Sampling in Combinatorial Spaces with SurVAE Flow Augmented MCMC

Priyank Jaini
Bosch-Delta Lab
University of Amsterdam

Didrik Nielsen
Technical University of Denmark

Max Welling
Bosch-Delta Lab
University of Amsterdam

Abstract

Hybrid Monte Carlo is a powerful Markov Chain Monte Carlo method for sampling from complex continuous distributions. However, a major limitation of HMC is its inability to be applied to discrete domains due to the lack of gradient signal. In this work, we introduce a new approach based on augmenting Monte Carlo methods with Sur-VAE Flows to sample from discrete distributions using a combination of neural transport methods like normalizing flows and variational dequantization, and the Metropolis-Hastings rule. Our method first learns a continuous embedding of the discrete space using a surjective map and subsequently learns a bijective transformation from the continuous space to an approximately Gaussian distributed latent variable. Sampling proceeds by simulating MCMC chains in the latent space and mapping these samples to the target discrete space via the learned transformations. We demonstrate the efficacy of our algorithm on a range of examples from statistics, computational physics and machine learning, and observe improvements compared to alternative algorithms.

1 Introduction

The ability to draw samples from a known distribution is a fundamental computational challenge. It has applications in diverse fields like statistics, probability, and stochastic modeling where these methods are useful for both estimation and inference. These are

further useful within the frequentist inference framework to form confidence intervals for a point estimate. Sampling procedures are also standard in the Bayesian setting for exploring posterior distributions, obtaining credible intervals, and solving inverse problems. The workhorse algorithms in these settings are simulation-based, amongst which the Markov Chain Monte Carlo Brooks et al. (2011) method is the most broadly used method. Impressive advances have been made – both in increasing efficiency and reducing computation costs – for sampling using Monte Carlo methods over the past century. However, problems in discrete domains still lack an efficient general-purpose sampler.

In this paper, we consider the problem of sampling from a known discrete distribution. Inspired by the recent success of deep generative models – particularly neural transport methods like normalizing flows (Tabak and Vanden-Eijnden, 2010; Tabak and Turner, 2013; Rezende and Mohamed, 2015)– in unsupervised learning, we propose a new approach to design Monte Carlo methods to sample from a discrete distribution based on augmenting MCMC with a *transport map*. Informally, let $\pi(\boldsymbol{\theta})$ be a discrete target distribution and $p(\mathbf{z})$ be a simple source density from which it is easy to generate independent samples e.g. a Gaussian distribution. Then, a transport map $\mathbf{T}$ from $p(\mathbf{z})$ to $\pi(\boldsymbol{\theta})$ is such that if $\mathbf{z}_i \sim p(\mathbf{z})$ then $\mathbf{T}(\mathbf{z}_i) \sim \pi(\boldsymbol{\theta})$.

The significance of having such a transport map $\mathbf{T}$ is particularly consequential: firstly, given such a map $\mathbf{T}$, we can generate samples from the target $\pi(\boldsymbol{\theta})$. Secondly, these samples can be generated cheaply irrespective of the cost of evaluating $\pi(\boldsymbol{\theta})$. Importantly, $\mathbf{T}$ affords us the ability to sample from the marginal and conditional distributions of $\pi(\boldsymbol{\theta})$ using $p(\mathbf{z})$ given an appropriate structure. Indeed, this idea of using neural transport maps based on normalizing flows have been explored by Parno and Marzouk (2018) for continuous densities by learning a diffeomorphic transformation $\mathbf{T}$ from the source density p to the target density $\pi(\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^d$.

In this paper, we extend this to discrete domains us-

ing the recently proposed SurVAE Flow framework (Nielsen et al., 2020). We first learn a transport map from the discrete space to a continuous space using a surjective transformation $\mathbf{Q}$. However, such a continuous space is often highly multi-modal with unfavorable geometry for fast mixing of MCMC algorithms. Thus, we learn an additional normalizing flow $\mathbf{T}_\phi$ that transforms this *complex* continuous space to a simple latent space with density $p(\mathbf{z})$ where sampling is easy. We finally sample from the desired target distribution by generating samples from the latent space and using the learned transformations $\mathbf{Q}$ and $\mathbf{T}$ to *push-forward* these samples to the target space. Our complete implementation allows parallelization over multiple GPUs, improving efficiency and reducing computation time.

The main contributions of this paper are:

- We present a SurVAE augmented general purpose MCMC solver for combinatorial spaces.

- We propose a new learning objective compared to previous methods in normalizing flows to train a transport map that "Gaussianizes" the discrete target distribution.

The rest of the manuscript is organized as follows: We begin in §2 by presenting a brief background on normalizing flows and setting up our main problem. In §3, we provide details of our method which consists of two parts: learning the transport map and, generating samples from the target distribution. Subsequently, in §4, we put our present work in perspective to other approaches to sampling in discrete spaces. Finally, we perform empirical evaluation on a diverse suite of problems in §5 to demonstrate the efficacy of our method. Our implementation is available at https://github.com/priyankjaini/discFlowMH

2 Preliminaries and Setup

In this section, we setup our main problem, provide key definitions and notations, and formulate the idea for an MCMC sampler augmented with normalizing flows for sampling in discrete spaces.

