A Gradient Based Strategy for Hamiltonian Monte Carlo Hyperparameter Optimization

Andrew Campbell [*1] **Wenlong Chen** [*2] **Vincent Stimper** [*3 4] **José Miguel Hernández-Lobato** [3] **Yichuan Zhang** [5]

Abstract

Hamiltonian Monte Carlo (HMC) is one of the most successful sampling methods in machine learning. However, its performance is significantly affected by the choice of hyperparameter values. Existing approaches for optimizing the HMC hyperparameters either optimize a proxy for mixing speed or consider the HMC chain as an implicit variational distribution and optimize a tractable lower bound that can be very loose in practice. Instead, we propose to optimize an objective that quantifies directly the speed of convergence to the target distribution. Our objective can be easily optimized using stochastic gradient descent. We evaluate our proposed method and compare to baselines on a variety of problems including sampling from synthetic 2D distributions, reconstructing sparse signals, learning deep latent variable models and sampling molecular configurations from the Boltzmann distribution of a 22 atom molecule. We find that our method is competitive with or improves upon alternative baselines in all these experiments.

1. Introduction

Hamiltonian Monte Carlo (HMC) is a very popular Markov Chain Monte Carlo (MCMC) method for generating approximate samples from complex probability distributions. It finds wide use in machine learning (Neal, 2011) and across the broader statistical community (Carpenter et al., 2017). Unfortunately, HMC's performance depends heavily on the choice of hyperparameters such as the proposal step size. A step size that is too large can lead to unstable dynamics, while a step size that is too small may result in random walk behaviour and highly correlated samples. Furthermore, the step sizes may need to be tuned on a dimension by dimension basis depending on the scale and shape of the target distribution (Neal, 2011).

Traditionally, samples are produced by running a single HMC chain for a long time compensating for imperfect hyperparameter choices through ergodicity. However, it is becomingly increasingly attractive to run many short MCMC chains in parallel to make better use of parallel compute hardware (Hoffman & Ma, 2020). In this case, it is even more important to choose good hyperparameters that encourage fast mixing. This approach also offers the novel opportunity to choose different hyperparameter values for each step in the chain, providing more tuning flexibility.

Fully exploiting this opportunity in practice is a challenge. MCMC hyperparameters are commonly tuned according to the average acceptance probability but it is unclear how this can scale up to tuning every hyperparameter individually. Looking to backpropagation's success in training neural networks, a gradient based method would seem most appropriate. Unfortunately, there is no universal tractable metric to quantify the performance of HMC that we can optimize for. We must therefore make a choice about which approximate metric is best suited for this application.

One approach is to use a proxy for the mixing speed of the chain. Levy et al. (2018) make use of a variation of the expected squared jumped distance (Pasarica & Gelman, 2010) to encourage proposals to make large moves in space. Alternatively, one can draw upon ideas from Variational Inference (VI) (Jordan et al., 1999), which matches an approximate distribution q to the target distribution p by maximizing the Evidence Lower Bound or ELBO. This is equivalent to minimizing the KL-divergence between q and p.

Salimans et al. (2015); Wolf et al. (2016) use VI to obtain a training objective in which q is the joint distribution of all HMC samples along the chain. For tractability, they introduce an auxiliary inference distribution approximating the reverse dynamics of the chain. The looseness of their ELBO then depends on the KL-divergence between the auxiliary inference distribution and the true reverse dynamics. As the chain length increases so does the dimensionality of

[*]Equal contribution [1]Department of Statistics, University of Oxford [2]Baidu, Inc. [3]Department of Engineering, University of Cambridge [4]Max Planck Institute for Intelligent Systems [5]Boltzbit Ltd. Correspondence to: Andrew Campbell <campbell@stats.ox.ac.uk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

these distributions, resulting in a looser and looser bound. This is problematic as, for longer chains, the optimized hyperparameters are encouraged to fit the imperfect auxiliary distribution as opposed to the target. In practice, Salimans et al. avoid this problem by only considering very short HMC chains, which limits the flexibility of their method.

We overcome these issues by considering the *marginal* distribution of the final state in the chain as our variational q. In this case, the ELBO can be broken down into the sum of the tractable expectation with respect to q of the log target density (up to a normalization constant) and the intractable entropy of q. During optimization, the entropy term prevents a fully flexible q from collapsing to a point mass maximizing the log target density. However, a HMC chain, by construction, cannot collapse to such a point mass. We argue that optimization can still be successful whilst ignoring the entropy term, provided the initial distribution of the chain is broad enough. In practice, we achieve this by inflating the initial proposal distribution by a scaling that is independently tuned by minimizing a discrepancy measure between p and q (Gong et al., 2021).

We empirically compare our method with alternative baselines on a wide range of tasks. We first consider sampling from a collection of 2D toy distributions. We then focus on more challenging approximate inference problems: reconstructing sparse signals, training deep latent variable models on MNIST and FashionMNIST and finally, sampling molecular configurations from the Boltzmann distribution of the 22 atom molecule Alanine Dipeptide. Our results show that our method is competitive with or can improve upon alternative tuning methods for HMC on all these problems.

2. Background

2.1. Hamiltonian Monte Carlo

HMC (Neal, 1993) aims to draw samples from an n-dimensional target distribution $p(x) = \frac{1}{\mathcal{Z}} p^*(x)$ where $\mathcal{Z}$ is the (usually unknown) normalization constant. It introduces an auxiliary variable $\nu \in \mathbb{R}^n$, referred to as the momentum, which is distributed according to $\mathcal{N}\big(\nu; \mathbf{0}, \text{diag}(\boldsymbol{m})\big)$, with the resulting method sampling on the extended space (x, ν). HMC progresses by first sampling an initial state from some initial distribution and then iteratively proposing new states and accepting/rejecting them according to an acceptance probability. To propose a new state, first, a new value for the momentum is drawn from $\mathcal{N}\big(\nu; \mathbf{0}, \text{diag}(\boldsymbol{m})\big)$, then, we simulate Hamiltonian Dynamics with Hamiltonian, $H(x, \nu) = -\log p^*(x) + \frac{1}{2}\nu^T \text{diag}(\boldsymbol{m})^{-1}\nu$ arriving at new state (x', ν'). This new state is accepted with probability $\min\big[1, \exp(-H(x', \nu') + H(x, \nu))\big]$. Otherwise we reject the proposed state and remain at the starting state. The Hamiltonian Dynamics are simulated using a numerical integrator, with leapfrog (Hairer et al., 2003) being a popular choice. L leapfrog updates are taken to propose a new state, with the update equations at step k being

