Score-Based Metropolis-Hastings Algorithms

Anonymous authors
Paper under double-blind review

Abstract

In this paper, we introduce a new approach for integrating score-based models with the Metropolis-Hastings algorithm. Score-based diffusion models have emerged as state-of-the-art methods for generative modeling, achieving remarkable performance in accurately learning the score function from data. However, these models lack an explicit energy function, making the Metropolis-Hastings adjustment step inaccessible. Consequently, the unadjusted Langevin algorithm is often used for sampling using estimated score functions. The lack of an energy function prevents the application of the Metropolis-adjusted Langevin algorithm and other Metropolis-Hastings methods, restricting the use of acceptance-function-based algorithms. We address this limitation by introducing a new loss function based on the *detailed balance condition*, allowing the estimation of the Metropolis-Hastings acceptance probabilities given a learned score function. We demonstrate the effectiveness of the proposed method for various scenarios, including sampling from score-based diffusion models and heavy-tail distributions.

1 Introduction

Learning to sample from intractable probability distributions is a longstanding challenge in machine learning and statistics (Goodfellow et al., 2020; Kingma, 2013; Ho et al., 2020; Dinh et al., 2016; Song et al., 2021). In particular, score-based models have emerged as a powerful class of generative algorithms, underpinning state-of-the-art (SOTA) models such as DALL · E 2 and Stable Diffusion (Rombach et al., 2022). Given samples from a target distribution, these models first estimate the score function —the gradient of the log-density of the data distribution (Hyvärinen, 2005)—using techniques like sliced score matching (Song et al., 2020) and denoising score matching (Vincent, 2011; Song & Ermon, 2019; Reu et al.). Subsequently, they leverage the learned score function and score-based sampling methods, primarily the Unadjusted Langevin Algorithm (ULA), to sample from the target distribution (Grenander & Miller, 1994; Roberts & Tweedie, 1996).

In parallel, a wealth of literature focuses on sampling from densities known only up to a normalizing constant (Hastings, 1970; Wang et al., 2024). A particular instance of this is the Metropolis-Hastings (MH) method, which provides a condition for accepting or rejecting new samples based on knowledge of a function proportional to the density (Robert et al., 2004). The MH method constructs a Markov chain based on a proposal distribution and an acceptance function, ensuring convergence when the detailed balance condition is satisfied. Various MH algorithms have been introduced, defined by their choice of proposal distributions, such as the Random Walk Metropolis (RW) algorithm (Metropolis et al., 1953), the Metropolis-adjusted Langevin algorithm (MALA) (Roberts & Tweedie, 1996; Roberts & Stramer, 2002), and the preconditioned Crank-Nicolson (pCN) algorithm (Hairer et al., 2014). While these methods are theoretically well-grounded, they require knowledge of the unnormalized density, making them impractical when only samples or an estimated score function are available.

A natural question arises when considering score-based modeling: despite the extensive research on MH sampling, its integration with score-based methods remains largely unexplored. The key technical challenge lies in the absence of an unnormalized density when only the score function (or its estimate) is available. This work proposes a unifying framework that bridges this gap, enabling the application of MH algorithms using only samples and a learned score function. Consequently, this framework can leverage the strengths of both MH and score-based methods. Our analysis is motivated by two key questions:

A) Can we estimate the acceptance function of an MH algorithm *solely* from an (estimated) score function and samples?
B) Does including an MH adjustment step *improve sample quality* for score-based models?

We answer these questions affirmatively by developing an approach to estimate an acceptance function from data. Our contributions are as follows:

- We introduce Score Balance Matching, a method for estimating the acceptance function in the MH algorithm using only an estimated score function.
- We use the learned acceptance function to incorporate MH steps into score-based samplers like ULA, enhancing the quality and diversity of generated samples.
- We evaluate our method's effectiveness and its robustness to hyperparameters by comparing it to ULA across diverse scenarios.

2 Related Work

A large body of related work focuses on estimating a score function and developing efficient sampling strategies based on it. While most approaches aim to accelerate convergence to the target distribution (e.g. through improved numerical integrators), we consider our work as a parallel approach, incorporating an acceptance function to refine samples. This work lies at the intersection of the literature on score-based models and Metropolis-Hastings methods. We first review recent advances in both areas.

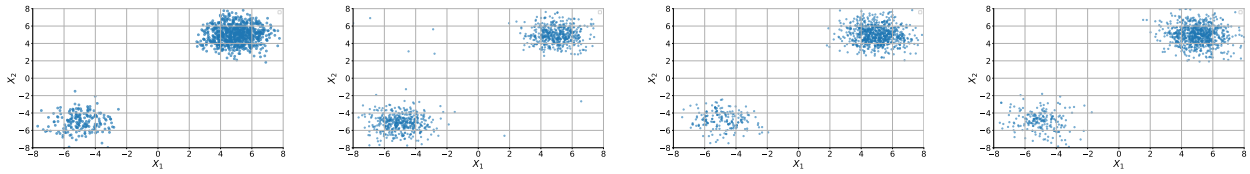
Score-Based Models. Many successful generative models sample from an estimated score function (Song & Ermon, 2019; Song et al., 2021). These methods are trained by minimizing the Fisher divergence between the true and predicted score (Hyvärinen, 2005). Significant effort has been devoted to understanding the efficacy of these sampling algorithms, with some papers investigating the asymptotic error of score-based diffusion models, as in (Chen et al., 2023; Zhang et al., 2024). Different sampling schemes have been proposed to improve sampling speed and quality (Dockhorn et al., 2021; Lu et al., 2022; Chen et al., 2024). These methods have relied heavily on various score-based samplers such as ULA or the reverse-time SDE of a diffusion process (Song et al., 2021).

Metropolis-Hastings. Metropolis-Hastings algorithms are a class of MCMC algorithms that use an acceptance function to sample from target distributions (Metropolis et al., 1953; Hastings, 1970). These algorithms generate a sequence of samples by proposing a candidate state from a proposal distribution and accepting or rejecting the candidate based on an acceptance criterion that ensures convergence to the target distribution. The efficiency of MH algorithms heavily depends on the choice of the proposal distribution, and significant research effort has focused on designing efficient proposal mechanisms (Rosenthal et al., 2011; Song et al., 2017; Titsias & Dellaportas, 2019; Davies et al., 2023; Lew et al., 2023).

Recent developments in MH algorithms have explored the use of adaptive proposals to improve convergence rates and reduce autocorrelation in the generated samples (Andrieu & Thoms, 2008; Haario et al., 2001). These adaptive MH methods adjust the proposal distribution during the sampling process to better capture the geometry of the target distribution, leading to faster and more accurate sampling (Roberts & Rosenthal, 2009; Brooks et al., 2011; Hirt et al., 2021; Biron-Lattes et al., 2024).

3 Score-Based Metropolis-Hastings

In this section, we first review score matching and MH algorithms in Sections 3.1 and 3.2 before we introduce Score Balance Matching in Section 3.3, a novel approach for learning the acceptance function from the score function. Let $X \sim p$ be a random variable, with support $\text{supp}(X) = \mathcal{X} \subset \mathbb{R}^d$, where p denotes its probability density function. We assume that p is unknown, and only samples of X are observed. Denote these observed samples by $\{x^{(i)}\}_{i=1}^N$, where N is the sample size.



(a) Original samples, weights $(0.8, 0.2)$.

(b) ULA samples, weights $(0.52, 0.48)$.

(c) RW samples, weights $(0.79, 0.21)$.

(d) MALA samples, weights $(0.79, 0.21)$.

Figure 1: Comparison of sampling methods for a mixture of two Gaussians: (a) Original distribution, (b) ULA with the true scores, (c) RW with the true distribution, and (d) MALA with true scores and distribution. The plots demonstrate the impact of slow mixing between modes, highlighting the challenges in low-density regions. We report the empirical weights of the mixture given by each sampling method.

3.1 Score Matching

Score-based models are grounded in score matching (Hyvärinen, 2005), which minimizes the Fisher divergence between the estimated and true gradients of the log data distribution, defined as follows.

