Concrete Score Matching: Generalized Score Matching for Discrete Data

Chenlin Meng*
Stanford University
chenlin@cs.stanford.edu

Kristy Choi*
Stanford University
kechoi@cs.stanford.edu

Jiaming Song
NVIDIA
jiamings@nvidia.com

Stefano Ermon
Stanford University, CZ Biohub
ermon@cs.stanford.edu

Abstract

Representing probability distributions by the gradient of their density functions has proven effective in modeling a wide range of continuous data modalities. However, this representation is not applicable in discrete domains where the gradient is undefined. To this end, we propose an analogous score function called the "Concrete score", a generalization of the (Stein) score for discrete settings. Given a predefined neighborhood structure, the Concrete score of any input is defined by the rate of change of the probabilities with respect to local directional changes of the input. This formulation allows us to recover the (Stein) score in continuous domains when measuring such changes by the Euclidean distance, while using the Manhattan distance leads to our novel score function in discrete domains. Finally, we introduce a new framework to learn such scores from samples called Concrete Score Matching (CSM), and propose an efficient training objective to scale our approach to high dimensions. Empirically, we demonstrate the efficacy of CSM on density estimation tasks on a mixture of synthetic, tabular, and high-dimensional image datasets, and demonstrate that it performs favorably relative to existing baselines for modeling discrete data.

1 Introduction

When estimating a statistical model, the representation of the underlying probability distribution has profound implications on the downstream modeling task. Likelihood-based model families such as Variational Autoencoders (VAEs) [1, 2, 3, 4], normalizing flows [5, 6, 7, 8, 9, 10], and diffusion models [11, 12, 13] are forced to either approximate the intractable normalizing constant or utilize restrictive model architectures; implicit models [14, 15, 16, 17] try to capture the underlying sampling process by relying on unstable adversarial training procedures. On the other hand, representing the distribution as the gradient of the log probability density function—also known as the (Stein) score—allows us to circumvent such issues. This is key to score matching's success on a wide range of continuous data modalities [18, 19, 20]. In fact, its recent resurgence has led to significant advances in machine learning applications of density estimation, such as image generation [21, 13] and audio synthesis [22, 23], among others.

However, score matching approaches that rely on modeling the gradient of the data distribution are inherently designed for continuous data; such methods hinge on the *existence* of the gradient, which is undefined for discrete domains [19]. Given the ubiquitous nature of structured, discrete data in

*Joint first author

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

our world today such as graphs, text, genomic sequences, images, we desire an approach that would allow us to expand the successes of score-based generative models into discrete domains.

To address this challenge, we first introduce the Concrete score: a generalization of the (Stein) score that is amenable to both continuous and discrete data types[2]. Given a predefined neighborhood structure, the Concrete score leverages structural information in the data to construct *surrogate gradient information* about the discrete space by considering the similarity between two neighboring examples. More precisely, the Concrete score of any input is defined by the rate of change of the probabilities with respect to directional changes of the input. In direct analogy to the continuous (Stein) score, the intuition is that the Concrete score should remain small when two examples are "close" to one another; when the two examples are very different, the Concrete score will be large. We find that measuring such changes by the Euclidean distance recovers the (Stein) score in continuous domains, while using the Manhattan distance leads to our novel score function in discrete domains.

Using this definition of the Concrete score, we then introduce a framework to learn such scores from samples called Concrete Score Matching (CSM). To successfully scale our method to high dimensions, we also propose an efficient training objective as well as several variations of our approach that improve performance in practice. Empirically, we demonstrate the efficacy of CSM on a wide range of density estimation tasks on a mixture of synthetic, tabular, and high-dimensional image datasets, and demonstrate that it performs favorably relative to existing baselines for modeling discrete data.

The contributions of our work can be summarized as follows:

1. We propose the Concrete score, a generalization of the (Stein) score to discrete domains. We construct the Concrete score such that it leverages structural information in the data to capture surrogate "gradient" information over discrete spaces.
2. We introduce a novel score matching framework for estimating such scores from samples called Concrete Score Matching (CSM). We then outline several connections between CSM and existing (continuous) score matching techniques, such denoising score matching (DSM).
3. We propose efficient learning objectives that allow our method to scale gracefully to high-dimensional datasets, and show how to port over the recent successes in continuous score matching approaches to discrete domains.

2 Preliminaries

Let $p_{\text{data}}(\mathbf{x})$ be the unknown data distribution over $\mathcal{X} \in \mathbb{R}^D$, for which we have access to i.i.d. samples $\{\mathbf{x}_i\}_{i=1}^N \sim p_{\text{data}}(\mathbf{x})$. The (Stein) score of $p_{\text{data}}(\mathbf{x})$ is the first order derivative of the log data density function $\boldsymbol{s}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x})$. The goal of score matching is to learn an unnormalized density $q_\theta(\mathbf{x})$ indexed by $\theta \in \Theta$ such that the estimated score function $\boldsymbol{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log q_\theta(\mathbf{x})$ is close to $\boldsymbol{s}(\mathbf{x})$.

In practice, we leverage a *score network* $\boldsymbol{s}_\theta : \mathbb{R}^D \to \mathbb{R}^D$ that is trained to minimize the Fisher divergence between $q_\theta(\mathbf{x})$ and $p_{\text{data}}(\mathbf{x})$ [20, 21]. Although the original objective is intractable to compute due to its dependence on the ground truth data scores $\boldsymbol{s}(\mathbf{x})$, we can leverage integration by parts [18] to simplify the objective as follows:

$$\mathcal{J}_{SM}(\theta) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\|\boldsymbol{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x}))\right] + \text{const.} \tag{1}$$

where $\text{tr}(\cdot)$ denotes the matrix trace, and the optimal score model satisfies $\boldsymbol{s}_{\theta^*}(\mathbf{x}) \approx \boldsymbol{s}(\mathbf{x})$. While the trace term in Eq. 1 remains problematic—it requires an expensive evaluation of the Hessian of the log-density function—recent works [20, 13] leverage the Skilling-Hutchinson trace estimator [25, 26] or directional derivatives [27] to efficiently approximate the training objective.

Despite their key role in successfully scaling score-based generative models to high-dimensional datasets, we note two critical limitations of existing score matching techniques: (1) $\mathcal{X}$ must be continuous; and (2) the density function $p_{\text{data}}(\mathbf{x})$ must be differentiable. This poses a significant challenge in discrete domains, where both requirements fail to hold.

[2] We borrow the terminology from "concrete mathematics" [24].

3 The Concrete Score

The above limitations prevent the direct application of score matching techniques to discrete data. We thus propose the Concrete score, a generalization of the (Stein) score for discrete settings. The key intuition is that although the gradient is undefined in discrete spaces, we can still construct a *surrogate* for the gradient by leveraging local directional changes to the input. We do so by exploiting special neighborhood structures in the data, and elaborate upon the necessary conditions on these structures to guarantee that our surrogate gradient indeed recovers a valid score function that (1) completely characterizes the distribution, and (2) is amenable for parameter estimation.

3.1 Constructing Surrogate Gradients for Discrete Data

To be more precise, let $p_{\text{data}}(\mathbf{x})$ be the data distribution over $\mathcal{X}$. We denote $\mathcal{N} : \mathcal{X} \to \mathcal{X}^K$ as the function mapping each example $\mathbf{x} \in \mathcal{X}$ to a set of neighbors, such that $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}_{n_1}, ..., \mathbf{x}_{n_k}\}$ and $K = |\mathcal{N}(\mathbf{x})|$ [3]. This neighborhood induces a particular graphical structure onto the support of $p_{\text{data}}(\mathbf{x})$, which we call the "neighborhood-induced graph", that will play a key role in constructing the surrogate gradient. We provide a formal definition below.

Definition 1 (Neighborhood-induced graph)**.** *Let $p_{\text{data}}(\mathbf{x})$ be the data distribution over $\mathcal{X}$ and $\mathcal{N}$ be the function mapping each node $\mathbf{x} \in \mathcal{X}$ to its set of neighbors. The neighborhood-induced graph $\mathcal{G}$ is the directed graph which results from adding a directed edge from $\mathbf{x}$ to each node in its neighborhood set $\mathbf{x}_n \in \mathcal{N}(\mathbf{x})$, for all $\mathbf{x} \in \text{supp}(\text{p}_{\text{data}}(\mathbf{x}))$.*

An important point is that the neighborhood structure can be asymmetric, since the neighborhood-induced graph is a directed graph. This implies that there may exist cases where $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}_1\}$ does not necessarily imply that $\mathcal{N}(\mathbf{x}_1) = \{\mathbf{x}\}$. We provide an intuitive example below.

Example 1. *When $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_5\}$, and the neighborhood structure $\mathcal{N}$ is defined as: $\mathcal{N}(\mathbf{x}_0) = \{\emptyset\}$ and $\mathcal{N}(\mathbf{x}_i) = \{\mathbf{x}_0\}$ $\forall i = 1, \ldots, 5$, the neighborhood-induced graph is the star graph in Figure 1.*

There exist a wide range of neighborhood structures, all of which yield different neighborhood-induced graphs. We visualize five common structures in Figure 1.

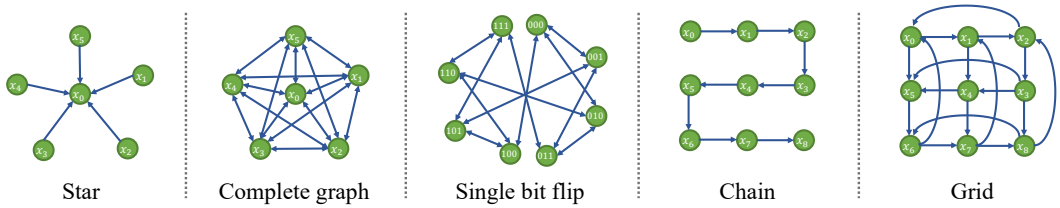


Figure 1: Examples of common neighborhood structures and their corresponding (connected) neighborhood-induced graphs, where the arrows point to the neighbors of a given node. Note that the neighborhood structure is directed.

Such graphs provide insight into the kinds of neighborhood structures that are amenable to our framework. One necessary characteristic of such graphs is *connectedness*. In fact, the five common graphical structures (among others) shown in Figure 1 are all weakly connected graphs. We emphasize that connectedness does not take the directionality of the underlying graph $\mathcal{G}$'s edges into account—it does not matter whether a node $\mathbf{x}$ has an incoming or outgoing edge.

With the above definitions in place, we can now define the surrogate gradient as the local differences between a set of examples (similar to directional derivative in the continuous case). Using this notion of the "gradient", we construct the Concrete score as the rate of change of the probabilities with respect to these local directional changes in the input $\mathbf{x}$ as defined below.

Definition 2 (Concrete score)**.** *Let $\mathcal{N}$ be a function mapping each point $\mathbf{x}$ to its set of neighbors $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}_{n_1}, ..., \mathbf{x}_{n_k}\}$. Then, the Concrete score $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x}; \mathcal{N}) : \mathcal{X} \to \mathbb{R}^{|\mathcal{N}(\mathbf{x})|}$ for a given distribution $p_{\text{data}}(\mathbf{x})$ evaluated at $\mathbf{x}$ is:*

$$\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x}; \mathcal{N}) \triangleq \left[\frac{p_{\text{data}}(\mathbf{x}_{n_1}) - p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})}, ..., \frac{p_{\text{data}}(\mathbf{x}_{n_k}) - p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} \right]^T. \tag{2}$$

[3] We consider a fixed number of neighbors K for each $\mathbf{x}$ to simplify our notation, but note that this can be generalized to a variable number of neighbors for every $\mathbf{x}$.

Although we have defined the Concrete score, we have yet to justify whether it is indeed a suitable score function for estimating $p_{\text{data}}(\mathbf{x})$. The necessary condition, which we call *completeness* [28], ensures that the Concrete score preserves enough information about $p_{\text{data}}(\mathbf{x})$ such that we can successfully learn $p_{\text{data}}(\mathbf{x})$ from data samples using score matching. We make this statement more precise in the following theorem.

