Generative Marginalization Models

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

Abstract

We introduce *marginalization models*, a new family of generative model for high-dimensional discrete data. They offer scalable and flexible generative modeling with tractable likelihoods through explicit modeling of all induced marginal distributions. Marginalization models enable fast evaluation of arbitrary marginal probabilities with a single forward pass of the neural network, which overcomes a major limitation of methods with exact marginal inference such as autoregressive models (ARMs). They also support scalable training for any-order generative modeling that previous methods fail to achieve under the setting of *distribution matching* to a given desired probability (specified by an unnormalized probability function such as energy function or reward function). We demonstrate the effectiveness of the proposed model on a variety of discrete data distributions, including binary images, language, physical systems, and molecules, on both likelihood maximization and distribution matching tasks. Marginalization models achieve orders of magnitude speedup in evaluation of the probability mass function. For distribution matching, marginalization models enable scalable training of any-order generative models that previous methods fail to achieve.

1 Introduction

Deep generative models have seen remarkable progress in multiple fields, encompassing image generation, audio syntheis, natural language modeling and science discovery. Most existing models do not support efficient probabilistic inference for key questions such as marginal probability $p(\mathbf{x}_s)$ and conditional probability $p(\mathbf{x}_u|\mathbf{x}_v)$, where s, u and v are disjoint subsets of the variables. The ability to answer such questions is important for many applications such as outlier detection (Ren et al., 2019; Mitchell et al., 2023), masked language modeling (Devlin et al., 2018; Yang et al., 2019), image inpainting (Yeh et al., 2017), and constrained protein/molecule design (Wang et al., 2022; Schneuing et al., 2022). Furthermore, the capacity to conduct such inference for arbitrary subsets of variables empowers users with control and flexibility in leveraging the model according to their specific needs and preferences. For instance, in protein design domain scientists may want to manually guide the generation of a protein from a user-defined secondary (local) structure under a particular path over the relevant variables, which is only feasible if the generative model can perform arbitrary marginal inference.

Towards this end, neural autoregressive models (ARMs) (Bengio & Bengio, 2000; Larochelle & Murray, 2011) are developed to facilitate conditional/marginal inference based on the idea of modeling a high-dimensional joint distribution as a factorization of univariate conditionals. Many efforts have been made to scale up ARMs and enable any-order generative modeling under the setting of maximum likelihood estimation (MLE) (Larochelle & Murray, 2011; Uria et al., 2014; Hoogeboom et al., 2021a), and great progress has been made in applications such as masked language modeling (Yang et al., 2019) and image inpainting (Hoogeboom et al., 2021a). However, marginal likelihood evaluation in most widely-used modern neural network architectures (e.g., Transformers (Vaswani et al., 2017) and U-nets (Ronneberger et al., 2015)) is limited by $\mathcal{O}(D)$ neural network passes, where D is the length of the sequence. This scaling makes it difficult to evaluate likelihoods on long sequences arising in data such as natural language and proteins. Additionally, in the setting of *distribution matching* (DM)—where a function such as a reward or energy can be evaluated pointwise and interpreted as an unnormalized (log) probability for the generative model to match—ARMs are limited to fixed-order generative modeling and lack scalability in training. The subsampling techniques developed for MLE to scale the training of conditionals are no longer applicable when matching log-probabilities. We provide an overview of generative modeling settings and ARMs in Appendix A and discuss the limitations of ARMs in Section 4.

To enable more scalability and flexibility in generative modeling of discrete data, we propose a new family of genera-

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

Accepted to ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling.

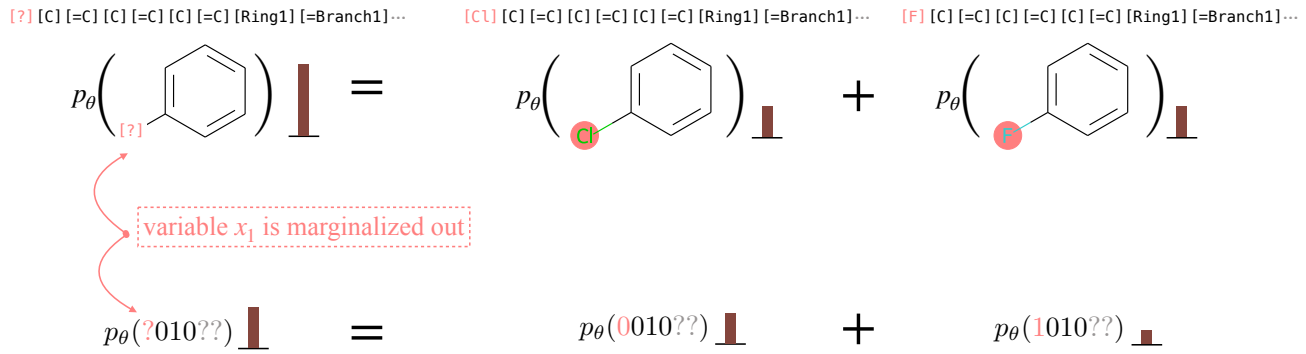

Figure 1. Marginalization models enable estimation of any marginal probability with a neural network θ that learns to "marginalize out" variables. The figure illustrates marginalization of a single variable on bit strings (representing molecules) with only two alternatives (versus K) for clarity. The bars represent probability masses.

tive 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 not only 1) scales up inference for any marginal, but also 2) enables any-order generative modeling and scalable training under both generative modeling 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 1); enforcing this constraint is one of the key contributions of this work.

We show that marginalization models can be trained under both maximum likelihood and distribution matching settings. We demonstrate the effectiveness of the proposed model in both settings on a variety of discrete data distributions, including binary images, text, physical systems, and molecules. We empirically demonstrate that marginalization models achieve orders of magnitude speedup in likelihood evaluation. For distribution matching, marginalization models enable scalable training of any-order generative models that previous methods fail to achieve.

2 Marginalization Models

We propose *marginalization models* (MaMs), a new type of generative model that enables both scalable any-order generative modeling and efficient marginal evaluation for both maximum likelihood and distribution matching. The flexibility and scalability of marginalization models are enabled by the explicit modeling of the marginal distribution and enforcing *marginalization self-consistency*.

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 (Weininger et al., 1989) and SELFIES (Krenn et al., 2020)). Moreover, vector representations are inherently applicable to any discrete problem, since it is feasible to encode any discrete object with a discrete random vector.

Definition We are interested in the discrete probability distribution $p(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_D)$ is a D-dimensional vector and each x_d takes K possible discrete values.

Marginalization Given a distribution $p(\mathbf{x})$, and $\mathbf{x} = [x_1, \ldots, x_D]$, where $x_d \in \{1, \ldots, K\}$. Let $\mathbf{x}_s$ be a subset of variables of $\mathbf{x}$, $\mathbf{x}_s \subseteq \{x_1, \ldots, x_D\}$. Denote the complement set of variables as $\mathbf{x}_{s^c} = \{x_1, \ldots, x_D\} \setminus \mathbf{x}_s$. The marginal of $\mathbf{x}_s$ is given by summing over all values of $\mathbf{x}_{s^c}$:

$$p(\mathbf{x}_s) = \textstyle\sum_{\mathbf{x}_{s^c}} p(\mathbf{x}_s, \mathbf{x}_{s^c}) \tag{1}$$

We refer to (1) as the *marginalization self-consistency* that any valid distribution should follow. The goal of a marginalization model θ is to estimate the marginals $p(\mathbf{x}_s)$ for any subset of variables $\mathbf{x}_s$ as closely as possible. To achieve this, we train a deep neural network p_θ that minimizes the distance of $p_\theta(\mathbf{x})$ and $p(\mathbf{x})$ on the full joint distribution while enforcing the marginalization self-consistency.

Parameterization To approximate arbitrary marginals over $\mathbf{x}_s$ with a single neural network forward pass, we additionally include the "marginalized out" variables $\mathbf{x}_{s^c}$ in the input by introducing a special symbol "?" to denote the missing values. By doing this, we create an augmented D-dimensional vector representation $\mathbf{x}_s^{\text{aug}} \in \mathcal{X}^{\text{aug}} \triangleq \{1, \ldots, K, ?\}^D$ and feed it to the NN. For example, for a binary vector of length 4, if $\mathbf{x}_s = \{x_1, x_3\}$ where $x_1 = 0$ and $x_3 = 1$, then $\mathbf{x}_s^{\text{aug}} = [0, ?, 1, ?]$ where "?" denotes x_2 and x_4 being marginalized out. From here onwards we will use $\mathbf{x}_s^{\text{aug}}$ and $\mathbf{x}_s$ interchangeably.

A marginalization model parameterized by a neural network θ takes in the augmented vector representation $\mathbf{x}_s^{\text{aug}} \in \{1, \ldots, K, ?\}^D$, and outputs the marginal log probability $f_\theta(\mathbf{x}_s) = \log p_\theta(\mathbf{x}_s)$ that must satisfy the marginalization

self-consistency constraints[1]:

$$\sum_{\mathbf{x}_{s^c}} p_\theta([\mathbf{x}_s, \mathbf{x}_{s^c}]) = p_\theta(\mathbf{x}_s) \quad \forall \mathbf{x}_s \in \{1, \dots, K, ?\}^D$$

where $[\mathbf{x}_s, \mathbf{x}_{s^c}]$ denotes the concatenation of $\mathbf{x}_s$ and $\mathbf{x}_{s^c}$. Given a random ordering of the variables $\sigma \in S_D$ where S_D defines the set of all permutations of $1, 2, \cdots, D$, the marginalization can be imposed over one variable at a time, which leads to the one-step marginalization constraints:

$$p_\theta(\mathbf{x}_{\sigma(<d)}) = \sum_{x_{\sigma(d)}} p_\theta(\mathbf{x}_{\sigma(\leq d)}), \quad (2)$$
$$\forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D].$$

Sampling Given the learned marginalization model, one can sample from the learned distribution by picking an arbitrary order σ and sampling one variable at a time. To evaluate the conditionals at each step of the generation, we can divide the marginal after the generation by the marginal before the generation. However, this can lead to invalid conditional if the marginalization self-consistency is not strictly enforced. Hence we use following normalized conditional:

$$p_\theta(x_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) = \frac{p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}])}{\sum_{x_{\sigma(d)}} p_\theta([\mathbf{x}_{\sigma(<d)}, x_{\sigma(d)}])}. \quad (3)$$

In this paper, we focus on the sampling procedure that generates one variable at a time, but marginalization models can also facilitate sampling multiple variables at a time in a similar fashion.