$$\begin{aligned}
\nu_{k+\frac{1}{2}} &= \nu_k + \frac{1}{2}\boldsymbol{\epsilon} \circ \nabla_{x_k} \log p^*(x_k)\,, \\
x_{k+1} &= x_k + \nu_{k+\frac{1}{2}} \circ \boldsymbol{\epsilon} \circ \frac{1}{\boldsymbol{m}}\,, \\
\nu_{k+1} &= \nu_{k+\frac{1}{2}} + \frac{1}{2}\boldsymbol{\epsilon} \circ \nabla_{x_{k+1}} \log p^*(x_{k+1})\,,
\end{aligned}$$

where $\frac{1}{\boldsymbol{m}} = (\frac{1}{m_1}, \ldots, \frac{1}{m_n})$ and $\circ$ denotes element wise multiplication. The step size, $\boldsymbol{\epsilon}$, and the mass, $\boldsymbol{m}$, are hyperparameters that need to be tuned for each problem the method is applied to. We note that in the usual definition of HMC, a single scalar valued ϵ is used. Our use of a vector $\boldsymbol{\epsilon}$ implies a different step size in each dimension which, with proper tuning, can improve performance by accounting for different scales across dimensions. The use of $\boldsymbol{\epsilon}$ does mean the procedure can no longer be interpreted as simulating Hamiltonian Dynamics, however, it can still be used as a valid HMC proposal (Neal, 2011). Further, $\boldsymbol{\epsilon}$ and $\boldsymbol{m}$ both correspond to an element-wise rescaling of x and so tuning both does not increase the expressivity of the method. However, we found empirically this overparameterization aided optimization. We do not consider the problem of choosing L in this work.

2.2. Variational Inference

VI approximates the target $p(x)$ with a tractable distribution $q_\phi(x)$ parameterized by ϕ. The value of ϕ is chosen as to minimise the Kullback-Leibler divergence with the target, $D_{\text{KL}}\big(q_\phi(x)||p(x)\big)$. As, typically, we know $p(x)$ only up to a normalization constant, we can equivalently choose ϕ by maximising the tractable ELBO:

$$\log \mathcal{Z} - D_{\text{KL}}\big(q_\phi(x)||p(x)\big) = \mathbb{E}_{q_\phi(x)}\big[\log p^*(x) - \log q_\phi(x)\big]\,.$$

3. Expected Log-Target Maximization

VI tunes the parameters of an approximate distribution to make it closer to the target. We build on this to obtain a tractable objective for HMC hyperparameter optimization. In the parallel HMC setting, we run multiple parallel HMC chains and take the final sample in each chain. Viewing this from the VI perspective, these final samples would be independent samples from an implicit variational distribution. If each chain starts at a sample from an initial distribution $q^{(0)}(x)$ and then runs T accept/reject cycles, we can denote the resulting implicit distribution as $q_\phi^{(T)}(x)$, where ϕ now represents the step-by-step hyperparameters $\phi = \{\boldsymbol{\epsilon}^{(1:T)}, \boldsymbol{m}^{(1:T)}\}$. Ideally, ϕ would be chosen as to

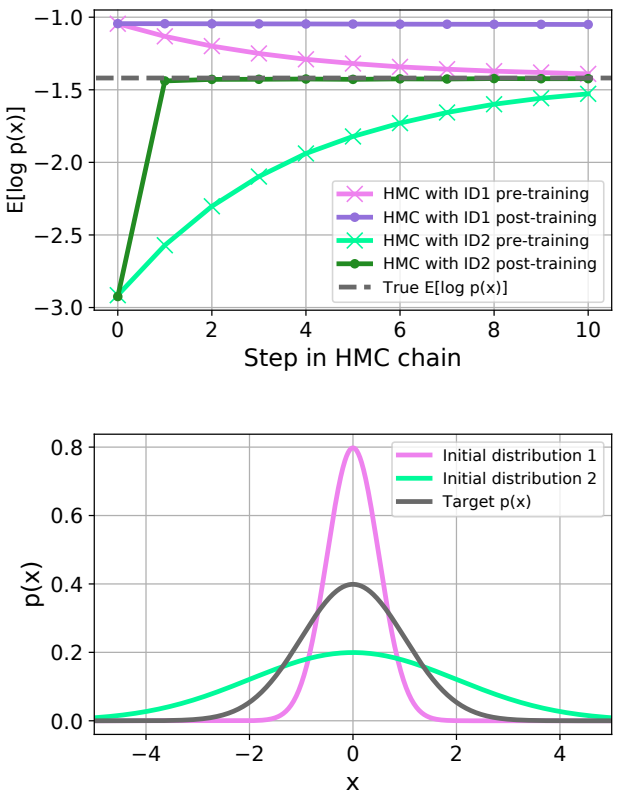


Figure 1. (Upper) Expected log target as a function of step t in the HMC chain for initial distribution 1, $\mathcal{N}(0, 0.25)$, and initial distribution 2, $\mathcal{N}(0, 4)$, before and after training. The 'true' value, $\mathbb{E}_p\left[\log p(x)\right]$ is also plotted. (Lower) The target pdf along with the two initial distributions.

maximise the ELBO:

$$\begin{aligned}\phi^* &= \underset{\phi}{\text{argmax}}\ \mathbb{E}_{q_\phi^{(T)}(x)}\left[\log p^*(x) - \log q_\phi^{(T)}(x)\right] \\ &= \underset{\phi}{\text{argmax}}\ \mathbb{E}_{q_\phi^{(T)}(x)}\left[\log p^*(x)\right] + H\left[q_\phi^{(T)}(x)\right] .\end{aligned}$$