Theorem 1 (Completeness)**.** *Let $p_{\text{data}}(\mathbf{x})$ be a (discrete) data distribution. Denote $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$ as the Concrete score of $p_{\text{data}}(\mathbf{x})$ with neighboring structure $\mathcal{N}$, and $\boldsymbol{c}_\theta(\mathbf{x};\mathcal{N})$ the Concrete score for a distribution $p_\theta(\mathbf{x})$ parameterized by $\theta \in \Theta$. When the graph induced by the neighborhood structure $\mathcal{N}$ is connected, $c_\theta(\mathbf{x};\mathcal{N}) = \boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$ implies that $p_\theta(\mathbf{x}) = p_{\text{data}}(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$.*

Proof sketch. If two nodes $\mathbf{x}$ and $\mathbf{x}'$ in a neighborhood-induced graph $\mathcal{G}$ share an edge, then their density ratio $p_{\text{data}}(\mathbf{x}')/p_{\text{data}}(\mathbf{x})$ can be uniquely identified using $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$. Thus when $\mathcal{G}$ is connected, the density ratio between any node pairs in $\mathcal{G}$ is uniquely identified given $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$. Therefore, knowing the density ratio between any two points uniquely identifies $p_{\text{data}}(\mathbf{x})$. □

We provide the full proof in Appendix A. We emphasize here that the connectedness of the neighborhood-induced graph (a kind of regularity condition on $\mathcal{N}$ and $p_{\text{data}}(\mathbf{x})$) was crucial for demonstrating the completness of the Concrete score.

Note that Theorem 1 only requires the neighborhood-induced graph to be connected. This implies that given a connected graph, we can augment it with additional edges—the new graph will still remain complete, but contain additional information in the augmented neighborhood structure that may help parameter estimation in practice. This is a phenomenon that we observe empirically in Section 5.3.

3.2 Connection to Stein Scores and Existing Score Matching Techniques

The form of the Concrete score in Eq. 2 suggests a natural connection with the Stein score in continuous domains. In the following proposition, we illustrate the connection between the two when we define the neighborhood structure to be similar to the grid in Figure 1. In particular, as the distance between neighboring nodes shrinks to zero, our Concrete score converges to the Stein score up to a multiplicative constant.

Proposition 1. *For $\mathbf{x} \in \mathbb{R}^D$, $p(\mathbf{x}) \in \mathcal{P}(\mathbb{R}^D)$, and $\delta > 0$, let the neighborhood structure $\mathcal{N}_\delta(\mathbf{x}) = \{\mathbf{x} + \delta \boldsymbol{e}_i\}_{i=1}^D$. Then, we have:*

$$\lim_{\delta \to 0} \frac{\boldsymbol{c}_p(\mathbf{x};\mathcal{N}_\delta)}{\delta} = \nabla_{\mathbf{x}} \log p(\mathbf{x}). \tag{3}$$

We note that while from this perspective the Concrete score can be seen as a finite-difference approximation to the continuous (Stein) score, CSM is different from finite difference score matching (FD-SM) [27]. We introduce a new family of score functions generalizable to discrete domains in the form of $(p(\mathbf{x}+\mathbf{v}) - p(\mathbf{x}))/p(\mathbf{x})$, where $\mathbf{v}$ is the direction from $\mathbf{x}$ pointing to its neighbor with length δ. On the other hand, FD-SM still attempts to match the Stein score, and approximates directional derivatives with finite difference using two neighbors in the form of $\log p(\mathbf{x}+\mathbf{v}) - \log p(\mathbf{x}-\mathbf{v})$. Nevertheless, the differences between the two forms are $o(\delta)$ as $\delta \to 0$.

3.3 Inference with Concrete Scores

To perform inference, we note that there exist several ways to define Markov chain Monte Carlo (MCMC) samplers that only rely on the Concrete score—we highlight the simplest setting of Metropolis-Hastings [29, 30, 31] for clarity of exposition. We first note that, by definition:

$$\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N}) + \mathbf{1} = \left[\frac{p_{\text{data}}(\mathbf{x}_{n_1})}{p_{\text{data}}(\mathbf{x})}, \ldots, \frac{p_{\text{data}}(\mathbf{x}_{n_k})}{p_{\text{data}}(\mathbf{x})}\right]^T \tag{4}$$

This implies that we can obtain the density ratios between the neighboring pairs $p_{\text{data}}(\mathbf{x}_{n_1})/p_{\text{data}}(\mathbf{x})$ and $p_{\text{data}}(\mathbf{x})/p_{\text{data}}(\mathbf{x}_{n_1})$ given the Concrete score. Given a proposal distribution q, our sampler will accept the proposed update $\mathbf{x}'$ with acceptance probability $A(\mathbf{x}'|\mathbf{x}) = \min\left(1, \frac{p_{\text{data}}(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p_{\text{data}}(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}\right)$. Specifically, we can choose q to sample uniformly among $\mathcal{N}(\mathbf{x})$ given a particular $\mathbf{x}$. When the neighborhood structure is symmetric, this selection leads to a simplified acceptance probability $A(\mathbf{x}'|\mathbf{x}) =$

$\min\left(1, \frac{p_{\text{data}}(\mathbf{x}')}{p_{\text{data}}(\mathbf{x})}\right)$. We note that when the underlying graph is connected, the chain is aperiodic (due to the presence of a rejection step), irreducible, and positive recurrent (since there is a finite number of discrete states), so the Markov chain is guaranteed to converge in the limit to the model distribution.

This means that we can also compute tighter bounds on log-likelihood estimates obtained by unnormalized probability models trained via CSM. In particular, we can compute both a lower bound estimated via Annealed Importance Sampling (AIS) [32] and a conservative upper bound estimated via the Reverse Annealed Importance Sampling Estimator (RAISE) [33]. This is because both AIS and RAISE allow us to approximate the intractable normalizing constant by constructing a sequence of intermediate distributions between our estimated target distribution and another proposal distribution, and we know that the Concrete score can be repurposed to obtain the necessary density ratios between neighboring pairs of distributions along this sequence.

4 Learning Concrete Scores with Concrete Score Matching

4.1 The Concrete Score Matching Objective

Next, under the assumption that $\mathcal{N}$ induces a weakly connected graph over the support of $p_{\text{data}}(\mathbf{x})$, we propose a new score matching framework called Concrete Score Matching (CSM) for estimating Concrete scores from data samples. To do so, we estimate $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x}; \mathcal{N})$ with a score model $\boldsymbol{c}_\theta(\cdot; \mathcal{N}) : \mathbb{R}^D \to \mathbb{R}^{|\mathcal{N}(\mathbf{x})|}$, where $|\mathcal{N}(\mathbf{x})|$ denotes the size of the neighborhood of $\mathbf{x}$. Our proposed training objective is quite natural, as it measures the average ℓ_2 difference between our score model $\boldsymbol{c}_\theta(\cdot; \mathcal{N})$ and the true score $\boldsymbol{c}_{p_{\text{data}}}(\cdot; \mathcal{N})$ similar to (continuous) score matching:

$$\mathcal{L}_{\text{CSM}}(\theta) = \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \left\| \boldsymbol{c}_\theta(\mathbf{x}; \mathcal{N}) - \boldsymbol{c}_{p_{\text{data}}}(\mathbf{x}; \mathcal{N}) \right\|_2^2 \tag{5}$$

In the following theorem, we demonstrate that Eq. 5 is indeed a principled learning objective: the estimated θ^* is a consistent estimator for the true underlying distribution $p_{\text{data}}(\mathbf{x})$.

Theorem 2 (Consistency)**.** *In the limit of infinite data and infinite model capacity, the optimal θ^* that minimizes Eq. 5 recovers the true Concrete score, or satisfies $\boldsymbol{c}_{\theta^*}(\mathbf{x}; \mathcal{N}) = \boldsymbol{c}_{p_{\text{data}}}(\mathbf{x}; \mathcal{N})$.*

Theorem 1 implies that the estimated (underlying) $p_{\theta^*}(\mathbf{x})$ from $c_{\theta^*}(\mathbf{x}; \mathcal{N})$ satisfies $p_{\theta^*}(\mathbf{x}) = p_{\text{data}}(\mathbf{x}) \; \forall \mathbf{x} \in \mathcal{X}$. Next, we note that Eq. 5 can be simplified to an objective that we can tractably optimize by removing its dependence on the unknown $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x}; \mathcal{N})$, similar in spirit to Eq. 1. Although the added flexibility of the score model may potentially lead to situations where it does not correspond to a valid probability distribution—similar to how continuous score models do not necessarily form a conservative vector field—this does not appear to hurt CSM's performance empirically.

Theorem 3. *Optimizing Equation (5) is equivalent to optimizing:*

$$\mathcal{J}_{\text{CSM}}(\theta) = \underbrace{\sum_{\mathbf{x}} \sum_{i=1}^{|\mathcal{N}(\mathbf{x})|} p_{\text{data}}(\mathbf{x}) \Big(\boldsymbol{c}_\theta(\mathbf{x}; \mathcal{N})_i^2 + 2\boldsymbol{c}_\theta(\mathbf{x}; \mathcal{N})_i \Big)}_{\mathcal{J}_1} - \underbrace{\sum_{\mathbf{x}} \sum_{i=1}^{|\mathcal{N}(\mathbf{x})|} 2 p_{\text{data}}(\mathbf{x}_{n_i}) \boldsymbol{c}_\theta(\mathbf{x}; \mathcal{N})_i}_{\mathcal{J}_2} \tag{6}$$

where $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}_{n_1}, ..., \mathbf{x}_{n_k}\}$ is the set of neighbors of $\mathbf{x}$.

We note that in Eq. 6, the objective $\mathcal{J}_{\text{CSM}}(\theta)$ can be approximated using Monte Carlo samples. We elaborate on the empirical training details in Section 4.2.

4.2 Efficient Training via Monte Carlo

In practice, Eq. 6 is still inefficient; it involves a summation over all $|\mathcal{N}(\mathbf{x})|$ neighbors for $\mathbf{x}$, which can be expensive for high-dimensional datasets. Therefore, we propose several modifications to the original training objective that we found to be crucial for scaling up and improving our approach.

We begin with the leftmost term $\mathcal{J}_1$. First, we approximate the outer expectation w.r.t. $p_{\text{data}}(\mathbf{x})$ with Monte Carlo samples from the empirical data distribution. Next, rather than evaluating the objective for all $|\mathcal{N}(\mathbf{x})|$ neighbors, we sample a neighbor $\mathbf{x}_{n_i}$ uniformly at random among $\mathcal{N}(\mathbf{x}_i)$ for

every $\mathbf{x}_i$ in a mini-batch. We then upweight the output from the model using this sample by $|\mathcal{N}(\mathbf{x})|$. We provide the unbiased training algorithm for the leftmost term in Algorithm 1. We note that is similar in spirit to sliced score matching [20], an approximation technique to make continuous score matching scalable to higher dimensions.

Next, we turn to the rightmost term $\mathcal{J}_2$. Similar to $\mathcal{J}_1$, we can estimate $\mathcal{J}_2$ using an unbiased estimator based on samples as demonstrated in Algorithm 2. We define $\mathcal{N}^{-1}(\mathbf{x}') = \{(\mathbf{x}, i) | \mathcal{N}(\mathbf{x})_i = \mathbf{x}'\}$ as the "reverse neighborhood" set, where an element $(\mathbf{x}, i) \in \mathcal{N}^{-1}(\mathbf{x}')$ indicates that $\mathbf{x}'$ is the i-th neighbor of $\mathbf{x}$. There could be multiple $\mathbf{x}$ with the same i in $\mathcal{N}^{-1}(\mathbf{x}')$ as in the case of the star graph. Computing and storing $\mathcal{N}^{-1}$ as a hash table mapping $\mathbf{x}$ to the set of $\mathcal{N}^{-1}(\mathbf{x})$ has a time and space complexity of at most $O(\sum_{\mathbf{x}} |\mathcal{N}(\mathbf{x})|)$ (*i.e.*, the number of edges). For special structures (*e.g.*, grid), we can design specific algorithms that bypass this process. We provide more details in Appendix C.