Learning marginals together with conditionals In training, we impose the marginalization self-consistency by minimizing the squared error of the constraints in (2). When the number of discrete values each x_d can take becomes large, each marginalization constraint in (2) requires K NN forward passes, which makes training challenging to scale when K is large. To address this issue, we augment the marginalization models with conditionals parameterized by ϕ. The marginalization constraints in (2) can be broken into K parallel marginalization constraints based on conditionals and marginals that allows subsampling the constraints during training:

$$p_\theta(\mathbf{x}_{\sigma(<d)}) p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) = p_\theta(\mathbf{x}_{\sigma(\leq d)}), \quad (4)$$
$$\forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D]. \quad (5)$$

And we denote the squared error penalty associated with a given marginalization self-consistency constraint to be:

$$\mathcal{L}^{\text{Mar}}_{\theta,\phi}(\mathbf{x}, d, \sigma) = \Big(\log p_\theta(\mathbf{x}_{\sigma(<d)}) + \log p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) - \log p_\theta(\mathbf{x}_{\sigma(\leq d)}) \Big)^2.$$

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

During training, we need to specify a distribution $q(\mathbf{x})$ for subsampling the marginalization constraints to optimize on. In practice, it can be set to the distribution we are interested to perform marginal inference on, such as p_{data} or the distribution of the generative model $p_{\theta,\phi}$.

3 Training the Marginalization Models

Maximum likelihood In this setting, we train MaM with the maximum likelihood objective in (9) while additionally enforcing the marginalization constraints in (2):

$$\max_{\theta,\phi} \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log p_\theta(\mathbf{x}) \quad (6)$$
$$\text{s.t.} \quad p_\theta(\mathbf{x}_{\sigma(<d)}) p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}) = p_\theta(\mathbf{x}_{\sigma(\leq d)}), \quad (7)$$
$$\forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D].$$

TWO-STAGE TRAINING A typical way to solve the above optimization problem is to convert the constraints into a penalty term and optimize the penalized objective, but we empirically found the learning to be slow and unstable. Instead, we identify an alternative two-stage optimization formulation that is theoretically equivalent to (6), but leads to more efficient training:

Theorem 3.1. *Solving the optimization problem in (6) is equivalent to the following two-stage optimization procedure, under mild assumption about the neural networks used being universal approximators:*
Stage 1: $\max_\phi \ \mathbb{E}_{\mathbf{x}} \mathbb{E}_\sigma \sum_{d=1}^D \log p_\phi \left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)} \right)$, *where* $\mathbf{x} \sim p_{data}$ *and* $\sigma \sim \mathcal{U}(S_D)$.
Stage 2: $\min_\theta \ \mathbb{E}_{\mathbf{x}} \mathbb{E}_\sigma \mathbb{E}_d \ \mathcal{L}^M_{\theta,\phi}(\mathbf{x}, d, \sigma)$, *where* $\mathbf{x} \sim q(\mathbf{x})$, $\sigma \sim \mathcal{U}(S_D)$ *and* $d \sim \mathcal{U}(1, \cdots, D)$.

The intuition is that the chain rule of probability implies that there is a one-to-one correspondence between optimal conditionals and marginals: $\log p_\theta(\mathbf{x}) = \sum_{d=1}^D \log p_\phi \left(x_{\sigma(d)} | \mathbf{x}_{\sigma(<d)} \right)$ for any σ and $\mathbf{x}$. By assuming neural networks are universal approximators, we can first optimize for the optimal conditionals, and then optimize for the corresponding optimal marginals. This is equivalent to first fitting an order-agnostic autoregressive model (Uria et al., 2014; Hoogeboom et al., 2021a) and then distilling that model into the marginalization network.

Distribution matching In this setting, we train MaM using the distribution matching objective in (10) with a penalty term to enforce the marginalization constraints in (2):

$$\min_{\theta,\phi} \quad D_{\text{KL}} \left(p_\theta(\mathbf{x}) || p(\mathbf{x}) \right) + \lambda \mathbb{E}_{\mathbf{x}} \mathbb{E}_\sigma \mathbb{E}_d \mathcal{L}^{\text{Mar}}_{\theta,\phi}(\mathbf{x}, d, \sigma),$$

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

SCALABLE TRAINING We use REINFORCE to estimate

the gradient of the KL divergence term:

$$\nabla_\theta D_{\mathrm{KL}}(p_\theta(\mathbf{x})||p(\mathbf{x})) \tag{8}$$
$$\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log p_\theta(\mathbf{x}^{(i)}) \left(\log p_\theta(\mathbf{x}^{(i)}) - \log f(\mathbf{x}^{(i)})\right)$$

For the penalty term, we subsample the ordering σ and step d for each data $\mathbf{x}$.

PERSISTENT MCMC SAMPLING We need cheap and effective samples from p_θ in order to perform REINFORCE, so a persistent set of Markov chains are maintained by randomly picking an ordering and taking block Gibbs sampling steps using the conditional distribution $p_\phi(\mathbf{x}_{\sigma(d)}|\mathbf{x}_{\sigma(<d)})$ (algorithm in Appendix C), in similar fashion to persistent contrastive divergence used in training RBMs (Tieleman, 2008). The samples from the conditional distribution p_ϕ serve as approximate samples from p_θ when they are close to each other. Otherwise, we can additionally use importance sampling for adjustment.

4 Addressing Limitations of ARMs

We discuss in more detail about how MaMs address some limitations of ARMs. The first one is general to both settings, while the latter two are specific to distribution matching.

Non-scalable evaluation of likelihoods: Due to sequential conditional modeling, evaluation of any arbitrary marginal $p(\mathbf{x}_o)$ with ARMs requires applying the NN up to D times, which is inefficient in time and memory for high-dimensional data. In comparison, MaMs are able to estimate any arbitrary marginal with one NN forward pass.

Lack of support for any-order training The training objectives in distribution matching measures the distance in terms of $\log p_\phi(\mathbf{x})$ and $\log p(x)$. However, unless the model is perfectly self-consistent, it will not be the case that $\log p_\phi(\mathbf{x}) = \mathbb{E}_\sigma \log p_\phi(\mathbf{x}|\sigma)$. In particular, we would not expect it to work to have an objective that takes an expectation over orderings σ and uses $\log p_\phi(\mathbf{x}|\sigma)$ to match the target $\log p(\mathbf{x})$, i.e., $\mathbb{E}_\sigma \mathbb{E}_{p_\phi(\cdot|\sigma)} d(p_\phi(\cdot|\sigma), p) \neq \mathbb{E}_{p_\phi} d(p_\phi, p)$. The MaM self-consistency constraint addresses this issue, while ARMs need to be trained with a preset order to minimize the distance between $\log p_\phi(\mathbf{x}|\sigma)$ and the target density $\log p(\mathbf{x})$. MaMs are not limited to fixed ordering because marginals are order-agnostic and we can optimize over expectation of orderings for the marginalization self-consistency constraints.

Non-scalable training When minimizing the difference between $\log p_\phi(\mathbf{x}|\sigma)$ and the target $\log p(\mathbf{x})$, ARMs need to sum conditionals to evaluate $\log p_\phi(\mathbf{x}|\sigma)$. One might consider subsampling one-step conditionals $p_\phi(x_{\sigma(d)}|\mathbf{x}_{\sigma(<d)}$ to estimate $p_\phi(\mathbf{x})$, but this leads to high variance of the REINFORCE gradient in (8) due to the product of the score function and distance terms, which are both high variance (We validate this in experiments, see Figure 2). Consequently, training ARMs for distribution matching necessitates a sequence of D conditional evaluations to compute the gradient of the objective function. This constraint leads to an effective batch size of $B \times D$, significantly limiting the scalability of ARMs to high-dimensional problems. Furthermore, obtaining Monte Carlo samples from ARMs for the REINFORCE gradient estimator is slow when the dimension is high. Due to the fixed input ordering, this process requires D sequential sampling steps, making more cost-effective sampling approaches like persistent MCMC infeasible. Marginalization models circumvent this challenge by directly estimating the log-likelihood with the marginal neural network. Additionally, the support for any-order modeling enables an efficient sampling strategy during training through the utilization of persistent MCMC methods.

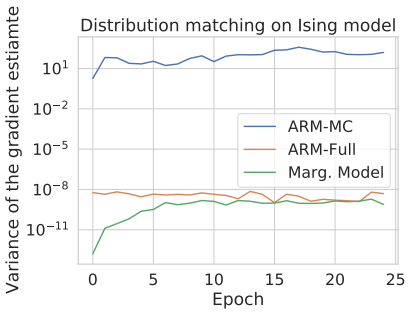

Figure 2. Approximating $\log p_\phi(\mathbf{x})$ with one-step conditional (ARM-MC) results in extremely high gradient variance during DM training.

5 Experiments

We conduct experiments with marginalization models (MaM) on both MLE and DM settings for discrete problems including binary images, text, molecules and phyiscal systems. We consider the following baselines: Any-order ARM (AO-ARM) (Hoogeboom et al., 2021a), ARM (Larochelle & Murray, 2011), GFlowNet (Bengio et al., 2021a; Zhang et al., 2022), and Discrete Flow (Tran et al., 2019)[2]. Only MaM and (AO-)ARM support marginal inference, so ARM will be the major focus of comparison. Discrete flow allows exact likelihood evaluation while GFlowNet needs to approximate the likelihood with sum using importance samples. For evaluating AO-ARM, we can either use it as an ensemble model (AO-ARM-E) by averaging the likelihood estimate from several random orderings or as a single model (AO-ARM-S) that picks a single random ordering.

Maximum likelihood estimation We report the negative test likelihood (bits/digit), likelihood estimate quality

[2]Results are only reported on text8 for discrete flow since there is no public code implementation.

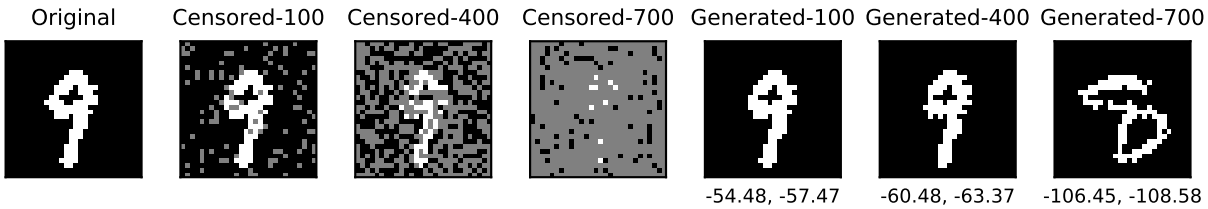

Figure 3. An example of the data generated (with $100/400/700$ pixels masked) for comparing the quality of likelihood estimate. Numbers below the images are LL estimates from MaM's marginal network (left) and AO-ARM-E's ensemble estimate (right).

and likelihood inference time per minibatch for **BINARY-MNIST**, and additionally on **MOLECULAR SET** and **TEXT8** in Appendix D. For MaM, we use the conditional network to evaluate the likelihood on test data, as marginals are not strictly valid and thus cannot be directly compared in terms of numerical values.