Let p, q be two probability density functions (w.r.t. the Lebesgue measure) over the source domain $\mathsf{Z} \subseteq \mathbb{R}^d$ and the target domain $\mathsf{X} \subseteq \mathbb{R}^d$, respectively. Normalizing flows learn a *diffeomorphic* transformation $\mathbf{T} : \mathsf{Z} \to \mathsf{X}$ that allows to represent q using p via the change of variables formula (Rudin, 1987):

$$q(\mathbf{x}) = p(\mathbf{z})/|\mathbf{T}'(\mathbf{z})|$$
$$= p(\mathbf{T}^{-1}\mathbf{x})/|\mathbf{T}'(\mathbf{T}^{-1}\mathbf{x})|,$$

where $|\mathbf{T}'(\mathbf{z})|$ is the (absolute value) of the Jacobian (determinant of the derivative) of $\mathbf{T}$. In other words, we can obtain a new random variable $\mathbf{x} \sim q$ by pushing the source random variable $\mathbf{z} \sim p$ through the map $\mathbf{T}$. When we only have access to an i.i.d. sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \sim q$, we can learn $\mathbf{T}$ and thus q through maximum likelihood estimation:

$$\max_{\mathbf{T} \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \Big[-\log |\mathbf{T}'(\mathbf{T}^{-1}\mathbf{x}_i)| + \log p(\mathbf{T}^{-1}\mathbf{x}_i) \Big].$$

where $\mathcal{F}$ is a class of diffeomorphic mappings. Conveniently, we can choose *any* source density p to facilitate estimation e.g. standard normal density on $\mathsf{Z} = \mathbb{R}^d$ (with zero mean and identity covariance) or uniform density over the cube $\mathsf{Z} = [0, 1]^d$.

This "push-forward" idea has played an important role in optimal transport theory (Villani, 2008) and has been used successfully for Monte carlo simulations. For example Marzouk et al. (2016); Parno and Marzouk (2018); Peherstorfer and Marzouk (2018); Hoffman et al. (2019); Albergo et al. (2019) have used normalizing flows for continuous random variables to address the limitations of HMC which suffers from the chain to mix slowly between distant states when the geometry of the target density is unfavourable.

Specifically, Parno and Marzouk (2018) addressed this problem of sampling in a space with difficult geometry by learning a diffeomorphic transport map that transforms the original random variable to another random variable with a simple distribution. Concretely, let our interest be to sample from $\pi(\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$. We can proceed by learning a diffeomorphic map $\mathbf{T} : \mathsf{Z} \to \Theta$ such that $\tilde{p}(\mathbf{z}) = \pi(\boldsymbol{\theta}) \cdot |\mathbf{T}'(\mathbf{z})|$ where $\mathbf{z} = \mathbf{T}^{-1}(\boldsymbol{\theta})$ such that $p(\mathbf{z})$ has a simple geometry amenable to efficient MCMC sampling. Thus, samples can be generated from $\pi(\boldsymbol{\theta})$ by running MCMC chain in the z-space and pushing these samples onto the Θ-space using $\mathbf{T}$. The transformation $\mathbf{T}$ can be learned by minimizing the KL-divergence between a fixed distribution with simple geometry in the z-space e.g. a standard Gaussian and $\tilde{p}(\mathbf{z})$ above. The learning phase attempts to ensure that the distribution $\tilde{p}(\mathbf{z})$ is approximately close to the fixed distribution with easy geometry so that MCMC sampling is efficient. A limitation of these works that use diffeomorphic transformations augmented samplers is that they are restricted to continuous random variables. This is primarily because the flow models used in these works can only learn density functions over continuous variables.

Uria et al. (2013) introduced the concept of *dequantization* to extend normalizing flows to discrete random variables. They consider the problem of estimating the discrete distribution $\pi(\boldsymbol{\theta})$ given samples $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n\} \sim \pi$ by "lifting" the discrete space Θ

to a continuous one X by filling the gaps in the discrete space with a uniform distribution i.e. $\mathbf{x} \in \mathsf{X}$ is such that $\mathbf{x} := \boldsymbol{\theta} + \mathbf{u}|\boldsymbol{\theta}$ where $\boldsymbol{\theta} \in \Theta$ and $\mathbf{u}|\boldsymbol{\theta} \sim \mathsf{Uniform}(0,1)$. Subsequently, they learn the continuous distribution $p_{\mathsf{model}}(\mathbf{x})$ over X using a normalizing flow by maximizing the log-likelihood of the continuous model $p_{\mathsf{model}}(\mathbf{x})$. Theis et al. (2015) showed that maximizing likelihood of the continuous model is equivalent to maximizing a lower bound on the log-likelihood for a certain discrete model $\pi_{\mathsf{model}} := \int_{\mathbf{u}} p_{\mathsf{model}}(\boldsymbol{\theta}+\mathbf{u})\mathrm{d}\mathbf{u}$ on the original discrete data. Thus, this learning procedure cannot lead to the continuous model degenerately collapsing onto the discrete data, because its objective is bounded above by the log-likelihood of a discrete model. Ho et al. (2019) extended the idea of uniform dequantization to propose *variational dequantization* where in $\mathbf{u}|\boldsymbol{\theta} \sim q(\mathbf{u}|\boldsymbol{\theta})$ instead of the uniform distribution. They learn the dequantizing distribution, $q(\mathbf{u}|\boldsymbol{\theta})$ from data by treating it as a variational distribution. They subsequently estimate $\pi(\boldsymbol{\theta})$ by optimizing the following lower bound:

$$\mathbb{E}_{\boldsymbol{\theta} \sim \pi}[\log \pi_{\mathsf{model}}(\boldsymbol{\theta})] \geq \mathbb{E}_{\substack{\boldsymbol{\theta} \sim \pi \\ \mathbf{u} \sim q(\mathbf{u}|\boldsymbol{\theta})}}\left[\log \frac{p_{\mathsf{model}}(\mathbf{x})}{q(\mathbf{u}|\boldsymbol{\theta})}\right] \quad (1)$$

Recently, Nielsen et al. (2020) introduced SurVAE Flows that extends the framework of normalizing flows to include surjective and stochastic transformations for probabilistic modelling. In the SurVAE flow framework, dequantization can be seen as a *rounding* surjective map $\mathbf{Q}_\lambda : \mathsf{X} \to \Theta$ with parameters λ such that $\boldsymbol{\theta} := \lfloor \mathbf{x} \rceil$ where the forward transformation is a discrete surjection $P(\boldsymbol{\theta}|\mathbf{x}) = \mathbb{I}(\mathbf{x} \in \mathcal{B}(\boldsymbol{\theta}))$, for $\mathcal{B}(\boldsymbol{\theta}) = \{\boldsymbol{\theta} + \mathbf{u}|\mathbf{u} \in [0,1)^d\}$. The inverse transformation $\mathbf{Q}_\lambda^{-1} := q(\mathbf{x}|\boldsymbol{\theta})$ is stochastic with support in $\mathcal{B}(\boldsymbol{\theta})$. Thus, these ideas of using dequantization to learn a discrete probability distribution can be viewed as learning a transformation $\mathbf{Q}_\lambda^{-1}$ that transforms a discrete space Θ to a continuous space X.

