A Gradient Based Strategy for Hamiltonian Monte Carlo Hyperparameter Optimization Supplementary Material

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

1. Analysis of Gradients

1.1. Problem Setup

In this section we analyse the effect of the accept/reject step on the gradients obtained through the reparameterization trick. We will work on a generic form of the objective which can be written as

$$\mathbb{E}_{\boldsymbol{x}_T}\left[\mathcal{L}(\boldsymbol{x}_T)\right] \tag{1}$$

where $\boldsymbol{x}_T \in \mathbb{R}^n$ is the final sample from a T step HMC chain and $\mathcal{L}$ is some loss function. For the objective we propose, $\mathcal{L}(\boldsymbol{x}_T)$ would be $\log p^*(\boldsymbol{x}_T)$. We are looking to differentiate this objective with respect to the step size at some layer k: $\boldsymbol{\epsilon}_k \in \mathbb{R}^n$, $k \in [1, T]$. We will do this using the reparameterization trick. The primitive variables we will use for this are

$$\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0) \tag{2}$$

$$\boldsymbol{\nu}_t \sim \mathcal{N}(0, \boldsymbol{I}) \quad \text{for} \quad t \in [0, T-1] \tag{3}$$

$$u_t \sim \mathcal{U}(0, 1) \quad \text{for} \quad t \in [1, T] \tag{4}$$

For the analysis that follows we will assume all masses are unity and consider only the differentiation with respect to the step sizes. All ideas easily transfer to the non-unity mass case. For the reparameterization trick, in order to exchange the order of integration and differentiation, we need to make sure the mapping from the primitive variables to $\boldsymbol{x}_T$ is smooth. This is not the case with the accept/reject step, however, just for the purpose of this analysis we can approximate it with a sigmoid, the approximation being improved as the sigmoid becomes sharper. Thus we can make inferences about the gradient behaviour when we use the normal accept/reject step by examining the effect in the limit as the sigmoid becomes infinitely sharp. Using this formulation, we will write the accept/reject step as

$$\boldsymbol{x}_t = \boldsymbol{x}'_t \mathbb{S}(u_t, p^{(t)}_{MH}) + \boldsymbol{x}_{t-1}(1 - \mathbb{S}(u_t, p^{(t)}_{MH})) \tag{5}$$

where $\boldsymbol{x}'_t$ is the proposed new state from the leapfrog operator applied for L steps,

$$\boldsymbol{x}'_t = LF(\boldsymbol{x}_{t-1}, \boldsymbol{\nu}_{t-1}, \boldsymbol{\epsilon}_t) \tag{6}$$

$p^{(t)}_{MH}$ is the Metropolis-Hastings acceptance ratio,

$$p^{(t)}_{MH} = \min\{1, \exp(-H(\boldsymbol{x}'_t, \boldsymbol{\nu}'_t) + H(\boldsymbol{x}_{t-1}, \boldsymbol{\nu}_{t-1}))\} \tag{7}$$

and $\mathbb{S}(u_t, p^{(t)}_{MH})$ is a sigmoid that will approximate the indicator function: $\mathbb{I}\{u_t < p^{(t)}_{MH}\}$.

$$\mathbb{S}(u_t, p^{(t)}_{MH}) = \frac{1}{1 + \exp(-s(p^{(t)}_{MH} - u_t))} \tag{8}$$

[*]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.

s is a 'sharpness' parameter, as $s \to \infty$ the sigmoid becomes a better and better approximation to the indicator function centered on $p_{MH}^{(t)}$.

1.2. Derivation of Gradient Estimator

We begin by taking gradients of the objective using the reparameterization trick

$$\frac{\partial}{\partial \boldsymbol{\epsilon}_k} \mathbb{E}_{\boldsymbol{x}_T}[\mathcal{L}(\boldsymbol{x}_T)] \tag{9}$$

$$= \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T}} \left[\frac{\partial}{\partial \boldsymbol{\epsilon}_k} \mathcal{L}(\boldsymbol{x}_T) \right] \tag{10}$$

$$= \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T}} \left[\frac{\partial \boldsymbol{x}_T}{\partial \boldsymbol{\epsilon}_k} \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] \tag{11}$$

We use 'denominator layout' for vector derivatives i.e. $\frac{\partial \boldsymbol{x}_T}{\partial \boldsymbol{\epsilon}_k}$ is a matrix with element ij corresponding to $\frac{\partial (\boldsymbol{x}_T)_j}{\partial (\boldsymbol{\epsilon}_k)_i}$.

To find $\frac{\partial \boldsymbol{x}_T}{\partial \boldsymbol{\epsilon}_k}$

$$\boldsymbol{x}_T = \boldsymbol{x}'_T \mathbb{S}(u_T, p_{MH}^{(T)}) + \boldsymbol{x}_{T-1}(1 - \mathbb{S}(u_T, p_{MH}^{(T)})) \tag{12}$$

$$\frac{\partial \boldsymbol{x}_T}{\partial \boldsymbol{\epsilon}_k} = \frac{\partial \boldsymbol{x}'_T}{\partial \boldsymbol{\epsilon}_k} \mathbb{S}_T + \frac{\partial \mathbb{S}_T}{\partial p_{MH}^{(T)}} \frac{\partial p_{MH}^{(T)}}{\partial \boldsymbol{\epsilon}_k} \boldsymbol{x}'^T_T + (1 - \mathbb{S}_T) \frac{\partial \boldsymbol{x}_{T-1}}{\partial \boldsymbol{\epsilon}_k} - \frac{\partial \mathbb{S}_T}{\partial p_{MH}^{(T)}} \frac{\partial p_{MH}^{(T)}}{\partial \boldsymbol{\epsilon}_k} \boldsymbol{x}^T_{T-1} \tag{13}$$

$$\frac{\partial \boldsymbol{x}'_T}{\partial \boldsymbol{\epsilon}_k} = \frac{\partial \boldsymbol{x}_{T-1}}{\partial \boldsymbol{\epsilon}_k} \frac{\partial LF(\boldsymbol{x}_{T-1}, \boldsymbol{\nu}_{T-1}, \boldsymbol{\epsilon}_T)}{\partial \boldsymbol{x}_{T-1}} + \underbrace{\frac{\partial LF(\boldsymbol{x}_{T-1}, \boldsymbol{\nu}_{T-1}, \boldsymbol{\epsilon}_T)}{\partial \boldsymbol{\epsilon}_k}}_{\text{zero if } k \neq T} \tag{14}$$