BINARY-MNIST In order to evaluate the quality of marginal likelihood estimates, we employ a controlled experiment where we randomly mask out portions of a test image and generate multiple samples with varying levels of masking (see Figure 3 for a visualization). This process allows us to obtain a set of distinct yet comparable samples, each associated with a different likelihood value. The likelihood evaluation results are shown in Table 1. For each model, we evaluate the likelihood of the generated samples and compare that with AO-ARM-Ensemble's estimate since it achieves the best likelihood on test data. We report the Pearson correlation and inference time of the likelihood estimates from the given model against that from AO-ARM-Ensemble. MaM achieves close to 4 order of magnitude speed-up while only at slightly worse or comparable quality.

Model	NLL (bpd) ↓	Pearson ↑	LL inf. time (s) ↓
AO-ARM-E-UNet	**0.148**	1.0	661.98 ± 0.49
AO-ARM-S-UNet	0.149	**0.993**	132.40 ± 0.03
MaM-UNet	0.149	**0.993**	**0.018 ± 0.00**
GflowNet-MLP	0.189	–	–

Table 1. Performance Comparison on Binary-MNIST

Distribution matching training We compare with ARM that uses sum of fixed-ordering conditionals to evaluate the log likelihood for matching. We additionally demonstrate the failure of ARM-MC (in Appendix E), which uses a one-step conditional to estimate the log-likelihood. ARM can be regarded as the golden standard of learning autoregressive conditionals, since its gradient is the most informative. MaM uses marginal network to evaluate $\log p_\theta(\mathbf{x})$ and subsamples a one-step marginalization constraint to minimize for each $\mathbf{x}$ in the batch. The effective batch size for ARM and GFlowNet is $B \times \mathcal{O}(D)$ for batch of size B, and $B \times \mathcal{O}(1)$ for ARM-MC and MaM . MaM and ARM optimizes KL divergence using REINFORCE with baseline. GFlowNet minimizes squared distance (Zhang et al., 2022).

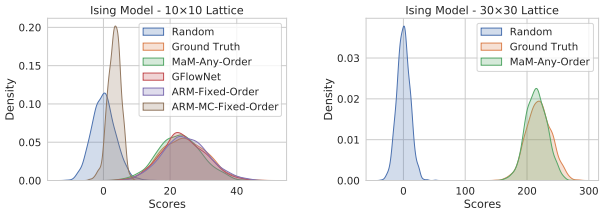

Figure 4. Ising model.

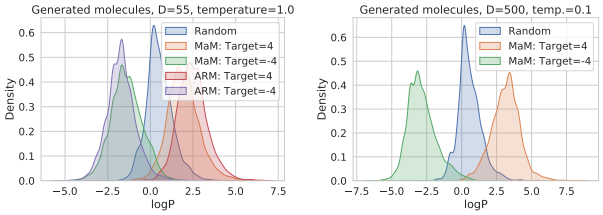

Figure 5. Molecular target property matching.

ISING MODEL Ising models (Ising, 1925) model interacting spins and are widely studied in mathematics and physics (see MacKay (2003)). The spins of the D sites are represented a D-dimensional binary vector and its distribution is $p^*(\mathbf{x}) \propto \exp\left(-\mathcal{E}_J(\mathbf{x})\right)$ where $\mathcal{E}_\mathbf{J}(\mathbf{x}) \triangleq -\mathbf{x}^\top \mathbf{J}\mathbf{x} - \boldsymbol{\theta}^\top \mathbf{x}$, with $\mathbf{J}$ the binary adjacency matrix.

Figure 4 plots the negative energies of generated samples. As described in Section 4, the ARM-MC gradient suffers from high variance and fails to converge. MaM is able to match the distribution fairly well and works for much larger dimension (30×30) while ARM fails to scale.

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 $p^*(\mathbf{x}) \propto \exp(-(g(\mathbf{x}) - g^*)^2/\tau)$, 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 5 (Table 8 and additional figures in Appendix E.4). MaM is able to match the distribution fairly well as compared to ARM but with a small gap. However, MaMs support flexible any-order modeling (see Figure 18) and works for much larger dimension ($D = 500$ in this example) that ARMs/GFlowNets fail to scale to.

References

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.

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, 2021a.

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.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. GFlowNet foundations. *arXiv preprint arXiv:2111.09266*, 2021b.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

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

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

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.

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

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.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

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

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.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

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.

Hoogeboom, E., Gritsenko, A. A., Bastings, J., Poole, B., Berg, R. v. d., and Salimans, T. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021a.

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, 2021b.

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

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

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

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.

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.

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.

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.

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.

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.

MacKay, D. J. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

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.

Mahoney, M. Large text compression benchmark, 2011.

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.

Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

OpenAI. ChatGPT: Language model for conversational ai. https://openai.com/blog/chatgpt, 2023.

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.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, 30, 2017.

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.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 689–690. IEEE, 2011.

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.

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.

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.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538. PMLR, 2015.

Rippel, O. and Adams, R. P. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

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.

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.

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.

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.

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.

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.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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

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.

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.

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.

Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475. PMLR, 2014.

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.

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.

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.

Wu, D., Wang, L., and Zhang, P. Solving statistical mechanics using variational autoregressive networks. *Physical review letters*, 122(8):080602, 2019.

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.

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.

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.

A Background

We first review two prevalent generative modeling settings. Then we introduce autoregressive models and how they can be trained for any-order modeling in maximum likelihood estimation.

Maximum likelihood (MLE) Given a dataset $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ drawn from a data distribution $p = p_{\text{data}}$, we aim to learn the distribution $p_\theta(\mathbf{x})$ that maximizes the probability of the data under our model. Mathematically, we aim to learn the parameters $\theta^\star$ that maximize the log-likelihood:

$$\theta^\star = \arg\max_\theta \ \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log p_\theta(\mathbf{x})\right] \approx \arg\max_\theta \ \tfrac{1}{N}\textstyle\sum_{i=1}^N \log p_\theta(\mathbf{x}^{(i)}) \tag{9}$$

which is also 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 (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song & Ermon, 2019)) and language (e.g., BERT (Devlin et al., 2018), GPT (Radford et al., 2019)) where we can empirically draw observed data from the distribution.

Distribution matching (DM) In this setting, we do not have data from the distribution of interest. Instead, we have access to the unnormalized (log) probability mass function f, usually in the form of reward function or energy function, that are defined by us or by physical systems to specify how likely a sample is. For example, we can define the target PMF to be $f(\mathbf{x}) = \exp(r(x)/\tau)$ where $r(x)$ is the reward function and $\tau > 0$ is a temperature parameter. This expresses the intuitive idea that we would like the model to assign higher probability to data with larger reward. For example, in conversation systems such as ChatGPT (Ouyang et al., 2022; OpenAI, 2023), $r(x)$ can express how well a response fits the user's preference. In molecular design applications, scientists can specify the reward according to how close a particular sample's measured or calculated properties are to some functional desiderata.

Mathematically, we aim to learn the parameters θ such that $p_\theta(\mathbf{x}) \approx f(\mathbf{x})/Z$, where Z is the normalization constant of f. Two common training criteria are to minimize the KL divergence (Noé et al., 2019; Wu et al., 2019; Damewood et al., 2022):

$$\min_\theta \ D_{\text{KL}}\left(p_\theta\left(\mathbf{x}\right) \middle\| f(\mathbf{x})/Z\right) = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\log p_\theta\left(\mathbf{x}\right) - \log f(\mathbf{x})/Z\right] \tag{10}$$

or the squared distance between the two log probabilities over a distribution of interest $q(\mathbf{x})$ (Bengio et al., 2021a; Zhang et al., 2022):

$$\min_\theta \ \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\left[\log p_\theta\left(\mathbf{x}\right) - \log f(\mathbf{x})/Z\right]^2 . \tag{11}$$

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

$$\log p(\mathbf{x}) = \textstyle\sum_{d=1}^D \log p\left(x_d \mid \mathbf{x}_{<d}\right), \tag{12}$$

where $\mathbf{x}_{<d} = \{x_1, \ldots, x_{d-1}\}$. Recently there has been great success in applying autoregressive models to discrete data, such as natural language, proteins (Shin et al., 2021; Lin et al., 2023; Madani et al., 2023), and molecules (Segler et al., 2018; Flam-Shepherd et al., 2022). Due to their sequential nature, evaluation of (joint/marginal) likelihood requires up to D neural network evaluations, which is costly for long sequences.

Any-order ARMs (AO-ARMs) Under the MLE setting, Uria et al. (2014) propose to learn the conditionals of ARMs for arbitrary orderings $\sigma \in S_D$, where S_D denotes the set of all permutations of $\{1, \ldots, D\}$. The model ϕ can be trained by maximizing a lower-bound objective (Uria et al., 2014; Hoogeboom et al., 2021a) that takes an expectation under a uniform distribution on orderings. This objective allows scalable training of AO-ARMs, leveraging efficient parallel evaluation of multiple one-step conditionals in one forward pass with architectures such as the U-Net (Ronneberger et al., 2015) and Transformers (Vaswani et al., 2017). However, training of AO-ARMs remains a challenge under the DM setting, which we will discuss in details in Section 4.

B Related Work

Autoregressive models Autoregressive models (ARMs) decompose a joint probability distribution into a sequence of conditional probabilities (Bengio & Bengio, 2000; Larochelle & Murray, 2011). Recent developments in deep learning have greatly advanced the performance of ARMs across different modalities, including images, audio, and text. Any-order

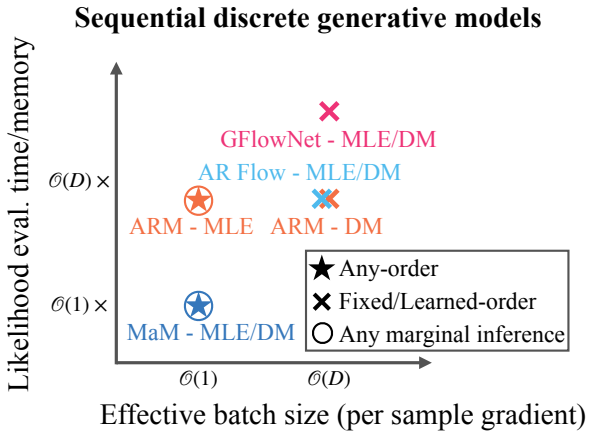

Figure 6. Scalability of sequential discrete generative models. The y-axis unit is # of NN forward passes.