Definition 1 (Fisher Divergence)**.** *The Fisher divergence $D_\nabla(p_1 \| p_2)$ between two probability distributions p_1 and p_2 is defined as:*

$$D_\nabla(p_1 \| p_2) = \mathbb{E}_{x \sim p_1} \left[\| \nabla_x \log p_1(x) - \nabla_x \log p_2(x) \|^2 \right].$$

Given a class of hypothesis function $\mathcal{S} \subset \{s : \mathcal{X} \to \mathbb{R}^d\}$, score matching aims to find $s^* \in \mathcal{S}$ that minimizes the Fisher divergence to the data distribution p:

$$s^* \in \arg\min_{s \in \mathcal{S}} \mathbb{E}_{x \sim p} \left[\| \nabla_x \log p(x) - s(x) \|^2 \right] \tag{1}$$

Since the score function is not directly observed, Hyvärinen proved that minimizing the loss function in equation 1 is equivalent to minimizing the following loss function:

$$\mathbb{E}_{x \sim p} \left[\frac{1}{2} \| s(x) \|^2 + \operatorname{tr}\left(\nabla_x s(x) \right) \right]. \tag{2}$$

Several variants of this loss have been proposed to circumvent the cumbersome estimation of the Hessian term in equation 2, including sliced score matching and denoising score matching, defined as follows:

Definition 2 (Sliced Score Matching)**.** *Sliced score matching (SSM) projects the score function onto random directions $v \sim \mathcal{N}(0, I_d)$:*

$$\mathbb{E}_{x \sim p, v \sim \mathcal{N}(0, I_d)} \left[\frac{1}{2} \| s(x) \|^2 + v^\top \nabla_x s(x) v \right]. \tag{3}$$

Definition 3 (Denoising Score Matching)**.** *Given a noise distribution $p_\sigma(\tilde{x}|x)$, the loss function is:*

$$\mathbb{E}_{x \sim p, \tilde{x} \sim p_\sigma(\tilde{x}|x)} \left[\| \nabla_{\tilde{x}} \log p_\sigma(\tilde{x}|x) - s(\tilde{x}) \|^2 \right]. \tag{4}$$

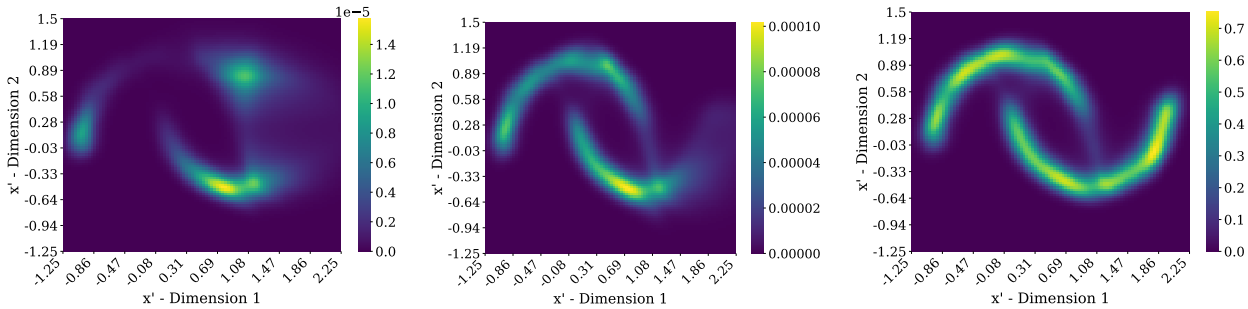
3.2 Metropolis-Hastings

MH involves an *acceptance function* $a : \mathcal{X} \times \mathcal{X} \to [0, 1]$, which defines the probability $a(x', x)$ of transitioning from the current $x \in \mathcal{X}$ to a proposal $x' \in \mathcal{X}$. A common choice of the acceptance function is given by:

$$a(x', x) = \min \left\{ 1, \frac{p(x') q(x|x')}{p(x) q(x'|x)} \right\} \tag{5}$$

where $p(x)$ is the target density and $q(x|x')$ is the proposal distribution. The MH algorithm proceeds as follows:

- **Initialize:** Set $x_1 \in \mathcal{X}$, number of iterations T, and proposal distribution $q(x'|x)$.



(a) $\lambda = 0$ (b) $\lambda = 0.1$ (c) $\lambda = 1.0$

Figure 2: Visualizing the acceptance probabilities for different values of x' for a given initial state $x = (0.0, 0.0)$. We plot the results for training an acceptance network with three different regularization values $\lambda \in \{0, 0.1, 1.0\}$.

- **For** $t = 1, \ldots, T$:
  - Propose $x' \sim q(x'|x_t)$.
  - Compute $a(x', x_t)$ in equation 5 as the acceptance probability.
  - Draw $u \sim \text{Uniform}(0, 1)$. If $u \leq a(x', x_t)$, accept $x_{t+1} = x'$, otherwise set $x_{t+1} = x_t$.

Note that for computing $a(x', x_t)$ in equation 5, a function proportional to $p(x)$ is needed.

Detailed Balance. Although the acceptance function is often defined by equation 5, a sufficient condition for ensuring convergence to the target distribution is that the acceptance ratio satisfies the detailed balance condition:

$$\forall x, x' \in \mathcal{X}, \quad \frac{a(x', x)}{a(x, x')} = \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}. \tag{6}$$

In particular, any acceptance function satisfying the detailed balance condition above ensures that the Markov chain has the target distribution $p(x)$ as its unique stationary distribution. Note that the standard acceptance function in equation 5 is one specific solution that satisfies this condition.

3.3 Score Balance Matching

We now introduce our new objective function called Score Balance Matching (SBM). Let q be a proposal distribution and let $\mathcal{A}$ be the set of valid acceptance functions for a given proposal q and target p.

$$\mathcal{A} = \{a : \mathcal{X} \times \mathcal{X} \to [0, 1] \mid a \text{ satisfies equation 6 almost surely}\}.$$

The SBM objective is defined as follows:

$$\begin{aligned}\mathcal{L}(a) = \mathbb{E}_{x \sim p, x' \sim q(\cdot|x)} \Big[&\|\nabla \log a(x', x) - \nabla \log a(x, x') \\ &- \nabla \log p(x') + \nabla \log p(x) \\ &- \nabla \log q(x \mid x') + \nabla \log q(x' \mid x)\|^2 \Big],\end{aligned} \tag{7}$$

where ∇ denotes the gradient with respect to $x, x' \in \mathbb{R}^d$, i.e., $\nabla = \nabla_{x,x'}$. The terms $\nabla \log p(x)$ and $\nabla \log p(x')$ represent the score functions at x and x', which can be approximated with a score model (e.g., obtained from score matching). We can express these score functions as:

$$\nabla \log p(x) = \nabla_{x,x'} \log p(x) = (\nabla_x \log p(x), \mathbf{0})^\top,$$

and similarly,

$$\nabla \log p(x') = (\mathbf{0}, \nabla_{x'} \log p(x'))^\top,$$

where $\mathbf{0} \in \mathbb{R}^d$ is the zero vector. Finally, we note that the gradients of the proposal distribution $q(x'|x)$ are typically known by design when the proposal function is specified.

Proposition 1. *We have that, for every function $a : \mathcal{X} \times \mathcal{X} \to [0, 1]$*

$$\mathcal{L}(a) = 0 \iff a \in \mathcal{A}. \tag{8}$$

Therefore, if we minimize SBM to zero, we are guaranteed to have a valid acceptance function. In practice, the acceptance function a can be parameterized using an expressive class of functions and optimized to minimize SBM. Since we estimate the acceptance function from finite samples, we establish a generalization bound to quantify the approximation error when learning $a(x', x)$ using a function class $\mathcal{F}$ (e.g., neural networks). We start by making the following assumptions.

Assumption 1. *$\mathcal{X}$ is compact.*

Assumption 2. *$\nabla \log p$ and $\nabla \log q$ are continuous on $\mathcal{X} \times \mathcal{X}$.*

Assumption 3. *$\forall a \in \mathcal{F}, \nabla \log a$ is continuous on $\mathcal{X} \times \mathcal{X}$.*

We note that Assumption 3 can be achieved by design. However, for many applications, Assumption 1 may not hold. To this end, we propose to employ the fact that the score function is scale invariant:

$$\nabla_x \log p(x) = \nabla_x \log p^*(x), \quad \forall x \in \mathcal{X}.$$

where, $\mathcal{X} \subset \mathbb{R}^d$ is chosen as a large compact subset of the support and p^* is a truncated density:

$$p^*(x) = \frac{p(x)}{Z} \mathbb{1}\left(x \in \mathcal{X}\right),$$

where $Z = \int_{x \in \mathcal{X}} p(x)dx$. In other words, by replacing the true target p with its truncated version p^* for SBM, Assumption 1 can also hold. Next we show the finite sample guarantee.

Proposition 2 (Finite-Sample Generalization Bound)**.** *Let $\widehat{\mathcal{L}_N}(a)$ be the empirical SBM loss computed over N i.i.d. training samples. Then, under Assumptions 1, 2, and 3, we have that with probability at least $1 - \delta$,*

$$\sup_{a \in \mathcal{F}} \left| \mathcal{L}(a) - \widehat{\mathcal{L}_N}(a) \right| \leq 2\mathcal{R}_N(\mathcal{F}) + \mathcal{O}\left(\sqrt{\frac{\log(1/\delta)}{N}} \right), \tag{9}$$

where $\mathcal{R}_N(\mathcal{F})$ is the empirical Rademacher complexity of class $\mathcal{F}$.