On the last line we have separated $\boldsymbol{\epsilon}_k$'s contributions to the gradient. The first is if $\boldsymbol{\epsilon}_k$ affects $\boldsymbol{x}_{T-1}$ and the second is $\boldsymbol{\epsilon}_k$'s effect on this current leapfrog step (only non-zero if $k = T$). Substituting back in we get

$$\frac{\partial \boldsymbol{x}_T}{\partial \boldsymbol{\epsilon}_k} = \frac{\partial \boldsymbol{x}_{T-1}}{\partial \boldsymbol{\epsilon}_k} \left[\frac{\partial LF_T}{\partial \boldsymbol{x}_{T-1}} \mathbb{S}_T + \mathbb{I}(1 - \mathbb{S}_T) \right] + \frac{\partial LF_T}{\partial \boldsymbol{\epsilon}_k} \mathbb{S}_T + \frac{\partial \mathbb{S}_T}{\partial p_{MH}^{(T)}} \frac{\partial p_{MH}^{(T)}}{\partial \boldsymbol{\epsilon}_k} (\boldsymbol{x}'^T_T - \boldsymbol{x}^T_{T-1}) \tag{15}$$

We can now recurse this equation. Define

$$\alpha_t = \frac{\partial LF_t}{\partial \boldsymbol{x}_{t-1}} \mathbb{S}_t + \mathbb{I}(1 - \mathbb{S}_t) \tag{16}$$

$$\beta_t = \frac{\partial LF_t}{\partial \boldsymbol{\epsilon}_k} \mathbb{S}_t \quad (\beta_t = 0 \text{ for } k \neq t) \tag{17}$$

$$\gamma_t = \frac{\partial \mathbb{S}_t}{\partial p_{MH}^{(t)}} \frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k} (\boldsymbol{x}'^T_t - \boldsymbol{x}^T_{t-1}) \tag{18}$$

$$\frac{\partial \boldsymbol{x}_T}{\partial \boldsymbol{\epsilon}_k} = \beta_k \alpha_{k+1} \dots \alpha_T + \sum_{t=k}^{T} \gamma_t \alpha_{t+1} \dots \alpha_T \tag{19}$$

where we slightly abuse notation and let $\alpha_{t+1} \dots \alpha_T = \mathbb{I}$ when $t = T$. Substitute this back into the equation for the overall gradient

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T}} \left[(\beta_k \alpha_{k+1} \dots \alpha_T + \sum_{t=k}^{T} \gamma_t \alpha_{t+1} \dots \alpha_T) \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] \tag{20}$$

$$= \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T}} \left[\beta_k \alpha_{k+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] + \sum_{t=k}^{T} \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T}} \left[\gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] \tag{21}$$

Therefore, the true gradient estimator for a single sample is

$$\underbrace{\beta_k \alpha_{k+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T}}_{\text{autodiff term}} + \underbrace{\sum_{t=k}^{T} \gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T}}_{\text{bias term}} \tag{22}$$

The first term is the gradient estimator that is obtained from standard automatic differentiation which simply accumulates gradients for leapfrog steps that are accepted. The second term is a bias term that is caused by the step size's effect on the acceptance probability. We now ask what is the behaviour of the bias term as $s \to \infty$.

1.3. Expected Value and Variance of Bias Term

First, we examine the expected value of the bias term. Picking out just one term of the summation,

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T}} \left[\gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] \tag{23}$$

$$= \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T}} \left[\frac{\partial \mathbb{S}_t}{\partial p_{MH}^{(t)}} \frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k} (\boldsymbol{x}_t'^T - \boldsymbol{x}_{t-1}^T) \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] \tag{24}$$

The problematic term is $\frac{\partial \mathbb{S}_t}{\partial p_{MH}^{(t)}}$. As a function of u_t, it becomes more and more peaked as $s \to \infty$. However, the integral with respect to u_t is bounded:

$$\frac{\partial \mathbb{S}_t}{\partial p_{MH}^{(t)}} = s \frac{\exp(-s(p_{MH}^{(t)} - u_t))}{(1 + \exp(-s(p_{MH}^{(t)} - u_t)))^2} \tag{25}$$

$$\int_0^1 s \frac{\exp(-s(p_{MH}^{(t)} - u_t))}{(1 + \exp(-s(p_{MH}^{(t)} - u_t)))^2} du_t \tag{26}$$

The limits are taken to be $[0, 1]$ because $u_t \sim \mathcal{U}(0, 1)$. Let $y = \exp(-s(p_{MH}^{(t)} - u_t))$

$$\int_{\exp(-sp_{MH}^{(t)})}^{\exp(-s(p_{MH}^{(t)} - 1))} \frac{1}{(1+y)^2} dy = \frac{1}{1 + \exp(-sp_{MH}^{(t)})} - \frac{1}{1 + \exp(-s(p_{MH}^{(t)} - 1))} \tag{27}$$

For $0 < p_{MH}^{(t)} < 1$, as $s \to \infty$ the first fraction tends to 1 and the second tends to 0. We also know that $\frac{\partial \mathbb{S}_t}{\partial p_{MH}^{(t)}}$ as a function of u_t becomes more and more peaked around $p_{MH}^{(t)}$. Therefore it acts more and more like $\delta(u_t - p_{MH}^{(t)})$. Thus the bias terms tend towards

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\nu}_{0:T-1}, u_{1:T/t}} \left[\frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k} (\boldsymbol{x}_t'^T - \boldsymbol{x}_{t-1}^T) \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \middle| u_t = p_{MH}^{(t)} \right] \tag{28}$$

which is evidently non-zero (we have neglected the case where $p_{MH}^{(t)} = 1$ but this simply introduces another non-zero term which doesn't change our conclusions).