(Order-agnostic) ARMs were first introduced in (Uria et al., 2014) by training with the any-order lower-bound objective for the maximum likelihood setting and recently seen in ARDM (Hoogeboom et al., 2021a) with state-of-the-art performance for any-order discrete modeling of image/text/audio.

Discrete diffusion models Discrete diffusion models learn to denoise from a latent base distribution into the data distribution. Sohl-Dickstein et al. (2015) first proposed diffusion for binary data and was extended in Hoogeboom et al. (2021b) for categorical data and both works adds uniform noise in the diffusion process. A wider range of transition distributions was proposed in Austin et al. (2021) and insert-and-delete diffusion processes have been explored in Johnson et al. (2021). Hoogeboom et al. (2021a) explored the connection between ARMs and diffusion model 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 (Rippel & Adams, 2013; Tabak & Turner, 2013; Dinh et al., 2014; Sohl-Dickstein et al., 2015; Rezende & Mohamed, 2015; Dinh et al., 2016; Kingma et al., 2016; Papamakarios et al., 2017). They have been extended to discrete data (Tran et al., 2019; Hoogeboom et al., 2019) 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.

GFlowNets GFlowNets (Bengio et al., 2021a;b) formulate the problem of generation as matching the probability flow at terminal states to the target normalized density. Zhang et al. (2022) proposes to apply GFlowNets to discrete data and additionally train an energy function from data. GFlowNets assume certain generation paths through a DAG, which limits its flexibility in sampling/generation. Compared to autoregressive models, this approach does not scale as well during training and its exact likelihood evaluation is intractable. We note that GFlowNet training also enforces a local detailed balance condition that is similar in spirit to the marginalization self-consistency presented here.

Probabilistic circuits Probabilistic circuits were recently proposed as a framework for tractable probabilistic models that unify tractable probabilistic models, including Chow-Liu trees (Chow & Liu, 1968), arithmetic circuits (Darwiche, 2003), sum-product networks (Poon & Domingos, 2011), etc. This type of model uses circuits that follow smoothness and decomposibilty structural properties such that probability densities/masses are tractable. The expressiveness of the model is limited by the allowed circuit structures.

C Additional Technical Details

C.1 Proof of Theorem 3.1

Proof. From the single-step marginalization self-consistency in (5), we have

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

Therefore we can rewrite the optimization in (6) as:

$$\begin{aligned} \max_{\phi} \quad & \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \mathbb{E}_{\sigma\sim\mathcal{U}(S_D)} \sum\nolimits_{d=1}^{D} \log p_\phi \left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right) \\ \text{s.t.} \quad & p_\theta(\mathbf{x}_{\sigma(<d)}) p_\phi(\mathbf{x}_{\sigma(d)} | \mathbf{x}_{\sigma(<d)}) = p_\theta(\mathbf{x}_{\sigma(\leq d)}), \ \forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D]. \end{aligned} \tag{13}$$

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

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

Then p^* is also the optimal solution to (13) the marginalization constraints are automatically satisfied by p^* since it is a valid distribution. From the universal approximation theorem (Hornik et al., 1989; Hornik, 1991; Cybenko, 1989), we can use separate neural networks to model p_θ (marginals) and p_ϕ (conditionals), and obtain optimal solution to (13) 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 (13):

$$p_{\phi^*} \left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right) = p^* \left(x_{\sigma(d)} \mid \mathbf{x}_{\sigma(<d)}\right), \quad \forall \, \mathbf{x}, \sigma \tag{14}$$

$$p_{\theta^*}(\mathbf{x}_s) = p^*(\mathbf{x}_s) Z_{\theta^*}, \quad \forall \, \mathbf{x}, s \subseteq \{1, \cdots, D\} \tag{15}$$

$$p_{\theta^*}(\mathbf{x}_{\sigma(<d)}) p_{\phi^*}(\mathbf{x}_{\sigma(d)} | \mathbf{x}_{\sigma(<d)}) = p_{\theta^*}(\mathbf{x}_{\sigma(\leq d)}), \ \forall \sigma \in S_D, \mathbf{x} \in \{1, \cdots, K\}^D, d \in [1:D] \tag{16}$$

where Z_{θ^*} is the normalization constant of p_{θ^*} and is equal to $p_{\theta^*}([?? \cdots ?])$. 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 (14). Then we can obtain the optimal θ^* by solving for condition (16) given the optimal conditionals ϕ^*:

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

□

C.2 Algorithms

We present the algorithms for training MaM for maximum likelihood and distribution matching settings in Algorithm 1 and Algorithm 2.

D Experiment Details for Maximum Likelihood Estimation

D.1 Dataset details

Binary MNIST Binary MNIST is a dataset introduced in (Salakhutdinov & Murray, 2008) that stochastically set each pixel to 1 or 0 in proportion to its pixel intensity. We use the training and test split of Salakhutdinov & Murray (2008) provided in https://github.com/yburda/iwae/tree/master. (Burda et al., 2015).

Algorithm 1 MaM MLE training

Input: Data $\mathcal{D}_{\text{train}}$, $q(\mathbf{x})$, network θ and ϕ
Stage 1: Train ϕ with the lower bound objective used in AO-ARM (Uria et al., 2014; Hoogeboom et al., 2021a)
for minibatch $\mathbf{x} \sim \mathcal{D}_{\text{train}}$ **do**
  Sample $\sigma \sim \mathcal{U}(S_D)$, $d \sim \mathcal{U}(1, \cdots, D)$
  $\mathcal{L} \leftarrow \sum_{j \in \sigma(\geq d)} \log p_\phi \left(x_j | \mathbf{x}_{\sigma(<d)} \right)$
  $\mathcal{L} \leftarrow \frac{D}{D-d+1} \mathcal{L}$
  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 (5)
  Update θ with gradient of $\mathcal{L}$
end for

Algorithm 2 MaM DM training

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 $\sigma \sim \mathcal{U}(S_D)$
  Sample $\mathbf{x}' \sim p_\phi(\mathbf{x}_{\sigma(\leq M)} | \mathbf{x}_{\sigma(>M)})$
  {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 (5), 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
  $\mathbf{x} \leftarrow \mathbf{x}'$
end for

To evaluate the quality of the likelihood estimates, we employ a controlled experiment where we randomly mask out portions (100, 400, and 700 pixels) of a test image and generate multiple samples with varying levels of masking (refer to Figure 3). We repeat this for 160 (randomly subsampled) test images and created a dataset of 640 sets of comparable images. To further test the quality of the marginal likelihood estimates on partially observed images, we curate a dataset of 160 sets of partial test images ($7 \sim 9$ images in each set) by randomly subsampling from the test set and masking the upper half of the images. To make sure the partial images are comparable but different in their log-likelihood, in each set, we remove samples that have a log-likelihood close to another sample within the threshold of 5.0.

Molecular Sets The molecules in MOSES are represented either in SMILES (Weininger et al., 1989) or SELFIES (Krenn et al., 2020) 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.

D.2 Training details

Binary MNIST

- 0, 1 and "?" are represented by a scalar value ("?" takes the value 0) and additionally a mask indicating if it is a "?".
- U-Net with 4 ResNet Blocks 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 28×28 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 Hoogeboom et al. (2021a) 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 finetune not only on the

additional MLP but also on the downsampling blocks to get a good estimate of the marginal probability, which shows marginal network and conditional network rely on different features to make the final prediction.

- Batch size is 128, Adam is used with learning rate 0.0001. Gradient clipping is set to 100. Both AO-ARM and MaM conditionals are trained for 600 epochs. The MaM marginals are finetuned from the trained conditionals for 100 epochs.

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 "?" 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.

Compute

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

D.3 Binary MNIST

We report the negative test likelihood (bits/digit), likelihood estimate quality and likelihood inference time per minibatch (of size 16) in Table 2. To keep GPU memory usage the same, we sequentially evaluate the likelihood for ARMs. Both MaM and AO-ARM use a U-Net architecture with 4 ResNet Blocks interleaved with attention layers. GFlowNets fail to scale to large architectures as U-Net, hence we report GFlowNet results using an MLP from Zhang et al. (2022). For MaM, we use the conditional network to evaluate the likelihood on test data, since marginals are not strictly valid but only approximations. The marginal network is used for evaluating likelihood quality and inference time.

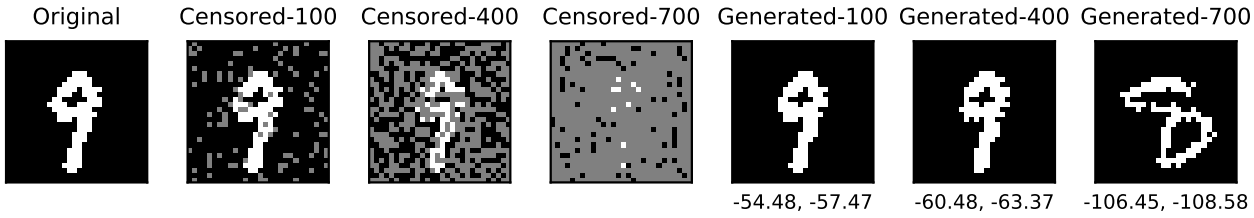

Figure 7. An example of the data generated (with $100/400/700$ pixels masked) for comparing the quality of likelihood estimate. Numbers below the images are LL estimates from MaM's marginal network (left) and AO-ARM-E's ensemble estimate (right).

Evaluating marginal likelihood estimate quality

In order to evaluate the quality of marginal likelihood estimates, we employ a controlled experiment where we randomly mask out portions of a test image and generate multiple samples with varying levels of masking (in Figure 7). This process allows us to obtain a set of distinct yet comparable samples, each associated with a different likelihood value. For each model, we evaluate the likelihood of the generated samples and compare that with AO-ARM-Ensemble's estimate since it achieves the best likelihood on test data. We report the Spearman's correlation and Pearson correlation of the likelihood estimates from the given model against that from AO-ARM-Ensemble. MaM achieves close to 4 order of magnitude speed-up while only at slightly worse quality.