For the scope of this paper, we will instantiate our method on three representative MH algorithms, specified by their proposal functions: the *Random Walk Metropolis-Hastings* (RW), the *Metropolis-Adjusted Langevin Algorithm* (MALA), and the *Preconditioned Crank-Nicolson* (pCN) algorithm. We next briefly review these methods.

RW: The proposal function for RW is a Gaussian centered at the current state, i.e., $q(x'|x) = \mathcal{N}\left(x, \sigma^2 I\right)$.

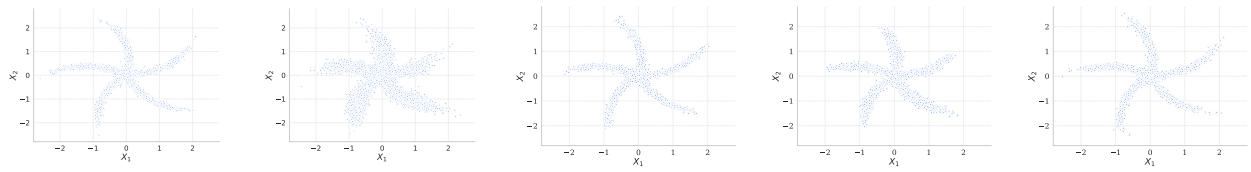
MALA: The proposal distribution in MALA uses both the current state and the gradient of the log-probability, i.e., $q(x'|x) = \mathcal{N}\left(x + \frac{\varepsilon^2}{2} \nabla \log p(x), \varepsilon^2 I\right)$.

pCN: The proposal function for pCN is given by $x' = \sqrt{1 - \beta^2} x + \beta \xi$, where $\xi \sim \mathcal{N}(0, I)$.

When the acceptance function is learned using SBM, we refer to these methods as **Score RW**, **Score MALA**, and **Score pCN**, respectively. While we illustrate our results on these three MH algorithms, we note that *the proposed framework is general* and can be applied to other proposal functions. In Appendix A, we include another approach to approximate the acceptance function.

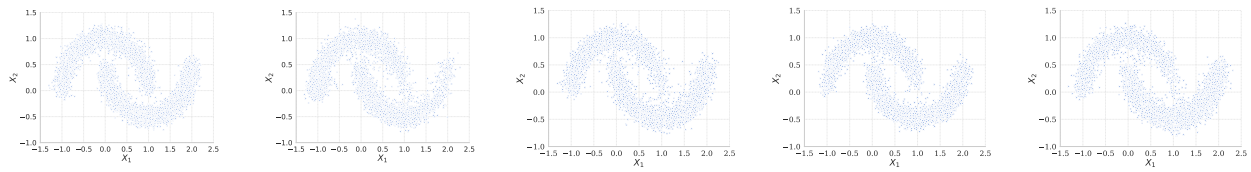
3.4 Motivation

Our approach is motivated by three key challenges. First, traditional MH algorithms typically require the knowledge of the unnormalized density, while it is common in generative modeling to assume that only the



(a) Original samples. (b) ULA. (c) Score RW. (d) Score MALA (e) Score pCN.

Figure 3: Comparison of different methods on the Pinwheel dataset.



(a) Original samples. (b) ULA. (c) Score RW. (d) Score MALA. (e) Score pCN.

Figure 4: Comparison of different methods on the Moons dataset.

score function can be accurately estimated. Second, ULA, widely used for sampling from score-based models, suffers from discretization errors that can lead to bias and slow-mixing. Consequently, by incorporating an MH adjustment step, we address a key limitation observed in ULA when sampling from distributions with multiple modes, particularly when these modes are separated by regions of low density. Specifically, when the data distribution is a mixture $p(x) = \pi p_1(x) + (1-\pi)p_2(x)$, where $p_1(x)$ and $p_2(x)$ are distinct and largely disjoint distributions, the gradient of the log probability density, $\nabla_x \log p(x)$, becomes misleading. In regions where $p_1(x)$ dominates, the score is driven solely by $p_1(x)$, and similarly for $p_2(x)$. Consequently, ULA, which relies on these score gradients, fails to correctly sample from the mixture distribution as it does not properly account for the mixing proportions π and $1-\pi$. This leads to an incorrect estimation of the relative densities between the modes. The faster mixing time of MALA compared to Langevin has been well-established in the literature as in (Dwivedi et al., 2019; Wu et al., 2022). To illustrate this, we conduct experiments to generate a mixture of Gaussian distributions with $p_1 = \mathcal{N}\left((5,5)^\top, I\right)$ and $p_2 = \mathcal{N}\left((-5,-5)^\top, I\right)$, with $\pi = 0.8$. Figure 1 shows that ULA produces samples that *misrepresent the relative weights* of the modes. In contrast, standard MH algorithms *significantly alleviate this problem*, generating samples with mode proportions that closely match the mixture weights.

4 Algorithm

In this section, we formulate the loss function and training algorithm for the acceptance function, which we model using neural networks.

Entropy Regularization. In addition to the original objective function introduced in equation 7, our algorithm involves an entropy regularization term. We first motivate it with the following result.

Proposition 3. *Let $a : \mathcal{X} \times \mathcal{X} \to [0,1]$ and let $M \geq 1$ be a scaling factor. Then,*

$$\mathcal{L}(a) = 0 \implies \mathcal{L}(\frac{a}{M}) = 0$$

It follows from Proposition 3 that there are infinitely many solutions to the detailed balance condition, including cases where the acceptance probabilities can become infinitesimally small. While each of these acceptance functions is a mathematically valid solution to the detailed balance condition, in practice, sampling with these low-acceptance probability functions can lead to always rejecting the proposal, resulting in a failure to converge within a practical timeframe. Moreover, these infinitesimal values often cause training instabilities in the neural network that parameterizes the acceptance function.

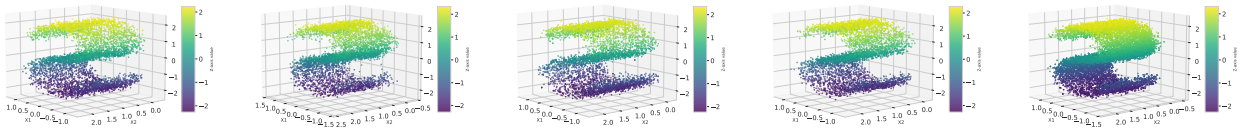
Therefore, if we *solely* optimize the loss function in equation 7, we risk converging to minima that are legitimate solutions but impractical due to their very low acceptance probabilities. To address this issue, we propose adding an entropy regularization term to our loss function. The modified loss function is given by:

$$\mathcal{L}_r(a) = \mathbb{E}\Big[\|\nabla \log a(X', X) - \nabla \log a(X, X') - \tilde{s}(X') + \tilde{s}(X) - \nabla \log q(X \mid X') + \nabla \log q(X' \mid X)\|^2\Big] + \lambda\, \mathbb{E}\left[H(a(X', X))\right], \tag{10}$$

where:

$$H(a(X', X)) = \log(a(X', X))a(X', X) + \log(1 - a(X', X))\,(1 - a(X', X))\,.$$

and $\lambda \geq 0$ is the weight of the regularization term. This regularization term encourages acceptance functions with higher entropy acceptance probabilities.



(a) Original samples. (b) ULA. (c) Score RW. (d) Score MALA. (e) Score pCN.

Figure 5: Comparison of different methods on the S-curve dataset.

To illustrate the importance of the regularization term, we present the results using the Moons dataset from `scikit-learn` (Pedregosa et al., 2011). Based on an estimated score function, we estimate an acceptance function to sample from the Moons dataset using the proposed Score RW algorithm. Training is done for different values of the regularization parameter λ, and the results are plotted in Figure 2. It can be observed that the acceptance probability is significantly enhanced with the regularization whereas training without the entropy regularization results in acceptance probability close 0. Although this acceptance probability remains mathematically valid (i.e., it will still converge to the target distribution), the convergence time may become impractically long.