We now examine the variance of the bias term.

$$\mathbb{V} \left[\sum_{t=k}^{T} \gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] = \sum_{t=k}^{T} \mathbb{V} \left[\gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] + \text{cross terms} \tag{29}$$

$$\mathbb{V} \left[\gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right] = \underbrace{\mathbb{E} \left[\gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \left(\gamma_t \alpha_{t+1} \dots \alpha_T \frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T} \right)^T \right]}_{\text{ⓐ}} - \underbrace{\boldsymbol{\mu}\boldsymbol{\mu}^T}_{\text{finite}} \tag{30}$$

$$ⓐ = \mathbb{E}\left[\left(\frac{\partial \mathbb{S}_t}{\partial p_{MH}^{(t)}}\right)^2 \frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k}(\boldsymbol{x}_t'^T - \boldsymbol{x}_{t-1}^T)\alpha_{t+1}\dots\alpha_T \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_T}\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_T}\right)^T \alpha_T^T \dots \alpha_{t+1}^T(\boldsymbol{x}_t' - \boldsymbol{x}_{t-1})\left(\frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k}\right)^T\right] \tag{31}$$

This time we have to look at $\left(\frac{\partial \mathbb{S}_t}{\partial p_{MH}^{(t)}}\right)^2$. Again writing the integral,

$$\int_0^1 \left(s\frac{\exp(-s(p_{MH}^{(t)} - u_t))}{(1+\exp(-s(p_{MH}^{(t)} - u_t)))^2}\right)^2 du_t \tag{32}$$

As before, let $y = \exp(-s(p_{MH}^{(t)} - u_t))$

$$\int_{\exp(-sp_{MH}^{(t)})}^{\exp(-s(p_{MH}^{(t)}-1))} s\frac{y}{(1+y)^4}dy = s\Bigg[\left(-\frac{1}{2}\frac{1}{(1+e^{-s(p_{MH}^{(t)}-1)})^2} + \frac{1}{3}\frac{1}{(1+e^{-s(p_{MH}^{(t)}-1)})^3}\right) - \tag{33}$$

$$\left(-\frac{1}{2}\frac{1}{(1+e^{-sp_{MH}^{(t)}})^2} + \frac{1}{3}\frac{1}{(1+e^{-sp_{MH}^{(t)}})^3}\right)\Bigg] \tag{34}$$

For $0 < p_{MH}^{(t)} < 1$, as $s \to \infty$, this value tends towards $\frac{s}{6}$ giving infinite variance for this bias term in the limit.

From this analysis, we have found that the true gradient estimator can be split into a term that corresponds to a standard application of automatic differentiation and a bias term originating from the dependence of the acceptance probability on the hyperparameters. As the sigmoid is made sharper to better approximate the indicator function, the bias term's expected value does not vanish and furthermore it's variance diverges. In this work, we wish to keep the accept/reject step and so ignore the bias term in our calculations for the gradient. We have found good empirical success making this approximation, the intuition being that the comparative magnitude of this bias is small in practice. We now examine why this is the case.

1.4. Order of Magnitude Analysis

In equation (28) we found that the expected value of the t^{th} bias term in the summation from equation (22) depends on the value of $\frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k}$. Expanding this term

$$\frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k} = \frac{\partial}{\partial \boldsymbol{\epsilon_k}}\Big[\min\{1, \exp(-H(\boldsymbol{x}_t', \boldsymbol{\nu}_t') + H(\boldsymbol{x}_{t-1}, \boldsymbol{\nu}_{t-1}))\}\Big] \tag{35}$$

$$= -\frac{\partial \Delta H_t}{\partial \boldsymbol{\epsilon}_k}\exp(-\Delta H_t)\mathbb{I}\{\exp(-\Delta H_t) < 1\} \tag{36}$$

where $\Delta H_t = H(\boldsymbol{x}_t', \boldsymbol{\nu}_t') - H(\boldsymbol{x}_{t-1}, \boldsymbol{\nu}_{t-1})$ is the error in the value of the Hamiltonian due to the leapfrog discretization. Consider first the case when $k = t$. It is known that the Hamiltonian error for the leapfrog integrator has order ϵ^3 (Neal, 2011) - 'local error' applies in this case as we have a fixed number of leapfrog steps as opposed to a fixed integration time. The derivative $\frac{\partial \Delta H_k}{\partial \boldsymbol{\epsilon_k}}$ will therefore have order ϵ^2. For the case $t > k$, $\boldsymbol{\epsilon_k}$ will affect the acceptance probability only through its effect on $\boldsymbol{x}_{t-1}$,

$$\frac{\partial p_{MH}^{(t)}}{\partial \boldsymbol{\epsilon}_k} = -\frac{\partial \boldsymbol{x}_{t-1}}{\partial \boldsymbol{\epsilon}_k}\frac{\partial \Delta H_t}{\partial \boldsymbol{x}_{t-1}}\exp(-\Delta H_t)\mathbb{I}\{\exp(-\Delta H_t) < 1\} \tag{37}$$

Therefore we only get the derivative $\frac{\partial \Delta H_t}{\partial \boldsymbol{x}_{t-1}}$ which will have order ϵ^3.