Table 2. Performance Comparison on Binary-MNIST

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

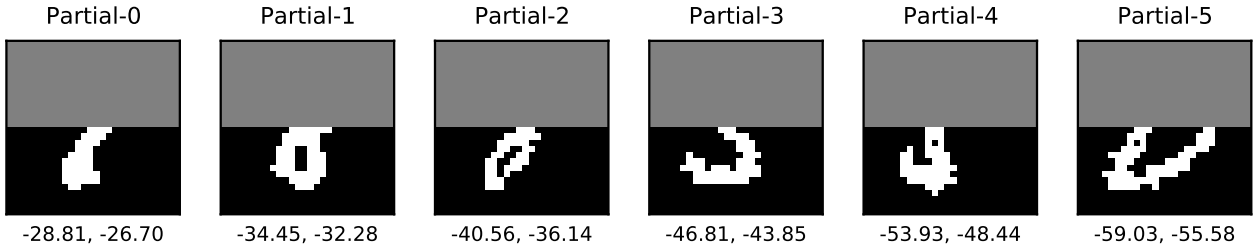

Figure 8. An example set of partial images for evaluating marginal likelihood estimate quality. The numbers in the captions show the log-likelihood calculated using learned marginals (left) v.s. learned conditionals (right)

Table 3. Performance Comparison on Binary-MNIST partial images

Model	Spearman's ↑	Pearson ↑	LL inference time (s) ↓
AO-ARM-E-U-Net	1.0	1.0	248.96 ± 0.14
AO-ARM-S-U-Net	**1.0**	**0.997**	49.75 ± 0.03
MaM-U-Net	0.998	0.995	**0.02 ± 0.00**

We addtionally evaluate the m arginal Likelihood estimate on partial Binary MNIST images. Figure 8 illustrates an example set of partial images that we evaluate and compare likelihood estimate from MaM against ARM. Table 3 contains the comparison of the marginal likelihood estimate quality and inference time.

Generated image samples

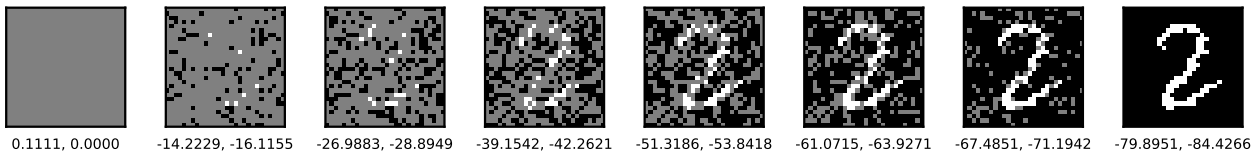

Figure 9. An example of the trajectory every 112 step when generating an MNIST digit following a random order. The future pixels are generated by conditioning on the existent filled-in pixels. The numbers in the captions show the log-likelihood calculated using learned marginals (left) v.s. learned conditionals (right)

Figure 9 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 10.

D.4 Molecular Sets

We test generative modeling of MaM on a benchmarking molecular dataset (Polykovskiy et al., 2020) refined from the ZINC database (Sterling & Irwin, 2015). Same metrics are reported as Binary-MNIST. Likelihood quality is measured similarly but on random groups of test molecules instead of generated ones. The generated molecules from MaM and AO-ARM are comparable to standard state-of-the-art molecular generative models, such as CharRNN (Segler et al., 2018), JTN-VAE (Jin

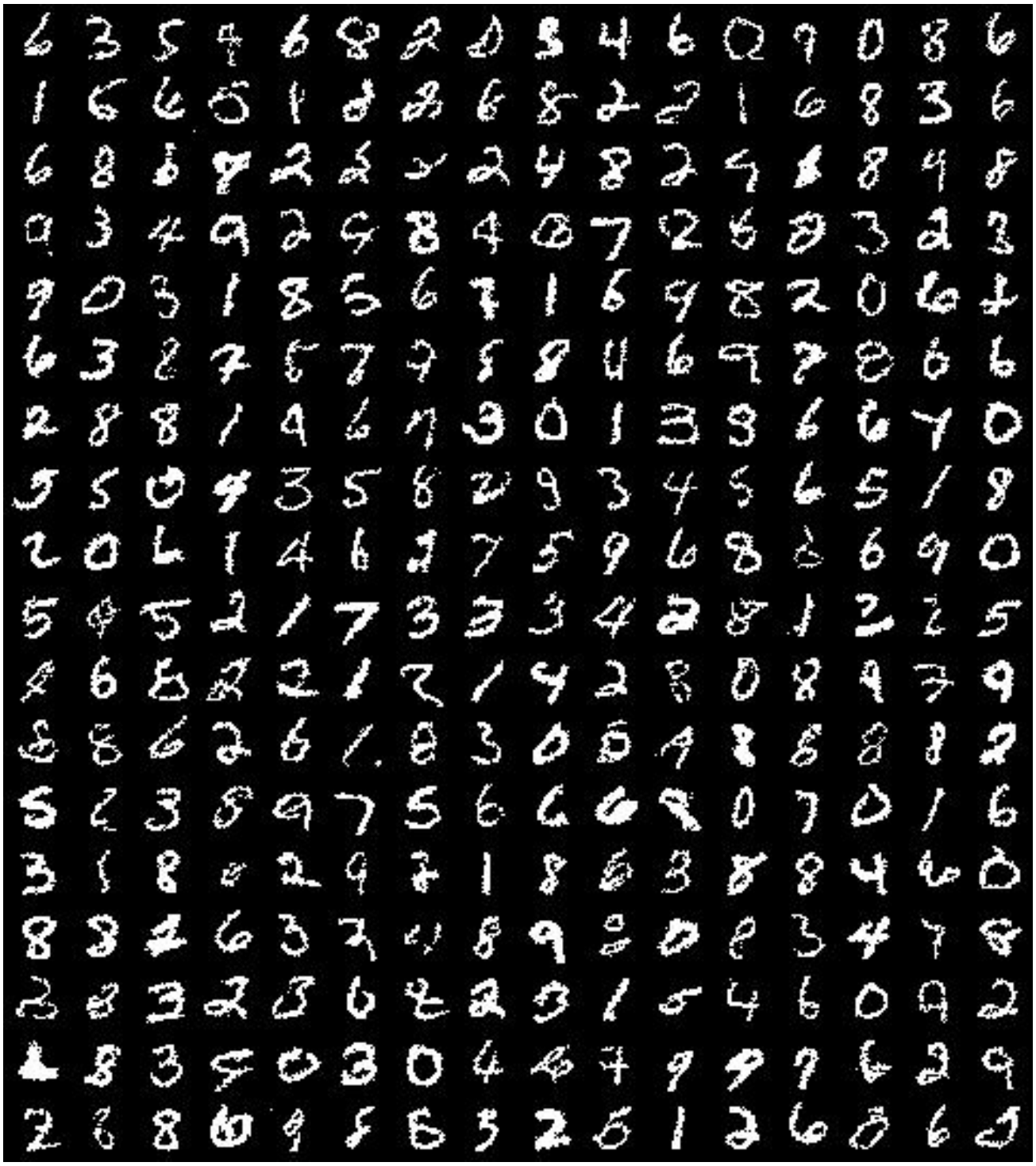

Figure 10. Generated samples: Binary MNIST

et al., 2018), and LatentGAN (Prykhodko et al., 2019) (in Table 5 and Figure 11), with additional controllability and flexibility in any-order generation. MaM supports much faster marginal inference, which is useful for domain scientists to reason about likelihood of substructures.

Table 4. Performance Comparison on Molecular Sets

Model	**NLL (bpd)** ↓	**Spearman's** ↑	**Pearson** ↑	**LL inf. time (s)** ↓
AO-ARM-E-Transfomer	**0.652**	1.0	1.0	96.87± 0.04
AO-ARM-S-Transfomer	0.655	0.996	0.994	19.32± 0.01
MaM-Transfomer	0.655	**0.998**	**0.995**	**0.006±0.00**

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 5 and Figure 11. Details of the baseline methods are provided in Polykovskiy et al. (2020). 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).

Table 5. Performance Comparison on MOSES

Model	**Valid**↑	**Unique 10k**↑	**Frag Test**↓	**Scaf TestSF**↑	**Int Div1**↑	**Int Div2**↑	**Filters**↑	**Novelty**↑
Train	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

Generated molecule samples

Figure 12 and 13 plot the generated molecules from MaM-SMILES and MaM-SELFIES.

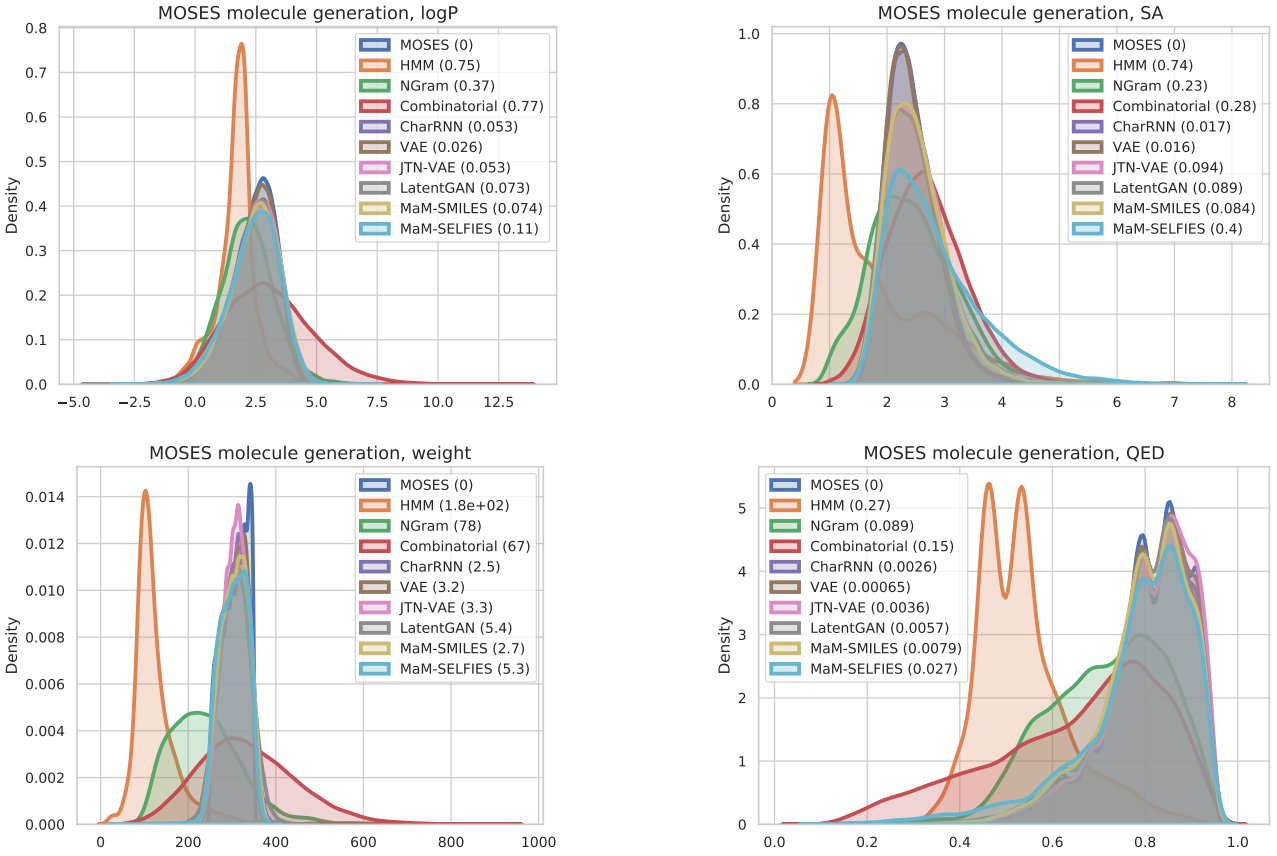

Figure 11. KDE plots of lipophilicity (logP), Synthetic Accessibility (SA), Quantitative Estimation of Drug-likeness (QED), and molecular weight for generated molecules. $30,000$ molecules are generated for each method.