Implementation. Typically, only samples from the target distribution p are observed, and the true $\nabla \log p$ is unknown. However, parametrizing the score with a neural network has proven effective in practice. We assume access to a score approximation $\tilde{s}\,(x) \approx \nabla \log p\,(x)$, estimated via score matching. Note that this is often the case in practice where an estimated score function $\tilde{s}$ is used in conjunction with ULA for sampling. The final loss function used to train the acceptance network in our score MH framework consists of two components: the primary loss term, which enforces the reversibility condition, and an entropy regularization term. The primary loss is designed to minimize the difference between the gradients of the log acceptance probability and the gradients of the log posterior and proposal distributions.

$$\mathcal{L}_{\text{emp}}(a) = \frac{1}{N}\sum_{i=1}^{N} l_i(a) + \frac{\lambda}{N}\sum_{i=1}^{N} h_i(a) \tag{11}$$

with

$$l_i(a) = \Big\|\nabla \log a\left(x'^{(i)}, x^{(i)}\right) - \nabla \log a\left(x^{(i)}, x'^{(i)}\right) - \tilde{s}\left(x'^{(i)}\right) + \tilde{s}\left(x^{(i)}\right) - \nabla \log q\left(x^{(i)} \mid x'^{(i)}\right) + \nabla \log q\left(x'^{(i)} \mid x^{(i)}\right)\Big\|^2$$

and

$$h_i(a) = \log a\left(x'^{(i)}, x^{(i)}\right) \cdot a\left(x'^{(i)}, x^{(i)}\right) + \log\left(1 - a\left(x'^{(i)}, x^{(i)}\right)\right) \cdot \left(1 - a\left(x'^{(i)}, x^{(i)}\right)\right).$$

Algorithm 1 Acceptance Loss Function

Input: Acceptance net $a(x', x)$, Score Net s, batch of samples b_x, batch of proposals $b_{x'}$, regularization parameter λ, gradient clipping threshold C.

Step 1: Compute gradients $\nabla \log a(x', x)$ and $\nabla \log a(x, x')$ using Autograd for every $(x, x') \in b_x \times b_{x'}$.

Step 2: Compute score gradients $s(x)$ and $s(x')$ for every $(x, x') \in b_x \times b_{x'}$.

Step 3: Compute proposal gradients $\nabla q(x|x')$ and $\nabla q(x'|x)$ for every $(x, x') \in b_x \times b_{x'}$.

Step 4: Clip all gradients to avoid instability using threshold C.

Step 5: Compute the loss $\mathcal{L}_{\text{emp}}$ as defined in equation 11.

Output: $\mathcal{L}_{\text{emp}}$

and where $x'^{(i)} = \alpha v' + (1 - \alpha)\tilde{x}$, with $\tilde{x} \sim p$ and $v' \sim q(\cdot \mid \tilde{x})$, where $\alpha \in [0, 1]$ controls how much x' deviates from the data distribution towards the proposal. In the first epochs of training, we begin with smaller values of α to favor data from the true distribution, and then gradually increase α to estimate the acceptance function around the proposal region. This approach is motivated by the fact that the score function may not be well-estimated outside of the data region, and starting with smaller α helps prevent biasing the training with poorly learned scores. We present the pseudo-code for the proposed loss function in Algorithm 1 and the training procedure in Algorithm 2. We made several design choices that we observed to be beneficial for learning stable acceptance probabilities:

- Using residual blocks akin to He et al. (2016).
- Employing smooth activation functions, e.g., GeLU.
- Clipping the gradients of the log acceptance function and the gradients of the log densities. This enables numerical stability and is also motivated by assumptions 2 and 3.

Algorithm 2 Training the Acceptance Network

Input: Acceptance Network a, Score Network s, dataset D, number of epochs N, sequence $\{\alpha_i\}_{i=1}^{N} \subset [0, 1]$ (increasing).

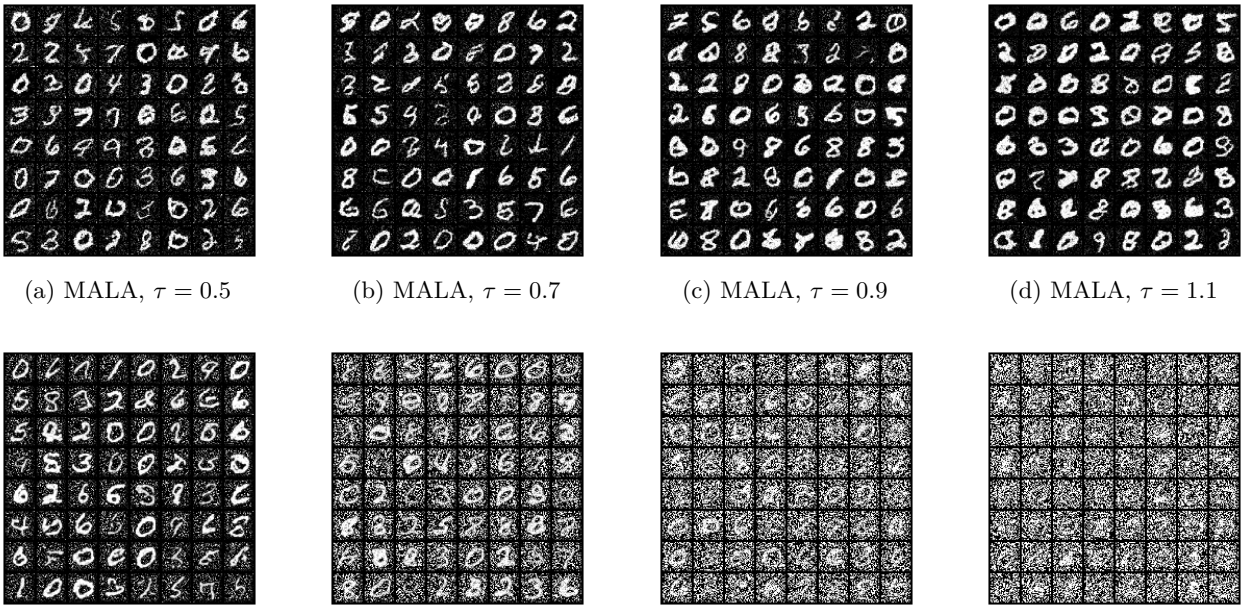
For each epoch $i = 1$ **to** N**:**

1. Sample a batch b_x and a batch $b_{\tilde{x}}$ from D.
2. For each $\tilde{x} \in b_{\tilde{x}}$, sample $v' \sim q(\cdot \mid \tilde{x})$ and set $x' = \alpha_i v' + (1 - \alpha_i)\tilde{x}$.
3. Compute the loss as defined in Algorithm 1.
4. Update the Acceptance Network a using the Adam optimizer.

Output: Trained Acceptance Network a.

Table 1: Quantitative comparison of methods across the datasets using the following metrics: Wasserstein-1 (W1), Wasserstein-2 (W2), and Maximum Mean Discrepancy (MMD).

Dataset	**ULA**			**Score RW**			**Score MALA**			**Score pCN**		
Metric	**W1**	**W2**	**MMD**	**W1**	**W2**	**MMD**	**W1**	**W2**	**MMD**	**W1**	**W2**	**MMD**
Moons	0.143	0.096	0.054	0.019	0.007	0.004	**0.018**	**0.004**	**0.002**	0.039	0.021	0.012
Pinwheel	0.106	0.062	0.022	**0.012**	**0.001**	**0.001**	0.086	0.064	0.027	0.016	0.007	0.003
S-curve	0.082	0.055	0.016	0.050	0.015	**0.004**	**0.046**	**0.014**	**0.004**	0.090	0.051	0.017
Swiss Roll	2.881	14.409	0.811	2.270	8.470	0.400	**1.399**	**4.575**	**0.245**	14.167	113.824	6.903



(a) MALA, $\tau = 0.5$ (b) MALA, $\tau = 0.7$ (c) MALA, $\tau = 0.9$ (d) MALA, $\tau = 1.1$

(e) ULA, $\tau = 0.5$ (f) ULA, $\tau = 0.7$ (g) ULA, $\tau = 0.9$ (h) ULA, $\tau = 1.1$

Figure 6: MNIST: comparing annealed MALA (top) and annealed ULA (bottom) across τ values 0.5, 0.7, 0.9, and 1.1. Results show that MALA is more robust to step size. Both algorithms were run for 100 denoising steps, with an additional 1000 steps at the smallest noise level. τ is a fixed parameter that controls the adaptive step size.

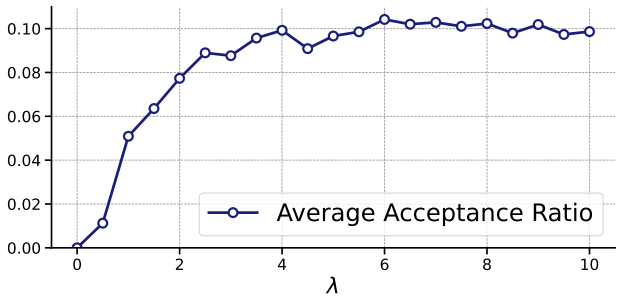


Figure 7: Average acceptance ratio as a function of λ. The average acceptance ratio is calculated by evaluating the acceptance network over a uniform grid of x' values, with ranges $[-1.5, 2.25]$ for the first dimension and $[-1, 1.5]$ for the second dimension, while keeping $x = (0, 0)$ fixed. The mean acceptance ratio is then obtained by averaging uniformly over the grid.

5 Empirical Results

In this section, we present empirical results on various datasets (Moons, Pinwheel, S-curve, Swiss Roll, and MNIST), to demonstrate the validity of our approach. Further results on extreme value distributions are in Appendix B.