Algorithm 1 An unbiased estimator of $\mathcal{J}_1$

Input: p_{data}, $\mathcal{N}$, $\boldsymbol{c}_\theta$ (model).
1. Sample $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$.
2. Sample $i \sim \text{Uniform}\{1, \ldots, |\mathcal{N}(\mathbf{x})|\}$.

Output: $|\mathcal{N}(\mathbf{x})| \cdot \big(\boldsymbol{c}_\theta(\mathbf{x}; \mathcal{N})_i^2 + 2\boldsymbol{c}_\theta(\mathbf{x}; \mathcal{N})_i\big)$.

Algorithm 2 An unbiased estimator of $\mathcal{J}_2$

Input: p_{data}, $\mathcal{N}$, $\boldsymbol{c}_\theta$ (model).
1. Sample $\mathbf{x}' \sim p_{\text{data}}(\mathbf{x})$.
2. Sample $(\mathbf{x}, i) \sim \text{Uniform}(\mathcal{N}^{-1}(\mathbf{x}'))$.

Output: $2 \cdot |\mathcal{N}^{-1}(\mathbf{x}')| \cdot \boldsymbol{c}_\theta(\mathbf{x}; \mathcal{N})_i$.

We leverage the grid structure in our experiments, where the neighbors of $\mathbf{x}$ are defined to be $\mathcal{N}(\mathbf{x}) = \{\mathbf{x} \pm \mathbf{e}_d\}_{d=1}^D$. Algorithm 1 and Algorithm 2 allow us to efficiently train the model by sampling a dimension $d \in [D]$ and flipping the bit for the dth entry of $\mathbf{x}$ for each $\mathbf{x}_n$.

4.3 Denoising Concrete Score Matching

Finally, we propose another training objective to approximate Eq. 6, with a focus on computational efficiency. Similar to denoising score matching (DSM) for (Stein) score estimation [19], we derive a denoising counterpart for Concrete score estimation called "Denoising Concrete Score Matching" (D-CSM). Specifically, given a discrete data distribution $p_{\text{data}}(\mathbf{x})$ and a discrete noise distribution $\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})$, we define the perturbed data distribution $\tilde{p}(\tilde{\mathbf{x}}) = \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x})\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})$, and the posterior $q(\mathbf{x}|\tilde{\mathbf{x}}) = \frac{p_{\text{data}}(\mathbf{x})\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})}{\tilde{p}(\tilde{\mathbf{x}})}$. Then, we can show that the Concrete score of the perturbed data distribution $\tilde{p}(\tilde{\mathbf{x}})$ can be obtained via $\boldsymbol{c}_{\tilde{p}(\tilde{\mathbf{x}})}(\tilde{\mathbf{x}}; \mathcal{N}) = \sum_{\mathbf{x}} \boldsymbol{c}_{\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})}(\tilde{\mathbf{x}}; \mathcal{N}) q(\mathbf{x}|\tilde{\mathbf{x}})$. This property allows us to obtain the D-CSM objective, as shown in the following theorem:

Theorem 4 (Denoising Concrete Score Matching)**.** *The objective:*

$$\mathcal{J}_{\text{D-CSM}}(\theta) = \sum_{\mathbf{x}, \tilde{\mathbf{x}}} p_{\text{data}}(\mathbf{x})\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x}) \left\| \boldsymbol{c}_\theta(\tilde{\mathbf{x}}; \mathcal{N}) - \boldsymbol{c}_{q(\tilde{\mathbf{x}}|\mathbf{x})}(\tilde{\mathbf{x}}; \mathcal{N}) \right\|_2^2 \tag{7}$$

is minimized when $\boldsymbol{c}_\theta(\tilde{\mathbf{x}}; \mathcal{N}) = \boldsymbol{c}_{p(\tilde{\mathbf{x}})}(\tilde{\mathbf{x}}; \mathcal{N})$.

We draw a direct analogy to DSM in order to provide additional insight into D-CSM. Recall that in DSM, there exists a noise distribution $q(\tilde{\mathbf{x}}|\mathbf{x})$ (*i.e.* $\mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$) such that the magnitude of Stein scores captures the distance between the perturbed $\tilde{\mathbf{x}}$ and its clean counterpart $\mathbf{x}$. D-CSM enjoys a similar interpretation. In particular, consider an input space $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{Z}^D$ with the neighborhood structure $\mathcal{N}(\mathbf{x}) = \{\mathbf{x} + \mathbf{v} | \mathbf{v} \in \{-1, 0, 1\}^D\}$. If we define $q(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{2^{D+k}}$ where k is the Manhattan distance between $\tilde{\mathbf{x}}$ and $\mathbf{x}$, then the Concrete score $\boldsymbol{c}_{\tilde{p}(\tilde{\mathbf{x}})}(\tilde{\mathbf{x}}; \mathcal{N})$ captures how close $\mathbf{x}$ is to $\tilde{\mathbf{x}}$ as measured by the Manhattan distance. Therefore, D-CSM can also be understood through the lens of a denoiser. We provide additional experimental results exploring the performance of D-CSM in Appendix B.3.

5 Experimental Results

In this section, we evaluate the performance of CSM on synthetic, tabular, and discrete image datasets on a variety of sampling and density estimation tasks. We provide additional details on specific experimental settings and hyperparameter configurations in Appendix C.

5.1 Baselines

In our experiments, we compare CSM to two relevant baselines for modeling discrete data: Ratio Matching and Discrete Score Matching with Marginalization (`Discrete Marginalization`). For conciseness, we provide additional details and derivations for their training objectives in Appendix D.

Ratio Matching. Similar to score matching, Ratio Matching [28] leverages the fact that ratios of probabilities are independent of the intractable normalizing constant (due to cancellation). The method then seeks to match the ground truth density ratios $\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}_{-d})} = \frac{q_\theta(\mathbf{x})}{q_\theta(\mathbf{x}_{-d})}$ where $\mathbf{x}_{-d}$ denotes the vector $\mathbf{x}$ with the dth entry bit-flipped (e.g. from 0 to 1).

Discrete Score Matching with Marginalization (Discrete Marginalization). The Discrete Marginalization baseline is another way of estimating discrete probability distributions with score matching as in [28]. However, we note that this approach is difficult to scale because it requires us to marginalize over the data dimension, which may also cause instabilities during training.

5.2 1-D Discrete Data

In this experiment, we consider a 16-category 1-D data distribution as shown in Figure 2. We parameterize our Concrete score model $\boldsymbol{c}_\theta(\mathbf{x}, \mathcal{N})$ using a shallow MLP model with Tanh activations, and use the Cycle neighborhood structure during training. We demonstrate that the learned Concrete score model can faithfully capture the true data distribution when trained with CSM. As shown on the left side of Figure 2, we find that CSM generates samples (blue) using MH that almost perfectly matches the samples drawn from $p_{\text{data}}(\mathbf{x})$ (green).

We also observe in Appendix A.3 that the Concrete score on $p_{\text{data}}(\mathbf{x})$ can recover the Stein score of a data distribution perturbed with triangular noise. Even though the Concrete score model is trained on discrete data (green), this connection allows us to sample with Langevin dynamics using the recovered Stein score. On the right side of Figure 2, we show how the samples generated using Langevin dynamics (smoothed orange) closely match the triangular noise-perturbed data distribution (gray). Then, we leverage a closed-form denoising formula for the perturbed distribution (Appendix A.3) to recover the clean data distribution (green) from samples obtained via Langevin dynamics (smoothed orange). The resulting denoised samples (orange) are labeled as `Denoised` in Figure 2. We provide more details and discussion in Appendix A.3.

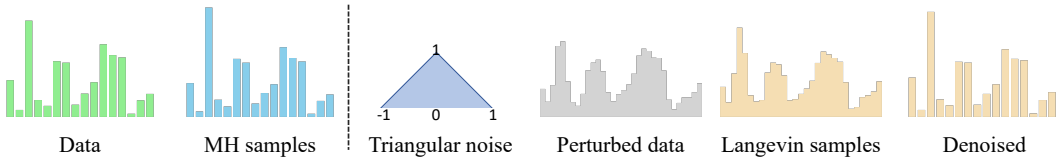


Figure 2: Sampling results from a toy 1-D discrete dataset. CSM recovers the true data distribution $p_{\text{data}}(\mathbf{x})$ using Metropolis-Hastings (left). The Concrete score can also be used to recover the Stein score of the triangular noise-perturbed data distribution, allowing for sampling with Langevin dynamics. Such samples can be denoised to recover the original (clean) data distribution (right).

5.3 Sampling with Toy 2-D Multi-Class Datasets

Next, we consider three 2-D toy benchmark datasets with multiple modes and discontinuities as commonly used in the density estimation literature [34, 35]. We quantize the data into 91×91 bins to obtain the discrete training data for the experiment (see Figure 3). We compare our method against the two baselines: (1) Ratio Matching [36]; and (2) Discrete Marginalization [28]. In particular, we found that the original equations in [28] were incorrect, and provide results using the correct training objectives (denoted as `Ratio-fixed` and `Marginal-fixed`) in Figure 3. We provide a step-by-step derivation addressing this issue with the correct expressions for the training objectives in Appendix D.

For a fair comparison, we use the same model architecture and training configurations across all methods. We use the grid neighborhood structure for training CSM. As shown in Figure 3, the samples from CSM best capture the shape of the underlying data distributions across all three datasets (leftmost column). Baselines implemented using the original equations in [28] indeed demonstrate poor performance on the density estimation task (`Ratio` and `Marginal` columns) in Figure 3.

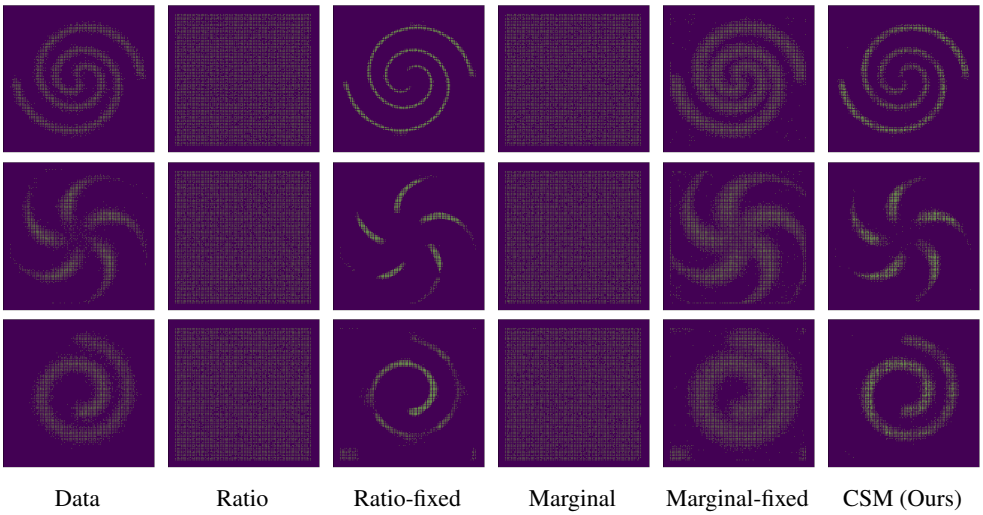


Figure 3: Sampling results on toy 2-D benchmark datasets. We find that CSM produces the highest quality samples across all 3 datasets relative to all baselines.