Comparing this to the 'autodiff' term in equation (22) used for optimization, we see the post multiplication by $\alpha_{k+1}\dots\alpha_T\frac{\partial \mathcal{L}(\boldsymbol{x}_T)}{\partial \boldsymbol{x}_T}$ is the same between the autodiff term and the k^{th} bias term. The difference is introduced with β_k,

$$\beta_k = \frac{\partial LF_k}{\partial \boldsymbol{\epsilon}_k}\mathbb{S}_k \tag{38}$$

It is easy to see this will have order ϵ^0. Examining just the final of L leapfrog steps

$$\boldsymbol{x}_L = \boldsymbol{x}_{L-1} + \boldsymbol{\nu}_{L-\frac{1}{2}} \circ \boldsymbol{\epsilon}_k \tag{39}$$

$$\frac{\partial LF_k}{\partial \boldsymbol{\epsilon}_k} = \frac{\partial \boldsymbol{x}_L}{\partial \boldsymbol{\epsilon}_k} = \frac{\partial \boldsymbol{x}_{L-1}}{\partial \boldsymbol{\epsilon_k}} + \frac{\partial \boldsymbol{\nu}_{L-\frac{1}{2}}}{\partial \boldsymbol{\epsilon}_k}\text{diag}(\boldsymbol{\epsilon}_k) + \text{diag}(\boldsymbol{\nu}_{L-\frac{1}{2}}) \tag{40}$$

We see we have already obtained a term independent of $\boldsymbol{\epsilon}_k$.

In summary, we have shown that the bias term for the true gradient estimator is of order ϵ^2 whereas the 'autodiff' term has order ϵ^0. Since in practice we keep ϵ small to obtain high acceptance probabilities, the relative effect of ignoring the bias term is small. We note here that it is useful to keep the accept/reject step even though most steps are accepted because in the case we enter an unstable regime we would have no mechanism to reject a sample that is the result of divergent dynamics. This could then throw off any sample based estimate involving this sample. Furthermore, without the accept/reject step, we lose all ergodic guarantees, however, this is of less importance when we only consider short chains.

2. Auxiliary Optimization Objectives Details

2.1. α-divergence

The original α divergence estimator was proposed by Hernández-Lobato et al. (2016). To reduce the variance of the estimator, we use a doubly reparameterized gradient estimator for the Monte-Carlo α-divergence objective (Tucker et al., 2019). We derive the form of this estimator here. For an unnormalized target $p^*(\mathbf{x})$ and approximation $q_{\boldsymbol{\phi}}(\mathbf{x})$, let $w_i = \frac{p^*(\mathbf{x}_i)}{q_{\boldsymbol{\phi}}(\mathbf{x}_i)}$ and $\mathbf{x}_{1:K} \sim \prod_i p_{\boldsymbol{\phi}}(\mathbf{x}_i)$ can be sampled using reparameterization trick: $\mathbf{x}_i = \mathbf{x}(\boldsymbol{\epsilon}_i; \boldsymbol{\phi})$, then minimizing α-divergence is equivalent to maximizing the following objective:

$$L_\alpha = \mathbb{E}_{\mathbf{x}_{1:K}}[\log \frac{1}{K}\sum_{i=1}^K w_i^\alpha] = \mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\log \frac{1}{K}\sum_{i=1}^K w_i^\alpha] \tag{41}$$

The gradient of L_α is

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}}\mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\log \frac{1}{K}\sum_{i=1}^K w_i^\alpha] &= \mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\sum_{i=1}^K \frac{1}{\sum_{j=1}^K w_j^\alpha}\nabla_{\boldsymbol{\phi}} w_i^\alpha] \\
&= \mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\sum_{i=1}^K \frac{\alpha w_i^{\alpha-1}}{\sum_{j=1}^K w_j^\alpha}\nabla_{\boldsymbol{\phi}} w_i] \\
&= \mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\sum_{i=1}^K \frac{\alpha w_i^{\alpha-1}}{\sum_{j=1}^K w_j^\alpha}(\nabla_{\mathbf{x}_i} w_i \nabla_{\boldsymbol{\phi}}\mathbf{x}_i - w_i \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}_i))] \\
&= \mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\sum_{i=1}^K \frac{\alpha w_i^{\alpha}}{\sum_{j=1}^K w_j^\alpha}(\nabla_{\mathbf{x}_i} \log w_i \nabla_{\boldsymbol{\phi}}\mathbf{x}_i - \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}_i))]
\end{aligned} \tag{42}$$

Now we would like to rewrite the term $\mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\sum_{i=1}^{K}\frac{\alpha w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{x}_i)]$, which can contribute significant variance to the gradient estimator (Roeder et al., 2017), to a form with less variance.

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\sum_{i=1}^{K}\frac{\alpha w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{x}_i)] &= \sum_{i=1}^{K}\mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\frac{\alpha w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{x}_i)] \\ &= \sum_{i=1}^{K}\mathbb{E}_{\mathbf{x}_{1:K}}[\frac{\alpha w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{x}_i)] \\ &= \alpha\sum_{i=1}^{K}\mathbb{E}_{\mathbf{x}_{-i}\mathbf{x}_i}[\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{x}_i)]\end{aligned} \tag{43}$$

where $\mathbf{x}_{-i}$ represents $\mathbf{x}_{1:i-1}\cup\mathbf{x}_{i+1:K}$. The equation above can be rewritten by taking advantage of the equivalence between the REINFORCE gradient and the reparameterization trick gradient (proved in (Tucker et al., 2019)):

$$\begin{aligned}\mathbb{E}_{\mathbf{x}_i}[\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{x}_i)] &= \mathbb{E}_{\boldsymbol{\epsilon}_i}[\nabla_{\mathbf{x}_i}(\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha})\nabla_{\boldsymbol{\phi}}\mathbf{x}_i] \\ &= \alpha\mathbb{E}_{\boldsymbol{\epsilon}_i}[(\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha} - \frac{w_i^{2\alpha}}{(\sum_{j=1}^{K} w_j^\alpha)^2})\nabla_{\mathbf{x}_i}\log w_i\nabla_{\boldsymbol{\phi}}\mathbf{x}_i]\end{aligned} \tag{44}$$