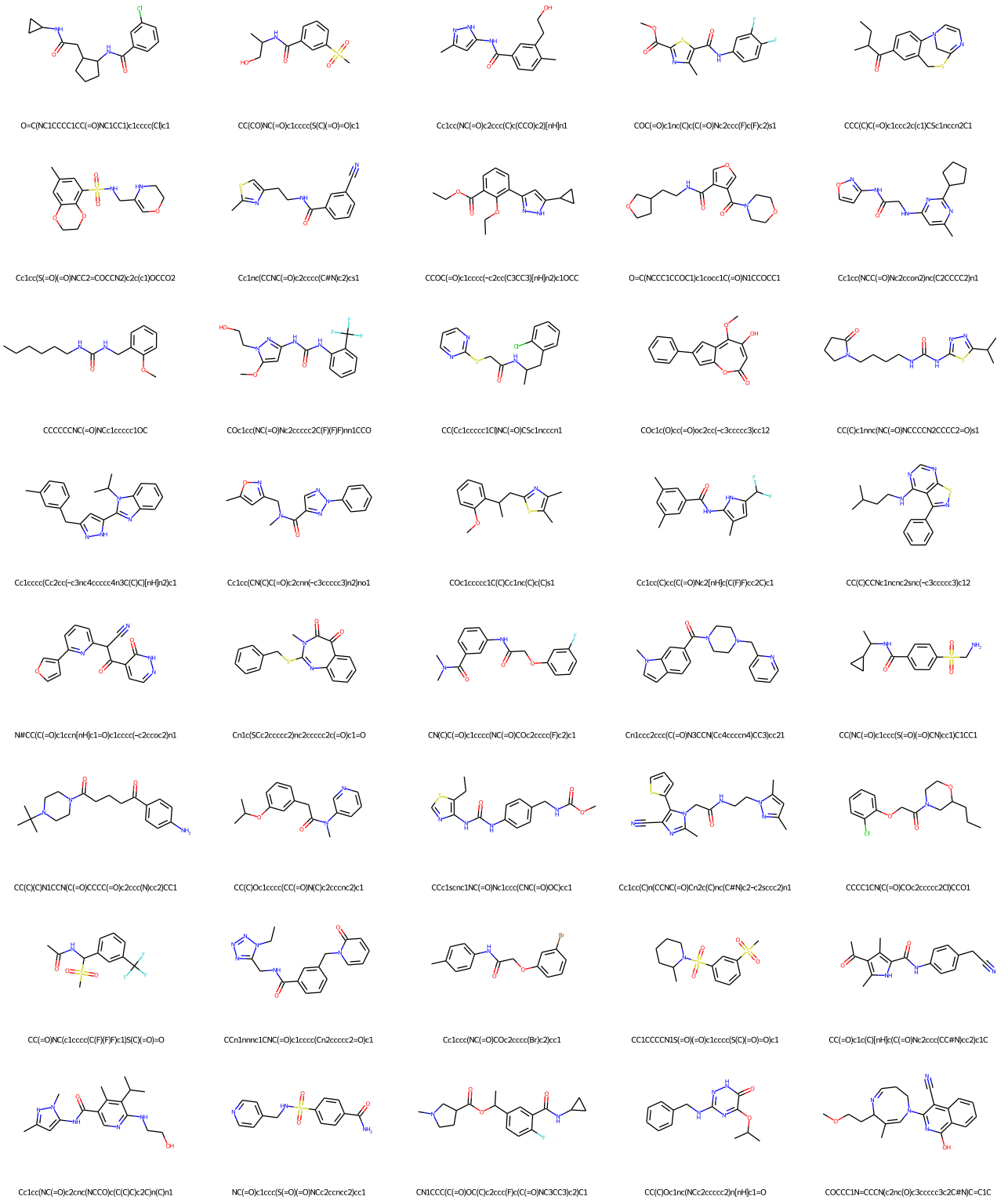

Figure 12. Generated samples from MaM-SMILES: MOSES

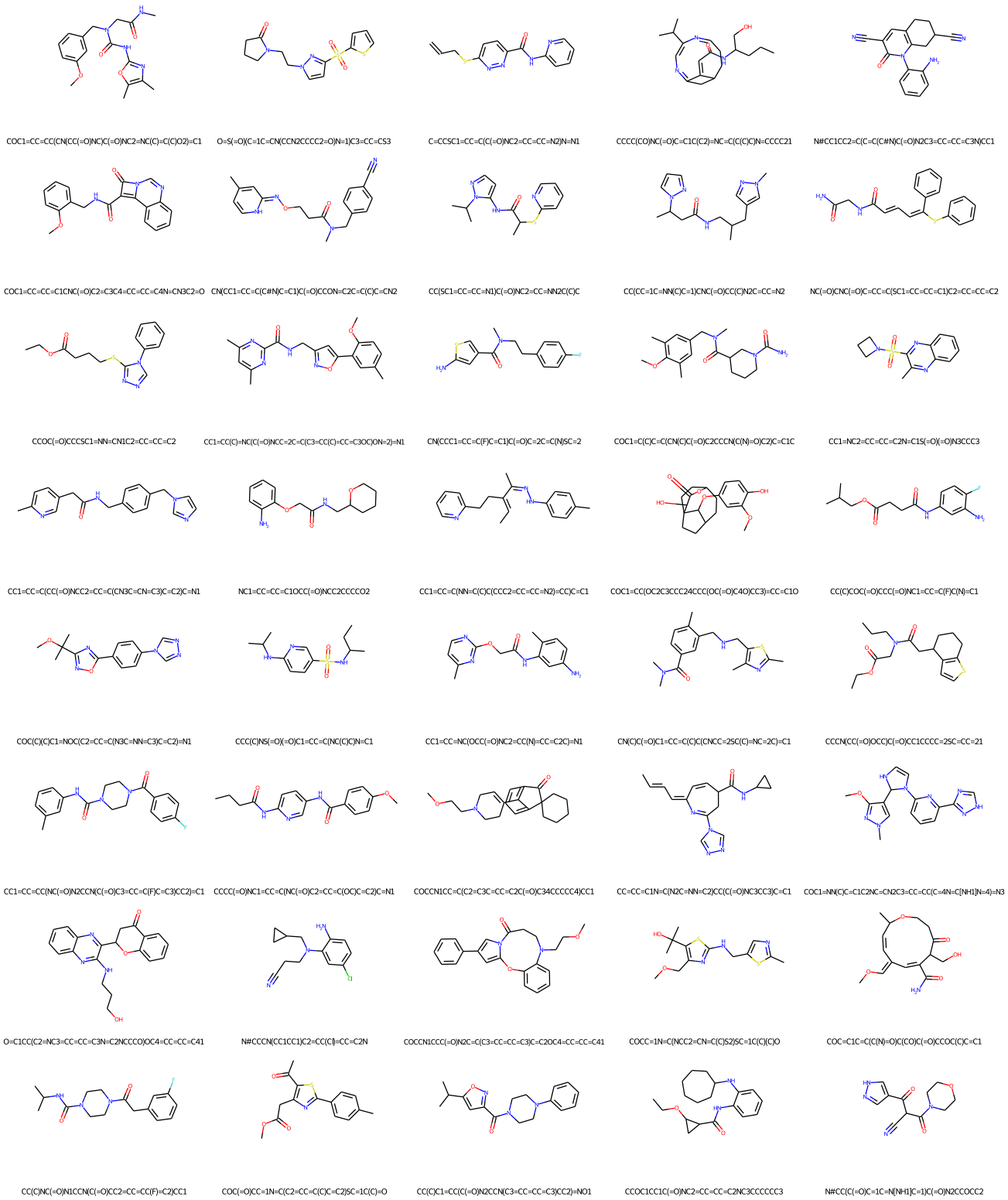

Figure 13. Generated samples from MaM-SELFIES: MOSES

D.5 text8

Table 6. Performance Comparison on text8

Model	NLL (bpc) ↓	Spearman's ↑	Pearson ↑	LL inf. time (s) ↓
Discrete Flow (8 flows)	**1.23**	–	–	–
AO-ARM-E-Transfomer	1.494	1.0	1.0	207.60 ± 0.33
AO-ARM-S-Transfomer	1.529	**0.982**	**0.987**	41.40 ± 0.01
MaM-Transfomer	1.529	0.937	0.945	$\mathbf{0.005 \pm 0.000}$

Text8 (Mahoney, 2011) is a character level language modeling task with 100M characters split into chunks of 250 character. We follow the same procedure as Binary-MNIST and report the same metrics in Table 6. The results of discrete flow is from Tran et al. (2019) (for which unfortunately no open-source implementations exist to measure other metrics).

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
```

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
```

E Experiment Details for Distribution Matching Training

E.1 Dataset details

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 Grathwohl et al. (2021) 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.

E.2 Training details

Ising model and molecule generation with target property

- Ising model input are of $\{0, 1, ?\}$ 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.

We compare with ARM that uses sum of conditionals to evaluate $\log p_\phi$ with fixed forward ordering and ARM-MC that uses a one-step conditional to estimate $\log p_\phi$. ARM can be regarded as the golden standard of learning autoregressive conditionals, since its gradient is the most informative. MaM uses marginal network to evaluate $\log p_\theta$ and subsamples a one-step marginalization constraint for each data point in the batch. The effective batch size for ARM and GFlowNet is $B \times \mathcal{O}(D)$ for batch of size B, and $B \times \mathcal{O}(1)$ for ARM-MC and MaM . MaM and ARM optimizes KL divergence using REINFORCE gradient estimator with baseline. GFlowNet minimizes squared distance following Zhang et al. (2022).