Datasets	Ratio Matching (↑)	Discrete Marginalization (↑)	CSM (Ours) (↑)
NLTCS	-6.15	-6.21	**-6.13**
Plants	-15.44	-19.03	**-14.02**
Jester	-56.49	-57.06	**-54.91**
Amazon Diaper	**-10.69**	-42.52	-11.13
Amazon Feeding	**-12.09**	-35.96	-12.65
Amazon Gifts	-4.57	-4.28	**-4.22**
Amazon Media	**-10.22**	-13.77	-10.30
Amazon Toys	-9.83	-16.34	**-9.30**

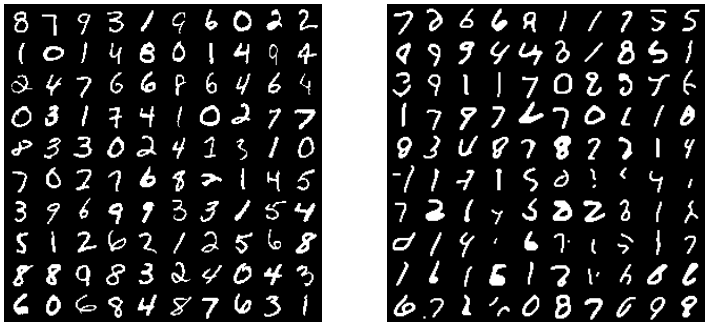
Table 1: Log-likelihood comparisons on discrete discrete tabular datasets. Higher is better. We find that CSM demonstrates good performance, almost always outperforming or performing comparably relative to the `Ratio Matching` and `Discrete Marginalization` baselines.

5.4 Likelihood Evaluation on Discrete Tabular Data

We then evaluate the performance of CSM on density estimation tasks for a wide range of tabular (discrete) datasets drawn from both the Twenty Datasets [37] and the Amazon Baby Registries benchmarks [38]. In order to evaluate likelihoods, we directly parameterize the density with a discrete autoregressive model (MADE [39]), but train the model using the baseline approaches and CSM. For a fair comparison, we use the same model architecture and experimental configurations across all methods. Similar to our previous experiments, we use the grid neighborhood structure for training CSM. As shown in Table 1, our approach (`CSM`) demonstrates favorable performance relative to the `Ratio Matching` and `Discrete Marginalization` baselines. We provide additional experimental results in Appendix B.1 and more experimental details in Appendix C.

5.5 Generative Modeling with High-dimensional Images

For our final experiment, we demonstrate that we can scale CSM to achieve good performance on complex, high-dimensional binarized image datsets. We experiment with the MNIST [40] dataset, which has 784 dimensions. We directly parameterize the Concrete score function with a U-Net [41], and train the model using the CSM based on a grid neighborhood structure. Similar to [21], we perturb the image with different levels of discrete (Categorical) noise, and train the models at different noise levels with annealing. After the Concrete score model is trained, we sample from the model using Metropolis-Hastings as discussed in Section 3.3 and follow a similar sample initialization process as [21]. We provide more details in Appendix C. We present the samples from our model in Figure 4b. We observe that our CSM approach with Metropolis-Hastings is able to generate samples that look similar to the digit examples in the training set (Figure 4a).



(a) Binary training images (b) MNIST samples (MH)

Figure 4: Image samples from CSM on the binarized MNIST dataset. We observe that CSM is able generate high-quality samples on MNIST using Metropolis-Hastings.

6 Related Work

Score Matching. Our work builds on the body of literature on score matching [18, 19] and score-based generative modeling [20, 21, 13]. Notably, we build upon the generalization of score matching to discrete data [36, 42, 28]. Our score function can be viewed through the lens of generalized score matching as in [28], where the Concrete score serves as a particular instantiation of the linear operator in their framework. However, we provide a novel perspective in terms of constructing the Concrete score via surrogate gradient information over a structured discrete space, and provide efficient training objectives for CSM. Our work is also closely related to score matching with finite difference (FD-SM) [27] as discussed in Section 3.2. However, rather than approximating the gradient with finite differences for continuous data, we leverage the first-order forward difference to construct our Concrete score function in discrete domains. In addition, CSM bears similarities to a concurrent work on scaling up Ratio Matching [28] for discrete energy-based models (EBMs) [43] called RMwGGIS—our approach can be viewed as a different way to train discrete EBMs (via the Concrete score).

Generative Modeling for Discrete Data. CSM also provides another way to train generative models for discrete data. A related work is [44], which trains a score-based model on distributions over graphs. However, they perturb the adjacency matrices with Gaussian noise and use the continuous variant of DSM. Although there exist several model families for learning binary and Categorical probability distributions such as normalizing flows [45, 46, 47], Sum-Product Networks (SPNs) [48, 49], denoising diffusion probabilistic models [50], discrete (latent) [51, 52] EBMs [43], and Generative Flow Networks (GFNs) [53], there does not yet exist a discrete score-based model that can scale to high-dimensional discrete datasets. Finally, our approach bears similarities with Gibbs with Gradients (GWG) [54]. In particular, GWG augments existing MCMC samplers with gradient information by leveraging local structure to estimate likelihood ratios for transitioning to the next state. However, GWG utilizes gradients from the probability mass functions of the underlying discrete distributions, in contrast to the way in which we construct the Concrete score function.

7 Conclusion

We introduced Concrete Score Matching (CSM), a novel framework for learning discrete probability distributions via score matching. We proposed to leverage particular kinds of structural information in the data to construct *surrogate gradient information* about the discrete space, and used this to define a valid score function. We also introduced several modifications to the original training objective that allowed CSM to scale gracefully to high-dimensional datasets, and demonstrated that CSM performs well on a variety of sampling and density estimation tasks.

However, this work is not without limitations. Since CSM depends on the neighborhood structure, certain types of graphs may work better for score matching in practice than others. Additionally, our efficient training objectives may suffer from high variance when scaling to high dimensions for particular kinds of neighborhood-induced graphs, such as the Star graph. For future work, although we fixed the number of neighbors K for each $\mathbf{x}$ in our experiments, it would be interesting to adaptively determine the optimal number of neighbors to use for each $\mathbf{x}$ during training. We also

believe that CSM can be generalized to leverage more complex neighborhood structures than those we explored in the paper. Additionally, empirically investigating the performance of D-CSM with Langevin dynamics relative to CSM would be interesting future work.

Broader Impact. This work introduces a novel score function—the Concrete score—that is defined over discrete spaces, as well as a corresponding score matching (CSM) framework that can scale to high-dimensional discrete datasets. This leads to empirical performance improvements over a range of density estimation and sampling tasks, and does not have a direct consequence on societal issues. However, we note that CSM could serve as the basis for developing more powerful generative models of structured, discrete data. Although this could lead to tangible benefits (e.g. improved generative modeling of text data), we should be mindful to take the usual precautions required for generative modeling research (e.g. against the development of deepfakes).

Acknowledgements and Funding Disclosure

We thank the anonymous reviewers for insightful discussions and feedback. KC is supported by the Qualcomm Innovation Fellowship and the Two Sigma Diversity PhD Fellowship. This research was supported by NSF (#1651565), AFOSR (FA95501910024), ARO (W911NF-21-1-0125), ONR, DOE, CZ Biohub, and Sloan Fellowship.

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[2] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[3] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[4] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[5] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[6] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

[7] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[8] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

[9] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[10] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[11] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[13] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[15] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[18] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[19] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[20] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[21] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[22] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

[23] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[24] Ronald L Graham, Donald E Knuth, Oren Patashnik, and Stanley Liu. Concrete mathematics: a foundation for computer science. *Computers in Physics*, 3(5):106–107, 1989.

[25] John Skilling. The eigenvalues of mega-dimensional matrices. In *Maximum Entropy and Bayesian Methods*, pages 455–466. Springer, 1989.

[26] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[27] Tianyu Pang, Kun Xu, Chongxuan Li, Yang Song, Stefano Ermon, and Jun Zhu. Efficient learning of generative models via finite-difference score matching. *Advances in Neural Information Processing Systems*, 33:19175–19188, 2020.

[28] Siwei Lyu. Interpretation and generalization of score matching. *arXiv preprint arXiv:1205.2629*, 2012.

[29] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[30] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[31] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[32] Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

[33] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Accurate and conservative estimates of mrf log-likelihood using reverse annealing. In *Artificial Intelligence and Statistics*, pages 102–110. PMLR, 2015.

[34] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[35] Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Uncertainty in artificial intelligence*, pages 1263–1273. PMLR, 2020.

[36] Aapo Hyvarinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on neural networks*, 18(5):1529–1531, 2007.

[37] Jan Van Haaren and Jesse Davis. Markov network structure learning: A randomized feature generation approach. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[38] Jennifer A Gillenwater, Alex Kulesza, Emily Fox, and Ben Taskar. Expectation-maximization for learning determinantal point processes. *Advances in Neural Information Processing Systems*, 27, 2014.

[39] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

[40] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[42] Aapo Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

[43] Meng Liu, Haoran Liu, and Shuiwang Ji. Gradient-guided importance sampling for learning discrete energy-based models. 2021.

[44] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.

[45] Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pages 7673–7682. PMLR, 2019.

[46] Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. *Advances in Neural Information Processing Systems*, 32, 2019.

[47] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34, 2021.

[48] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

[49] Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

[50] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[51] Fan Bao, Chongxuan Li, Kun Xu, Hang Su, Jun Zhu, and Bo Zhang. Bi-level score matching for learning energy-based latent variable models. *Advances in Neural Information Processing Systems*, 33:18110–18122, 2020.

[52] Fan Bao, Kun Xu, Chongxuan Li, Lanqing Hong, Jun Zhu, and Bo Zhang. Variational (gradient) estimate of the score function in energy-based latent variable models. In *International Conference on Machine Learning*, pages 651–661. PMLR, 2021.

[53] Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. *arXiv preprint arXiv:2202.01361*, 2022.

[54] Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, pages 3831–3841. PMLR, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Sections 3-4.1
   (b) Did you include complete proofs of all theoretical results? [Yes] Appendix A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will release the code publicly upon publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Section 5 and Appendix C
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Appendix C
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Section 5
   (b) Did you mention the license of the assets? [No] We only used publicly available datasets for our experiments.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Proofs of Theoretical Results

In this section, we provide proofs and additional discussions of the theoretical results.

A.1 Concrete Score Matching

Theorem 1 (Completeness)**.** *Let $p_{\text{data}}(\mathbf{x})$ be a (discrete) data distribution. Denote $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$ as the Concrete score of $p_{\text{data}}(\mathbf{x})$ with neighboring structure $\mathcal{N}$, and $\boldsymbol{c}_\theta(\mathbf{x};\mathcal{N})$ the Concrete score for a distribution $p_\theta(\mathbf{x})$ parameterized by $\theta \in \Theta$. When the graph induced by the neighborhood structure $\mathcal{N}$ is connected, $c_\theta(\mathbf{x};\mathcal{N}) = \boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$ implies that $p_\theta(\mathbf{x}) = p_{\text{data}}(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$.*

Proof. If two nodes $\mathbf{x}$ and $\mathbf{x}'$ in a neighborhood-induced graph $\mathcal{G}$ share an edge, then their density ratio $p_{\text{data}}(\mathbf{x}')/p_{\text{data}}(\mathbf{x})$ and $p_{\text{data}}(\mathbf{x})/p_{\text{data}}(\mathbf{x}')$ can be uniquely identified using $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$ based on the definition:

$$\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N}) + \mathbf{1} = \left[\frac{p_{\text{data}}(\mathbf{x}_{n_1})}{p_{\text{data}}(\mathbf{x})}, ..., \frac{p_{\text{data}}(\mathbf{x}_{n_k})}{p_{\text{data}}(\mathbf{x})}\right]^T. \tag{8}$$

For simplicity, we name the elements in $\mathcal{X}$: $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. Note that when $\mathcal{G}$ is a weakly connected graph, any node pair $(\mathbf{x}_1, \mathbf{x}_n)$ is connected via a path in the graph. Denoting the path between these two nodes as $\mathbf{x}_1 \to ... \to \mathbf{x}_n$, we can obtain the density ratio for any neighboring pairs on the path using the definition of $\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$. Therefore the density ratio $p_{\text{data}}(\mathbf{x}_n)/p_{\text{data}}(\mathbf{x}_1)$ can be computed using the products of the density ratios for all neighboring data pairs on the path. This implies that when $\mathcal{G}$ is weakly connected, we can uniquely recover the density ratios $\{p_{\text{data}}(\mathbf{x}_1)/p_{\text{data}}(\mathbf{x}_1), p_{\text{data}}(\mathbf{x}_2)/p_{\text{data}}(\mathbf{x}_1), \ldots, p_{\text{data}}(\mathbf{x}_N)/p_{\text{data}}(\mathbf{x}_1)\}$, which uniquely determine $p_{\text{data}}(\mathbf{x})$. Therefore, knowing the density ratio between any two points uniquely identifies $p_{\text{data}}(\mathbf{x})$.

