLL of 2.88 bpd (if we continue training to 3000 epochs, test NLL will get close to 2.69 bpd shown in the AO-ARM literature [24]). Test NLL is compared in Table 2. MaM achieve highly correlations in terms of $\log p$ estimate when compared with AO-ARM $\log p$'s. The marginal self-consistency error is averaged ~ 0.3 in $\log p$ values. Generated samples are shown in Figure 17 and Figure 18.

Figure 17. . CIFAR-10: conditional generation.

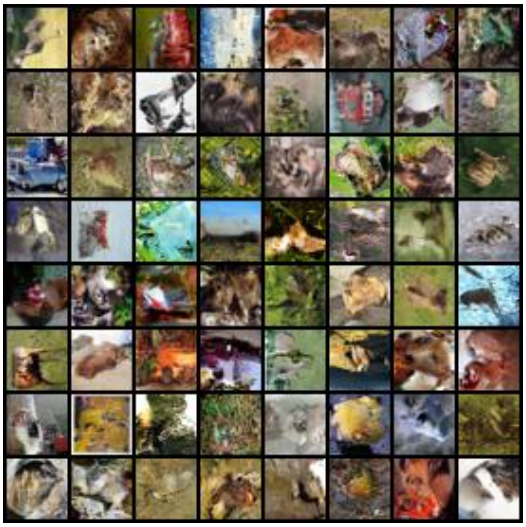


Figure 18. . CIFAR-10: generated samples. Note that sometimes images are flipped because MaM is trained on augmented images.

C.3.2. IMAGENET32

We train MaMs conditionals for 16 epochs and train 3 more epochs for fitting the marginals. MAM achieves a test NLL of 2.88 bpd (if we continue training to 3000 epochs, test NLL will get close to 2.69 bpd shown in the AO-ARM literature [24]). Test NLL is compared in Table 3. MaM achieve highly correlations in terms of $\log p$ estimate when compared with AO-ARM $\log p$'s. The marginal self-consistency error is averaged ~ 0.3 in $\log p$ values. Generated samples are shown in Figure 19 and Figure 20.

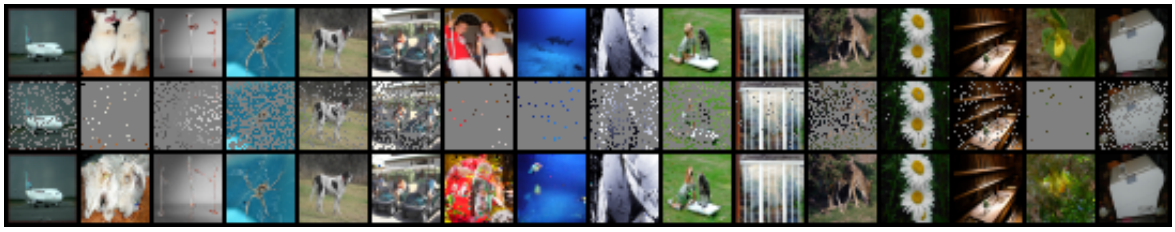


Figure 19. . ImageNet32: conditional generation.

C.3.3. BINARY MNIST

Likelihood estimate on partial Binary MNIST images

Figure 22 illustrates an example set of partial images that we evaluate and compare likelihood estimate from MAM against ARM. Table 8 contains the comparison of the marginal likelihood estimate quality and inference time.

Likelihood estimate on synthetic Binary MNIST images

Figure 20. . ImageNet32: generated samples.

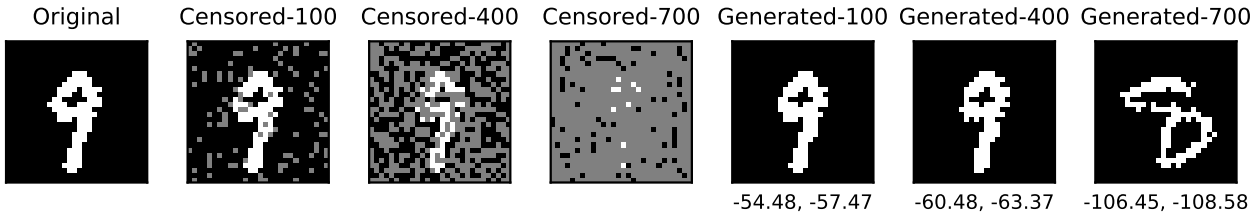


Figure 21. An example of the data generated (with $100/400/700$ pixels masked) for comparing the quality of likelihood estimate. Numbers below the images are LL estimates from MAM's marginal network (left) and AO-ARM-E's ensemble estimate (right).

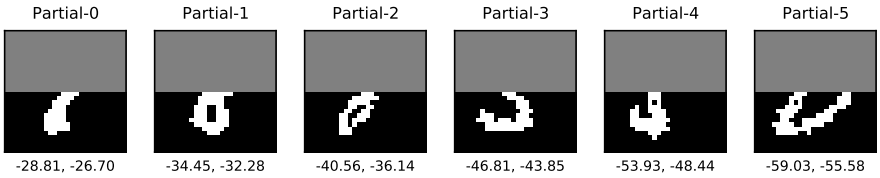


Figure 22. An example set of partial images for evaluating marginal likelihood estimate quality. The numbers in the captions show the log-likelihood calculated using learned marginals (left) v.s. learned conditionals (right)

Table 8. Marginal estimates on Binary-MNIST partial images

	Pearson ↑	Marg. inf. time (s) ↓
AO-ARM	0.997	49.75 ± 0.03
MAM	0.995	0.02 ± 0.00

Figure 21 illustrates an example of "synthetic" MNIST images generated from masked MNIST images that we evaluate and compare likelihood estimate from MAM against ARM. Table 9 shows the marginal likelihood estimate shows strong correlation with actual $\log p$ from ARM, demonstrating strong generalizing to data on the manifold but not seen during training.