Results Analysis. The results in Table 1 illustrate the performance of four sampling methods: ULA, Score-based RW, Score-based MALA, and Score-based pCN, across four datasets using three evaluation metrics: Wasserstein-1 (W1), Wasserstein-2 (W2), and Maximum Mean Discrepancy (MMD). Figures 3, 4, and 5 illustrate the results for low-dimensional data. Overall, score-based methods achieve lower W1 and W2 values compared to ULA across all datasets. This trend is especially noticeable on the Moons and Pinwheel datasets, where the W2 metric significantly outperforms that of the ULA method. For more complex datasets

like Swiss Roll, the score MH algorithms show noticeable improvements, with Score RW and Score MALA achieving significantly lower W1 and MMD scores when compared to the other methods.

Stability with respect to Step Size: Score MALA vs. ULA. The results demonstrate that Score MALA is more robust to variations in step size compared to ULA as seen in Figures 6 and 8.

Effect of Entropy Regularization. We study the effect of the entropy regularization term on the Moons dataset. As shown in Figure 7, the relationship between the entropy regularization parameter λ and the average acceptance ratio indicates a significant impact on the sampling efficiency. As λ increases from 0 to around 4, the average acceptance ratio rises sharply, suggesting that stronger entropy regularization enhances learning more practical acceptance functions. Beyond $\lambda = 4$, the acceptance ratio plateaus. This demonstrates that the performance remains robust for larger values of the entropy regularization term.

Diffusion Models Application. We evaluate the performance of the the proposed method with denoising score matching applied to the MNIST dataset using annealed ULA and annealing with score-based MALA. Annealing (Song & Ermon, 2019; Song et al., 2021) involves a sequence of noise scales, progressively refining the samples from high noise to low noise, thus improving the stability and efficiency of score-based sampling. Further details on the annealing process and its implementation are provided in the appendix. Our results presented in Figures 6 and 8 indicate that the proposed method exhibits greater stability with respect to step size variations compared to annealed ULA, as measured by the sensitivity parameter τ. We present further details in Appendix E.

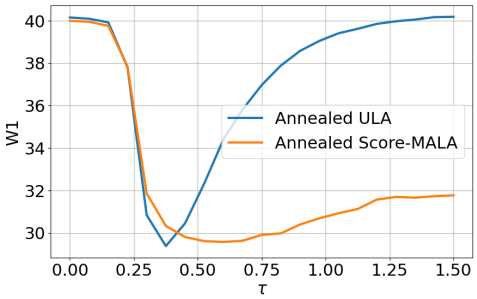


Figure 8: MNIST data generation comparing annealed ULA and annealed score-based MALA. τ is a fixed parameter that controls the adaptive step size.

6 Discussion

In this work we describe a method for estimating the acceptance function given only knowledge of samples and an estimated score function. We demonstrated the successful estimation of an acceptance function applicable across various MH sampling algorithms. While we empirically evaluate on three particular instances, this work opens the door for further implementations and techniques for sampling using acceptance functions. Additionally, we discussed a particular class of acceptance functions that are amenable for efficient sampling using entropy regularization. The proposed methods provide new avenues for combining MH within the context of score-based generative modeling.

Limitations. The main limitation of the method is the additional computational cost required to train the acceptance network. Furthermore, more advanced architectures need to be proposed to extend this framework to higher-resolution image data. Additionally, further theoretical analysis is needed to provide more appropriate priors on which acceptance function should be chosen.

References

Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and computing*, 18: 343–373, 2008.

Miguel Biron-Lattes, Nikola Surjanovic, Saifuddin Syed, Trevor Campbell, and Alexandre Bouchard-Côté. automala: Locally adaptive metropolis-adjusted langevin algorithm. In *International Conference on Artificial Intelligence and Statistics*, pp. 4600–4608. PMLR, 2024.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations*, 2023.

Yuzhu Chen, Fengxiang He, Shi Fu, Xinmei Tian, and Dacheng Tao. Adaptive time-stepping schedules for diffusion models. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

Laurence Davies, Robert Salomone, Matthew Sutton, and Chris Drovandi. Transport reversible jump proposals. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6839–6852. PMLR, 25–27 Apr 2023.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. *arXiv preprint arXiv:2112.07068*, 2021.

Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-hastings algorithms are fast. *Journal of Machine Learning Research*, 20(183):1–42, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, 7(2): 223–242, April 2001.

Martin Hairer, Andrew M. Stuart, and Sebastian J. Vollmer. Spectral gaps for a metropolis–hastings algorithm in infinite dimensions. *The Annals of Applied Probability*, 24(6):2455–2490, December 2014.

W. Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 1970.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Marcel Hirt, Michalis Titsias, and Petros Dellaportas. Entropy-based adaptive hamiltonian monte carlo. *Advances in Neural Information Processing Systems*, 34:28482–28495, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alexander K. Lew, George Matheos, Tan Zhi-Xuan, Matin Ghavamizadeh, Nishad Gothoskar, Stuart Russell, and Vikash K. Mansinghka. Smcp3: Sequential monte carlo with probabilistic program proposals. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pp. 7061–7088, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

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.

Mehryar Mohri. Foundations of machine learning, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Teodora Reu, Francisco Vargas, Anna Kerekes, and Michael M Bronstein. To smooth a cloud or to pin it down: Expressiveness guarantees and insights on score matching in denoising diffusion models. In *The 40th Conference on Uncertainty in Artificial Intelligence*.

Christian P Robert, George Casella, Christian P Robert, and George Casella. The metropolis—hastings algorithm. *Monte Carlo statistical methods*, pp. 267–320, 2004.

Gareth O Roberts and Jeffrey S Rosenthal. Examples of adaptive mcmc. *Journal of computational and graphical statistics*, 18(2):349–367, 2009.

Gareth O Roberts and Osnat Stramer. Langevin diffusions and metropolis-hastings algorithms. *Methodology and computing in applied probability*, 4:337–357, 2002.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, December 1996.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Jeffrey S Rosenthal et al. Optimal proposal distributions and adaptive mcmc. *Handbook of Markov Chain Monte Carlo*, 4(10.1201), 2011.

Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. *Advances in neural information processing systems*, 30, 2017.

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

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*, pp. 574–584. PMLR, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Michalis Titsias and Petros Dellaportas. Gradient-based adaptive markov chain monte carlo. *Advances in neural information processing systems*, 32, 2019.

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

Liwei Wang, Xinru Liu, Aaron Smith, and Aguemon Y Atchade. On cyclical mcmc sampling. In *International Conference on Artificial Intelligence and Statistics*, pp. 3817–3825. PMLR, 2024.

Keru Wu, Scott Schmidler, and Yuansi Chen. Minimax mixing time of the metropolis-adjusted langevin algorithm for log-concave sampling. *Journal of Machine Learning Research*, 23(270):1–63, 2022.

Kaihong Zhang, Heqi Yin, Feng Liang, and Jingbo Liu. Minimax optimality of score-based diffusion models: Beyond the density lower bound assumptions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60134–60178. PMLR, 21–27 Jul 2024.

Appendix

In this appendix, we first discuss an alternative approximation of the acceptance function that relies solely on the score functions, derived using the Taylor expansion of the log densities. Next, we consider sampling with score-based MALA for heavy-tailed distributions. Third, we provide the proofs for the theoretical results. Finally, we present additional details of the experiments and further empirical results.

A Taylor Score-based Metropolis-Hastings

In this section, we propose an approximation of the acceptance function based on the Taylor expansion of the log densities.

For an acceptance function $a(x'x)$ as defined in equation 5, the ratio $r(x', x) = \frac{p(x')q(x|x')}{p(x)q(x'|x)}$ can be rewritten as:

$$\log\left(r(x', x)\right) = \log p(x') - \log p(x) + \log q(x \mid x') - \log q(x' \mid x) \tag{12}$$

Let $\tau = x' - x$. Assuming that $\|\tau\|$ is sufficiently small, we perform a second-order Taylor series expansion of the logarithmic terms around the point x.