Plugging 44 into 42, we can get

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\log\frac{1}{K}\sum_{i=1}^{K} w_i^\alpha] &= \alpha\sum_{i=1}^{K}\mathbb{E}_{\boldsymbol{\epsilon}_i}[(\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha} - \alpha\frac{w_i^\alpha}{(\sum_{j=1}^{K} w_j)^\alpha} + \alpha\frac{w_i^{2\alpha}}{(\sum_{j=1}^{K} w_j^\alpha)^2})\nabla_{\mathbf{x}_i}\log w_i\nabla_{\boldsymbol{\phi}}\mathbf{x}_i] \\ &= \alpha\sum_{i=1}^{K}\mathbb{E}_{\boldsymbol{\epsilon}_i}[((1-\alpha)\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha} + \alpha(\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha})^2)\nabla_{\mathbf{x}_i}\log w_i\nabla_{\boldsymbol{\phi}}\mathbf{x}_i]\end{aligned} \tag{45}$$

Therefore, the gradient of DReG-L_α with respect to $\boldsymbol{\phi}$ is

$$\alpha\mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}[\sum_{i=1}^{K}((1-\alpha)\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha} + \alpha(\frac{w_i^\alpha}{\sum_{j=1}^{K} w_j^\alpha})^2)\nabla_{\mathbf{x}_i}\log w_i\nabla_{\boldsymbol{\phi}}\mathbf{x}_i] \tag{46}$$

Thus in Algorithm 1 in the main text, $\nabla_\psi D_\alpha(q_\psi^{(0)}(x)\,||\,p(x))$ corresponds to the a sample based estimator of (46) with $w_i = \frac{p^*(x_i)}{q_\psi^{(0)}(x_i)}$.

2.2. SKSD

This discrepancy metric is taken from recent work by Gong et al. (2021). In that work, they derive the following scalable discrepancy metric

$$SK_{\max}(q,p) = \sum_{\boldsymbol{r}\in O_r}\sup_{\boldsymbol{g}_r}\mathbb{E}_{q(\boldsymbol{x})q(\boldsymbol{x}')}[h_{p,r,g_r}(\boldsymbol{x},\boldsymbol{x}')]$$

where

$$\begin{aligned}h_{p,r,g}(\boldsymbol{x},\boldsymbol{y}) =& s_p^r(\boldsymbol{x})k_{rg}(\boldsymbol{x}^T\boldsymbol{g},\boldsymbol{y}^T\boldsymbol{g})s_p^r(\boldsymbol{y}) + \boldsymbol{r}^T\boldsymbol{g}s_p^r(\boldsymbol{y})\nabla_{\boldsymbol{x}^T\boldsymbol{g}}k_{rg}(\boldsymbol{x}^T\boldsymbol{g},\boldsymbol{y}^T\boldsymbol{g}) + \\ & \boldsymbol{r}^T\boldsymbol{g}s_p^r(\boldsymbol{x})\nabla_{\boldsymbol{y}^T\boldsymbol{g}}k_{rg}(\boldsymbol{x}^T\boldsymbol{g},\boldsymbol{y}^T\boldsymbol{g}) + (\boldsymbol{r}^T\boldsymbol{g})^2\nabla^2_{\boldsymbol{x}^T\boldsymbol{g},\boldsymbol{y}^T\boldsymbol{g}}k_{rg}(\boldsymbol{x}^T\boldsymbol{g},\boldsymbol{y}^T\boldsymbol{g})\end{aligned}$$

$s_p^r(\boldsymbol{x}) = (\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x}))^T\boldsymbol{r}$ is the projection of the score using $\boldsymbol{r}\in O_r$ a vector from an orthogonal basis (we use the one-hot vectors) and $k_{rg}(\boldsymbol{x}^T\boldsymbol{g},\boldsymbol{y}^T\boldsymbol{g})$ is some kernel operating on scalar values. They show that $SK_{\max}$ is equal to 0 if and only if $p=q$ almost everywhere (Corollary 3.1).

For our experiments, we use a sample based estimate of this discrepancy. Using the notation from our main paper, the estimator is

$$\text{SKSD}(x_{1:N}^{(T)}, \text{score}(x)) = \frac{1}{N(N-1)} \sum_{\boldsymbol{r} \in O_r} \sum_{1 \leq i \neq j \leq N} h_{p,r,g_r}(x_i^{(T)}, x_j^{(T)})$$

To perform optimization over $\boldsymbol{g}_r$ values, we again utilize SGD on this sample based estimator. We optimize $\boldsymbol{g}_r$ jointly with s, maximizing with respect to each $\boldsymbol{g}_r$ vector and minimizing with respect to s.

For the kernel $k_{rg}(\boldsymbol{x}^T\boldsymbol{g}, \boldsymbol{y}^T\boldsymbol{g})$, we use the Radial Basis Function kernel.

$$k_{rg}(\boldsymbol{x}^T\boldsymbol{g}, \boldsymbol{y}^T\boldsymbol{g}) = \exp\left(-\frac{|\boldsymbol{x}^T\boldsymbol{g} - \boldsymbol{y}^T\boldsymbol{g}|^2}{\sigma}\right)$$

The bandwidth σ is set to be the median squared pairwise distance calculated on a batch of $\boldsymbol{x}^T\boldsymbol{g}$ samples.

3. 2D Distributions Experiment

As described in the main text we use 30-step HMC chains to draw samples from the targets. The initial distribution trained through $\alpha = \{0, 1\}$ divergence is trained to convergence. The HMC hyperparameters are tuned using the maxELT objective. We apply 200 gradient updates using this objective for the Gaussian and Laplace targets and 500 for the others.