Generated samples

Table 9. Marginal estimates on Binary-MNIST "synthetic" images

	Pearson ↑
AO-ARM	0.993
MAM	0.993

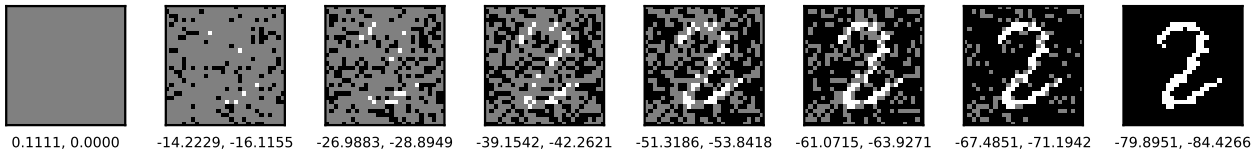


Figure 23. An example of the trajectory every 112 step when generating an MNIST digit following a random order. The future pixels are generated by conditioning on the existent filled-in pixels. The numbers in the captions show the log-likelihood calculated using learned marginals (left) v.s. learned conditionals (right)

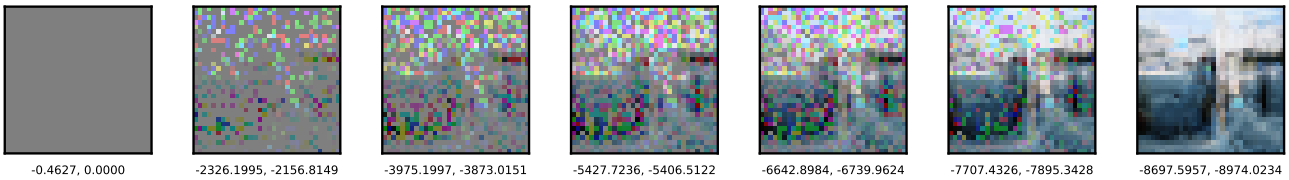


Figure 24. An example of the trajectory when generating an ImageNet image following a random order. The future pixels are generated by conditioning on the existent filled-in pixels. The numbers in the captions show the log-likelihood calculated using learned marginals (left) v.s. learned conditionals (right).

Figure 23 shows how a digit is generated pixel-by-pixel following a random order. We show generated samples from MAM using the learned conditionals ϕ in Figure 25.

C.4. Additional results on MOSES

C.4.1. Comparing MAM with SOTA on MOSES molecule generation

We compare the quality of molecules generated by MAM with standard baselines and state-of-the-art methods in Table 11 and Figure 26. Details of the baseline methods are provided in [55]. MAM-SMILES/SELFIES represents MAM trained on SMILES/SELFIES string representations of molecules. MAM performs either better or comparable to SOTA molecule generative modeling methods. The major advantage of MAM and AO-ARM is that their order-agnostic modeling enables generation in any desired order of the SMILES/SELFIES string (or molecule sub-blocks).

C.4.2. Generated molecular samples

Figure 27 and 28 plot the generated molecules from MAM-SMILES and MAM-SELFIES.

C.4.3. Out-of-distribution test on differentiating Drug vs. Photodiode

We challenged the model with a more difficult OOD task: distinguishing between tyrosine kinase inhibitors (a specific type of drug) and organic photodiodes from focused chemical spaces while the MAM model is trained on a general chemical space of drug-like compounds. The tyrosine kinase inhibitors should be considered by the model to have higher likelihood given that it has more similar properties (such as moderate weight and lipophilicity) to the molecules in ZINC.

We created 1000 pairs consisting of one of each using datasets from Subramanian et al. [74], controlling for other factors like SMILES length and chemical space to increase difficulty. Despite this, MAM's marginals correctly identified the drug molecule 74% of the time (vs. 79% for AO-ARM), with 90% alignment between marginal estimates and AO-ARM $\log p$'s.