Whilst the first term in this expression can be estimated directly via Monte Carlo, the entropy term, $H\left[q_\phi^{(T)}(x)\right]$, is intractable. To get around this, we should consider the purpose of the two terms during optimization. Maximizing the first term encourages $q_\phi^{(T)}(x)$ to produce samples that are in the high probability regions of the target, i.e., ensuring that $q_\phi^{(T)}(x)$ is high where $\log p^*(x)$ is high. The entropy term acts as a regularizer preventing $q_\phi^{(T)}(x)$ from simply collapsing to a point mass at the mode of $p(x)$. The key observation of our method is that HMC already fulfills this regularization role because the implicit distribution it defines is not fully flexible. If $q_\phi^{(T)}(x)$ were to collapse to a point mass, this would require the hyperparameters to be such that the HMC scheme always guides the chain state to the same point in space, irrespective of what initial position is sampled from $q^{(0)}(x)$. This is unreasonable for practical problems. Therefore, we propose optimizing ϕ simply by maximizing the expected log target density under the final state of the chain:

$$\phi^* = \underset{\phi}{\text{argmax}}\ \mathbb{E}_{q_\phi^{(T)}(x)}\left[\log p^*(x)\right] . \tag{1}$$

Although HMC does have a regularization effect, removing the entropy term does have some implications that we must consider. Namely, if the initial distribution, $q^{(0)}(x)$, is concentrated in a very high probability region of the target, $p(x)$, then optimizing the objective in (1) will not encourage HMC to explore the full target. Conversely, it would encourage the chains to remain in this region of high probability, close to their initial sampling point, which is undesirable behaviour. The key to avoiding this problem is to choose an initial distribution that has a sufficiently wide coverage of the target. We discuss methods for doing this in Section 3.3.

3.1. Demonstration on a Toy Problem

We now demonstrate the ideas of the previous section on a very simple toy problem. Here the target is a 1-dimensional normal distribution $\mathcal{N}(0, 1)$ which we attempt to sample from using a 10 step HMC chain with each step consisting of 5 leapfrog updates. We initialize the chain either with a narrow initial distribution $\mathcal{N}(0, 0.25)$ or a wide initial distribution $\mathcal{N}(0, 4)$, as shown in the bottom plot in Figure 1. We keep $\boldsymbol{m}$ constant for all steps but train one step size $\boldsymbol{\epsilon}_t$ for each HMC step. The top of Figure 1 plots the progression of $\mathbb{E}_{q_\phi^{(t)}}\left[\log p(x)\right]$ during sampling for these two cases, before and after hyperparameter training. Before training, when using the narrow initial distribution, $\mathbb{E}_{q_\phi^{(t)}}\left[\log p(x)\right]$ initially starts above the true value but converges from above as the marginal HMC distribution, $q_\phi^{(t)}$, spreads to cover the target. However, after training according to (1), all the step sizes have become very small causing the HMC chains to remain at their initial sampled positions which is obviously detrimental for convergence. To avoid this, we can use a wide initial distribution. The top plot in Figure 1 shows that, in this case, tuning hyperparameters according to (1) greatly speeds up convergence.

3.2. Estimating the Gradients

We solve (1) using stochastic gradient descent (SGD), with gradients computed using a Monte Carlo approximation and the reparameterization trick (Rezende et al., 2014; Kingma & Welling, 2014). Here, we are forced to make another approximation since the discontinuous accept/reject step introduces a non-differentiability in the Monte Carlo estimate of the objective. We can either ignore this and use biased gradients or remove the accept/reject step altogether as other works have suggested (Salimans et al., 2015; Caterini et al., 2018). We opt for the former since we wish to work with HMC in its original form with the accept/reject step.

Our empirical results confirm that we are able to successfully optimize the hyperparameters whilst using biased gradients. The intuition behind this is that the bias originates from the effect of the hyperparameters on the acceptance probability through the leapfrog discretization error. In practice, this discretization error and its gradient with respect to the hyperparameters is small due to the leapfrog integrator having second order accuracy (Leimkuhler & Reich, 2004). This results in a small overall bias. More concrete arguments justifying this can be found in the supplement. Finally, to improve computational efficiency, we avoid the calculation of second-order gradients by stopping backpropagation of the gradient through x_k in $\nabla_{x_k} \log\, p^*(x_k)$ during the leapfrog updates. We find this has little impact on convergence and can lead to $5\times$ speedups in execution time.

3.3. Tuning the Initial Distribution

For our method to be useful, the initial distribution $q^{(0)}(x)$ must provide a sufficient coverage of $p(x)$. However, a $q^{(0)}(x)$ that is overly spread out is undesirable because the burn-in time is then unnecessarily long. We describe below a method for choosing $q^{(0)}(x)$ that aims to achieve an optimal trade-off between these two requirements.

The main idea is to use a variational approximation to the target as the initial distribution, as done by Hoffman (2017). This should be a distribution that can be easily sampled from and easily tuned to fit the target, e.g. a Gaussian or a normalizing flow (Tabak & Vanden-Eijnden, 2010; Rezende & Mohamed, 2015). Rather than use the standard ELBO for tuning $q^{(0)}(x)$, we use α-divergence minimization (Hernández-Lobato et al., 2016). The α value dictates the mass covering behaviour of the resulting approximation, with $\alpha = 0$ corresponding to the standard mode seeking $D_{\text{KL}}\big(q^{(0)}(x)||p(x)\big)$ minimization and $\alpha = 1$ corresponding to the mass covering $D_{\text{KL}}\big(p(x)||q^{(0)}(x)\big)$ minimization. We compare both α values in our experiments. The α-divergence is very useful in this context as it can provide a mass covering approximation without the use of samples from the target. However, if samples from $p(x)$ are available, then $q^{(0)}(x)$ can alternatively be tuned on those samples via maximum likelihood (ML), which is also mass-covering.

The previous approaches will produce a $q^{(0)}(x)$ that fits the target, but do not guarantee that it will be broad enough. To address this, we allow our method to automatically adjust the width of the initial distribution as necessary to keep $q_\phi^{(T)}(x)$ as closely matched to $p(x)$ as possible. This is achieved by applying a scalar scale factor s centered around the mean μ of $q^{(0)}(x)$ to each sample x_i from this distribution, i.e. $\hat{x}_i = s(x_i - \mu) + \mu$. We tune s by minimizing the Sliced Kernelized Stein Discrepancy (Gong et al., 2021) or SKSD[1] between the final state distribution $q_\phi^{(T)}(x)$ and $p(x)$ The SKSD requires only samples from $q_\phi^{(T)}$ and gradients of the target, $\nabla_x \log p^*(x)$. This objective encourages suitable values of s because if s is too small then the tuning of ϕ by solving (1) (which occurs jointly with the tuning of s) will result in a degenerate $q_\phi^{(T)}(x)$ far from the target. The SKSD measures this discrepancy and provides a learning signal for increasing s. Conversely, if s is too large then $q_\phi^{(T)}(x)$ will also be far from the target since the HMC chain will not be able to compensate for the poor initialization. The SKSD will then favor to decrease s. Finally, as in the leapfrog updates, we stop the gradient computations through x in $\nabla_x \log\, p^*(x)$ when evaluating SKSD to avoid the calculation of second-order gradients.