3.1. Baselines

The first baseline is taken from Hoffman (2017), where the step size in each dimension k is given by $\sigma_k \epsilon_0$ with σ_k being the standard deviation in dimension k, as estimated by a Gaussian fitted by minimizing the $\alpha = 0$-divergence and ϵ_0 being adjusted as to control the minimum acceptance probability over a batch of parallel chains. In line with Hoffman (2017), we set this minimum acceptance probability to 0.25.
The second baseline is the method from Ruiz and Titsias (2019), where the initial factorized Gaussian is tuned by minimizing a novel divergence metric which incorporates feedback from the final states of HMC, and the HMC step size in each dimension is defined as in Hoffman (2017), but this time we adjust ϵ_0 to ensure that the mean acceptance probability over a batch of parallel chains is equal to 0.65.
The final baseline is the No-U-Turn Sampler (Hoffman and Gelman, 2014)[1]. We use the dual averaging variant of the No-U-Turn Sampler, which includes a method for tuning the step size as to encourage an equivalent of the HMC average acceptance probability towards a target $\delta \in (0, 1)$. We set $\delta = 0.2$ in our experiments.

3.2. Equations for 2D targets

We list the probability density functions for all targets in Table 1.

3.3. Comparison between $-\mathbb{E}_p\big[\log p^*(x)\big]$ and $-\mathbb{E}_{q_\phi^{(T)}}\big[\log p^*(x)\big]$

Here, we compare $-\mathbb{E}_p\big[\log p^*(x)\big]$ with $-\mathbb{E}_{q_\phi^{(T)}}\big[\log p^*(x)\big]$ to quantify the mode seeking behaviour of the different methods. We display our results in Table 2. We see that the SKSD helps the method better match the ground truth and on some distributions e.g. Gaussian, helps prevent $-\mathbb{E}_{q_\phi^{(T)}}\big[\log p^*(x)\big]$ from underestimating $-\mathbb{E}_p\big[\log p^*(x)\big]$. This may occur with a narrow initial distribution ($\alpha = 0$) because the optimization will encourage the chains to remain in the region of high log target, artificially inflating the $\mathbb{E}_{q_\phi^{(T)}}\big[\log\ p^*(x)\big]$ value.

4. Sparse Signal Recovery Experiment

Here we cover in more detail the setup for the sparse signal recovery experiment. We repeat the overall description of the model for ease of exposition. 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

$$y = Xw + e \tag{47}$$

[1]We use the implementation from https://github.com/mfouesneau/NUTS in our experiments.

Table 1. Unnormalized log densities for the 2D distributions used in the first experiment.

Name	Unnormalized log density, $\log p^*(x)$
Gaussian	$-\frac{1}{2}\left(\frac{32}{19}x_1^2 - \frac{60}{19}x_1x_2 + \frac{40}{19}x_2^2\right)$
Laplace	$-\lvert x_1 - 5\rvert - \lvert x_2 - 5\rvert$
Dual Moon	$-3.125\left(\sqrt{x_1^2 + x_2^2} - 2\right)^2 + \log\left[\exp\left(-0.5\left(\frac{x_1+2}{0.6}\right)^2\right) + \exp\left(-0.5\left(\frac{x_1-2}{0.6}\right)^2\right)\right]$
Mixture	$\log\left[\sum_{i=1}^{7}\exp\left(-0.5\left[\left(x_1 - 5\cos\left(\frac{2i\pi}{7}\right)\right)^2 + \left(x_2 - 5\sin\left(\frac{2i\pi}{7}\right)\right)^2\right]\right)\right]$
Wave 1	$\log\left[\exp\left(-0.5\left[\frac{x_2+\sin(0.5\pi x_1)}{0.35}\right]^2\right) + \exp\left(-0.5\left[\frac{-x_2-\sin(0.5\pi x_1)+3\exp\left(-\frac{0.5}{0.36}(x_1-1)^2\right)}{0.35}\right]^2\right)\right]$
Wave 2	$\log\left[\exp\left(-0.5\left[\frac{x_2+\sin(0.5\pi x_1)}{0.4}\right]^2\right) + \exp\left(-0.5\left[\frac{-x_2-\sin(0.5\pi x_1)+\frac{3}{1+\exp\left(-\frac{x_1-1}{0.3}\right)}}{0.35}\right]^2\right)\right]$

Table 2. $-\mathbb{E}_{q_\phi^{(T)}}\left[\log p^*(x)\right]$ values for the baselines and the 4 variations of our method on each of the synthetic test distributions. The ground truth value $-\mathbb{E}_p\left[\log p^*(x)\right]$ is found using rejection sampling.

	Gaussian	Laplace	Dual Moon	Mixture	Wave 1	Wave 2
Ground-truth	2.8083	2.0075	0.8511	0.9282	-0.1703	0.1483
$\alpha = 0$	2.4911	1.9937	1.0315	0.9216	0.0463	0.1314
$\alpha = 1$	2.7861	2.0431	2.9218	0.9197	-0.1050	0.6938
SKSD & $\alpha = 0$	2.8390	2.0074	0.8439	0.9174	-0.1293	0.2500
SKSD & $\alpha = 1$	2.8183	2.0281	0.7987	0.9198	-0.1814	1.2002
$\min \bar{p} = 0.25$	3.0148	1.9140	4.1515	2.4894	1.4220	1.0397
Ruiz and Titsias	2.7993	2.0012	2.8102	0.9765	-0.0709	0.1661
NUTS	2.7862	1.9955	0.7828	0.9379	-0.1979	0.0911

with a fixed measurement matrix $X \in \mathbb{R}^{n\times d}$ (each row being a uniform sample from the unit hypersphere) and additive Gaussian noise $e \in \mathbb{R}^n$, $e \sim \mathcal{N}(0, \sigma_0^2 \boldsymbol{I})$. For the Bayesian approach we also need a prior on w that expresses our knowledge of the sparsity of w. We use the horseshoe prior for this which can be written as

$$p(w) = \prod_{i=1}^{d} \int \mathcal{N}(w_i; 0, \tau^2\lambda_i^2) C^+(\lambda_i; 0, 1) d\lambda_i$$

where $C^+(\lambda_i; 0, 1) = 2\pi^{-1}(1 + \lambda_i^2)^{-1}$ is a positive Cauchy distribution. This involves an intractable integral and so we are forced to sample from the joint posterior $p(w, \lambda|y, X)$ keeping only the w samples in order to obtain samples from $p(w|y, X)$. Specifically, this means that for HMC the unnormalized posterior target we use is given by

$$\begin{aligned} p(w, \lambda|y, X) &\propto p(y|w, \lambda, X)p(w|\lambda)p(\lambda) \\ &= \mathcal{N}(y; Xw, \sigma_0^2 \boldsymbol{I}) \prod_{i=1}^{d} \mathcal{N}(w_i; 0, \tau^2\lambda_i^2) C^+(\lambda_i; 0, 1) \end{aligned}$$