In this paper, we study the problem of sampling in discrete spaces i.e. let $\Theta \subseteq \{1, 2, \cdots, K\}^d := [K]^d$ be a discrete random variable in d-dimensions with probability mass function (potentially unnormalized) $\pi(\boldsymbol{\theta})$, $\boldsymbol{\theta} \in \Theta$. Given access to a function that can compute $\pi(\boldsymbol{\theta})$, $\forall \boldsymbol{\theta} \in \Theta$, we aim to generate samples $\langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n \rangle \sim \pi(\boldsymbol{\theta})$. The aforementioned works on normalizing flows for discrete data is not directly applicable in this regime. This is due to the fact that Uria et al. (2013); Ho et al. (2019) and Nielsen et al. (2020) estimate $\pi(\boldsymbol{\theta})$ given samples from the original distribution by maximizing the log-likelihood of a discrete model. However, we only have access to the function $\pi(\boldsymbol{\theta})$ and the learning method given in Equation (1) cannot be used in our setting. In the next section, we address this by extending the ideas of neu-

ral transport MCMC (Parno and Marzouk, 2018; Hoffman et al., 2019; Albergo et al., 2019) and leveraging SurVAE Flows (Nielsen et al., 2020).

3 Flow Augmented MCMC

We discussed in Section 2 the utility of normalizing flows augmented Monte Carlo samplers for overcoming the difficulties posed by unfavourable geometry in the target continuous distribution, as evidenced by the works of Parno and Marzouk (2018); Hoffman et al. (2019) and Albergo et al. (2019). We will now introduce our method of SurVAE Flow augmented MCMC for sampling in combinatorial spaces.

Informally, our method proceeds as follows: We first define a rounding surjective transformation $\mathbf{Q}_\lambda : \mathsf{X} \to \Theta$ with parameters λ such that X is a continuous embedding of the Θ space with density $q(\mathbf{x})$. Since the continuous embedding may be highly multi-modal with potentially unfavourable geometry for efficient MCMC sampling, we define an additional diffeomorphic transformation, $\mathbf{T}_\phi : \mathsf{Z} \to \mathsf{X}$ with parameters ϕ, from a simple latent space Z to X. Subsequently, we learn these transformations via maximum likelihood estimation. This concludes the learning phase of our algorithm. Finally, we generate samples from $\pi(\boldsymbol{\theta})$ by running MCMC chains to generate samples from the learned distribution over Z and pushing-forward these samples to the Θ space using $\mathbf{Q}_\lambda$ and $\mathbf{T}_\phi$. We elaborate each of these steps below.

$\mathbf{Q}_\lambda$ is a surjective transformation that takes as input a real-valued vector $\mathbf{x} \in \mathsf{X} \subseteq \mathbb{R}^d$ and returns the *rounded* value of $\mathbf{x}$ i.e. $\boldsymbol{\theta} = \lfloor \mathbf{x} \rceil$. Thus, the forward transformation from X to Θ is deterministic. The inverse transformation from Θ to X is however stochastic since the random variable $\mathsf{X} := \Theta + \mathsf{U}_{|\boldsymbol{\theta}}$ where $\mathsf{U}_{|\boldsymbol{\theta}} \subseteq [0,1)^d$ is given by $\mathbf{x} := \boldsymbol{\theta} + \mathbf{u}$ where $\mathbf{u} \sim q(\mathbf{u}|\boldsymbol{\theta})$. Given access to $q(\mathbf{u}|\boldsymbol{\theta})$, we can evaluate the density $q(\mathbf{x})$ at a point $\mathbf{x} \in \mathbb{R}^d$ exactly as:

$$q(\mathbf{x}) = q(\boldsymbol{\theta} + \mathbf{u}) := \pi(\boldsymbol{\theta}) \cdot q(\mathbf{u}|\boldsymbol{\theta}) \quad (2)$$

since $\mathbf{x} \in \mathcal{B}(\boldsymbol{\theta})$. Thus, we need to learn $q(\mathbf{u}|\boldsymbol{\theta})$ in order to fully specify this surjective transformation. Since, $q(\mathbf{u}|\boldsymbol{\theta})$ can be any arbitrary continuous density, we learn $q(\mathbf{u}|\boldsymbol{\theta})$ using a normalizing flow i.e. we learn a diffeomorphic transformation $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta}) : \mathcal{E} \to \mathsf{U}_{|\boldsymbol{\theta}}$ where $\mathcal{E} \subseteq \mathbb{R}^d$ is standard Gaussian distributed. Under this setup, we get

$$q(\mathbf{u}|\boldsymbol{\theta}) = p(\boldsymbol{\varepsilon}) \cdot |\mathbf{T}_\lambda'(\boldsymbol{\varepsilon}; \boldsymbol{\theta})|^{-1} \quad (3)$$

Thus, learning the surjective transformation $\mathbf{Q}$ is equivalent to learning $q(\mathbf{u}|\boldsymbol{\theta})$ which reduces to learning a flow $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta})$. Thus, for brevity we use the in-

formal notation that the transformation $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta})$ quantizes the continuous space X to $\boldsymbol{\theta}$. Finally, using Equation (3), the density $q(\mathbf{x})$ in Equation (2) for a fixed $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta})$ can be written as:

$$q(\mathbf{x}) = \pi(\boldsymbol{\theta}) \cdot p(\boldsymbol{\varepsilon}) \cdot |\mathbf{T}'_\lambda(\boldsymbol{\varepsilon}; \boldsymbol{\theta})|^{-1} \qquad (4)$$