Given that the SKSD is a tractable objective that measures the discrepancy between $q_\phi^{(T)}(x)$ and $p(x)$, it is theoretically feasible to use the SKSD to optimize ϕ too. However, we found that the SKSD does not scale well when optimizing many parameters, which is why (1) is used instead. Note, however, that the SKSD works very well in practice when we only tune the single scalar parameter s.

We have empirically evaluated the method described in this section on a variety of applications and have found that it gives consistently good results.

3.4. Final optimization procedure

Algorithm 1 summarizes our optimization strategy, where Adam_update$(\eta, \nabla_\eta \mathcal{L}, i)$ returns the new value for η given by the i-th iteration of the Adam optimizer using gradient $\nabla_\eta \mathcal{L}$, $D_\alpha(q^{(0)}_{\psi_{i-1}}(x)\,||\,p(x))$ is an estimate of the α-divergence whose gradient is computed using doubly reparameterized gradient estimators (Tucker et al., 2019), SKSD$(x^{(T)}_{1:N}, \text{score}(x))$ estimates the sliced kernelized Stein discrepancy and HMC$_{\phi_{i-1}}(x_n^{(0)'}, \text{score}(x))$ runs an HMC chain with initial state $x_n^{(0)'}$, target score function score(x) and hyperparameters ϕ_{i-1}. Details on the computation of $D_\alpha(q^{(0)}_{\psi_{i-1}}(x)\,||\,p(x)), i)$ and SKSD$(x^{(T)}_{1:N}, \text{score}(x))$ are given in the Supplementary Material.

4. Experiments

Different experiments are performed to confirm that we can discover good hyperparameter settings by using the previously described method. We provide code for reproducing all our experiments on github [2].

[1] We use the metric called maxSKSD in Gong et al. (2021)

[2] https://github.com/VincentStimper/hmc-hyperparameter-tuning

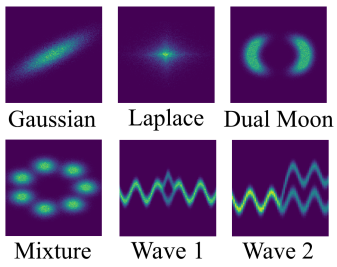


Figure 2. Histograms of 2D targets generated by rejection sampling.

	Gaussian	Laplace	Dual Moon	Mixture	Wave 1	Wave 2
maxELT $\alpha = 0$	0.0677	0.0005	0.2370	0.0004	0.0525	0.0462
maxELT $\alpha = 1$	0.0009	0.0004	0.8637	0.0010	0.0158	0.0801
maxELT $\alpha = 0$ SKSD	0.0008	0.0016	0.1684	0.0004	0.0020	0.0217
maxELT $\alpha = 1$ SKSD	0.0009	0.0014	0.2528	0.0004	0.0019	0.0317
Hoffman (2017)	0.0364	0.0005	1.1553	0.0846	0.9447	0.0465
Ruiz & Titsias (2019)	0.0003	0.0003	1.6290	0.0003	0.0024	0.2375
NUTS	0.0044	0.0016	0.2326	0.0023	0.0260	0.0965

Table 1. KSD between the HMC samples and the target distribution for the baselines and the 4 variations of our method on each of the synthetic target distributions (equations are given in the Supplementary Material).

Algorithm 1 Optimization Procedure

Input: Initial ψ_0, ϕ_0 and s_0, number of iterations I_1 and I_2 and number of samples N
Define score$(x) =$ stop_gradient$(\nabla_x \log p^*(x))$
 # Train initial distribution
for $i = 1$ **to** I_1 **do**
  $\psi_i \leftarrow$ Adam_update$(\psi_{i-1}, \nabla_{\psi_{i-1}} D_\alpha(q^{(0)}_{\psi_{i-1}}(x) \,||\, p(x)), i)$
end for
$\mu \leftarrow$ mean of $q^{(0)}_{\psi_{i-1}}(x)$
 # Train HMC hyperparameters
for $i = 1$, **to** I_2 **do**
  for $n = 1$, **to** N **do** *# This loop is vectorized in practice*
    Draw $x_n^{(0)} \sim q^{(0)}_{\psi}(x)$
    $x_n^{(0)'} \leftarrow s_{i-1}(x_n^{(0)} - \mu) + \mu$ *# Rescale samples*
    $x_n^{(T)} \leftarrow \text{HMC}_{\phi_{i-1}}(x_n^{(0)'}, \text{score}(x))$
  end for
  $\phi_i \leftarrow$ Adam_update$(\phi_{i-1}, \nabla_{\phi_{i-1}} \frac{1}{N} \sum_{n=1}^{N} \log p^*(x_n^{(T)}), i)$
  $s_i \leftarrow$ Adam_update$(s_{i-1}, \nabla_{s_{i-1}} \text{SKSD}(x_{1:N}^{(T)}, \text{score}(x)), i)$
end for
Return $\psi_{I_1}, \phi_{I_2}, s_{I_2}$

4.1. 2D distributions

We first focus on drawing approximate samples from a range of synthetic 2D target densities (Figure 2). We use 30-step HMC chains and a factorized Gaussian $q^{(0)}(x)$ that is trained by minimizing the α-divergence. We optimize step sizes and masses using (1), a procedure we refer to as 'maxELT' for maximizing the expected log target. Additionally, as an ablation study, we consider different initial distribution training strategies: $\alpha = 0$ or 1 and whether or not to tune the scaling s by minimizing the SKSD ($s = 1$ when not tuned). To quantify convergence to the target, we used the Kernelized Stein Discrepancy (KSD) (Liu et al., 2016; Chwialkowski et al., 2016) between the generated samples and the targets.