□

Theorem 2 (Consistency)**.** *In the limit of infinite data and infinite model capacity, the optimal θ^* that minimizes Eq. 5 recovers the true Concrete score, or satisfies $\boldsymbol{c}_{\theta^*}(\mathbf{x};\mathcal{N}) = \boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$.*

Proof. It is easy to see the optimal θ^* that minimizes the following equation

$$\mathcal{L}_{\text{CSM}}(\theta) = \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \left\|\boldsymbol{c}_\theta(\mathbf{x};\mathcal{N}) - \boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})\right\|_2^2 \tag{9}$$

satisfies $\boldsymbol{c}_{\theta^*}(\mathbf{x};\mathcal{N}) = \boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N})$ almost everywhere.

□

Theorem 3. *Optimizing Equation (5) is equivalent to optimizing:*

$$\mathcal{J}_{\text{CSM}}(\theta) = \underbrace{\sum_{\mathbf{x}} \sum_{i=1}^{|\mathcal{N}(\mathbf{x})|} p_{\text{data}}(\mathbf{x}) \Big(\boldsymbol{c}_\theta(\mathbf{x};\mathcal{N})_i^2 + 2\boldsymbol{c}_\theta(\mathbf{x};\mathcal{N})_i\Big)}_{\mathcal{J}_1} - \underbrace{\sum_{\mathbf{x}} \sum_{i=1}^{|\mathcal{N}(\mathbf{x})|} 2p_{\text{data}}(\mathbf{x}_{n_i})\boldsymbol{c}_\theta(\mathbf{x};\mathcal{N})_i}_{\mathcal{J}_2} \tag{6}$$

where $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}_{n_1}, ..., \mathbf{x}_{n_k}\}$ is the set of neighbors of $\mathbf{x}$.

Proof. Recall that by definition

$$\boldsymbol{c}_{p_{\text{data}}}(\mathbf{x};\mathcal{N}) \triangleq \left[\frac{p_{\text{data}}(\mathbf{x}_{n_1}) - p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})}, ..., \frac{p_{\text{data}}(\mathbf{x}_{n_k}) - p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})}\right]^T. \tag{10}$$

$$
\begin{aligned}
&\arg\min_{\theta} \mathcal{L}_{\mathrm{CSM}}(\theta) \\
&= \arg\min_{\theta} \sum_{\mathbf{x}} p_{\mathrm{data}}(\mathbf{x}) \left\| \boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N}) - \boldsymbol{c}_{p_{\mathrm{data}}}(\mathbf{x};\mathcal{N}) \right\|_2^2 \\
&= \arg\min_{\theta} \sum_{\mathbf{x}} p_{\mathrm{data}}(\mathbf{x}) \Big[\left\| \boldsymbol{c}_{p_{\mathrm{data}}}(\mathbf{x};\mathcal{N}) \right\|_2^2 - 2\boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N})^T \boldsymbol{c}_{p_{\mathrm{data}}}(\mathbf{x};\mathcal{N}) + \left\| \boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N}) \right\|_2^2 \Big] \\
&= \arg\min_{\theta} \sum_{\mathbf{x}} p_{\mathrm{data}}(\mathbf{x}) \Big[\left\| \boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N}) \right\|_2^2 - 2\boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N})^T \boldsymbol{c}_{p_{\mathrm{data}}}(\mathbf{x};\mathcal{N}) \Big] \\
&= \arg\min_{\theta} \underbrace{\sum_{\mathbf{x}} \sum_{i=1}^{|\mathcal{N}(\mathbf{x})|} p_{\mathrm{data}}(\mathbf{x}) \Big(\boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N})_i^2 + 2\boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N})_i \Big)}_{\mathcal{J}_1} - \underbrace{\sum_{\mathbf{x}} \sum_{i=1}^{|\mathcal{N}(\mathbf{x})|} 2p_{\mathrm{data}}(\mathbf{x}_{n_i}) \boldsymbol{c}_{\theta}(\mathbf{x};\mathcal{N})_i}_{\mathcal{J}_2} \\
&= \arg\min_{\theta} \mathcal{J}_{\mathrm{CSM}}(\theta)
\end{aligned}
$$

□

A.2 Denoising Concrete Score Matching

Property 1. $\boldsymbol{c}_{\tilde{p}(\tilde{\mathbf{x}})}(\tilde{\mathbf{x}};\mathcal{N}) = \sum_{\mathbf{x}} \boldsymbol{c}_{\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})}(\tilde{\mathbf{x}};\mathcal{N}) q(\mathbf{x}|\tilde{\mathbf{x}})$

Proof. For simplicity, given a distribution $p(\mathbf{x})$, we define the operator $L[p(\mathbf{x})] = \Big[p(\mathbf{x}_{n_1}) - p(\mathbf{x}), ..., p(\mathbf{x}_{n_k}) - p(\mathbf{x}) \Big]^T$, where $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}_{n_1}, \ldots, \mathbf{x}_{n_k}\}$ are the neighbors of $\mathbf{x}$. Recall that by definition, $\tilde{p}(\tilde{\mathbf{x}}) = \sum_{\mathbf{x}} p_{\mathrm{data}}(\mathbf{x}) \tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})$ and the posterior $q(\mathbf{x}|\tilde{\mathbf{x}}) = \frac{p_{\mathrm{data}}(\mathbf{x})\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})}{\tilde{p}(\tilde{\mathbf{x}})}$. We have

$$
\begin{aligned}
c_{\tilde{p}(\tilde{\mathbf{x}})}(\tilde{\mathbf{x}};\mathcal{N}) &= \frac{L[\tilde{p}(\tilde{\mathbf{x}})]}{\tilde{p}(\tilde{\mathbf{x}})} \\
&= \frac{L[\sum_{\mathbf{x}} p_{\mathrm{data}}(\mathbf{x}) \tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})]}{\tilde{p}(\tilde{\mathbf{x}})} \\
&= \frac{\sum_{\mathbf{x}} p_{\mathrm{data}}(\mathbf{x}) L[\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})]}{\tilde{p}(\tilde{\mathbf{x}})} \text{ (by linearity)} \\
&= \sum_{\mathbf{x}} \frac{L[\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})]}{\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})} \frac{p_{\mathrm{data}}(\mathbf{x})\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})}{\tilde{p}(\tilde{\mathbf{x}})} \\
&= \sum_{\mathbf{x}} c_{\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})}(\tilde{\mathbf{x}};\mathcal{N}) q(\mathbf{x}|\tilde{\mathbf{x}})
\end{aligned}
$$

□

Theorem 4 (Denoising Concrete Score Matching)**.** *The objective:*

$$
\mathcal{J}_{\mathrm{D-CSM}}(\theta) = \sum_{\mathbf{x},\tilde{\mathbf{x}}} p_{\mathrm{data}}(\mathbf{x}) \tilde{q}(\tilde{\mathbf{x}}|\mathbf{x}) \left\| \boldsymbol{c}_{\theta}(\tilde{\mathbf{x}};\mathcal{N}) - \boldsymbol{c}_{q(\tilde{\mathbf{x}}|\mathbf{x})}(\tilde{\mathbf{x}};\mathcal{N}) \right\|_2^2 \tag{7}
$$

is minimized when $\boldsymbol{c}_{\theta}(\tilde{\mathbf{x}};\mathcal{N}) = \boldsymbol{c}_{p(\tilde{\mathbf{x}})}(\tilde{\mathbf{x}};\mathcal{N})$.

Proof. The model $\boldsymbol{c}_{\theta}$ that minimizes the least squares

$$
\theta^* = \arg\min_{\theta} \sum_{\mathbf{x},\tilde{\mathbf{x}}} p_{\mathrm{data}}(\mathbf{x}) \tilde{q}(\tilde{\mathbf{x}}|\mathbf{x}) ||\boldsymbol{c}_{\theta}(\tilde{\mathbf{x}};\mathcal{N}) - c_{\tilde{q}(\mathbf{x}|\tilde{\mathbf{x}})}(\tilde{\mathbf{x}};\mathcal{N})||^2 \tag{11}
$$

satisfies $\boldsymbol{c}_{\theta}(\tilde{\mathbf{x}};\mathcal{N}) = \sum_{\mathbf{x}} c_{\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})}(\tilde{\mathbf{x}};\mathcal{N}) q(\mathbf{x}|\tilde{\mathbf{x}}) = c_{\tilde{p}(\tilde{\mathbf{x}})}(\tilde{\mathbf{x}};\mathcal{N})$ using Property 1. □

A.3 Connection to distribution perturbed with triangular noise

As discussed in Section 5.2, our approach can be used to denoise a given data distribution perturbed with triangular noise. For simplicity, we assume $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{Z}^D$. The argument proceeds in a similar manner to how denoising score matching can be used to compute the expected posterior $\mathbb{E}_{\mathbf{x}}[p(\mathbf{x}|\tilde{\mathbf{x}})]$, where $\tilde{\mathbf{x}}$ is the perturbed data.

Definition 3 (Triangular noise)**.** *We define the PDF of the D-dimensional triangular distribution with lower limit -1, upper limit 1, and mode 0 as the following:*

$$\mathcal{T}_D(\mathbf{x}) = \begin{cases} 0 & \mathbf{x} < -1 \\ \mathbf{x}+1 & -1 \leq \mathbf{x} \leq 0 \\ 1-\mathbf{x} & 0 < \mathbf{x} \leq 1 \\ 0 & \mathbf{x} > 1 \end{cases} \tag{12}$$

Lemma 1. *Given a D-dimensional discrete distribution $p_{\text{data}}(\mathbf{x})$, let $\tilde{p}(\tilde{\mathbf{x}})$ be the distribution of $p_{\text{data}}(\mathbf{x})$ when perturbed with triangular noise as defined in Equation (12).*

$$\tilde{p}(\tilde{\mathbf{x}}) \triangleq \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \mathcal{T}_D(\tilde{\mathbf{x}} - \mathbf{x}). \tag{13}$$

Then for any $\mathbf{x}, \mathbf{y} \sim p_{\text{data}}(\mathbf{x})$ we have $\frac{\tilde{p}(\mathbf{x})}{\tilde{p}(\mathbf{y})} = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{y})}$.

Proof. It is easy to see that for any $\mathbf{x}, \mathbf{y} \in \mathcal{X} \subseteq \mathbb{Z}^D$, we have

$$\tilde{p}(\mathbf{x}) = p_{\text{data}}(\mathbf{x}) \mathcal{T}_D(\mathbf{0}),\ \tilde{p}(\mathbf{y}) = p_{\text{data}}(\mathbf{y}) \mathcal{T}_D(\mathbf{0}) \tag{14}$$

because the width of the triangular noise is less than one. Thus $\frac{\tilde{p}(\mathbf{x})}{\tilde{p}(\mathbf{y})} = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{y})}$.