Taylor Expansion of $\log p(x')$ Around x: Expanding $\log p(x') = \log p(x + \tau)$ using a second-order Taylor series:

$$\log p(x + \tau) = \log p(x) + \nabla \log p(x) \cdot \tau + \frac{1}{2}\tau^\top \nabla^2 \log p(x)\, \tau + o(\|\tau\|^2) \tag{13}$$

Therefore,

$$\log p(x') - \log p(x) = \nabla \log p(x) \cdot \tau + \frac{1}{2}\tau^\top \nabla^2 \log p(x)\, \tau + o(\|\tau\|^2) \tag{14}$$

Taylor Expansion of $\log p(x)$ Around x': Similarly, expanding $\log p(x) = \log p(x' - \tau)$:

$$\log p(x' - \tau) = \log p(x') - \nabla \log p(x') \cdot \tau + \frac{1}{2}\tau^\top \nabla^2 \log p(x')\, \tau + o(\|\tau\|^2) \tag{15}$$

Rearranging gives:

$$\log p(x) - \log p(x') = -\nabla \log p(x') \cdot \tau + \frac{1}{2}\tau^\top \nabla^2 \log p(x')\, \tau + o(\|\tau\|^2) \tag{16}$$

Final Result Therefore, we have that,

$$\begin{aligned}
\log p(x') - \log p(x) &= \frac{1}{2}\left(\log p(x') - \log p(x)\right) + \frac{1}{2}\left(\log p(x') - \log p(x)\right) \\
&= \frac{1}{2}\left[\nabla \log p(x) \cdot \tau + \nabla \log p(x') \cdot \tau\right] \\
&\quad + \frac{1}{4}\tau^\top \left(\nabla^2 \log p(x) - \nabla^2 \log p(x')\right) \tau + o(\|\tau\|^2)
\end{aligned}$$

Simplifying, we obtain:

$$\log p(x') - \log p(x) = \frac{1}{2}\left(\nabla \log p(x) + \nabla \log p(x')\right) \cdot \tau + \frac{1}{4}\tau^\top \left(\nabla^2 \log p(x) - \nabla^2 \log p(x')\right) \tau + o(\|\tau\|^2) \tag{17}$$

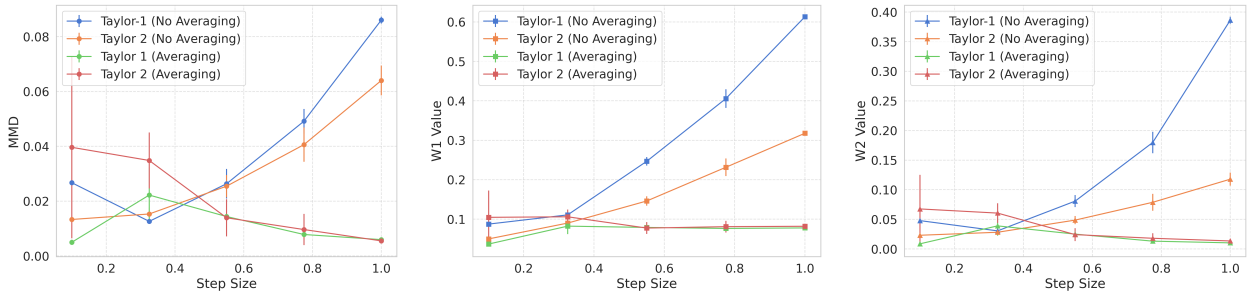
Combining the above results, we have:

$$\log r(x', x) = \frac{1}{2}\left(\nabla \log p(x) + \nabla \log p(x')\right) \cdot \tau + \frac{1}{4}\tau^\top \left(\nabla^2 \log p(x) - \nabla^2 \log p(x')\right) \tau + o(\|\tau\|^2) + \log q(x \mid x') - \log q(x' \mid x)$$

If the distance between the Hessian is negligible, we can approximate the acceptance function as

$$\log r(x', x) \approx \frac{1}{2}\left(\nabla \log p(x) + \nabla \log p(x')\right) \cdot \tau + \left[\log q(x \mid x') - \log q(x' \mid x)\right]$$

We denote this approximation of the acceptance function as Taylor-1 (Averaging). Figure 9 illustrates how this approximation can provide a better result compared to the first-order and second-order Taylor series from equation 14, while achieving very similar performance to the original equation 17, which utilizes the Hessian terms. This is particularly important because computing the Hessian numerically (by backpropagation through a score model) is computationally expensive and unstable.



(a) MMD Comparison (b) W1 Comparison (c) W2 Comparison

Figure 9: Comparison of Taylor approximations: Taylor-1 and Taylor-2 methods (with and without averaging) evaluated using MMD, W1, and W2 metrics. The experiments were conducted on the Moons dataset.

Additionally, Figure 10 compare the performance of Taylor-1 (Averaging) to score-based Metropolis-Hastings. We can see that for some range the Taylor approximation provides a very good approximation of the Acceptance function as it yields good sampling quality. However, even though some robustness is observed score-based MALA still outperforms the Taylor approximation.

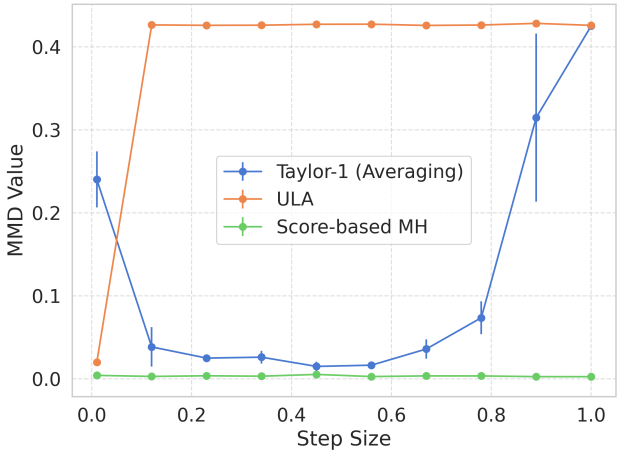


Figure 10: MMD distance comparison between ULA, Taylor-1 (Averaging), and Score-based MH.

B Metropolis-Hastings and Generalized Extreme Value Distributions

In this section, we study the effect of sampling using ULA, MALA, and score-based MALA algorithms. We analyze the impact of the adjustment step on distributions with heavy tails and hypothesize that as the tail becomes heavier, the adjustment step becomes increasingly more important. This observation suggests that, to accurately model and generate extreme events using score-based models, incorporating the adjustment step is essential.

Generalized Extreme Value (GEV) Distribution The Generalized Extreme Value (GEV) distribution is commonly used to model the maxima of datasets and is defined by its probability density function (PDF):

$$f(x;\xi,\mu,\sigma) = \begin{cases} \frac{1}{\sigma}\left(1+\xi\frac{x-\mu}{\sigma}\right)^{-1-\frac{1}{\xi}} \exp\left(-\left(1+\xi\frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}\right), & \text{if } \xi \neq 0, \\ \frac{1}{\sigma}\exp\left(-\frac{x-\mu}{\sigma}\right)\exp\left(-\exp\left(-\frac{x-\mu}{\sigma}\right)\right), & \text{if } \xi = 0, \end{cases}$$

where: - μ is the location parameter, - $\sigma > 0$ is the scale parameter, - ξ is the shape parameter (controls the tail heaviness).

The GEV distribution unifies three types of distributions: Gumbel ($\xi = 0$), Fréchet ($\xi > 0$), and Weibull ($\xi < 0$), making it highly flexible for extreme value analysis.

For the Fréchet distribution, moments of order k exist only if $k < \frac{1}{\xi}$. If $k \geq \frac{1}{\xi}$, the moment diverges, leading to infinite values. This property makes the Fréchet distribution particularly suited for modeling heavy-tailed distributions, as it captures the behavior of distributions with heavy tails and undefined higher-order moments.

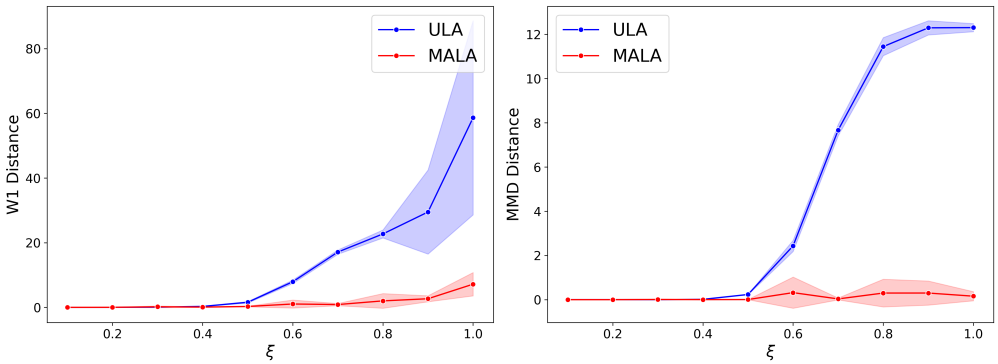
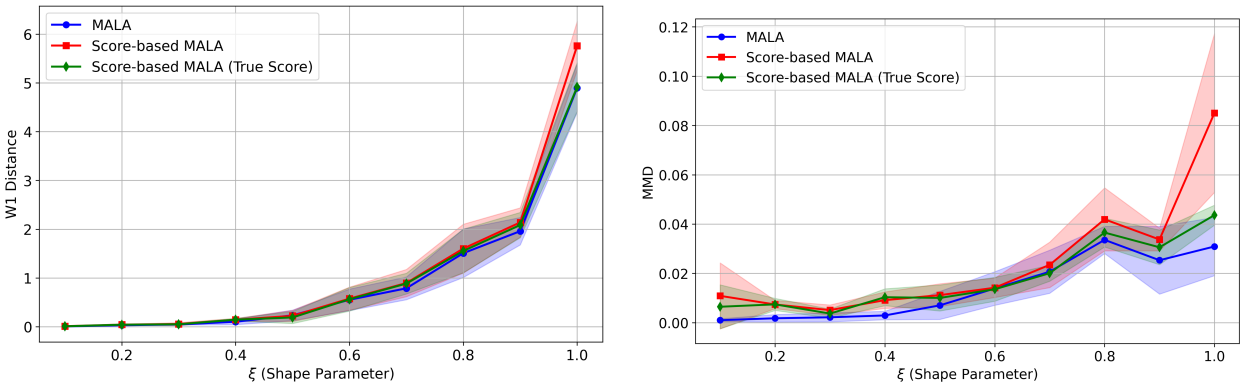


Figure 11: Comparison between ULA and MALA sampling methods for GEV$(0, 1, \xi)$, highlighting the effect of the adjustment step on heavy-tailed distributions. We fix the step size of both ULA and MALA to 0.1.