We include three baselines for reference. The first one is taken from Hoffman (2017) and initializes the HMC chains with an $\alpha = 0$ trained Gaussian distribution and tunes step sizes according to a minimum acceptance probability heuristic. The second baseline is the method from Ruiz & Titsias (2019) which initializes chains with a variational distribution, trained using a novel divergence metric. The HMC parameters are not tuned however. The final baseline is the popular No-U-Turn Sampler (Hoffman & Gelman, 2014). Full details regarding the baselines are given in the Supplementary Material.

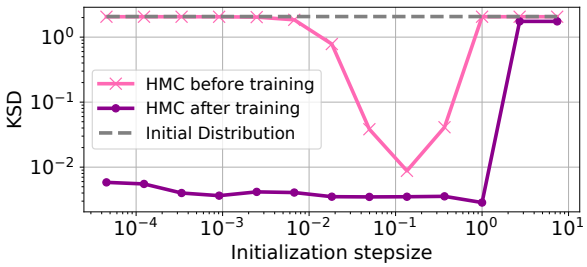


Figure 3. KSD versus step size used as initialization point for optimization. Wave 1 is the target distribution.

Results are shown in Table 1. Our method with automatic s scaling using the SKSD performs consistently well across distributions. For some simple targets, e.g. Gaussian and Laplace, it performs slightly worse than Ruiz & Titsias (2019), however, this method breaks down on the more complex targets, Dual Moon and Wave 2 whereas ours remains consistent. Furthermore, we fit the targets better when we tune s by minimizing the SKSD than when we do not. For some distributions, tuning s effectively helps prevent mode seeking behaviour. We confirm this quantitatively in the Supplementary Material by comparing expected log target values. We find that narrow initial distributions ($\alpha = 0$) often lead to excessively high log target values, but the tuning of s can prevent this pathology.

We also investigate the method's robustness to the initialization point. Figure 3 plots the KSD between the HMC samples and the target distribution (Wave 1) before and after training the hyperparameters for a large range of initialization step sizes. We find our method is largely invariant to

the initialization point provided it is not excessively large, in which case every step in the chain is rejected and there is no gradient signal for learning. However, we do no worse than the initial distribution as each sample remains at its initial sampling point.

4.2. Sparse Signal Recovery

We now consider higher dimensional target distributions. Specifically, posterior distributions in Bayesian compressed sensing (Donoho, 2006; Ji et al., 2008). The aim is to recover the sparse signal $w \in \mathbb{R}^d$ from measurements $y \in \mathbb{R}^n$ where $n < d$. The observation model is $y = Xw + e$ where $e \in \mathbb{R}^n$, $e \sim \mathcal{N}(0, \sigma_0^2 \boldsymbol{I})$, is additive Gaussian noise and $X \in \mathbb{R}^{n \times d}$ is a measurement matrix obtained by sampling its entries from a standard Gaussian distribution and then normalizing the rows to have unit Euclidean norm. We place a sparsity enforcing horseshoe prior on w and consider the posterior for w given y, X and σ_0^2. When $n << d$ this posterior is multi-modal representing multiple potential explanations for the available observations. In these experiments we fix $n = 6$ and $d = 64$.

We draw posterior samples using a 20 step HMC chain with 5 leapfrog iterations per step, initialized with a factorized Gaussian distribution that is trained with $\alpha = 1$. We do not use $\alpha = 0$ here since this method performs significantly worse in this problem due to its mode seeking tendency (Hernández-Lobato et al., 2015). We evaluate the quality of our samples by calculating their log marginal likelihood on test data generated using the same sparse signal as in the training set. The highest test log marginal likelihood is obtained when the generated samples come from the true posterior distribution.

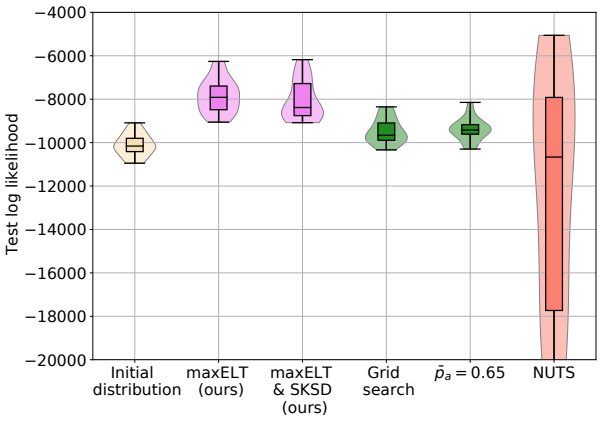


Figure 4. Violin-Box plot of test log marginal likelihoods on the sparse signal recovery problems. We draw 10,000 posterior samples to estimate the log marginal likelihood of 1000 test observations and repeat 20 times to obtain confidence bands.

We compare our method with three baselines that use the same initial distribution. The first two are parallel samplers where for each sample we first sample the initial distribution and then run 20 HMC accept/reject steps. The step sizes and masses are tuned either by a grid search to maximize the log marginal likelihood on a validation set or the step sizes are tuned such that the average acceptance probability is 0.65. The final baseline is sequential—the No-U-Turn sampler. To generate samples, we first sample the initial distribution once then run 15000 accept/reject steps, discarding the first 5000 samples. The hyperparameters are tuned using dual averaging during the burn-in period. Full experimental details are in the Supplementary Material.

The results are shown in Figure 4 and we see that our method generally outperforms the baselines. It is also noticeable that the sequential sampler (NUTS) has a very high variance between repeats. This is because it is very sensitive to the initialization point. NUTS performs well in the very few cases where the initialization is close to a posterior mode with good properties. However, if it is initialized close a sub-optimal mode, it will likely stay there for the whole run and find hyperparameters only suited to this local region giving a very poor log likelihood. The parallel samplers do not have this issue as they have a new initialization point for each sample. We also note that, in this case, using SKSD to tune s results in a slight degradation in performance. On this problem, $\alpha = 1$ is already well suited to find a good initial distribution and, indeed, the value for s tuned by SKSD was very close to 1.

4.3. Deep Latent Gaussian Models

We now focus on training Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014). These are generative models of the form $p(x, z) = p_\theta(x|z)\mathcal{N}(z; 0, \boldsymbol{I})$, explaining observed data x with a latent variable z and a likelihood $p_\theta(x|z)$ parameterized by a neural network $\text{NN}_\theta(z)$ with parameters θ and input z. The standard training in VAEs is to use a factorized Gaussian distribution $q_\psi(z|x)$ parameterized by $\text{NN}_\psi(x)$ to approximate $p_\theta(z|x)$ and train (θ, ψ) jointly by maximizing the ELBO.