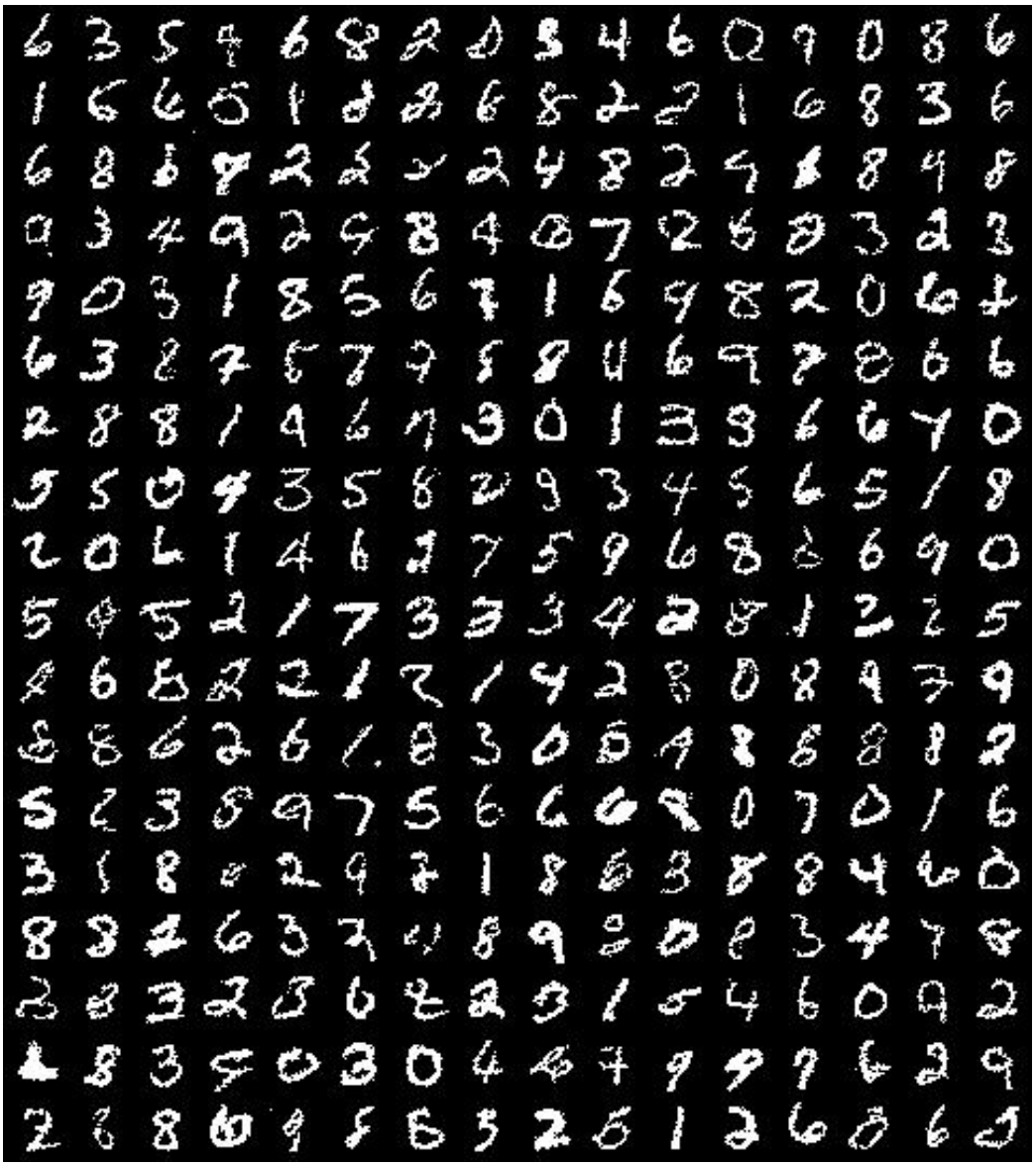


Figure 25. Generated samples: Binary MNIST

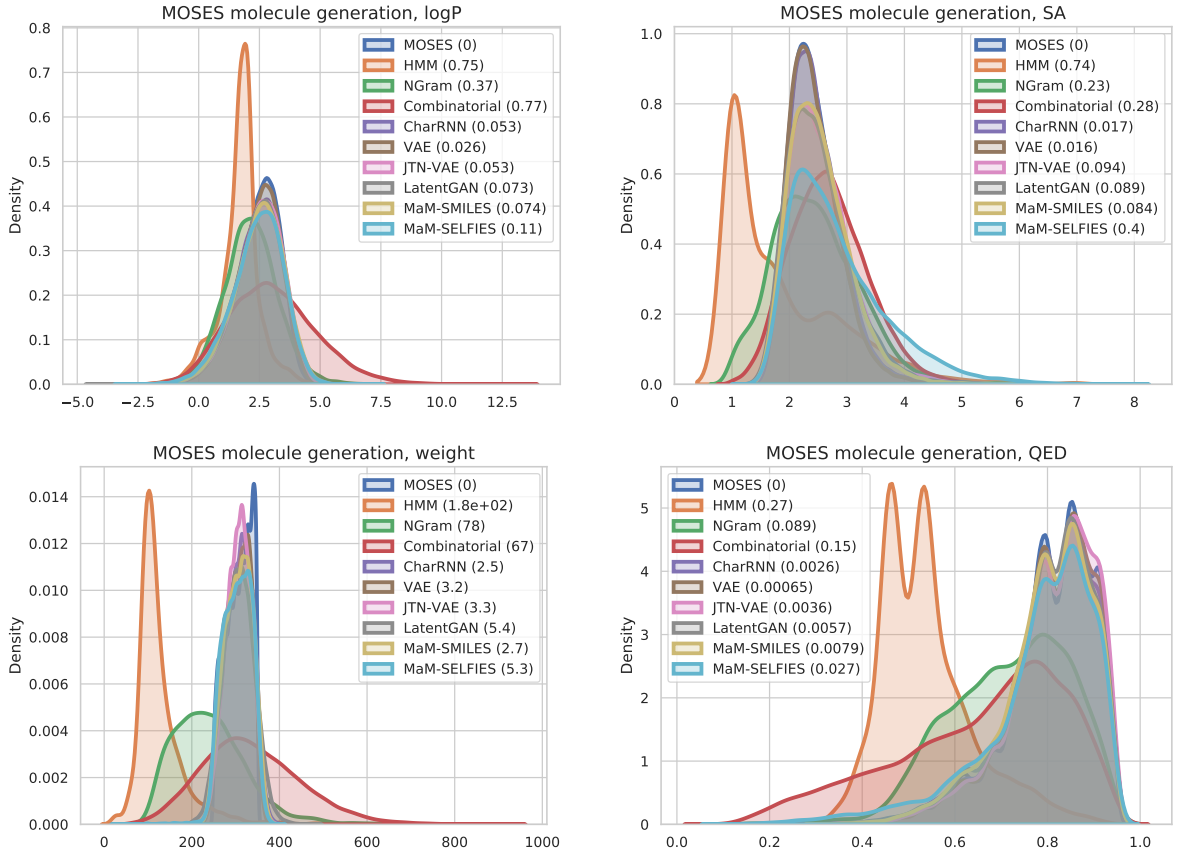


Figure 26. KDE plots of lipophilicity (logP), Synthetic Accessibility (SA), Quantitative Estimation of Drug-likeness (QED), and molecular weight for generated molecules. $30,000$ molecules are generated for each method.