In Figure12, we illustrate that score-based MALA achieves competitive performance compared to the ground truth MALA. While MALA uses the standard acceptance function defined in equation 5, score-based MALA learns a potentially different acceptance function.



(a) W1 Distance comparison.

(b) MMD comparison.

Figure 12: Comparison between score-based MALA (the score network is trained from data), score-based MALA (True Score) where the acceptance network is trained with the true score function, and the original MALA algorithm. Left: W1 distance. Right: MMD.

C Proofs

Proof of Proposition 1. We aim to prove that

$$\mathcal{L}(a) = 0 \iff a \in \mathcal{A}.$$

(Reverse Direction:) Suppose $a \in \mathcal{A}$, meaning that for every $x, x' \in \mathcal{X}$, $a(x', x)$ satisfies

$$\frac{a(x', x)}{a(x, x')} = \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}.$$

Taking the logarithm on both sides,

$$\log a(x', x) - \log a(x, x') = \log p(x') - \log p(x) + \log q(x \mid x') - \log q(x' \mid x).$$

Differentiating both sides with respect to x and x', we obtain

$$\nabla \log a(x', x) - \nabla \log a(x, x') = \nabla \log p(x') - \nabla \log p(x) + \nabla \log q(x \mid x') - \nabla \log q(x' \mid x).$$

Therefore we have that $\mathcal{L}(a) = 0$.

(Forward Direction:)

Suppose that $\mathcal{L}(a) = 0$, i.e., since the integrand is non-negative, we have almost surely for every $x, x' \in \mathcal{X}$

$$\nabla \log a(x', x) - \nabla \log a(x, x') = \nabla \log p(x') - \nabla \log p(x) + \nabla \log q(x \mid x') - \nabla \log q(x' \mid x),$$

then integrating this equality gives:

$$\log a(x', x) - \log a(x, x') = \log p(x') - \log p(x) + \log q(x \mid x') - \log q(x' \mid x) + C,$$

where $C \in \mathbb{R}$ is a constant. Therefore, the acceptance function $a(x, x')$ can be expressed as:

$$\frac{a(x', x)}{a(x, x')} = e^C \cdot \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}.$$

As e^C does not depend on x, setting $x = x'$, we get that

$$1 = e^C$$

Therefore we have that $C = 0$. Hence, $a \in \mathcal{A}$, which concludes the proof. □

Proof of Proposition 2. Let $\hat{\mathcal{L}}(a)$ be the empirical SBM loss computed over N i.i.d. training samples:

$$\hat{\mathcal{L}}(a) = \frac{1}{N} \sum_{i=1}^{N} \left\| \nabla \log a(x_i', x_i) - \nabla \log a(x_i, x_i') - \nabla \log p(x_i') + \nabla \log p(x_i) - \nabla \log q(x_i \mid x_i') + \nabla \log q(x_i' \mid x_i) \right\|^2.$$

All we need to prove is that the loss function is bounded in order to apply the standard uniform convergence bound from statistical learning theory (Mohri, 2018).

Proving the boundedness of the loss function. By Assumption 1, $\mathcal{X}$ is compact, which implies that $\mathcal{X} \times \mathcal{X}$ is also compact. Furthermore, by Assumptions 2 and 3, the gradient terms $\nabla \log a$, $\nabla \log p$, and $\nabla \log q$ are continuous over a compact domain. By the Extreme Value Theorem, they attain their maximum absolute values, denoted by constants C_1, C_2, and C_3, respectively:

$$\sup_{x,x' \in \mathcal{X}} \|\nabla \log a(x', x)\| \leq C_1, \quad \sup_{x,x' \in \mathcal{X}} \|\nabla \log p(x)\| \leq C_2, \quad \sup_{x,x' \in \mathcal{X}} \|\nabla \log q(x' \mid x)\| \leq C_3.$$

Thus, the SBM loss is bounded as follows:

$$\mathcal{L}(a) \leq \mathbb{E}_{x\sim p, x'\sim q(\cdot|x)} \left[(2C_1 + 2C_2 + 2C_3)^2\right] = C_4,$$

where $C_4 = (2C_1 + 2C_2 + 2C_3)^2$ is a finite constant.

Applying standard uniform convergence bounds (Mohri, 2018), for all $a \in \mathcal{F}$, with probability at least $1 - \delta$, we obtain

$$\sup_{a\in\mathcal{F}} \left|\mathcal{L}(a) - \hat{\mathcal{L}}(a)\right| \leq 2\mathcal{R}_N(\mathcal{F}) + \mathcal{O}\left(\sqrt{\frac{\log(1/\delta)}{N}}\right),$$

where $\mathcal{R}_N(\mathcal{F})$ is the Rademacher complexity of the function class $\mathcal{F}$. □

Proof of Proposition 3. Suppose $\mathcal{L}(a) = 0$. By Proposition 1, this is equivalent to $a \in \mathcal{A}$, i.e., a satisfies the detailed balance condition:

$$p(x)\, q(x' \mid x)\, a(x', x) = p(x')\, q(x \mid x')\, a(x, x').$$

Let $a_M = \frac{a}{M}$. Dividing both sides by $M \geq 1$ yields:

$$p(x)\, q(x' \mid x)\, a_M(x', x) = p(x')\, q(x \mid x')\, a_M(x, x'),$$

which is precisely the detailed balance condition for a_M. Hence $a_M \in \mathcal{A}$, and therefore $\mathcal{L}\left(\frac{a}{M}\right) = 0$. □

D Experiments Details

In this section, we provide descriptions of the datasets used to generate the empirical results from the main text, and outline the neural network architectures and hyperparameter choices. All presented empirical results were compiled using an NVIDIA RTX 3090 GPU.

D.1 Dataset Descriptions

We use four datasets generated using `scikit-learn` (Pedregosa et al., 2011) and custom code. The parameters provided ensure reproducibility of the results:

- **Moons**: This dataset consists of two interlocking crescent-shaped clusters. We generate 10000 samples with a noise level of 0.1 using `sklearn.datasets.make_moons(n_samples=10000, noise=0.1)`.
- **Pinwheel**: Data points are arranged in six spiral arms. We generate 10000 samples with a radial standard deviation of 0.5, tangential standard deviation of 0.05, and a rate of 0.25 (which controls the spread of the arms). The dataset is generated using custom code and the following parameters:
  - `num_classes=5`,
  - `radial_std=0.5`,
  - `tangential_std=0.05`,
  - `rate=0.25`.
- **S-curve**: This dataset consists of 10000 points distributed along an "S"-shaped 3D manifold with added Gaussian noise of 0.1. It is generated using `sklearn.datasets.make_s_curve(n_samples=10000, noise=0.1)`.
- **Swiss Roll**: This dataset contains 10000 points arranged along a 3D spiral-shaped surface, with Gaussian noise of 0.5 added. It is generated using `sklearn.datasets.make_swiss_roll(n_samples=10000, noise=0.5)`.
- **MNIST**: MNIST consists of grayscale images of handwritten digits (0-9), each of size 28×28. The dataset contains $60,000$ training samples and $10,000$ test samples. For our experiments, we normalize pixel values to $[-1, 1]$ and convert the images into PyTorch tensors.

All datasets are converted to PyTorch tensors for further processing. The experiments were performed using an NVIDIA RTX 3090 GPU.