□

Denote $\bar{\mathbf{x}}_d$ the integer such that the d-th index of $\tilde{\mathbf{x}}$, denoted $\tilde{\mathbf{x}}_d$, satisfies $\tilde{\mathbf{x}}_d \in [\bar{\mathbf{x}}_d, \bar{\mathbf{x}}_d + 1)$. Similar to Section 4.3, in the discrete case with perturbed triangular noise $\mathcal{T}_D(\mathbf{x})$, we define $\tilde{p}(\tilde{\mathbf{x}}) = \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \mathcal{T}_D(\tilde{\mathbf{x}}|\mathbf{x})$ and the posterior $q(\mathbf{x}|\tilde{\mathbf{x}}) = \frac{p_{\text{data}}(\mathbf{x}) \mathcal{T}_D(\tilde{\mathbf{x}}|\mathbf{x})}{\tilde{p}(\tilde{\mathbf{x}})}$, where $\mathcal{T}_D(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{T}_D(\tilde{\mathbf{x}} - \mathbf{x})$.

We have

$$\mathbb{E}_{\mathbf{x}}[q(\mathbf{x}|\tilde{\mathbf{x}})] = \sum_{\mathbf{x} \in \{\mathbf{x}_d \in \{\bar{\mathbf{x}}_d, \bar{\mathbf{x}}_d + 1\}\}} \mathbf{x} q(\mathbf{x}|\tilde{\mathbf{x}}) \tag{15}$$

$$= \sum_{\mathbf{x} \in \{\mathbf{x}_d \in \{\bar{\mathbf{x}}_d, \bar{\mathbf{x}}_d + 1\}\}} \mathbf{x} \frac{p_{\text{data}}(\mathbf{x}) \mathcal{T}_D(\tilde{\mathbf{x}}|\mathbf{x})}{p(\tilde{\mathbf{x}})} \tag{16}$$

Using the fact that $\mathcal{T}_D$ is independent among dimensions, we have

$$p_{\text{data}}(\mathbf{x}) \mathcal{T}_D(\tilde{\mathbf{x}}|\mathbf{x}) = p_{\text{data}}(\mathbf{x}) \prod_{d=1}^{D} \Big(\mathbb{1}[\mathbf{x}_d = \bar{\mathbf{x}}_d] * (\bar{\mathbf{x}}_d + 1 - \tilde{\mathbf{x}}_d) + \mathbb{1}[\mathbf{x}_d = \bar{\mathbf{x}}_d + 1] * (\tilde{\mathbf{x}}_d - \bar{\mathbf{x}}_d) \Big) \tag{17}$$

and

$$\tilde{p}(\tilde{\mathbf{x}}) = \sum_{\mathbf{x} \in \{\mathbf{x}_d \in \{\bar{\mathbf{x}}_d, \bar{\mathbf{x}}_d + 1\}\}} p_{\text{data}}(\mathbf{x}) \prod_{d=1}^{D} \Big(\mathbb{1}[\mathbf{x}_d = \bar{\mathbf{x}}_d] * (\bar{\mathbf{x}}_d + 1 - \tilde{\mathbf{x}}_d) + \mathbb{1}[\mathbf{x}_d = \bar{\mathbf{x}}_d + 1] * (\tilde{\mathbf{x}}_d - \bar{\mathbf{x}}_d) \Big) \tag{18}$$

Plugging in Equation (17) and Equation (18), we have

$$q(\mathbf{x}|\tilde{\mathbf{x}}) = \frac{\prod_{d=1}^{D} \Big(\mathbb{1}[\mathbf{x}_d = \bar{\mathbf{x}}_d] * (\bar{\mathbf{x}}_d + 1 - \tilde{\mathbf{x}}_d) + \mathbb{1}[\mathbf{x}_d = \bar{\mathbf{x}}_d + 1] * (\tilde{\mathbf{x}}_d - \bar{\mathbf{x}}_d) \Big)}{\sum_{\mathbf{y} \in \{\mathbf{y}_d \in \{\bar{\mathbf{x}}_d, \bar{\mathbf{x}}_d + 1\}\}} \frac{p_{\text{data}}(\mathbf{y})}{p_{\text{data}}(\mathbf{x})} \prod_{d=1}^{D} \Big(\mathbb{1}[\mathbf{y}_d = \bar{\mathbf{x}}_d] * (\bar{\mathbf{x}}_d + 1 - \tilde{\mathbf{x}}_d) + \mathbb{1}[\mathbf{y}_d = \bar{\mathbf{x}}_d + 1] * (\tilde{\mathbf{x}}_d - \bar{\mathbf{x}}_d) \Big)}. \tag{19}$$

Equation (19) means that given $\tilde{\mathbf{x}}$, we can compute the posterior in closed form using density ratios $\frac{p_{\text{data}}(\mathbf{y})}{p_{\text{data}}(\mathbf{x})}$, which can be evaluated using Concrete scores as long as the graph induced by the neighborhood structure is connected (based on Theorem 1). Knowing $p(\mathbf{x}|\tilde{\mathbf{x}})$ also allows us to sample from $p(\mathbf{x}|\tilde{\mathbf{x}})$ to perform denoising. Similarly, given $q(\mathbf{x}|\tilde{\mathbf{x}})$, we can also compute Equation (15) in closed form.

Recovering Stein Scores Given Concrete scores, we can uniquely recover the Stein score of $\tilde{p}(\mathbf{x})$, denoted $\mathbf{s}(\tilde{\mathbf{x}})$, using the following equation, where the d-th index of $\mathbf{s}(\tilde{\mathbf{x}})$ is defined as

$$\mathbf{s}(\tilde{\mathbf{x}})_d = \frac{p_{\text{data}}(\bar{\mathbf{x}}_d + 1) - p_{\text{data}}(\bar{\mathbf{x}}_d)}{p_{\text{data}}(\bar{\mathbf{x}}_d + 1) * (\tilde{\mathbf{x}}_d - \bar{\mathbf{x}}_d) + p_{\text{data}}(\bar{\mathbf{x}}) * (\bar{\mathbf{x}}_d + 1 - \mathbf{x}_d)} \tag{20}$$

$$= \frac{\frac{p_{\text{data}}(\bar{\mathbf{x}}_d+1)}{p_{\text{data}}(\bar{\mathbf{x}}_d)} - 1}{\frac{p_{\text{data}}(\bar{\mathbf{x}}_d+1)}{p_{\text{data}}(\bar{\mathbf{x}}_d)} * (\tilde{\mathbf{x}}_d - \bar{\mathbf{x}}_d) + (\bar{\mathbf{x}}_d + 1 - \mathbf{x}_d)}, \tag{21}$$

where $\mathbf{x}_d \in \mathbb{Z}$ and $\mathbf{x}_d \in [\mathbf{x}_d, \mathbf{x}_d + 1)$. Thus, the Stein score $\mathbf{s}(\tilde{\mathbf{x}})$ of $\tilde{p}(\mathbf{x})$ can be constructed using the density ratio from Concrete scores. Given the recovered Stein score $\mathbf{s}(\tilde{\mathbf{x}})$, we can perform Langevin dynamics to sample from $\tilde{p}(\mathbf{x})$, which gives the smoothed sample in Figure 2 (smoothed orange). We can then sample from $q(\mathbf{x}|\tilde{\mathbf{x}})$ using Equation (19) to perform closed-form denoising, which is shown in Figure 2 (orange).

A.4 Connection to Stein Scores

In this section, we study a special case of the Concrete score, as described in Proposition 1, to highlight its connection to both the continuous (Stein) score and existing score matching methods. Given a D-dimensional continuous data distribution $p(\mathbf{x})$ and $\delta > 0$, we define a particular neighborhood structure $\mathcal{N}(\mathbf{x}) = \{\mathbf{x}_{n_i}\}_{i=1}^{D}$ where $\mathbf{x}_{n_i} = p(\mathbf{x} + \delta \mathbf{e}_i)$ and $\mathbf{e}_i$ is the standard (one-hot) basis vector with the i-th element $\mathbf{e}_i = 1$. Then:

$$\frac{\boldsymbol{c}_\theta(\mathbf{x}, \mathcal{N})}{\delta} = \frac{1}{p(\mathbf{x})} \left[\frac{p(\mathbf{x} + \delta \mathbf{e}_1) - p(\mathbf{x})}{\delta}, \ldots, \frac{p(\mathbf{x} + \delta \mathbf{e}_D) - p(\mathbf{x})}{\delta} \right]^T \tag{22}$$

From Eq. 22, we can make two observations. First, we see that $\left[\frac{p(\mathbf{x}+\delta \mathbf{e}_1)-p(\mathbf{x})}{\delta}, \ldots, \frac{p(\mathbf{x}+\delta \mathbf{e}_D)-p(\mathbf{x})}{\delta} \right]^T$ approximates the directional derivative of $p(\mathbf{x})$ via a first-order forward difference operation. As a result, the scaled Concrete score function $\frac{\boldsymbol{c}_\theta(\mathbf{x},\mathcal{N})}{\delta}$ converges to $\boldsymbol{s}(\mathbf{x})$ in the limit of $\delta \to 0$:

$$\lim_{\delta \to 0} \frac{\boldsymbol{c}_\theta(\mathbf{x}, \mathcal{N})}{\delta} = \frac{1}{p(\mathbf{x})} \lim_{\delta \to 0} \left[\frac{p(\mathbf{x} + \delta \mathbf{e}_1) - p(\mathbf{x})}{\delta}, \ldots, \frac{p(\mathbf{x} + \delta \mathbf{e}_D) - p(\mathbf{x})}{\delta} \right]^T = \frac{\nabla_{\mathbf{x}} p(\mathbf{x})}{p(\mathbf{x})} = \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

which is precisely the definition of the Stein score.

B Additional Experimental Results

B.1 Likelihood Evaluation on Discrete Tabular Data

We present additional results from Section 5.4, where we trained a MADE model using standard maximum likelihood. We use the training setting as detailed in Appendix C. As shown in Table 2, this maximum likelihood baseline serves as an upper bound on performance across all score-based methods. We note that as in continuous (Stein) score matching, log-likelihoods and score matching losses are not always correlated even though they both theoretically converge to the optimal solution given infinite model capacity [20]. This is due to practical constraints such as model mis-specification or optimization challenges. We believe that this is also the case for discrete score matching, which explains why directly optimizing likelihood outperforms all other approaches in Table 2.

B.2 Neighborhood Structure Specification in Practice

In theory, our approach can use any neighborhood structure as long as the neighborhood-induced graph is connected (see Theorem 1). In practice, the choice of neighborhood structure can affect

Datasets	Ratio Matching (↑)	Discrete Marginalization (↑)	CSM (Ours) (↑)	Log-Likelihood (↑)
NLTCS	-6.15	-6.21	**-6.13**	-6.00
Plants	-15.44	-19.03	**-14.02**	-12.52
Jester	-56.49	-57.06	**-54.91**	-51.77
Amazon Diaper	**-10.69**	-42.52	-11.13	-9.82
Amazon Feeding	**-12.09**	-35.96	-12.65	-11.29
Amazon Gifts	-4.57	-4.28	**-4.22**	-3.43
Amazon Media	**-10.22**	-13.77	-10.30	-7.79
Amazon Toys	-9.83	-16.34	**-9.30**	-7.71

Table 2: Log-likelihood comparisons on discrete discrete tabular datasets. Higher is better. We find that CSM demonstrates good performance, almost always outperforming or performing comparably relative to the `Ratio Matching` and `Discrete Marginalization` baselines.

performance due to challenges in optimization and proper model specification. We provide an empirical analysis to build intuition on a series of 1-D synthetic datasets, and believe that the same intuition can generalize to higher dimensions.

First, we consider the 1-D distribution in Figure 5, where the data distribution does not contain any low density regions (regions with close to zero density). We parameterize the unnormalized probability distribution with softmax logits, and train the model using CSM. For the training objective, we explore 4 different neighborhood structures (3 different cycles as well as a fully-connected graph) as shown in Figure 5. In this setting, we observe that different neighborhood structures yield similar performances as evaluated by log-likelihood (see Figure 5).