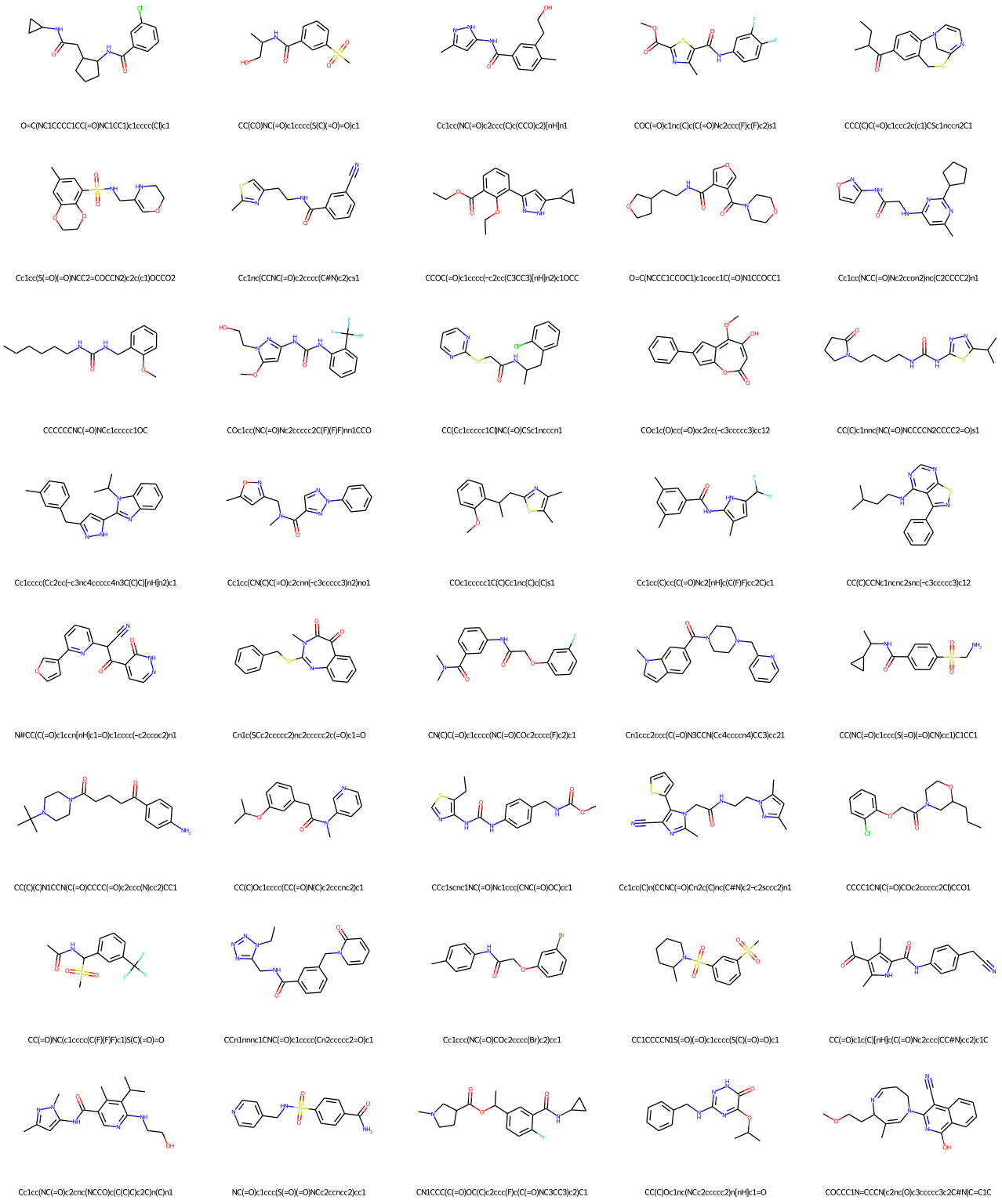


Figure 27. Generated samples from MAM-SMILES: MOSES

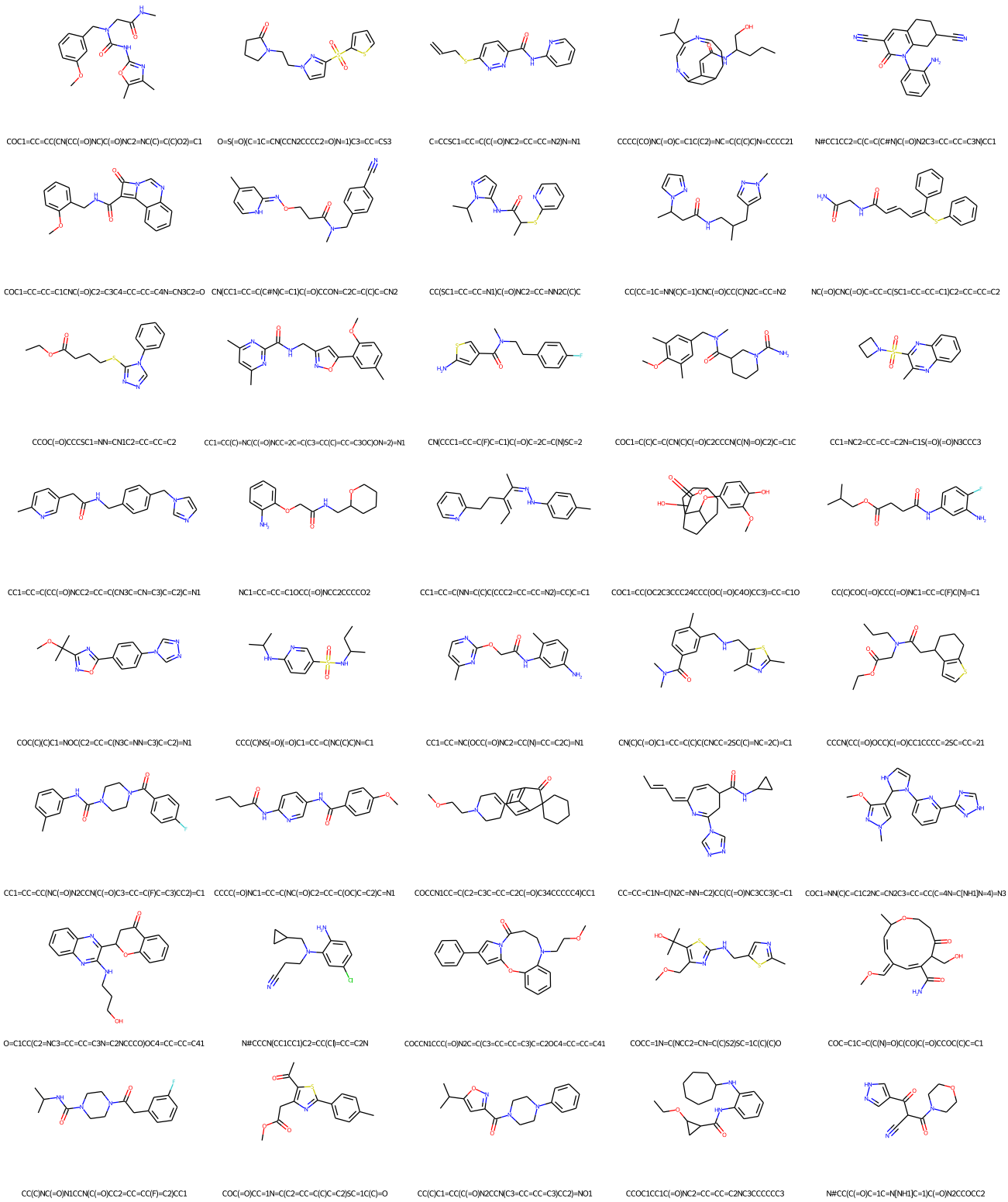


Figure 28. Generated samples from MAM-SELFIES: MOSES

Table 10. Character modeling on Molecular Sets

	NLL (bpc) ↓	Pearson ↑	Time (s) ↓
AO-ARM♦	0.655	0.994	19.32± 0.01
MAM♦	0.655	0.995	0.006±0.00

Table 11. Performance Comparison on MOSES

Model	**Valid↑**	**Unique 10k↑**	**Frag Test↓**	**Scaf TestSF↑**	**Int Div1↑**	**Int Div2↑**	**Filters↑**	**Novelty↑**
Training data	1.0	1.0	1.0	0.9907	0.8567	0.8508	1.0	1.0
HMM	0.076	0.5671	0.5754	0.049	0.8466	0.8104	0.9024	**0.9994**
NGram	0.2376	0.9217	0.9846	0.0977	**0.8738**	**0.8644**	0.9582	**0.9694**
CharRNN	**0.9748**	0.9994	**0.9998**	**0.1101**	**0.8562**	**0.8503**	**0.9943**	0.8419
JTN-VAE	**1.0**	**0.9996**	0.9965	**0.1009**	0.8551	0.8493	**0.976**	0.9143
MAM-SMILES	0.7192	**0.9999**	**0.9978**	**0.1264**	0.8557	0.8499	**0.9763**	**0.9485**
MAM-SELFIES	**1.0**	**0.9999**	**0.997**	0.0943	**0.8684**	**0.8625**	0.894	0.9155

C.5. Additional results on text8

C.5.1. Length extrapolation on Text8

In Table 7, we evaluated the model's ability to handle length extrapolation on Text8. We trained data with $D = 250$ and tested on sequences with $D = 300$.

Robustness MAM's predicted $\log p$ marginals maintain a high correlation with those calculated using AO-ARM conditionals, even on longer sequences. Its quality matches Parallel AO-ARM with 20 steps.

Graceful Extrapolation We observe the absolute errors in $\log p_\theta(x) - \log p_{\text{ARM}}(x)$ increase due to challenge from OOD prediction, but the variance of these errors remains surprisingly similar to that observed when $D = 250$. This indicates that MAM gracefully extrapolates the relative scales among $\log p$ values, explaining the high observed correlation in the Table.