Algorithm 3 Pinwheel Dataset Generation

1: **Input:** Number of samples n, number of classes K, radial standard deviation σ_r, tangential standard deviation σ_t, rotation rate α
2: **Output:** Dataset $\mathbf{X} \in \mathbb{R}^{n \times 2}$
3: Generate random class labels $\mathbf{y} \in \{0, 1, \ldots, K-1\}$ for n samples
4: Compute angles $\theta_k = \frac{2\pi k}{K}$ for each class $k \in \{0, 1, \ldots, K-1\}$
5: Generate radial components $r_i \sim \mathcal{N}(1, \sigma_r^2)$ for each sample i
6: Compute tangential noise $\delta_i \sim \mathcal{N}(0, \sigma_t^2)$ for each sample i
7: Compute angle for each point: $\phi_i = \theta_{y_i} + \alpha \cdot r_i$
8: Compute Cartesian coordinates:

$$x_i = r_i \cdot \cos(\phi_i) + \delta_i$$

$$y_i = r_i \cdot \sin(\phi_i) + \delta_i$$

9: Return the dataset $\mathbf{X} = \{(x_i, y_i)\}_{i=1}^n$

D.2 Neural Network Architectures

We now provide detailed descriptions of several different neural network architectures designed for learning the score function (Score Nets), and the acceptance networks (Acceptance Nets).

Score Nets. The Score Nets architecture is a simple feed-forward neural network consisting of an input layer, two hidden layers, and an output layer. The input dimension is the same as the output dimension, ensuring that the output matches the shape of the input data. The `Softplus` activation function is applied after each hidden layer to introduce non-linear transformations. The output layer does not use an activation function, as it directly produces the estimated score of the input.

Acceptance Nets. The Acceptance Nets architecture is a feed-forward neural network designed to compute the acceptance function for a pair of inputs. The network takes two input vectors, concatenates them, and passes the result through an initial fully connected layer with a customized hidden dimension. Following this, the network consists of three residual blocks, each containing fully connected layers with the same hidden dimension, enhanced by `GELU` activation functions to improve non-linearity and gradient flow. Finally, the output is passed through a fully connected layer, followed by `GELU` and a `Sigmoid` activation to ensure the output is between 0 and 1, representing the acceptance probability.

MNIST Architectures. For the MNIST Score Net, we employ a time-dependent UNet architecture with an encoder and decoder blocks and we use Softplus activations functions. For further details on the Score Network, please refer to the MNIST tutorial available at the following GitHub repository, as we use the same architecture `https://github.com/yang-song/score_sde`. For its acceptance network we use a Siamese-like and that takes two inputs (x, x'), embeds them via convolutional and residual layers, and processes them alongside learned time and step-size embeddings. We use GeLU activation functions. These branches do not share weights. The representation are then fed to a common branch. The acceptance network also encodes the time and step size parameters.

Annealed ULA and MALA for MNIST. We evaluate annealed ULA and annealed MALA for sampling high-quality image reconstructions. We employ a denoising score matching approach as presented in (Song & Ermon, 2019; Song et al., 2021).

D.3 Hyperparameters

Score Nets. We summarize the hyperparameters used to train the Score Nets in Table 2.

Acceptance Nets. We provide below the details for training the Acceptance network for the various score-based Metropolis-Hastings algorithms.

Table 2: Score-Net Training Hyperparameters

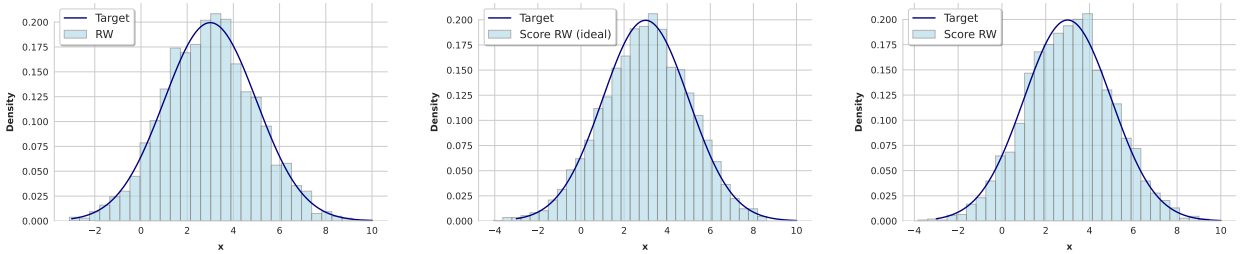
Dataset	**Optimizer**	**Learning Rate (LR)**	**Epochs**	**Hidden Dimension**
Moons	Adam	1×10^{-3}	5000	64
Pinwheel	Adam	5×10^{-4}	2000	512
S-curve	Adam	5×10^{-4}	2000	512
Swiss Roll	Adam	5×10^{-4}	2000	512

Table 3: Acceptance-Net Training Hyperparameters

Dataset	**Optimizer**	**LR**	**Hidden Dimension**	**Residual Layers**	**Epochs**	λ
Moons	Adam	5×10^{-4}	256	3	1000	2
Pinwheel	Adam	5×10^{-4}	256	4	200	2
S-curve	Adam	5×10^{-4}	512	4	200	2
Swiss Roll	Adam	5×10^{-4}	512	4	200	1

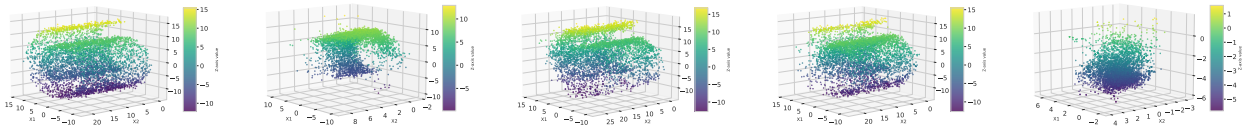
E Additional Empirical Results

In this section, we include additional empirical results for three datasets. First, we perform a sanity check using a synthetic dataset, demonstrating that training the acceptance network with a learned score achieves similar performance to training with the true score, as shown in Figure 13. Furthermore, we present results for generating the Swiss Roll dataset in Figure 14. Finally, we provide additional results for the MNIST dataset, illustrating the robustness of Annealed MALA to the step size in Figure 15.



(a) Standard Random Walk Sampling. (b) Score-based RW with true scores. (c) Score-based RW with learned score.

Figure 13: Comparison of sampling methods from a Gaussian distribution $\mathcal{N}(3, 2)$ using three approaches: (a) Standard Random Walk Sampling with the acceptance function defined in equation 5, (b) Score-based RW with an acceptance network trained using true scores, and (c) Score-based RW with an acceptance network trained using a learned score.



(a) Original samples. (b) ULA. (c) Score RW. (d) Score MALA. (e) Score pCN.

Figure 14: Comparison of different methods on the Swiss Roll dataset.

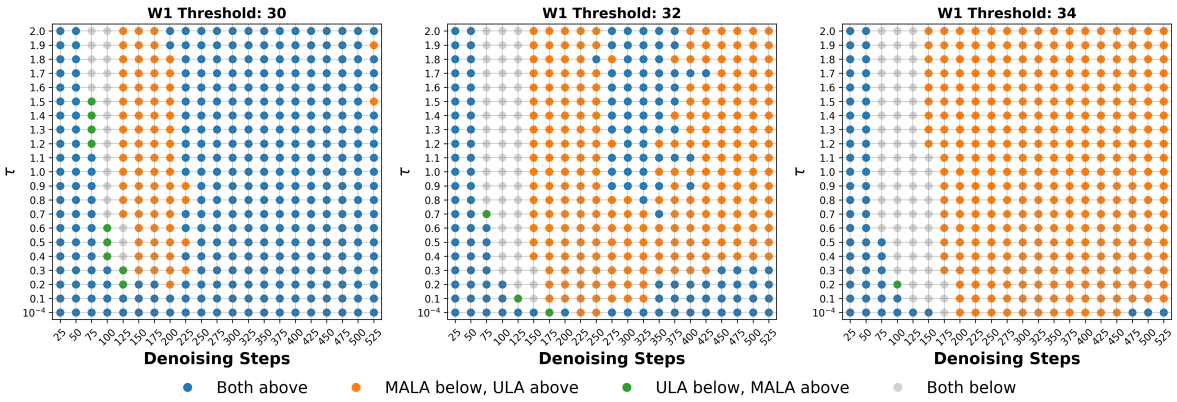


Figure 15: Comparison of Wasserstein-1 (W1) values for Annealed MALA and Annealed ULA across different τ and step size settings. The plots classify regions based on whether the W1 values for both methods exceed or fall below a given threshold. Each subplot corresponds to a different threshold value $(30, 32, 34)$. Colors indicate classification: (i) both methods above the threshold (blue), (ii) MALA below and ULA above (orange), (iii) ULA below and MALA above (green), and (iv) both methods below the threshold (gray). The x-axis represents the number of denoising steps, while the y-axis corresponds to the adaptive step size parameter τ. We can see that annealed MALA exhibits a more robust behavior across different τ levels.