E.3 Ising model

Ising models (Ising, 1925) model interacting spins and are widely studied in mathematics and physics (see MacKay (2003)). We study Ising model on a square lattice. The spins of the D sites are represented a D-dimensional binary vector and its distribution is $p^*(\mathbf{x}) \propto f^*(\mathbf{x}) = \exp\left(-\mathcal{E}_J(\mathbf{x})\right)$ where $\mathcal{E}_\mathbf{J}(\mathbf{x}) \triangleq -\mathbf{x}^\top \mathbf{J}\mathbf{x} - \boldsymbol{\theta}^\top \mathbf{x}$, with $\mathbf{J}$ the binary adjacency matrix. These models, although simplistic, bear analogies to the complex behavior of high-entropy alloys (Damewood et al., 2022).

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. 2000 ground truth samples are generated following Grathwohl et al. (2021) and we measure test negative log-likelihood on those samples. We also measure $D_{\text{KL}}(p_\theta(\mathbf{x})||p^*)$ by sampling from the learned model and evaluating $\sum_{i=1}^{M}(\log p_\theta(\mathbf{x}_i) - \log f^*(\mathbf{x}_i))$. Figure 4 contains KDE plots of $-\mathcal{E}_\mathbf{J}(\mathbf{x})$ for the generated samples.

As described in Section 4, the ARM-MC gradient suffers from high variance and fails to converge. Under a larger learning rate, it also tends to collapse and converge to a single sample. MaM-DM has significant speedup in inference time and is the only model that supports any-order generative modeling. The performance in terms of KL divergence and likelihood are slightly worse than models with fixed/learned order, which is expected since any-order modeling is harder than fixed-order modeling, and MaM-DM is solving a more complicated task of jointly learning conditionals and marginals. On a 30×30 ($D = 900$) Ising model, MaM-DM achieves a bpd of 0.835 on ground-truth samples while ARM and GFlowNet fails to scale. Distribution of generated samples is shown in Fig 4.

Table 7. Performance Comparison on Ising model (10×10)

Model	**NLL (bpd)** ↓	**KL divergence** ↓	**Inference time (s)** ↓
ARM-Forward-Order-MLP	0.79	**-78.63**	5.29±0.07e-01
ARM-MC-Forward-Order-MLP	24.84	-18.01	5.30±0.07e-01
MaM-Any-Order-MLP	0.80	-77.77	**3.75±0.08e-04**
GFlowNet-Learned-Order-MLP	**0.78**	-78.17	–

Generated samples

We compare ground truth samples and MaM samples in Figure 14 and 15.

E.4 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.

Table 8. Performance Comparison on Target Lipophilicity

Model	**KL divergence** ↓			
Distribution	$\text{logP} = 4, \tau = 1.0$	$\text{logP} = -4, \tau = 1.0$	$\text{logP} = 4, \tau = 0.1$	$\text{logP} = 4, \tau = 0.1$
ARM-FO-MLP	-174.25	-168.62	-167.83	-160.2
MaM-AO-MLP	-173.07	-166.43	-165.75	-157.59

Target property distribution matching on lipophilicity (logP)

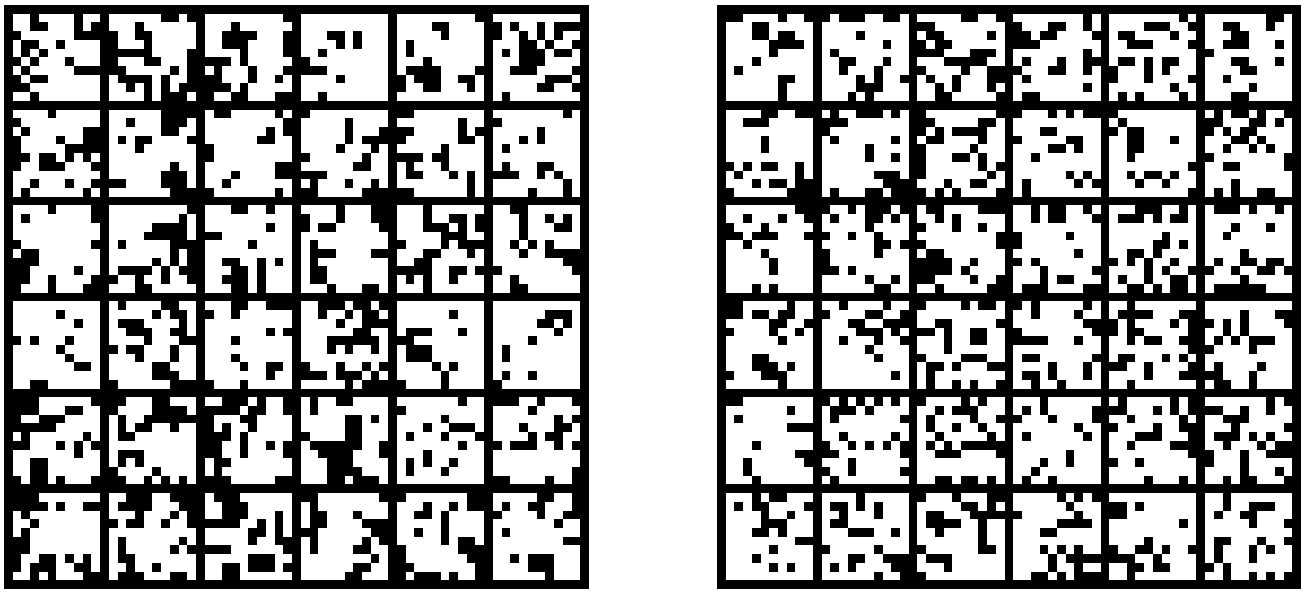

Figure 14. Samples: 10×10 Ising model. Ground truth (left) v.s. MaM (right).

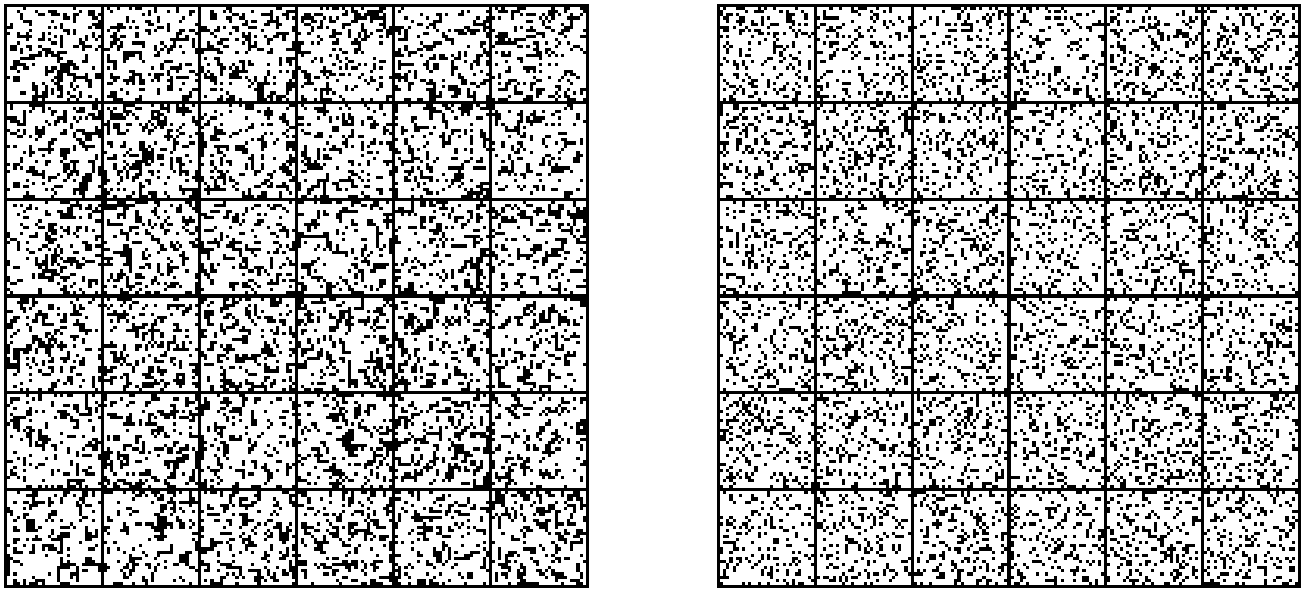

Figure 15. Samples: 30×30 Ising model. Ground truth (left) v.s. MaM (right).

We train ARM and MaM for lipophilicity of target values 4.0 and -4.0, both with $\tau = 1.0$ and $\tau = 0.1$ for molecule length $D = 55$. Both models are trained for 3000 iterations with batch size 512. Results in KL divergence is shown in Table 8. Findings are consistent with the Ising model experiments. There is a small gap in the performance of MaM against ARM, but MaM supports any-order modeling and scales to problems with much larger dimension.

Figure 16 shows 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 Figure 17. ARM fails to scale to $D = 500$ so only results from MaM are reported. In this case, logP values are shifted towards the target but their peaks are closer to 0 than when $D = 55$, due to the enlarged molecule space containing more molecules with logP around 0. Also, this is validated by the result when $\tau = 0.1$ and $\tau = 0.01$ for $D = 500$, where the peaks are more and more centered around the target values when we lower the temperature.

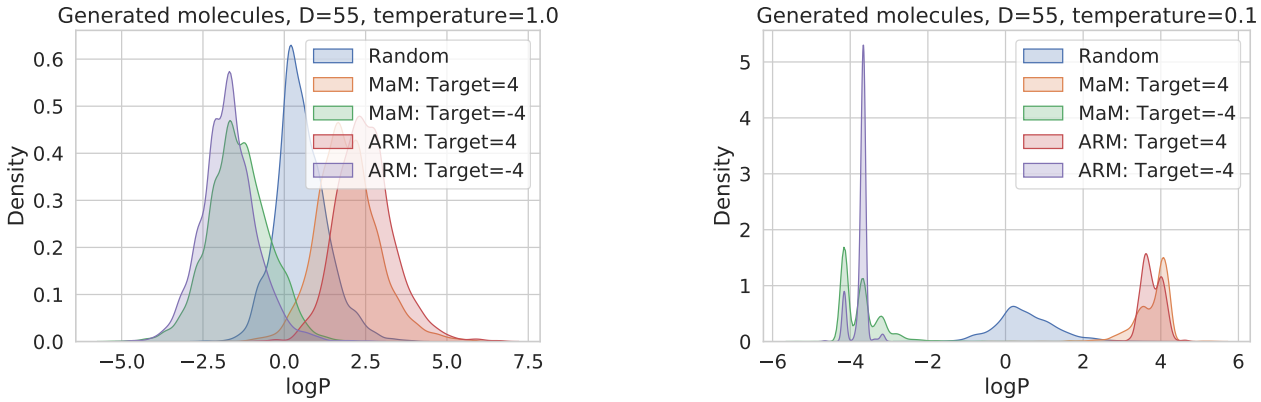

Figure 16. Target property matching with different temperatures. 2000 samples are generated for each method.