The density $q(\mathbf{x})$ over the continuous embedding X of Θ learned above can have any arbitrary geometry and may not be efficient for MCMC sampling. Thus, next we learn a diffeomorphic transformation $\mathbf{T}_\phi$ from a latent space Z to X such that using the change of variables formula we get:

$$\tilde{p}_{\phi,\lambda}(\mathbf{z}) = q(\mathbf{x}) \cdot |\mathbf{T}'_\phi(\mathbf{T}_\phi^{-1}\mathbf{x})| \qquad (5)$$

Our complete model, therefore, consists of transformations $\mathbf{T}_\phi$ and $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta})$ that transforms Z to Θ. The next challenge is to learn $\mathbf{T}_\phi$ and $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta})$ such that the induced density $\tilde{p}_{\phi,\lambda}(\mathbf{z})$ has a simple geometry for efficient MCMC sampling. We achieve this by forcing $\tilde{p}_{\phi,\lambda}(\mathbf{z})$ to be close to a standard Gaussian by minimizing the KL-divergence between $\tilde{p}_{\phi,\lambda}(\mathbf{z})$ and $p(\mathbf{z})$ where $p(\mathbf{z}) = (2\pi)^{-\frac{d}{2}} \exp(-\mathbf{z}^T\mathbf{z})$.

Concretely, let $\mathcal{L}(\lambda, \phi) := \mathsf{KL}\big(p(\mathbf{z}) \,||\, \tilde{p}_{\phi,\lambda}(\mathbf{z})\big)$. $\mathcal{L}(\lambda, \phi)$ can be approximated with the empirical average by generating i.i.d. samples $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m\} \sim p(\mathbf{z})$ giving:

$$\mathcal{L}(\lambda, \phi) \approx \frac{1}{m} \sum_{i=1}^m \log \frac{p(\mathbf{z}_i)}{\tilde{p}_{\phi,\lambda}(\mathbf{z}_i)}$$

We thus arrive at the following optimization problem:

$$\mathbf{T}_\lambda^*, \mathbf{T}_\phi^* := \arg\min_{\lambda,\phi} \ \mathcal{L}(\lambda, \phi) \approx \arg\max_{\lambda,\phi} \frac{1}{m} \sum_{i=1}^m \log \tilde{p}_{\phi,\lambda}(\mathbf{z}_i)$$

Using Equations (4) and (5), we can rewrite our final objective as:

$$\mathbf{T}_\lambda^*, \mathbf{T}_\phi^* = \arg\max_{\lambda,\phi} \left(\frac{1}{m} \sum_{i=1}^m \log \pi(\boldsymbol{\theta}_i) + \log p(\boldsymbol{\varepsilon}_i) \right.$$
$$\left. - \log |\mathbf{T}'_\lambda(\boldsymbol{\varepsilon}_i; \boldsymbol{\theta}_i)| + |\mathbf{T}'_\phi(\mathbf{T}_\phi^{-1}\mathbf{x}_i)| \right) \quad (6)$$

where $\forall i \in [m]$, $\mathbf{x}_i = \mathbf{T}_\phi(\mathbf{z}_i)$, $\boldsymbol{\theta}_i = \lfloor \mathbf{T}_\phi(\mathbf{z}_i) \rfloor$, $\mathbf{u}_i = \mathbf{x}_i - \boldsymbol{\theta}_i$ and $\boldsymbol{\varepsilon}_i = \mathbf{T}_\lambda^{-1}(\mathbf{u}_i)$.

Learning $\mathbf{T}_\lambda^*, \mathbf{T}_\phi^*$ results in a density $\tilde{p}_{\phi,\lambda}^*(\mathbf{z})$ that is approximately Gaussian with a landscape amenable to efficient MCMC sampling. Thus, our sampling phase consists of running an MCMC sampler of choice with target density $\tilde{p}(\mathbf{z})$ resulting in samples $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\} \sim \tilde{p}_{\phi,\lambda}^*(\mathbf{z})$. We can finally obtain samples $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n\} \sim \pi(\boldsymbol{\theta})$ as $\boldsymbol{\theta}_i = \lfloor \mathbf{T}_\phi^*(\mathbf{z}_i) \rfloor$.

We end this section with two important remarks.

Remark 1. *In our method, it is possible to by-pass the last step of sampling in the z-space using MCMC entirely since the trained flow is a generative model that can be used directly to generate samples from $\pi(\boldsymbol{\theta})$. This can indeed be done if the learned transformations $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta})^*$ and $\mathbf{T}_\phi^*$ result in the density $\tilde{p}(\mathbf{z})$ to be Gaussian. Thus, there is a natural trade-off here: we can spend only enough computation to train the flow to learn $\tilde{p}(\mathbf{z})$ that is suitable for fast-mixing of the MCMC chain and generate samples from $\tilde{p}(\mathbf{z})$ that is not Gaussian using any sampler of choice or we can spend a larger amount of compute to learn a flow that perfectly Gaussianizes the target density $\pi(\boldsymbol{\theta})$. Then, we can sample directly by sampling from a Gaussian in the z-space and using the learned transformations to obtain samples from $\pi(\boldsymbol{\theta})$.*

Remark 2. *As mentioned in Section 2, we can use any density $p(\mathbf{z})$ instead of a Gaussian for training the transformations $\mathbf{T}_\lambda(\cdot; \boldsymbol{\theta})$ and $\mathbf{T}_\phi$. The main motivation of our method is to "push-forward" $\pi(\boldsymbol{\theta})$ onto a space that is amenable to efficient sampling. An interesting future work might be to devise learning objectives that explicitly drive the learned density to have simple geometry that favours "off-the-shelf" samplers.*

4 Connection to previous works

In Section 3 we introduced a normalizing flow augmented MCMC sampler for combinatorial spaces. Our method combines surjective transformation for learning continuous relaxations for discrete spaces and normalizing flows that map the continuous space to a simpler discrete space for easy MCMC sampling. In this section, we put both these ideas of continuous relaxations of discrete spaces and neural transport methods for efficient MCMC sampling in to perspective with existing work.