C.5.2. Samples used for evaluating likelihood estimate quality

We show an example of a set of generated samples from masking different portions of the same text, which is then used for evaluating and comparing the likelihood estimate quality. Their log-likelihood calculated using the conditionals with the AO-ARM are in decreasing order. We use MAM marginal network to evaluate the log-likelihood and compare its quality with that of the AO-ARM conditionals.

Original text:

```
the subject of a book by lawrence weschler in one nine nine five entitled mr wilson s
cabinet of wonder and the museum s founder david wilson received a macarthur foundation
genius award in two zero zero three the museum claims to attract around six
```

Text generated from masking out 50 tokens:

```
 the_su_je_t of a b_ok_by_la_r_nce _es_h___ _n o___nine n_ne five entitled mr_wilson
s_cabinet of wonder and the museum s founder __vid w_l_o_ r__eive_ a macarthur fou__a__on
_e___s_awa_d in two _ero z_r_ _hree _he museum c_aims _o attr_ct ar_u_d s__
the subject of a book by lawrence heschell in one nine nine five entitled mr wilson s
cabinet of wonder and the museum s founder david wilson received a macarthur foundation
dennis award in two zero zero three the museum claims to attract around sev
```

Text generated from masking out 100 tokens:

```
_the_su_je_t _f __b__k__y_l__r_nc_ _es_h_____n o___nine n_ne five_
enti_l_d mr_wil_o_ __c_b__et of wond_r an_ _h_ mu_eu_ s f_u_der___vid
_w_l_______eive_ a_maca_thur f_u__a___n _e_____a_a_d __ two _er_ z_r_ _h_ee____ museum c_a_ms__o
__tr_ct ar_u__ ___
the subject of a book by lawrence bessheim in one nine nine five entitled mr wilson s
cabinet of wonder and the museum s founder david wilson received a macarthur foundation
leaven award in two zero zero three the museum claims to detract around the
```

Text generated from masking out 150 tokens:

```
_the__u____t__f __________l__r_nc_ _es_h_____n o___n__e n_ne__ive_
e_ti_l__m__wil______c____et of won___ an___________ s___u_der___vid_
w_________eiv__ a__a_a_th________a___n _e_____a_______ two__e__ z_r_ ___e_____ _use_m c_a_ms___
__tr_ct_a_____ ___
the tudepot of europe de laurence desthefs in one nine nine five entitled mr wild the
cabinet of wonder anne cedallica s founder david wright received arnasa the culmination
team sparked in two zero zero three the museum claims to retract athlet c a
```

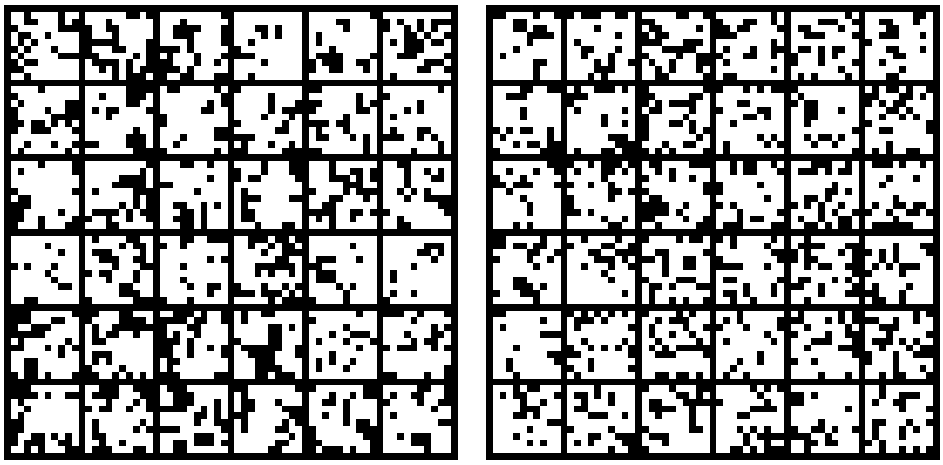


Figure 29. Samples: 10×10 Ising model. Ground truth (left) v.s. MAM (right).

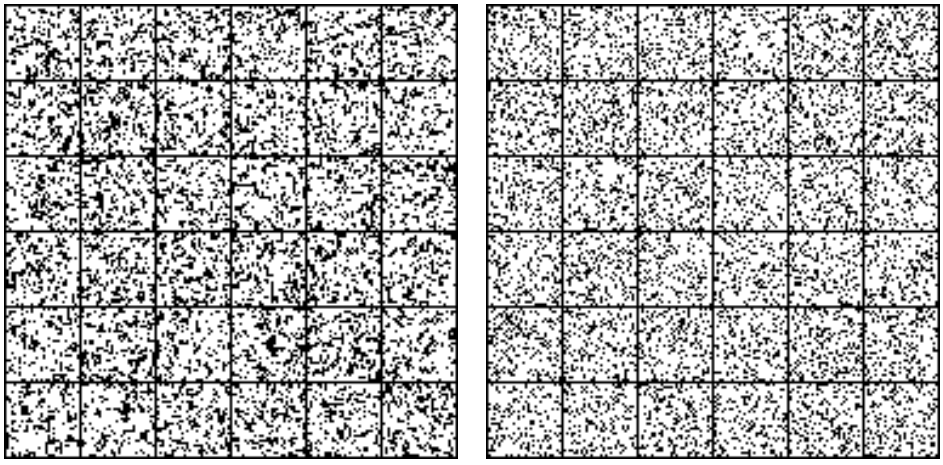


Figure 30. Samples: 30×30 Ising model. Ground truth (left) v.s. MAM (right).