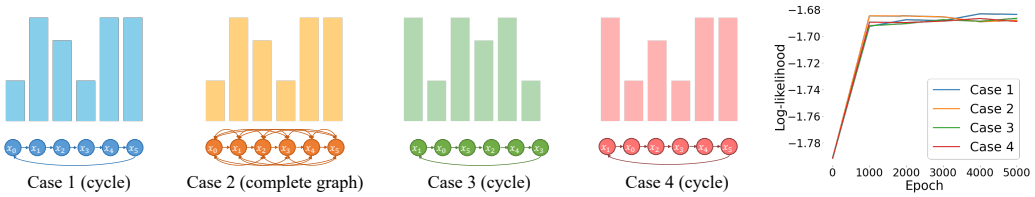


Figure 5: Examples of neighborhood structures and their corresponding log-likelihood values (higher is better) when trained with Concrete-SM.

Next, we consider a different 1-D data distribution which contains low density regions. As shown in Figure 6, we observe that the neighborhood structure in this setting plays a critical role in the final log-likelihoods: the complete graph in Case 2 outperforms all other structures. Interestingly, when the low-density regions are in between sets of high-density modes (as in Case 3 in Figure 6), we find that the model can still perform comparably relative to Case 2.

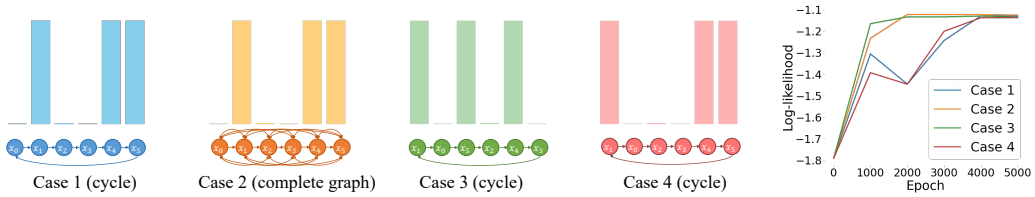


Figure 6: Examples of neighborhood structures and their corresponding log-likelihood values (higher is better) when trained with CSM.

This observation is similar to that of Stein score matching [21, 18]. Therefore, we believe that a similar noise annealing procedure followed by denoising CSM may potentially alleviate the effects of poorly chosen neighborhood structures for data distributions with low density regions. We will leave this investigation for future work. Such empirical results highlight that the best-performing structure in practice is often data-dependent. That is, a neighborhood graph which respects the particular dataset structure will perform the best out of all possible graph structures (analogous to the way in particular types of inductive biases are useful for solving specific tasks).

One interesting avenue for future research would be to draw inspiration from [43] and investigate whether it is possible to construct an "optimal" neighborhood structure for a given dataset. In this work, we uniformly sample a set of neighbors from $\mathcal{N}(\mathbf{x})$ when computing the CSM objective in practice (Algorithms 1 and 2), which can lead to issues with high variance during training. We hypothesize that an optimal proposal distribution over $\mathcal{N}(\mathbf{x})$ that minimizes variance may lead to insights on what the optimal neighborhood structure should be.

B.3 Denoising Concrete Score Matching

We present additional experiments on denoising concrete score matching (D-CSM) as introduced in Section 4.3. In particular, we consider two 2-D toy benchmark datasets as commonly used in the density estimation literature [34, 35]. We quantize the data into 91×91 equally-distanced bins to obtain the discrete training data (see `Clean data` in Figure 7). Similar to our previous experiments, we use the grid neighborhood structure as detailed in Section 5.3 for training via D-CSM.

For the discrete noise distribution $\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})$, we consider the following distribution where each marginal is an independent 1-D Categorical distribution defined as:

$$\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})_i = \begin{cases} w & \text{when } \tilde{\mathbf{x}}_i = \mathbf{x}_i \\ \frac{1-w}{90} & \text{when } \tilde{\mathbf{x}}_i \neq \mathbf{x}_i, \end{cases} \tag{23}$$

where $0 < w < 1$ and $\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})_i$ denotes the i-th entry of $\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})$. Note that higher values of w correspond to smaller noise levels. We use the same experimental setting as detailed in Appendix C.2, where we parameterize the un-normalized probability distribution with softmax logits, but train the model using D-CSM (see Equation (7)).

We present the results in Figure 7. We observe that samples from our model (green) matches samples from the ground truth noisy data distributions $\tilde{p}(\tilde{\mathbf{x}})$ (blue), indicating the practical effectiveness of D-CSM. We leave a more in-depth exploration of the D-CSM approach for future work.

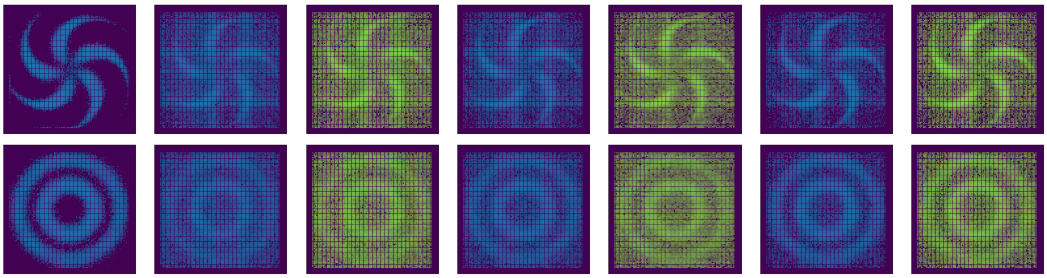


Figure 7: Samples from models trained with D-CSM. The blue samples are the ground truth samples from $\tilde{p}(\tilde{\mathbf{x}})$ obtained by perturbing the clean data distribution with $\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})$. As we increase w, the variance of the noise distribution $\tilde{q}(\tilde{\mathbf{x}}|\mathbf{x})$ decreases and the perturbed data distribution $\tilde{p}(\tilde{\mathbf{x}})$ becomes less noisy. The samples from our models (green) match the ground truth perturbed data samples (blue) across different values of w, demonstrating the effectiveness of D-CSM.

C Experimental Details

In this section, we provide more details for each of the experimental settings. We will release our code upon publication.

C.1 1-D Discrete Data

We discuss the experimental setting for Section 5.2. We use a 16-category 1-D data distribution with class labels from 0 to 15, and normalize the data to $[0, 1]$ before feeding it into the model. We use a 3-layer shallow MLP for the score network with 100 hidden units and Tanh activations. We use

the model to directly parameterize the Concrete score and define the neighborhood structure to be a Cycle structure as shown in Figure 8. We train the model using the Adam optimizer with learning rate 0.001 for 100,000 iterations. The Langevin samples and denoised samples in Figure 2 are obtained using the approach detailed in Appendix A.3.

C.2 2-D Multi-Class Datasets

For the 2-D experiments, we consider three 2-D toy benchmark datasets with multiple modes and discontinuities as commonly used in the density estimation literature [34, 35]. We quantize the data into 91×91 equally-distanced bins to obtain the discrete training data (see Figure 3). We use the grid neighborhood structure for training via CSM (see Figure 8). For a fair comparison, we use the same model architecture and training configurations across all methods. We directly parameterize the unnormalized probability distribution with softmax logits, but train the model using the baselines and CSM. Since there are 91×91 possible classes, our model directly parameterizes an unnormalized distribution with 91×91 possible values. We initialize the model to have a uniform probability across classes and train the model using the Adam optimizer with learning rate 0.0005. To sample from the model, we can either use likelihood sampling (by normalizing the model) or Metropolis-Hastings. We empirically observe that both approaches work well.

C.3 Discrete Tabular Data

For the tabular experiments, we consider tabular (discrete) datasets drawn from both the Twenty Datasets [37] and the Amazon Baby Registries benchmarks [38]. In order to evaluate likelihoods, we directly parameterize the probability distribution with a discrete autoregressive model (MADE [39]), but train the model using the baseline approaches and CSM. The MADE model uses 2 residual blocks with latent dimension 100 and Tanh activations. We train the model for 50000 iterations using the Adam optimizer with learning rate 0.0005. We use a grid neighborhood structure for CSM (see Figure 8). For a fair comparison, we use the same model architecture and experimental configurations across all methods. Similar to our previous experiments, we use the grid neighborhood structure for training via CSM (see Figure 8).

C.4 Discrete Image Data

We experiment with the binarized MNIST [40] dataset, which has 784 dimensions. We perturb the images $\mathbf{x}$ with binarized Gaussian noise $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma_i^2 I)$ $(i = 1, ..., 7)$. Specifically, given a sample from the noisy data distribution $\tilde{\mathbf{x}} \sim q(\tilde{\mathbf{x}}|\mathbf{x})$, we discretize $\tilde{\mathbf{x}}$ by mapping it to its nearest integer neighbor, and then take the modula by 2 to map $\tilde{\mathbf{x}}$ into binary bins. We note that this process is similar to applying Categorical noise. In our experiment, we use $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7\} = \{0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.65\}$. We normalize the data to lie in between $[-1, 1]$ before feeding it into the model. We directly parameterize the Concrete score function with a U-Net [41], and train the model using CSM based on a grid neighborhood structure. We train the models until convergence with the Adam optimizer using learning rate 0.0002.

We use Metropolis-Hastings for sampling from the model after training: in particular, we initialize the sample to be drawn from uniform binary noise then use the model corresponding to noise level σ_7 to perform Metropolis-Hastings updates. After that, we use the output from the model to initialize the input for sampling from the model with noise level σ_6. We then repeat the procedure and initialize the model corresponding to the previous noise level with the output from the next noise level—a process similar to [21]. We observe that each noise level requires around 100-800 steps to obtain reasonable results.

C.5 Training for Special Neighborhood Structures

In this section, we provide additional details on special neighborhood structures where we can design even more efficient training procedures than using Algorithm 1 and Algorithm 2. For simplicity, we name the elements in $\mathcal{X}$: $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$.

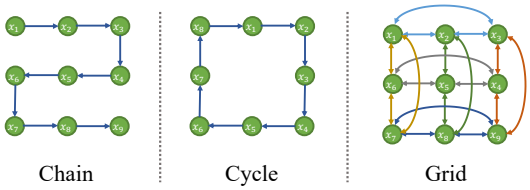


Figure 8: Special types of neighborhood structures considered in the paper.

Chain. When the neighborhood-induced graph is a Chain (Figure 8), the second term $\mathcal{J}_2$ in Equation (6) simplifies to the following via reparameterization:

$$\sum_{k=1}^{N-1} 2p_{\text{data}}(\mathbf{x}_{k+1})\boldsymbol{c}_\theta(\mathbf{x}_k;\mathcal{N})_1 = \sum_{k=2}^{N} 2p_{\text{data}}(\mathbf{x}_k)\boldsymbol{c}_\theta(\mathbf{x}_{k-1};\mathcal{N})_1. \tag{24}$$

Note that the subscript of $\boldsymbol{c}_\theta$ is always one since each node has at most one neighbor.

Cycle. Similarly, when the neighborhood-induced graph is a Cycle (Figure 8), the second term $\mathcal{J}_2$ becomes:

$$\sum_{k=1}^{N} 2p_{\text{data}}(\mathbf{x}_{k+1})\boldsymbol{c}_\theta(\mathbf{x}_k;\mathcal{N})_1 = \sum_{k=1}^{N} 2p_{\text{data}}(\mathbf{x}_k)\boldsymbol{c}_\theta(\mathbf{x}_{(k-1 \bmod N)};\mathcal{N})_1 \tag{25}$$

Thus for both Chains and Cycles, the term $\mathcal{J}_2$ in Equation (6) can be evaluated efficiently using samples from $p_{\text{data}}(\mathbf{x})$.

Grid. When the neighborhood-induced graph is a Grid (Figure 8), the second term $\mathcal{J}_2$ can be approximated efficiently by decomposing each dimension into smaller Cycles. Specifically, to estimate $\mathcal{J}_2$, we can uniformly sample a dimension d, and then use the same reparameterization approach used for the Cycle structure at each dimension.

D Additional Related Works and Corrections

In this section, we provide additional details about the baseline methods: in particular, we found that the original equations in [28] were incorrect. We elaborate upon them below.