Neural transport samplers for continuous spaces: As we discussed briefly in Section 2, the neural transport augmented sampler for continuous variables has been successfully used by Parno and Marzouk (2018), Peherstorfer and Marzouk (2018), Hoffman et al. (2019) and Albergo et al. (2019). A subtle difference between these works and our work here – apart from the major difference that the aforementioned works are applicable only for continuous domains – is the method of training the transport map itself. These methods train the transport map (or normalizing flow) by minimizing the Kullback-Liebler divergence between the target density $\pi(\boldsymbol{\theta})$ and the density $\tilde{\pi}(\boldsymbol{\theta})$ learned via $\mathbf{T}$ by pushing-forward a standard normal distribution. Albergo et al. (2019) additionally use the reverse KL-divergence for the target density and the approximate density for training in-

stead. We, on the other hand, learn by minimizing the KL-divergence in the latent space since we only have access to a lower-bound of the discrete density $\tilde{\pi}(\boldsymbol{\theta})$ (cf. Equation (1)).

Continuous relaxations of discrete spaces: The idea to relax the constraint that random variables of interest only take discrete values has been used extensively in combinatorial optimization (Pardalos, 1996). Such an approach is attractive since the continuous space affords the function with gradient information, contours, and curvatures that can better inform optimization algorithms. Surprisingly though, this idea has not received much attention in the MCMC setting. Perhaps the closest work to ours in the present manuscript is that of Zhang et al. (2012) who use the Gaussian Integral Trick (Hubbard, 1959) to transform discrete variable undirected models into fully continuous systems where they perform HMC for inference and the evaluation of the normalization constant. The Gaussian Integral Trick used by Zhang et al. (2012) can be viewed as specifying a fixed map from the discrete space to an augmented continuous space. However, this can result in a continuous space that is highly multi-modal and not amenable for efficient sampling.

Nishimura et al. (2020) on the other hand map to a continuous space using *uniform dequantization* (Uria et al., 2013) i.e. filling the space between points in the discrete space with uniform noise inducing parameters with piecewise constant densities. [1] They further propose a Laplace distribution for the momentum variables in dealing with discontinuous targets and argue this to be more effective than the Gaussian distribution. This work relies on the key theoretical insight of Pakman and Paninski (2013) that Hamiltonian dynamics with a discontinuous potential energy function can be integrated explicitly near the discontinuity such that it preserves the total energy. Pakman and Paninski (2013) used this to propose a sampler for binary distributions which was later extended to handle more general discontinuities by Afshar and Domke (2015). Dinh et al. (2017) also used this idea on settings where the parameter space involves phylogenetic trees. A major limitation of Pakman and Paninski (2013), Afshar and Domke (2015) and Dinh et al. (2017) is the fact that these method do not represent a general-purpose solver since they encounter computational issues when dealing with complex discontinuities. Similarly, the work of Nishimura et al. (2020) requires an integrator that works component-wise and is prohibitively slow for high dimensional problems. Furthermore, it is not clear if the the Laplace momentum based sampler leads to efficient exploration of the

continuous space which is highly multi-modal due to uniform dequantization.

Discrete MCMC samplers: A main limitation of embedding discrete spaces into continuous ones is that they can often destroy natural topological properties of the space under consideration e.g. space of trees, partitions, permutations etc. Titsias and Yau (2017) proposed an alternative approach called Hamming ball sampler based on informed proposals that are obtained by augmenting the discrete space with auxiliary variables and performing Gibbs sampling in this augmented space. However, potentially strong correlations between auxiliary variables and the chain state severely slows down convergence. Zanella (2020) tried to address this problem by introducing locally balanced proposals that incorporate local information about the target distribution. This framework was later called a *Zanella process* by Power and Goldman (2019). They used the insights in (Zanella, 2020) to build efficient, continuous-time, non-reversible algorithms by exploiting the structure of the underlying space through symmetries and group-theoretic notions. This helps them to build locally informed proposals for improved exploration of the target space. However, their method requires explicit knowledge of the underlying properties of the target space which is encoded in the sampler which can be problematic.

5 Experiments

We now present experimental results for our SurVAE Flow augmented MCMC on problems covering a range of discrete models applied in statistics, physics, and machine learning. These include a synthetic example of a discretized Gaussian Mixture Model, Ising model for denoising corrupted MNIST, quantized logistic regression on four real world datasets, and Bayesian variable selection for high-dimensional data. The code for our implementation as well as the experiments is available at https://github.com/priyankjaini/discFlowMH.

We compare our model to two baselines that include a random walk Metropolis-Hastings algorithm and Gibbs sampling. We further also compare to discrete HMC (dHMC) (Nishimura et al., 2020) although we use the original implementation released by the authors which is not parallelizable and implements using the numpy package (Harris et al., 2020) and R. In contrast, we implemented our method and the other baselines in Pytorch (Paszke et al., 2019) and is parallelizable to use multiple GPUs for fast and efficient computation.

For each experiment, we train our model to learn $\mathbf{T}_\lambda$

---

[1] This corresponds to $q(\mathbf{u}|\boldsymbol{\theta}) = \mathsf{Uniform}(0,1)$ in our method.

and $\mathbf{T}_\phi$ for $10,000$ iterations with a batch-size of 128, learning rate of 10^{-3} and optimize using the Adam optimizer. We run 128 parallel chains of Metropolis-Hastings algorithm for 10^5 steps with thinning of 10 (i.e. saving every tenth sample) with no burn-in since our learned flow already provides a good initialization for sampling. For fair comparison, we follow a similar setup for the baselines i.e. we run 128 parallel chains for each method for 10^5 steps with thinning of 10. We use 10^5 burn-in steps for the baselines in all the experiments. Additionally, for denoising binarized MNIST using the Ising model, we also experiment with different number of burn-in steps which are detailed in the results. For dHMC (Nishimura et al., 2020), we used the publicly available code released by the authors.