Our approximation to $p_\theta(z|x)$ is the HMC chain output distribution $q_\phi^{(T)}(z|x)$. The initial state of the HMC chain is sampled from $q_\psi^{(0)}(z|x)$, which is given by a factorized Gaussian distribution parameterized by $\text{NN}_\psi(x)$ and appropriately scaled by s as before. The parameters ψ are trained by minimizing the α-divergence and s is trained using the SKSD. We train ϕ and θ by jointly maximizing the objective in (1) with respect to these two parameters. We consider HMC chains of length 30 with 5 leapfrog iterations per step and tune a different step size parameter per dimension and per step in the HMC chain while the mass parameters are all kept constant and equal to 1 all throughout the HMC chain.

Note that we use only one set of HMC hyperparameters for

Table 2. Average test marginal log-likelihood and its standard error (SE) estimated using HAIS for different methods for MNIST and Fashion MNIST. For methods with a scale factor s, we report the final scale after training. We also report the best log-likelihood values from previous works.

	MNIST			Fashion MNIST		
Model	Scale	Mean	SE	Scale	Mean	SE
VAE	-	-85.08	0.22	-	-108.54	0.60
DReG-IWAE	-	-83.73	0.21	-	-104.48	0.58
maxELT $\alpha = 0$	1.0	-83.48	0.21	1.0	-104.08	0.58
maxELT $\alpha = 1$	1.0	-82.46	0.21	1.0	-103.57	0.58
maxELT $\alpha = 0$ SKSD	6.79	-81.91	0.20	5.58	-103.18	0.58
maxELT $\alpha = 1$ SKSD	3.90	-81.94	0.20	3.59	**-102.29**	0.57
Hoffman	-	**-81.74**	0.20	-	-103.04	0.58
Ruiz & Titsias	-	-82.45	0.21	-	-105.13	0.59
Salimans et al.	-	-81.94	-	-	-104.44	0.59
Caterini et al.	-	-82.62	-	-	-104.26	0.58

all $p_\theta(z|x)$ targets, independently of x. One could make the hyperparameters depend on x through an amortization network, but we found that this did not improve performance. We consider two benchmark datasets: MNIST and Fashion MNIST. As is common, we use binarized images and a Bernoulli likelihood $p_\theta(x|z)$ parameterized with the same convolutional architecture as Salimans et al. (2015). Full experimental details are in the Supplementary Material.

We evaluate the quality of our trained models using the marginal log-likelihood $\log p_\theta(x)$ on the test set estimated using Hamiltonian Annealed Importance Sampling or HAIS (Sohl-Dickstein & Culpepper, 2012). Results are shown in Table 2. We also report log-likelihood values for multiple baselines. Using the same neural architecture, we implemented the standard VAE and IWAE[3] models. We also implemented another method for tuning ϕ (Hoffman, 2017) where the step sizes are adjusted to make the minimum average acceptance probability in each minibatch equal to 0.25. Furthermore, we implemented the method for tuning ψ from Ruiz & Titsias (2019) while the HMC step sizes were tuned to ensure the mean average acceptance probability in each minibatch is equal to 0.65. We update θ as in our method and use the same number of leapfrog steps for a fair comparison. Finally, we report results for MNIST from Salimans et al. (2015) and Caterini et al. (2018) which include HMC hyperparameter tuning during training and use the same network architecture as us. These authors only evaluated on MNIST, so we reimplemented their methods for Fashion MNIST[4]. We confirm significant differences between the models using paired t-tests. Full results are in the Supplementary Material.

On both datasets, the HMC based methods achieve better performance than VAE or IWAE, showing that reducing the approximation bias of the variational distribution with HMC greatly helps. Furthermore, we see that adding the scale factor to our method significantly improves performance as this avoids degenerate behaviour when training the HMC hyperparameters. We note that, without any scaling s, $\alpha = 1$ outperforms $\alpha = 0$ due to $\alpha = 0$ resulting in a too narrow initial distribution. With scaling, the SKSD automatically widens the initial distribution making the performance of both methods similar. Finally, we observe that HMC based methods top out at similar log-likelihood values (within around one standard error). We believe this is due to the methods reaching the limits of the chosen neural network architecture on these datasets, with no more gains to be made from more accurate posterior approximations.

4.4. Molecular Configurations

Finally, we evaluate our method on the complex real-world problem of sampling equilibrium molecular configurations from the Boltzmann distribution of the molecule Alanine Dipeptide. The unnormalized target distribution for the atom coordinates x is $\mathrm{e}^{-u(x)}$, where u denotes the potential energy of the system, which can be obtained using the laws of physics. This problem is usually tackled via Molecular Dynamics (MD) simulations. Here, we aim to produce independent samples from $\mathrm{e}^{-u(x)}$ using our trained short HMC chains. We do not operate directly on the Cartesian coordinates but apply the coordinate transform presented by Noé et al. (2019), to map some of the Cartesian coordinates to bond lengths, bond angles, and dihedral angles. The final dimensionality for x is 60. For more details, we refer to the Supplementary Material.

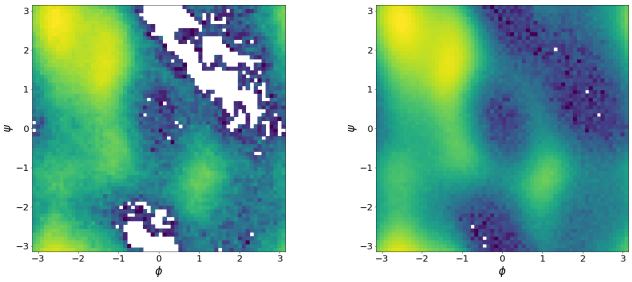
For the initial distribution, we use a normalizing flow based on real-valued non-volume preserving (RNVP) transformations (Dinh et al., 2017), followed by 50 HMC steps with 10 leapfrog iterations per step. We used different methods to train the flow and the HMC hyperparameters. The flow was trained with $\alpha = \{0, 1\}$-divergence and by ML. The latter was done using 10^5 training data samples obtained via a MD simulation[5]. As in the case with 2D densities and with sparse signals but unlike in the VAE case, we first trained the flows and then kept them fixed when tuning the HMC hyperparameters and the scale factor.