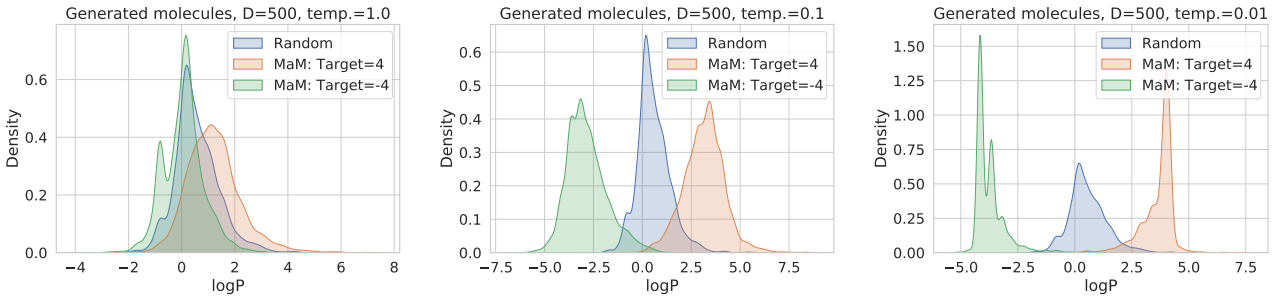

Figure 17. Target property matching with different temperatures. 2000 samples are generated for each method.

Conditionally generated samples

We illustrate how MaM enables flexible and controllable generation via any-order modeling in Figure 18. With MaM, we can pick an arbitrary molecular substructure and conditionally generate a molecule in a user-specified order.

More samples from conditionally generating towards low lipophilicity (target $= -4.0$, $\tau = 1.0$) from user-defined substructures of Benzene are shown in Figure 19 and Figure 20. We are able to generate from any partial substructures with any-order generative modeling of MaM. Figure 19 shows conditional generation from masking out the left 4 SELFIES characters. Figure 20 shows conditional generation from masking the right $4 \sim 20$ characters.

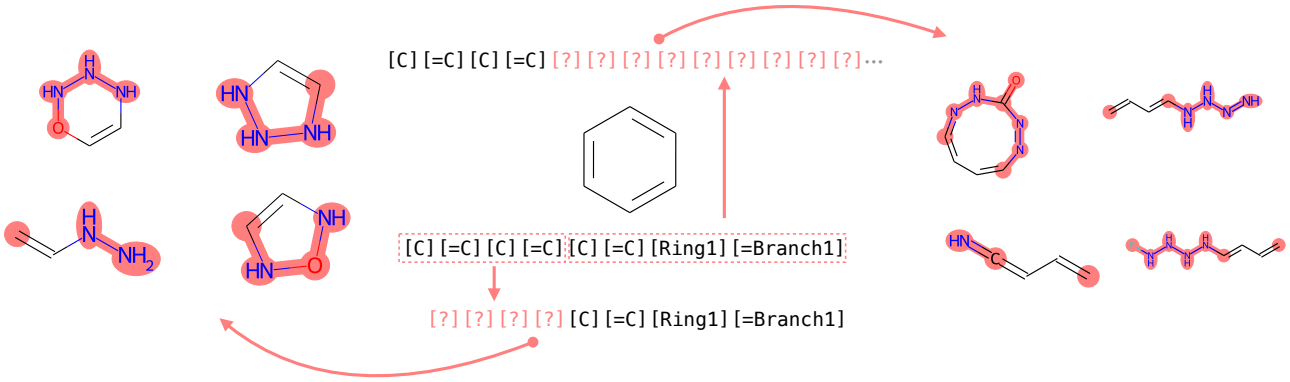

Figure 18. Conditionally generate towards low lipophilicity from user-defined substructures of Benzene. Left: Masking out the left 4 SELFIES characters. Right: masking the right 4-20 characters.

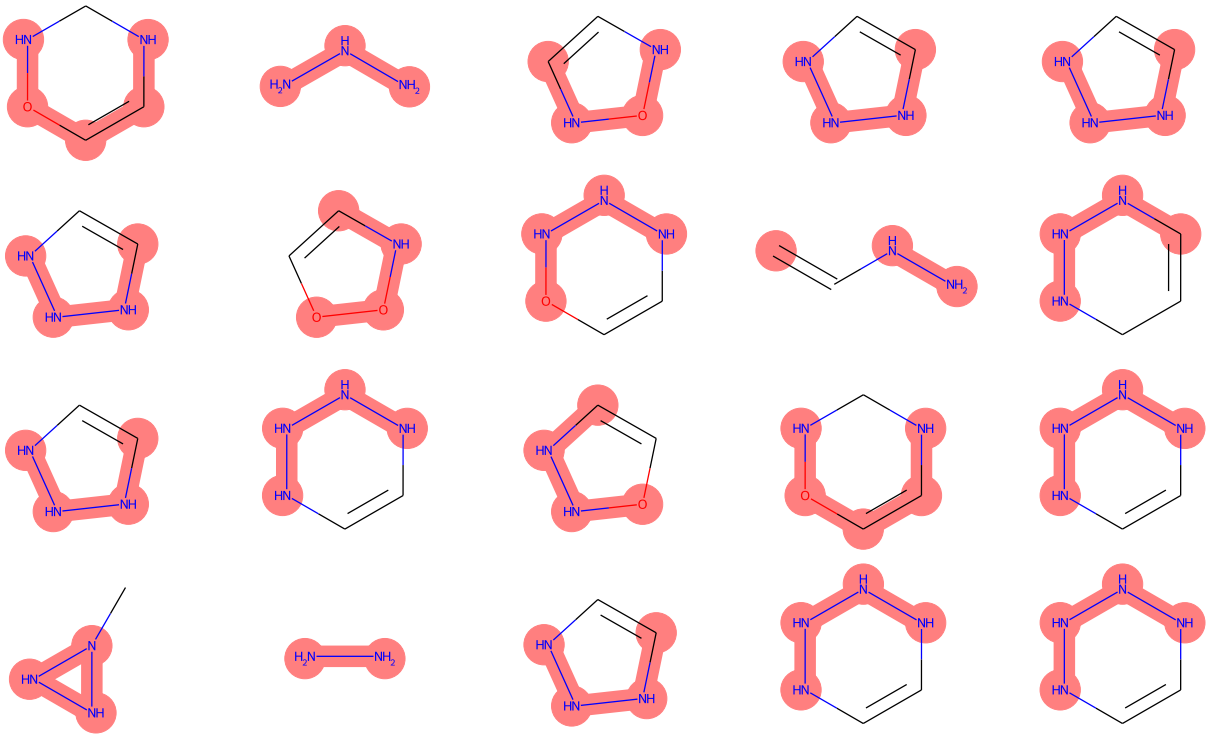

Figure 19. Generated samples from masking out the left 4 SELFIES characters of a Benzene.

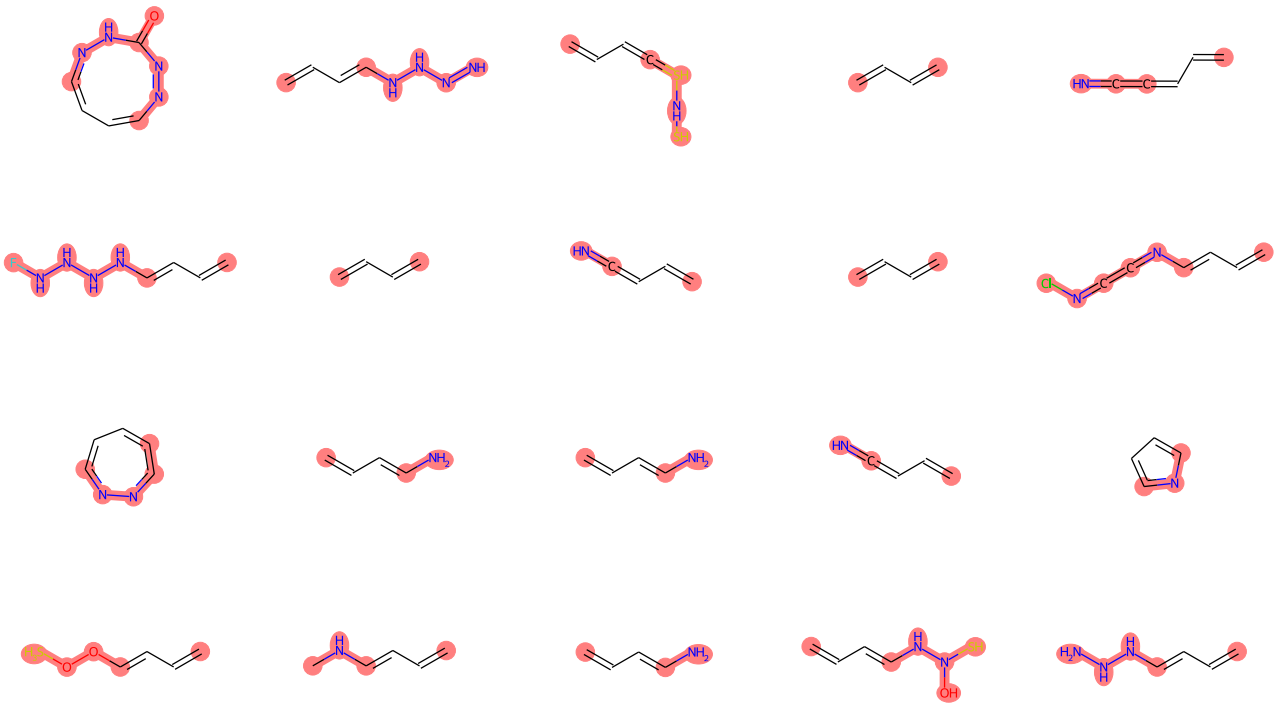

Figure 20. Generated samples from masking out the right 4-20 SELFIES characters of a Benzene.

F Limitations and Broader Impacts

The marginalization self-consistency in MaM is only softly enforced by minimizing the squared error on subsampled marginalization constraints. Therefore the marginal likelihood estimate is not guaranteed to be always (approximately) valid. In particular, as a deep learning model, it has the risk of overfitting and low robustness on unseen data manifold. In practice, it means one should not blindly apply it to data that is not seen by the training and expect the marginal likelihood estimate can be trusted.

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.