We compare the efficiencies for all these models by reporting both the mean effective sample size (ESS)[2] across dimensions, mean ESS across dimensions per minute, and the accuracy or (unnormalized) log-probability of the samples generated by each sampler for the corresponding downstream task. For our method (labeled as Flow + MH), we include the training time of the flow i.e. $(\mathbf{T}_\lambda, \mathbf{T}_\phi)$ when evaluating ESS per min to ensure a fair comparison. We compute the ESS for 16 chains at a time, resulting in 8 ESS estimates and report mean $\pm$ standard error across these 8 estimates. The major aims of our experimental evaluation presented here are two-fold: Firstly, our aim is to demonstrate the efficacy of our flow augmented MCMC over broad applications. In particular, we want to demonstrate that flow-based generative models present an attractive methodology to sample from a discrete distribution by modelling the distribution in a continuous space and sampling in a simpler latent space. Thus, in our experiments here we have restricted ourselves to using the basic Metropolis-Hastings (MH) algorithm for sampling to demonstrate the advantages of being able to learn a transport map from the discrete space to a simple latent space. By using a more sophisticated sampler like HMC (Duane et al., 1987; Neal et al., 2011), we could thus get even better results. Secondly, most samplers (and especially discrete space samplers) are either inefficient and/or are prohibitively slow for high-dimensional problems. Thus, through our experiments we want demonstrate that leveraging the advances of flows to learn high-dimensional distributions, our sampler is more efficient with significantly better results than the alternative methods for high-dimensional problems.

Discretized Gaussian Mixture Model We first illustrate our method on a 2D toy problem. We define

Figure 1: Toy problem. *Left:* Discretized Gaussian mixture target distributions. *Middle:* Approximation learned by flow. *Right:* Samples from MCMC chain.

a set of target distributions by discretizing Gaussian mixture models with 5 components using a discretization level of 6 bits. In Fig. 1 we compare samples from the target distributions, samples from the approximation learned by the flow and samples from the MCMC chain. We observe that the flow gives an initial rough approximation to the target density – thus greatly simplifying the geometry for the MCMC method. Next, the samples from the MCMC chain are indistinguishable from true samples from the target distribution. This also highlights the trade-off we described in Remark 1 in Section 3.

Ising Model for Denoising Binarized MNIST We illustrate the application of discrete samplers to un-directed graphs in this section by considering the removal of noise from a binary 28×28 MNIST image. We take an image from the MNIST dataset and binarize it such that the true un-corrupted image is described by a vector $\mathbf{y} \in \{0,1\}^{784}$. We obtain a corrupted image $\mathbf{x}$ by taking this unknown noise-free image $\mathbf{y}$ and randomly flipping the value of each pixel with probability 0.1. Given this corrupted image, our goal is to recover the original noise-free image. Following (Bishop, 2006, Section 8.3), we solve this by for-

---

[2]We chose the mean ESS as a metric because for our experiments the minimum ESS across features tended to be close to one due to variables being discrete.
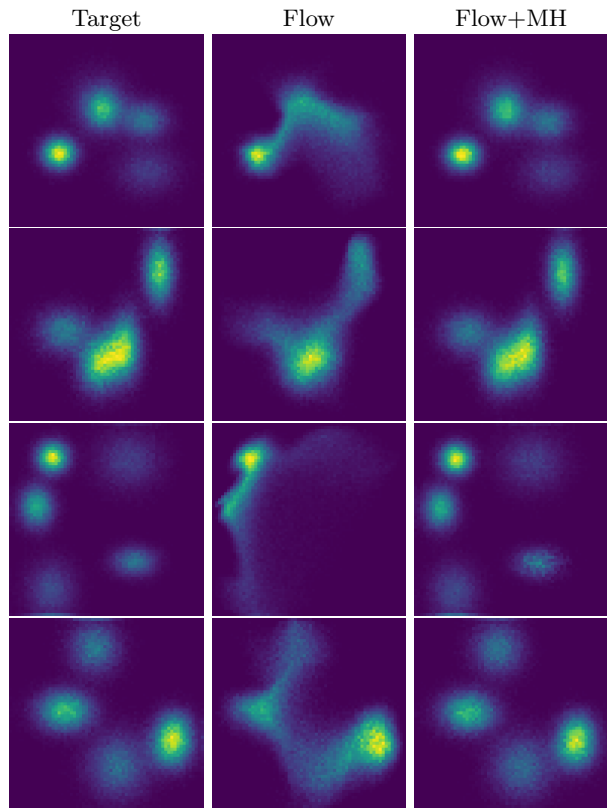
Table 1: Results for denoising binary MNIST using the Ising model. We report ESS per 10^4 samples, ESS per minute, and the average log-likelihood (unnormalized) for the generated samples.

Sampler	ESS	ESS/min	$\log \pi(\boldsymbol{\theta})$
Flow + MH (iters = 1k)	43.45 ± 5.07	0.15 ± 0.02	4025.67 ± 7.56
Flow + MH (iters = 2k)	$\mathbf{46.48 \pm 7.02}$	$\mathbf{0.15 \pm 0.02}$	4039.92 ± 6.05
Flow + MH (iters = 5k)	45.95 ± 5.34	0.13 ± 0.01	4054.25 ± 5.15
Flow + MH (iters = 10k)	28.52 ± 4.17	0.07 ± 0.01	4056.82 ± 5.64
Gibbs (burn-in = 10k)	7.131 ± 1.42	0.011 ± 0.002	4045 ± 22.79
Gibbs (burn-in = 20k)	6.09 ± 0.82	0.009 ± 0.001	4050.08 ± 16.56
Gibbs (burn-in = 50k)	7.65 ± 0.88	0.012 ± 0.001	4056.37 ± 9.94
Gibbs (burn-in = 100k)	7.82 ± 1.06	0.009 ± 0.001	4060.09 ± 7.56
discrete MH (burn-in = 10k)	26.24 ± 1.84	0.07 ± 0.01	3989.26 ± 99.77
discrete MH (burn-in = 20k)	12.01 ± 2.39	0.03 ± 0.01	4019.20 ± 44.22
discrete MH (burn-in = 50k)	1.90 ± 0.17	0.004 ± 0.001	4038.96 ± 17.95
discrete MH (burn-in = 100k)	2.40 ± 0.87	0.005 ± 0.002	4044.16 ± 16.03
dHMC	17.83 ± 1.49	0.022 ± 0.001	4038.68 ± 23.55