D.1 Ratio Matching with Discrete Data

We assume with slight abuse of notation that $\mathcal{X} \in \{0,1\}^D$, and $p_{\text{data}}(\mathbf{x})$ is the unknown discrete data distribution. Ratio matching [36, 28] was originally proposed as an alternative approach to score matching for learning discrete binary probability distributions from samples. Similar to score matching, it leverages the fact that ratios of probabilities are also independent of the intractable normalizing constant (due to cancellation), and seeks to match the ground truth density ratios $\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}_{-d})} = \frac{q_\theta(\mathbf{x})}{q_\theta(\mathbf{x}_{-d})}$ where $\mathbf{x}_{-d}$ denotes the vector $\mathbf{x}$ with the dth entry bit-flipped (e.g. from 0 to 1). Concretely, ratio matching minimizes the following objective:

$$\mathcal{J}_{RM}(\theta) = \tfrac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\sum_{d=1}^{D}\left(g\left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}_{-d})}\right) - g\left(\frac{q_\theta(\mathbf{x})}{q_\theta(\mathbf{x}_{-d})}\right)\right)^2 + \left(g\left(\frac{p_{\text{data}}(\mathbf{x}_{-d})}{p_{\text{data}}(\mathbf{x})}\right) - g\left(\frac{q_\theta(\mathbf{x}_{-d})}{q_\theta(\mathbf{x})}\right)\right)^2\right] \tag{26}$$

where $g(z) = \frac{1}{1+z}$ for $z \in \mathbb{R}^+$ is a nonlinear transformation of the ratios for numerical stability. The symmetrized objective tries to minimize the squared distance between the two density ratios.

As the original ratio matching approach [36, 28] is proposed for binary data, [28] extends ratio matching to multi-class data. Following the notation in [28], let $\mathbf{x}^{\setminus i}$ denote the vector formed by dropping the i-th element (which we call $\mathbf{x}_i$) from $\mathbf{x}$. To make things concrete, this means that $p(\mathbf{x}) = p(\mathbf{x}^{\setminus i}, \mathbf{x}_i)$. We also let $q_\theta(\xi_i|\mathbf{x}^{\setminus i})$ denote the conditional probability for the i-th index of $\mathbf{x}$ taking the value ξ_i, and let $q_\theta(\sim \xi_i|\mathbf{x}^{\setminus i})$ denote the conditional probability for the i-th index of $\mathbf{x}$ not taking the value ξ_i.

The generalized ratio matching objective proposed in [28] is:

$$\theta^* = \arg\min_\theta \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} \left[g\left(\frac{p(\xi_i, \mathbf{x}^{\setminus i})}{p(\sim \xi_i, \mathbf{x}^{\setminus i})}\right) - g\left(\frac{q_\theta(\xi_i, \mathbf{x}^{\setminus i})}{q_\theta(\sim \xi_i, \mathbf{x}^{\setminus i})}\right)\right]^2 \tag{27}$$

which can be simplified as:

$$\theta^* = \arg\min_\theta \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} (1 - q_\theta(\xi_i|\mathbf{x}^{\setminus i}))^2. \tag{28}$$

It is easy to see that the optimal θ^* for Equation (28) can be achieved independent of $p_{\text{data}}(\mathbf{x})$, which is problematic. We also empirically show that model trained with Equation (28) cannot learn meaningful features based on the data (see `Ratio` in Figure 3).

As an alternative, we provide a corrected multi-class ratio matching objective. Similar to [36, 42], we assume that all the probabilities are non-zero. For simplicity, we use $p(\mathbf{x})$ to denote $p_{\text{data}}(\mathbf{x})$. The objective for multi-class ratio matching can achieved by optimizing:

$$\theta^* = \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} \left[g\left(\frac{p(\xi_i, \mathbf{x}^{\setminus i})}{p(\sim \xi_i, \mathbf{x}^{\setminus i})}\right) - g\left(\frac{q_\theta(\xi_i, \mathbf{x}^{\setminus i})}{q_\theta(\sim \xi_i, \mathbf{x}^{\setminus i})}\right)\right]^2 \tag{29}$$

$$= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \left[(1 - q_\theta(\mathbf{x}_i|\mathbf{x}^{\setminus i}))^2 + \sum_{\xi_i \neq \mathbf{x}_i} q_\theta(\xi_i|\mathbf{x}^{\setminus i})^2\right]. \tag{30}$$

Proof.

$$
\begin{aligned}
\theta^* &= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} \left[g\left(\frac{p(\xi_i, \mathbf{x}^{\setminus i})}{p(\sim \xi_i, \mathbf{x}^{\setminus i})} \right) - g\left(\frac{q_\theta(\xi_i, \mathbf{x}^{\setminus i})}{q_\theta(\sim \xi_i, \mathbf{x}^{\setminus i})} \right) \right]^2 \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} \left[p(\sim \xi_i | \mathbf{x}^{\setminus i}) - q_\theta(\sim \xi_i | \mathbf{x}^{\setminus i}) \right]^2 \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} \left[p(\xi_i | \mathbf{x}^{\setminus i}) - q_\theta(\xi_i | \mathbf{x}^{\setminus i}) \right]^2 \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} \left[q_\theta^2(\xi_i | \mathbf{x}^{\setminus i}) - 2p(\xi_i | \mathbf{x}^{\setminus i}) q_\theta(\xi_i | \mathbf{x}^{\setminus i}) \right]^2 \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} q_\theta^2(\xi_i | \mathbf{x}^{\setminus i}) - \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} 2p(\xi_i | \mathbf{x}^{\setminus i}) q_\theta(\xi_i | \mathbf{x}^{\setminus i}) \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} q_\theta^2(\xi_i | \mathbf{x}^{\setminus i}) - \sum_{i=1}^{d} \sum_{\mathbf{x}^{\setminus i}, \mathbf{x}_i} p(\mathbf{x}) \sum_{\xi_i} 2p(\xi_i | \mathbf{x}^{\setminus i}) q_\theta(\xi_i | \mathbf{x}^{\setminus i}) \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} q_\theta^2(\xi_i | \mathbf{x}^{\setminus i}) - \sum_{i=1}^{d} \sum_{\mathbf{x}^{\setminus i}, \xi_i} 2p(\xi_i, \mathbf{x}^{\setminus i}) q_\theta(\xi_i | \mathbf{x}^{\setminus i}) \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} q_\theta^2(\xi_i | \mathbf{x}^{\setminus i}) - \sum_{i=1}^{d} \sum_{\mathbf{x}} 2p(\mathbf{x}) q_\theta(\mathbf{x}_i | \mathbf{x}^{\setminus i}) \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \frac{\sum_{\xi_i} q_\theta^2(\xi_i, \mathbf{x}^{\setminus i}) + 2q_\theta(\mathbf{x}_i, \mathbf{x}^{\setminus i}) q_\theta(\mathbf{x}^{\setminus i})}{q_\theta^2(\mathbf{x}^{\setminus i})} \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \frac{-q_\theta^2(\mathbf{x}^{\setminus i}) + \sum_{\xi_i \neq \mathbf{x}_i} q_\theta^2(\xi_i, \mathbf{x}^{\setminus i}) + (\sum_{\xi_i \neq \mathbf{x}_i} q_\theta(\xi_i, \mathbf{x}^{\setminus i}))^2}{q_\theta^2(\mathbf{x}^{\setminus i})} \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \frac{\sum_{\xi_i \neq \mathbf{x}_i} q_\theta^2(\xi_i, \mathbf{x}^{\setminus i}) + (q_\theta(\mathbf{x}^{\setminus i}) - q_\theta(\mathbf{x}_i, \mathbf{x}^{\setminus i}))^2}{q_\theta^2(\mathbf{x}^{\setminus i})} \\
&= \arg\min_\theta \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{i=1}^{d} \left[(1 - q_\theta(\mathbf{x}_i | \mathbf{x}^{\setminus i}))^2 + \sum_{\xi_i \neq \mathbf{x}_i} q_\theta^2(\xi_i | \mathbf{x}^{\setminus i}) \right].
\end{aligned}
$$

□

We provide the samples trained with Equation (29) in Figure 3 (`Ratio-fixed`). We observe that the samples appear to be more reasonable then those from a model trained with Equation (28) (`Ratio`).

D.2 Discrete Marginalization

Given a multi-class discrete data distribution $p_{\text{data}}(\mathbf{x})$, discrete marginalization [28] proposes to learn the data distribution by minimizing

$$
\sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \left[\left(\frac{\mathcal{M}_i p}{p} \right)^2 + \left(\frac{\mathcal{M}_i q_\theta}{q_\theta} \right)^2 - 2\mathcal{M}_i \left(\frac{\mathcal{M}_i q_\theta}{q_\theta} \right) \right], \tag{31}
$$

where $\mathcal{M}_i$ is the marginalization operator defined as $\mathcal{M}_i : \mathcal{F}^1 \to \mathcal{F}^1 : f(\mathbf{x}) \to \int_{\mathbf{x}_i} f(\mathbf{x}) d\mathbf{x}_i$ or $\sum_{\mathbf{x}_i} f(\mathbf{x})$ in the discrete case. Thus $\frac{\mathcal{M}_i p}{p} = \frac{1}{p_{\text{data}}(\mathbf{x}_i | \mathbf{x}^{\setminus i})}$ and $\frac{\mathcal{M}_i q_\theta}{q_\theta} = \frac{1}{q_\theta(\mathbf{x}_i | \mathbf{x}^{\setminus i})}$.

The authors further simplified Equation (31) to

$$\theta^* = \arg\min_\theta \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \sum_{\xi_i} \frac{1 - 2q_\theta(\xi_i|\mathbf{x}^{\setminus i})}{q_\theta^2(\xi_i|\mathbf{x}^{\setminus i})}. \tag{32}$$

However, we observe that this is a typo in Equation (32)—the optimal θ^* can be achieved while being independent of $p_{\text{data}}(\mathbf{x})$. In our experiments, we also observe that model trained with Equation (32) cannot learn meaningful representations of the data (see `Marginal` in Figure 3).

In the following, we provide the corrected simplified version of Equation (31).

Theorem 5. *Optimizing Equation (31) is equivalent to optimizing*

$$\theta^* = \arg\min_\theta \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \left[\frac{1}{q_\theta^2(\mathbf{x}_i|\mathbf{x}^{\setminus i})} - \sum_{\xi_i} \frac{2}{q_\theta(\xi_i|\mathbf{x}^{\setminus i})} \right]. \tag{33}$$

Proof.

$$\theta^* = \arg\min_\theta \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \left[\left(\frac{\mathcal{M}_i p}{p} \right)^2 + \left(\frac{\mathcal{M}_i q_\theta}{q_\theta} \right)^2 - 2\mathcal{M}_i \left(\frac{\mathcal{M}_i q_\theta}{q_\theta} \right) \right] \tag{34}$$

$$= \arg\min_\theta \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \left[\left(\frac{\mathcal{M}_i q_\theta}{q_\theta} \right)^2 - 2\mathcal{M}_i \left(\frac{\mathcal{M}_i q_\theta}{q_\theta} \right) \right] \tag{35}$$

$$= \arg\min_\theta \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{i=1}^{d} \left[\frac{1}{q_\theta^2(\mathbf{x}_i|\mathbf{x}^{\setminus i})} - \sum_{\xi_i} \frac{2}{q_\theta(\xi_i|\mathbf{x}^{\setminus i})} \right]. \tag{36}$$

In the last equation, we used the fact that $\sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \mathcal{M}_i (\frac{\mathcal{M}_i q_\theta}{q_\theta}) = \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{\xi_i} (\frac{\mathcal{M}_i q_\theta}{q_\theta}) = \sum_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \sum_{\xi_i} \frac{1}{q_\theta(\xi_i|\mathbf{x}^{\setminus i})}$, which directly follows from Lemma 4 in [28].

□

We provide the samples trained with Equation (33) in Figure 3 (`Marginal-fixed`). We observe that the samples appear to be more reasonable then those from a model trained with Equation (32) (`Marginal`).