In our experiment, we used $n = 6$, $d = 64$, $\sigma_0 = 0.005$, $\tau = 0.01$. Note this means the HMC chains will be sampling from a $2 \times 64 = 128$ dimensional posterior. We used chain lengths of 20 accept/reject steps each consisting of 5 leapfrog steps. We trained the chain using maxELT for 10^3 iterations with a batch size of 128 and learning rate 10^{-2}. For evaluation we draw an additional 1000 test observations i.e. we keep w the same but we create an extra 1000 rows of X and sample a 1000 dimensional noise vector to give 1000 more scalar observations. We then calculate the marginal likelihood of the test observations using the posterior samples from HMC using

$$p(y^*|X^*, y, X) = \mathbb{E}_{p(w|y,X)}\big[p(y^*|w, X^*)\big]$$

The log marginal likelihood can be safely calculated from a batch of posterior samples using the 'logsumexp' numerically safe function. We compared against three baselines in this experiment. The first two baselines are parallel samplers (like our method) meaning independently for each sample, we first sample the initial distribution and then run 20 accept/reject HMC steps. For the first of these, we set all masses to the same scalar value and all stepsizes to the same scalar value and grid searched over the possible values of these two scalars. To assess the best combination in the grid, we created a validation set of 1000 extra observations and picked the combination with the largest validation set marginal log likelihood. For the second, we kept the masses constant (at the initialization point for maxELT) and trained a single scalar stepsize that is used for all steps in the HMC chain such that the average acceptance probability was 0.65. The final sampler is a sequential sampler meaning we first sample the initial distribution once and then run 5000 accept/reject burn in steps before finally running 10000 accept/reject steps taking a sample from each of these 10000 steps. We use NUTS with dual averaging for this sequential sampler as in the 2D distribution experiment. We set the dual averaging target to be $\delta = 0.5$ which tunes the stepsizes during the initial 5000 step burn in phase.

5. Deep Latent Gaussian Model Experiment

5.1. Experimental Details

In these experiments, we set the dimension of the latent variable z to be 32. For our HMC variational distribution, $q_\phi^{(T)}(z|x)$, we set $T = 30$ and use 5 leapfrog updates. We only consider training the step sizes here and leave all masses at 1. As there are multiple parameters being optimized jointly, we summarise the entire method in this section for clarity. The parameters being optimized are: θ - the decoder neural network parameters, ϕ - the HMC step sizes (a total of $30 \times 32 = 960$ scalar values), ψ - the encoder ($q_\psi^{(0)}(z|x)$) neural network parameters and s - the scale factor used to scale samples from $q_\psi^{(0)}(z|x)$. There are then 4 optimization objectives:

$$\mathcal{L}_1 = \mathbb{E}_{q_\phi^{(T)}(z|x)}\big[\log p_\theta(x, z)\big] \quad \text{maxELT objective} \tag{48}$$

$$\mathcal{L}_2 = \text{SKSD}\big(z_{1:N}^{(T)}, \text{score}(z)\big) \quad \text{SKSD between } q_\phi^{(T)}(z|x) \text{ and } p_\theta(z|x) \tag{49}$$

$$\mathcal{L}_3 = \mathbb{E}_{q_\psi^{(0)}(z|x)}\big[\log p_\theta(x, z) - \log q_\psi^{(0)}(z|x)\big] \quad \text{Standard ELBO} \tag{50}$$

$$\mathcal{L}_4 = \mathbb{E}_{z_{1:k}\sim q_\psi^{(0)}(z|x)}\left[\sum_{i=1}^{k}\left(\frac{\omega^i}{\sum_{j=1}^{k}\omega^j}\right)^2 \log \omega^i\right],\ \omega^i = \frac{p_\theta(x, z^i)}{q_\psi^{(0)}(z^i|x)} \quad \text{DReG IWAE objective} \tag{51}$$

Using these 4 objectives, we then follow Algorithm 1 during training. In our experiments we set $I_1 = 10^5$ and $I_2 = 5 \times 10^4$. We have introduced pre-training steps before starting HMC optimization as these updates are very quick to do and make sure the HMC optimization has a reasonable starting point. We see that when $\alpha = 0$, we train ψ by maximising (50) which is equivalent to minimising the $\alpha = 0$ divergence with the target. When $\alpha = 1$ we use (51) as the objective for ψ which is the doubly reparameterized version of the IWAE objective (Tucker et al., 2019). This is equivalent to minimizing the $\alpha = 1$ divergence with the target, as shown by Hernández-Lobato (2016). For the case where we do not use the SKSD then we simply omit s from our model and omit the updates to s from Algorithm 1. For the VAE and IWAE baselines we simply run the pre-training steps for the full 1.5×10^5 steps using the $\alpha = 0$ or $\alpha = 1$ updates respectively.