mulating it as an Ising model and using Monte carlo samplers to sample $\boldsymbol{\theta}$ with energy function given by:

$$E(\boldsymbol{\theta}, \mathbf{x}) = -\beta \cdot \sum_{i,j} \boldsymbol{\theta}_i \boldsymbol{\theta}_j - \eta \cdot \sum_i \boldsymbol{\theta}_i \mathbf{x}_i$$

where $\boldsymbol{\theta}_i$ and $\mathbf{x}_i$ are the ith pixel in the sample under consideration $\boldsymbol{\theta}$ and corrupted image $\mathbf{x}$ respectively. We train the flow for iters = $\{10^3, 2 \times 10^3, 5 \times 10^3, 10 \times 10^3\}$ and run the baselines with burn $-$ in = $\{10^4, 2 \times 10^4, 5 \times 10^4, 10 \times 10^4\}$. We report the results in Table 1 and the samples in Figure 2-3. We The results evidently show that Flow augmented MCMC significantly outperforms other samplers on both ESS and the underlying downstream task demonstrating its ability to disentangle complex correlations present in high-dimensional problems by mapping it to a latent space whereas other coordinate-wise samplers are not able to handle this efficiently.

Quantized Logistic Regression Next, we consider the task of logistic regression where the learnable parameters and biases denoted by $\boldsymbol{\theta}$ taking discrete values. This problem is particularly applicable for training quantized neural networks where the weights take discrete values. However, here we restrict ourselves to a simpler version of quantized logistic regression on four real-world datasets from the UCI repository that include Iris, Breast Cancer, Wine, and Digits. For each dataset, we run 5-fold cross-validation – with results averaged across the 5 folds. In each fold, we train the
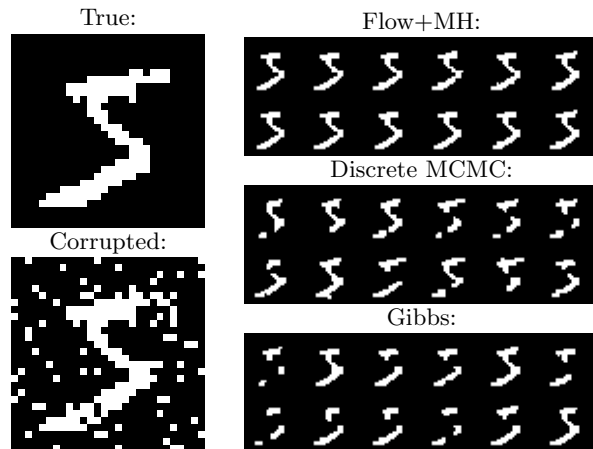

Figure 2: Ising model. *Left:* True MNIST digit and observed corrupted MNIST digit. *Right:* Denoised posterior samples from independent MCMC chains.

flow for 10^4 iterations and run discrete MCMC with a burn-in of 10^5 steps. We consider parameters quantized to 4 bits and report the results for these in Table 2. We notice that our method outperforms other methods on ESS but suffers in ESS/min due to the time it takes to train the flow.

Bayesian Variable Selection Here, we consider the problem of probabilistic selection of features in regression where the hierarchical framework allows for complex interactions making posterior exploration

Table 2: Results for quantized logisitic regression on four datasets. We report the ESS per 10^4 samples, ESS per minute, maximum accuracy, and the average log-likelihood for samples generated by each model.

Dataset	Parameters	Sampler	ESS	ESS/min	accuracy	$\log \pi(\boldsymbol{\theta})$
Iris	15	Flow + MH	**923.21 ± 37.29**	**58.64 ± 2.37**	97.4	-8.55 ± 1.77
		discrete MCMC	108.53 ± 1.44	26.14 ± 0.35	97.4	-8.54 ± 1.77
		dHMC	361.07 ± 16.97	32.85 ± 0.64	97.4	-8.40 ± 1.89
Wine	42	Flow + MH	**146.25 ± 3.51**	6.59 ± 0.16	97.8	-2.15 ± 1.63
		discrete MCMC	65.78 ± 0.46	15.05 ± 0.10	97.8	-2.15 ± 1.63
		dHMC	89.12 ± 11.57	4.53 ± 0.98	97.7	-3.54 ± 0.48
Breast Cancer	62	Flow + MH	**15.77 ± 0.49**	**0.56 ± 0.02**	96.8	-19.85 ± 3.40
		discrete MCMC	10.46 ± 0.12	2.29 ± 0.03	96.8	-20.14 ± 3.46
		dHMC	9.54 ± 0.28	1.38 ± 0.03	96.8	-21.19 ± 2.86

Table 3: Results for Bayesian variable selection on synthetic datasets of 100, 200, and 400 dimensions with only 10, 20, and 40 informative features, respectively. We report ESS per 10^4 samples, ESS per minute, and the average log-likelihood for samples generated by each model.

Setting	Sampler	ESS	ESS/min	$\log \pi(\boldsymbol{\theta})$
features $= 100$, features$_{informative} = 10$	Flow + MH	**436.48 ± 57.82**	**4.78 ± 0.63**	**-1054.97 ± 15.89**
	Gibbs	23.19 ± 22.20	0.12 ± 0.12	-1235.02 ± 13.11
features $= 200$, features$_{informative} = 20$	Flow + MH	**213.36 ± 29.37**	**0.89 ± 0.12**	**-2231.24 ± 8.28**
	Gibbs	8.78 ± 6.12	0.013 ± 0.009	-2485.91 ± 0.14
features $= 400$, features$_{informative} = 40$	Flow + MH	**141.45 ± 17.97**	**0.15 ± 0.02**	**-4627.58 ± 10.86**
	Gibbs	1.46 ± 0.23	$0.001 ± 10^{-4}$	-5306.61 ± 0.20

extremely difficult. Following Schäfer and Chopin (2013), we consider a hierarchical Bayesian model in which the target vector $\mathbf{y} \in \mathbb{Z}^d$ in linear regression is observed through:

$$\mathbf{y}|\beta, \boldsymbol{\theta}, \sigma^2, \mathbf{x} \sim \mathcal{N}(\mathbf{x} \cdot \boldsymbol{\theta} \cdot \beta, \sigma^2 I)$$

with features $\mathbf{x} \in \mathbb{R}^{d \times k}$, parameters $\beta \in \mathbb{R}^k$ and the binary vector $\boldsymbol{\theta}$ that indicates the features that are included in the model. To achieve a closed form for the marginal posterior that is independent of β and σ^2, we make the following choices following Power and Goldman (2019): $\pi(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) = \int \pi(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}, \beta, \sigma^2) \mathrm{d}\beta \, \mathrm{d}\sigma^2$ with conjugates $p(\beta|\sigma^2, \boldsymbol{\theta}) = \mathcal{N}(0, \nu^2\sigma^2 I_n \boldsymbol{\theta})$, $p(\sigma) = I\Gamma(\frac{2}{2}, \frac{\alpha w}{2})$ and, $p(\boldsymbol{\theta}) = \mathsf{Uniform}(\{0, 1\}^k)$ where $I\Gamma$ is the inverse-gamma distribution and we set ν, w, α as in George and McCulloch (1997). For this setting, we create high-dimensional synthetic datasets rather than the low-dimensional datasets usually used to demonstrate the efficacy of our method. We present the results in Table 3.

6 Conclusion

In this paper, we presented a flow based Monte Carlo sampler for sampling in combinatorial spaces. Our method learns a deterministic transport map from a discrete space to a simple continuous latent space where it is efficient to sample. Thereby, we sample from the discrete space by generating samples in the latent space and using the transport map to obtain samples in the discrete space. By learning a map to a simple latent space (like standard Gaussian), our method is particularly suited for high-dimensional domains where alternative samplers are not efficient due to the presence of more complex correlations. This is also reflected in our implementation which is faster and efficient as demonstrated by our results on a suite of experiments. In the future, it will be interesting to devise learning strategies for the transport map that explicitly pushes the latent space to have certain desirable properties for efficient sampling. Another di-

rection could be to extend the framework of SurVAE Flow layers to incorporate underlying symmetries and invariance in the target domain.

References

Afshar, H. M. and Domke, J. (2015). Reflection, Refraction, and Hamiltonian Monte Carlo. In *Advances in neural information processing systems*, pages 3007–3015.

Albergo, M., Kanwar, G., and Shanahan, P. (2019). Flow-based Generative Models for Markov Chain Monte Carlo in Lattice Field Theory. *Physical Review D*, 100(3):034515.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. springer.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Dinh, V., Bilge, A., Zhang, C., and Matsen IV, F. A. (2017). Probabilistic Path Hamiltonian Monte Carlo. *arXiv preprint arXiv:1702.07814*.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222.

George, E. I. and McCulloch, R. E. (1997). Approaches for Bayesian Variable Selection. *Statistica sinica*, pages 339–373.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. (2019). Flow++: Improving Flow-based Generative Models with Variational Dequantization and Architecture Design. *ICML, 2019*.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. (2019). Neutralizing Bad Geometry in Hamiltonian Monte Carlo using Neural Transport. *ICML, 2019*.

Hubbard, J. (1959). Calculation of Partition Functions. *Physical Review Letters*, 3(2):77.

Marzouk, Y., Moselhy, T., Parno, M., and Spantini, A. (2016). Sampling via Measure Transport: An Introduction. In Ghanem, R., Higdon, D., and Owhadi, H., editors, *Handbook of Uncertainty Quantification*, pages 1–41. Springer.

Neal, R. M. et al. (2011). MCMC using Hamiltonian Dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

Nielsen, D., Jaini, P., Hoogeboom, E., Winther, O., and Welling, M. (2020). SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows. *arXiv preprint arXiv:2007.02731*.

Nishimura, A., Dunson, D. B., and Lu, J. (2020). Discontinuous Hamiltonian Monte Carlo for Discrete Parameters and Discontinuous Likelihoods. *Biometrika*, 107(2):365–380.

Pakman, A. and Paninski, L. (2013). Auxiliary-Variable Exact Hamiltonian Monte Carlo Samplers for Binary Distributions. In *Advances in neural information processing systems*, pages 2490–2498.

Pardalos, P. M. (1996). Continuous Approaches to Discrete Optimization Problems. In *Nonlinear optimization and applications*, pages 313–325. Springer.

Parno, M. and Marzouk, Y. (2018). Transport Map Accelerated Markov Chain Monte Carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Peherstorfer, B. and Marzouk, Y. (2018). A Transport-based Multifidelity Pre-Conditioner for Markov Chain Monte Carlo.

Power, S. and Goldman, J. V. (2019). Accelerated Sampling on Discrete Spaces with Non-Reversible Markov Processes. *arXiv preprint arXiv:1912.04681*.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *ICML*.

Rudin, W. (1987). *Real and Complex Analysis*. McGraw-Hill, 3rd edition.

Schäfer, C. and Chopin, N. (2013). Sequential Monte Carlo on Large Binary Sampling Spaces. *Statistics and Computing*, 23(2):163–184.

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

Tabak, E. G. and Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233.

Theis, L., Oord, A. v. d., and Bethge, M. (2015). A Note on the Evaluation of Generative Models. *arXiv preprint arXiv:1511.01844*.

Titsias, M. K. and Yau, C. (2017). The Hamming Ball Sampler. *Journal of the American Statistical Association*, 112(520):1598–1611.

Uria, B., Murray, I., and Larochelle, H. (2013). RNADE: The Real-Valued Neural Autoregressive Density Estimator. In *Advances in Neural Information Processing Systems*, pages 2175–2183.

Villani, C. (2008). *Optimal Transport: Old and New*. Springer.

Zanella, G. (2020). Informed Proposals for Local MCMC in Discrete Spaces. *Journal of the American Statistical Association*, 115(530):852–865.

Zhang, Y., Ghahramani, Z., Storkey, A. J., and Sutton, C. A. (2012). Continuous Relaxations for Discrete Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 3194–3202.