Text generated from masking out 200 tokens:

```
_t__________f _________l_r____ ______ ____o_____e_n___iv__
e__i_l__ ___wil_____c____t___ w_____a_________________der____d_
w_____ ____e_______a_a_________a___n_e____a______t__________
___e_____ _u_e___c_a__s___ ___r_c__a_____ ___
the builder of the pro walter a a e sec press one nine nine five esciele the wild men
convert of wark flax notes the world undergroand whirl spiken america ascent and martin
decree a letter to the antler s default museum chafes in america ascent vis
```

C.6. Additional experiments on Ising model

Generated samples

We compare ground truth samples and MAM samples in Figure 29 and 30.

C.7. Additional experiments on molecule target generation

C.7.1. Target property energy-based training on lipophilicity (logP)

Figure 31 and 32 show the logP of generated samples of length $D = 55$ towards target values 4.0 and -4.0 under distribution temperature $\tau = 1.0$ and $\tau = 0.1$. For $\tau = 1.0$, the peak of the probability density (mass) appears around 2.0 (or -2.0) because there are more valid molecules in total with that logP than molecules with 4.0 (or -4.0), although a single molecule with 4.0 (or -4.0) has a higher probability than 2.0 (or -2.0). When the temperature is set to much lower ($\tau = 0.1$), the peaks concentrate around 4.0 (or -4.0) because the probability of logP value being away from 4.0 (or -4.0) quickly diminishes to zero. We additionally show results on molecules of length $D = 500$. In this case, logP values are shifted towards the target but their peaks are closer to 0 than when $D = 55$, possibly due to the enlarged molecule space containing more molecules with logP around 0. Also, this is validated by the result when $\tau = 0.1$ for $D = 500$, the larger design space allows for more molecules with logP values that are close to, but not precisely, the target value.

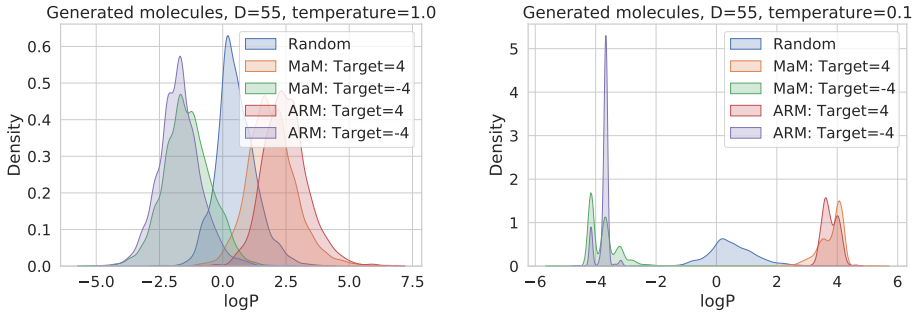


Figure 31. Target property matching with different temperatures. 2000 samples are generated for each method.

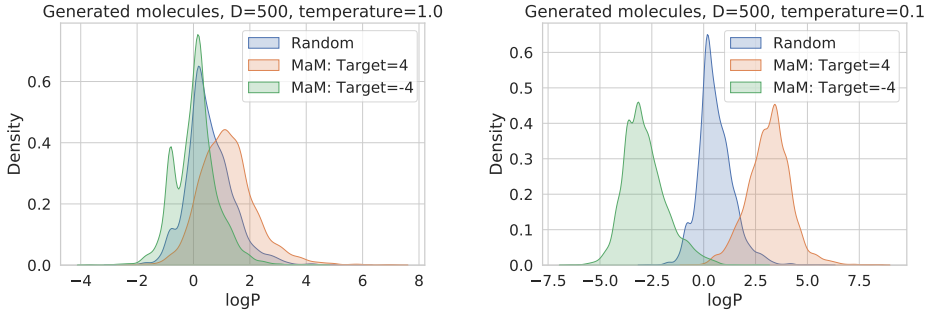


Figure 32. Target property matching with different temperatures. 2000 samples are generated for each method.

C.7.2. Conditionally generated samples

More samples from conditionally generating towards low lipophilicity (target $= -4.0$, $\tau = 1.0$) from user-defined substructures of Benzene. We are able to generate from any partial substructures with any-order generative modeling of MAM. Figure 33 shows conditional generation from masking out the left 4 SELFIES characters. Figure 34 shows conditional generation from masking the right $4 \sim 20$ characters.