For each initial distribution type, the HMC hyperparameters were tuned according to maxELT alone or by maxELT with SKSD scale training. As a baseline, we optimized the HMC parameters via grid search, keeping step sizes and masses

[3]We used the DReG estimator from (Tucker et al., 2019) for the IWAE.

[4]We reimplemented Salimans et al. (2015) ourselves and used the implementation from https://github.com/anthonycaterini/hvae-nips for Caterini et al. (2018)

[5]The same dataset was used to obtain the mean and variances required in the normalization step of the coordinate transform.



(a) Ground truth (MD) (b) maxELT & SKSD

Figure 5. Ramachandran plot of (a) the ground truth determined via a MD simulation and (b) samples from the model with the proposal trained with ML and HMC hyperparameters being tuned by maxELT & SKSD.

constant across dimensions and HMC step, varying these two parameters in a grid and picking the combination that gave the lowest median marginal KL-divergence to the MD training data. We also considered another baseline by adjusting the step size value such that the average acceptance probability was 0.65 (referred to as $\overline{p}_{\text{a}} = 0.65$). Further implementation details are given in the Supplementary Material.

A new MD simulation was run to obtain 10^6 ground truth samples for evaluating the performance of the different methods. For tractability, this is done by comparing the marginal distributions of the generated and ground truth samples. First, we use Ramachandran plots (Ramachandran et al., 1963), which are 2D histograms for the two dihedral angles in the bonds connecting an amino acid to the protein backbone. These plots are frequently used to analyse how proteins fold locally. In the Alanine Dipeptide case we can obtain a Ramachandran plot for the bond connecting the two amino acids forming this molecule. Two sample plots are shown in Figure 5. We compute KL divergences between the 2D histograms (Ramachandran plots) for the ground truth samples and for the samples generated by the different methods. The results are given in Figure 6. All the corresponding Ramachandran plots are shown in the Supplementary Material. Our method outperforms the baselines for all proposals except for the one trained with the $\alpha = 0$-divergence, where it improves upon grid search but performs worse than the $\overline{p}_{\text{a}} = 0.65$ baseline.

Finally, for each dimension of x and using kernel density estimation, we compute the KL-divergences between the 1D densities produced by the ground truth samples and by the samples generated by the different methods. We perform a Wilcoxon test to check whether the resulting 60 divergence values (one per dimension) are consistently lower for one method or another. Table 3 shows p-values for the case in which maxELT & SKSD is being tested for having lower KL-divergences than the baselines. In all but

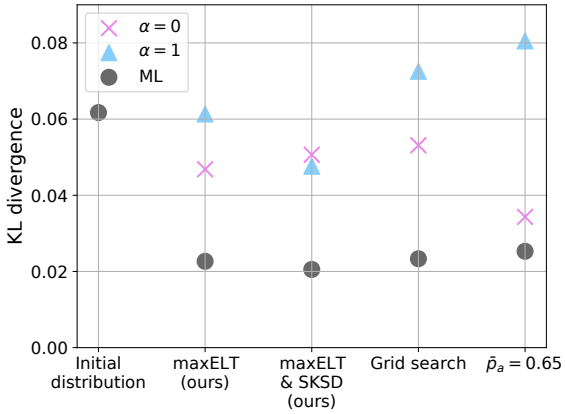


Figure 6. Visualization of the KL-divergences of the proposal and the models where different HMC hyperparameter tuning schemes were used.

Table 3. P-values of the Wilcoxon test with the alternative hypothesis that the model with HMC parameters tuned by maxELT & SKSD leads to lower KL-divergences of the marginals with respect to the ground truth than the respective baseline.

	Grid search	$\overline{p}_{\text{a}} = 0.65$
$\alpha = 0$	0.0030	0.39
$\alpha = 1$	0.0030	0.0040
ML	0.010	5.1e-5

one case maxELT & SKSD leads a significant improvement over the baselines and in the remaining case the two methods are on par with each other. All the other remaining p-values can be found in the Supplementary Material.

5. Discussion and Related Work

Our experiments show a general trend that, when we solely optimize the objective in (1), the value of α used significantly affects performance. However, when applying the SKSD scaling, this difference becomes smaller, showing that this technique is useful for automatically finding a suitable initial distribution.

A limitation of the method is that the hyperparameter optimization procedure must be run before samples can be produced. Therefore, our method is most applicable in cases where the optimization time is outweighed by the time savings gained through better hyperparameter choices and a faster mixing speed. This is the case for difficult problems where choosing reasonable hyperparameters is highly non-trivial and also for problems where a large number of samples need to be produced for a downstream task.

Previous works have also used VI to obtain an objective for the gradient based tuning of HMC hyperparameters. As discussed previously, Salimans et al. (2015); Wolf et al. (2016)

consider the entire joint distribution of the HMC chain as the variational distribution. The higher dimensionality increases the looseness of the variational bound, making the method highly sensitive to the length of the chain and the accuracy of a reverse dynamics approximation. Our experiments in section 4.3 show we can consider much longer chains and perform just as well without needing a reverse approximation. Caterini et al. (2018) also construct an alternative ELBO for HMC. However, they only sample the auxiliary momentum variables once at the start of the chain which reduces the empirical performance of HMC.

In contrast to these methods, some gradient based tuning techniques do not use ideas from VI but instead optimize a proxy for mixing speed. Levy et al. (2018) generalize the standard leapfrog integrator used in HMC with multi-layer perceptrons which are then trained by maximizing a modified version of the expected squared jumped distance. We improve upon this objective by directly optimizing convergence speed by using gradient information from the target distribution itself. It would be an interesting direction to use our objective to train this generalised leapfrog operator. Finally, Titsias & Dellaportas (2019) consider the gradient based tuning of the Markov transition operator, $p_\phi(x_t|x_{t-1})$, in the case of a single long MCMC chain. They optimize with respect to the expected acceptance probability for the next step in the chain, regularized with the entropy of $p_\phi(x_t|x_{t-1})$. Unfortunately, as this entropy is intractable when using the leapfrog algorithm as the transition operator, it cannot be applied to HMC in its current form.

There are also many non-gradient based heuristics for tuning HMC hyperparameters. The popular No-U-Turn Sampler (Hoffman & Gelman, 2014) can adaptively set the number of leapfrog steps L to avoid U-turns and find a global constant for the step sizes by adjusting the average acceptance rate. In section 4.1, we found we can outperform NUTS even though we do not adaptively set L. With our objective, we can tune individual step sizes (and masses) for each dimension and step in the chain, allowing for a much higher degree of granular control over the algorithm. Furthermore, we do not need to rely on 'rules of thumb' such as standard acceptance rate targets used in many algorithms (Hoffman & Gelman, 2014; Hoffman, 2017) but we can automatically tune all continuous hyperparameters using information from the target distribution directly.

Other works use MCMC as part of a hybrid inference scheme. Ruiz & Titsias (2019) propose a novel objective to improve the training of the initial distribution using MCMC samples. However, their objective cannot be used for tuning MCMC as it encourages final samples to be close to the initial distribution. Hoffman et al. (2019) use normalizing flows to warp the target distribution such that it is close to an isotropic Gaussian and thus easy to sample from using HMC. Though, they must still use gradient-free optimization on a heuristic to tune the HMC parameters, one could investigate performance if hyperparameters were instead tuned using our differentiable objective.

Our work also builds upon methods from statistical mechanics. The Boltzmann Generator (Noé et al., 2019) opened up this line of research by using a normalizing flow to sample molecular configurations. We found we can improve upon this by using the flow as the initial distribution for HMC, fine tuning the flow samples with our short chains. Our Alanine Dipeptide experiment comes from the recent work of Wu et al. (2020) on stochastic normalizing flows, which consist of stochastic steps interspersed between deterministic steps in a normalizing flow. It is possible to combine such models with our approach, using our objective to tune hyperparameters within the stochastic layers. We leave this extension to future work.

6. Conclusion

In this work, we presented a new objective motivated by VI that can be easily used for the gradient-based optimization of HMC hyperparameters. We provided a fully automatic method for choosing an initial distribution for the HMC chain that reduces burn-in time and aids optimization. Evaluating on multiple real-world problems, we found our method is competitive with or improves upon existing methods for tuning hyperparameters. We hope this encourages further work applying this idea to other methods that use HMC and that would benefit from increased convergence speed.

Acknowledgements

Andrew Campbell acknowledges support from the EPSRC CDT in Modern Statistics and Statistical Machine Learning (EP/S023151/1). José Miguel Hernández Lobato acknowledges support from a Turing AI Fellowship under grant EP/V023756/1. Yichuan Zhang acknowledges support from Samsung Research, Samsung Electronics Co., Seoul, Republic of Korea and from Boltzbit Ltd.

References

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

Caterini, A. L., Doucet, A., and Sejdinovic, D. Hamiltonian variational auto-encoder. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Chwialkowski, K., Strathmann, H., and Gretton, A. A kernel

test of goodness of fit. *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations (ICLR)*, 2017.

Donoho, D. L. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. doi: 10.1109/TIT.2006.871582.

Gong, W., Li, Y., and Hernández-Lobato, J. M. Sliced kernelized stein discrepancy. *International Conference on Learning Representations (ICLR)*, 2021.

Hairer, E., Lubich, C., and Wanner, G. Geometric numerical integration illustrated by the Störmer–Verlet method. *Acta Numerica*, 12:399–450, 2003.

Hernández-Lobato, J., Hernández-Lobato, D., and Suárez, A. Expectation propagation in linear regression models with spike-and-slab priors. *Machine Learning*, 99:437–487, 2015.

Hernández-Lobato, J. M., Li, Y., Rowland, M., Bui, T., Hernandez-Lobato, D., and Turner, R. Black-box alpha divergence minimization. *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Hoffman, M. and Gelman, A. The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 2014.

Hoffman, M. and Ma, Y. Black-box variational inference as a parametric approximation to Langevin dynamics. *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

Hoffman, M. D. Learning deep latent Gaussian models with Markov Chain Monte Carlo. *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Ji, S., Xue, Y., and Carin, L. Bayesian compressive sensing. *IEEE Transactions on signal processing*, 56(6):2346–2356, 2008.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Kingma, D. P. and Welling, M. Auto-encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Leimkuhler, B. and Reich, S. *Simulating Hamiltonian dynamics*. Number 14. Cambridge university press, 2004.

Levy, D., Hoffman, M. D., and Sohl-Dickstein, J. Generalizing Hamiltonian Monte Carlo with neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Liu, Q., Lee, J., and Jordan, M. A kernelized stein discrepancy for goodness-of-fit tests. *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Neal, R. M. Probabilistic Inference using Markov Chain Monte Carlo Methods. Technical report, University of Toronto, Department of Computer Science, 1993.

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

Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), September 2019. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aaw1147.

Pasarica, C. and Gelman, A. Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, 20(1):343–364, 2010.

Ramachandran, G. N., Ramakrishnan, C., and Sasisekharan, V. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7(1):95–99, July 1963. ISSN 0022-2836. doi: 10.1016/S0022-2836(63)80023-6.

Rezende, D. and Mohamed, S. Variational Inference with Normalizing Flows. *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proceedings of the International Conference on Machine Learning (ICML)*, 2014.

Ruiz, F. and Titsias, M. A contrastive divergence for combining variational inference and MCMC. *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Salimans, T., Kingma, D., and Welling, M. Markov Chain Monte Carlo and variational inference: Bridging the gap. *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Sohl-Dickstein, J. and Culpepper, B. J. Hamiltonian annealed importance sampling for partition function estimation. Technical report, Redwood Center for Theoretical Neuroscience University of California, Berkeley, 2012.

Tabak, E. G. and Vanden-Eijnden, E. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010. ISSN 15396746, 19450796. doi: 10.4310/CMS.2010.v8.n1.a11.

Titsias, M. and Dellaportas, P. Gradient-based adaptive Markov Chain Monte Carlo. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Tucker, G., Lawson, D., Gu, S., and Maddison, C. Doubly reparameterized gradient estimators for Monte Carlo objectives. *International Conference on Learning Representations (ICLR)*, 2019.

Wolf, C., Karl, M., and van der Smagt, P. Variational inference with Hamiltonian Monte Carlo. *arXiv preprint arXiv:1609.08203*, 2016.

Wu, H., Köhler, J., and Noé, F. Stochastic Normalizing Flows. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.