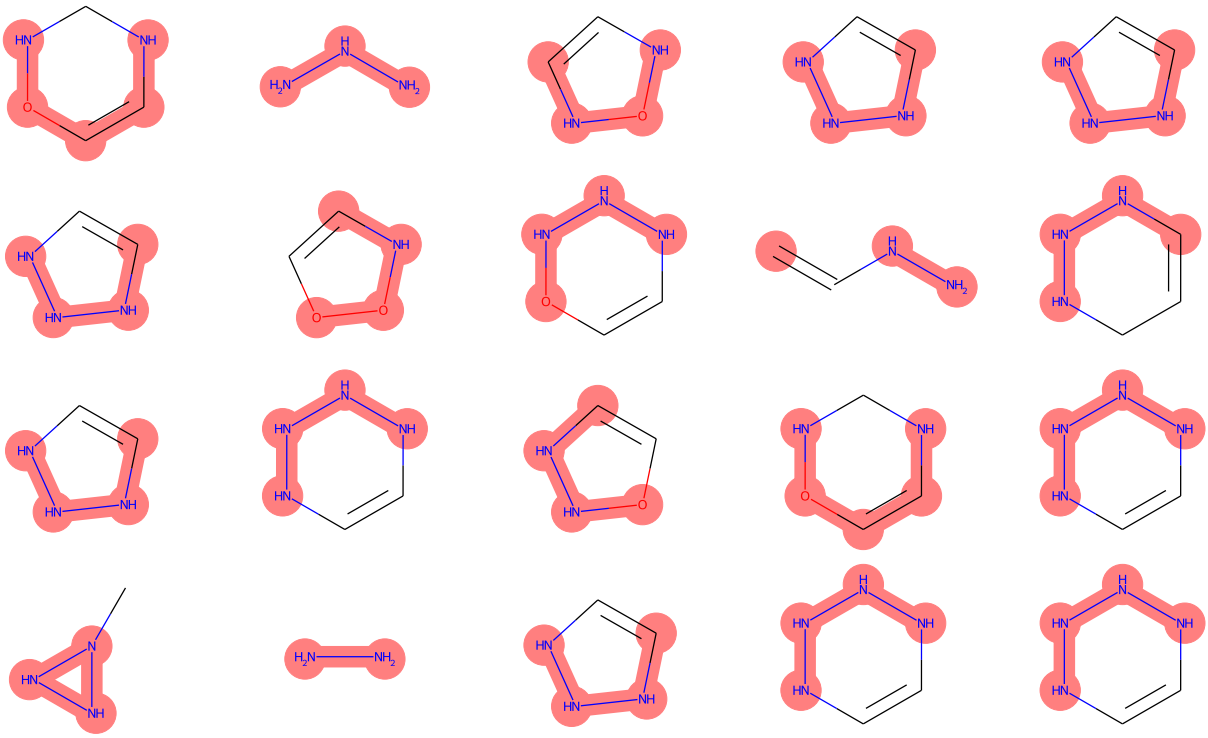


Figure 33. Generated samples from masking out the left 4 SELFIES characters of a Benzene. Shaded region are the impainted structures.

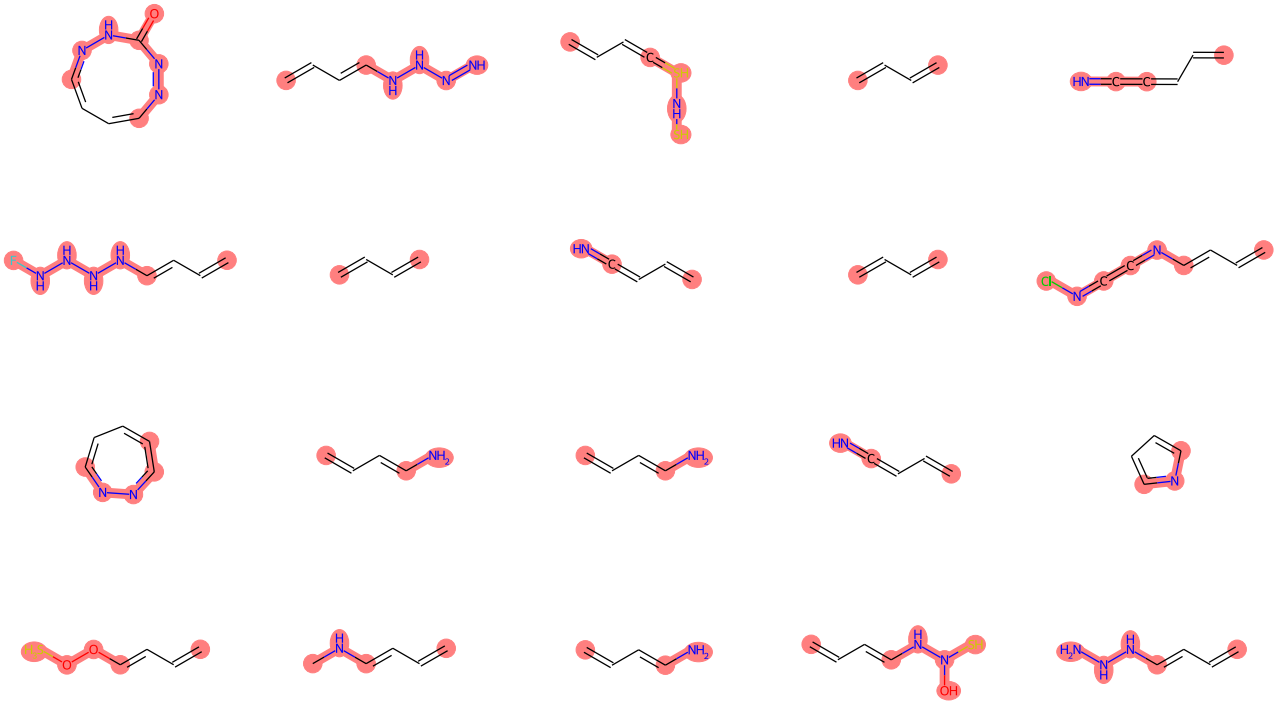


Figure 34. Generated samples from masking out the right 4-20 SELFIES characters of a Benzene. Shaded region are the impainted structures.