Algorithm 1 DLGM Training Algorithm

Input: Initial θ_0, ψ_0, ϕ_0 and s_0, number of iterations I_1 and I_2, data batch size N_1, sample batch size N_2
Define score$(x) =$ stop_gradient$(\nabla_z \log p_\theta(x, z))$
 # Pre-training steps
for $i = 1$ **to** I_1 **do**
  if $\alpha = 0$ **then**
    $\theta_i \leftarrow$ Adam_update$(\theta_{i-1}, \nabla_{\theta_{i-1}} \mathcal{L}_3, i)$
    $\psi_i \leftarrow$ Adam_update$(\psi_{i-1}, \nabla_{\psi_{i-1}} \mathcal{L}_3, i)$
  else if $\alpha = 1$ **then**
    $\theta_i \leftarrow$ Adam_update$(\theta_{i-1}, \nabla_{\theta_{i-1}} \mathcal{L}_4, i)$
    $\psi_i \leftarrow$ Adam_update$(\psi_{i-1}, \nabla_{\psi_{i-1}} \mathcal{L}_4, i)$
  end if
end for
$s_{I_1-1} \leftarrow s_0$
$\phi_{I_1-1} \leftarrow \phi_0$
 # HMC-training steps
for $i = I_1$ **to** $I_1 + I_2$ **do**
  $x \leftarrow$ minibatch of size N_1
  for $j = 1$ **to** N_1 **do** *# These loops are vectorized in practice*
    $\mu_j \leftarrow$ mean of $q^{(0)}_{\psi_{i-1}}(z|x_j)$
    for $k = 1$ **to** N_2 **do**
      $z^{(0)}_{jk} \sim q^{(0)}_{\psi_{i-1}}(z|x_j)$
      $z^{(0)'}_{jk} \leftarrow s_{i-1}(z^{(0)}_{jk} - \mu_j) + \mu_j$
      $z^{(T)}_{jk} \leftarrow \text{HMC}_{\phi_{i-1}}(z^{(0)'}_{jk}, \text{score}(x_j))$
    end for
  end for
  $\theta_i \leftarrow$ Adam_update$(\theta_{i-1}, \nabla_{\theta_{i-1}} \mathcal{L}_1, i)$
  $\phi_i \leftarrow$ Adam_update$(\phi_{i-1}, \nabla_{\phi_{i-1}} \mathcal{L}_1, i)$
  $s_i \leftarrow$ Adam_update$(s_{i-1}, \nabla_{s_{i-1}} \mathcal{L}_2, i)$
  if $\alpha = 0$ **then**
    $\psi_i \leftarrow$ Adam_update$(\psi_{i-1}, \nabla_{\psi_{i-1}} \mathcal{L}_3, i)$
  else if $\alpha = 1$ **then**
    $\psi_i \leftarrow$ Adam_update$(\psi_{i-1}, \nabla_{\psi_{i-1}} \mathcal{L}_4, i)$
  end if
end for

5.2. Effectiveness of the Scaling

To demonstrate the scaling's effectiveness, we run our optimization scheme on the MNIST dataset for a range of fixed scale factors and then compute $\mathbb{E}_{p(z|x)}\big[\log p_\theta(x, z)\big] - \mathbb{E}_{q^{(T)}_\phi(z|x)}\big[\log p_\theta(x, z)\big]$ with samples from $p(z|x)$ found through HAIS (Sohl-Dickstein and Culpepper, 2012). The results for $\alpha = 0$ and $\alpha = 1$ are shown in Figure 1. For small scales, the metric is negative implying $q^{(T)}_\phi(z|x)$ is oversampling high probability regions of the target with a higher scale factor alleviating this issue. Figure 1 also shows the scale found by SKSD training when this is included in the optimization run, we see it

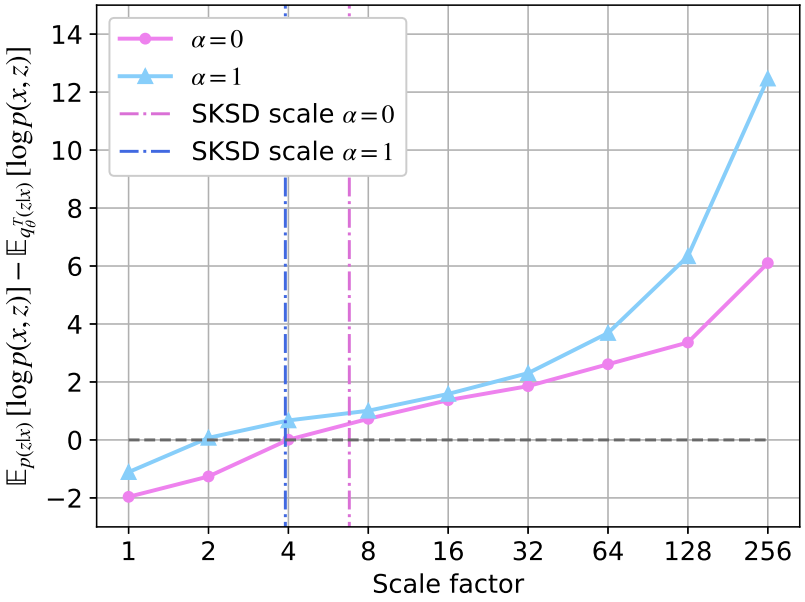


Figure 1. $\mathbb{E}_{p(z|x)}\left[\log p_\theta(x,z)\right] - \mathbb{E}_{q_\phi^{(T)}(z|x)}\left[\log p_\theta(x,z)\right]$ averaged over 200 randomly chosen MNIST test images for a range of fixed scalings used during training. The ground truth posterior $p(z|x)$ is estimated using 100 HAIS samples. The final scales found when running the training with the SKSD are also plotted.

can find an appropriate scale factor that is large enough to prevent this pathology whilst also ensuring stable performance.

5.3. Deep Latent Gaussian Model paired t-tests

We carry out a paired t-test for each model pairing with the null hypothesis that the two population means of the log-likelihoods $\log p(x)$ are equal. Log-likelihood values are paired between models by observed data point x. The p-values for the tests on the MNIST dataset are given in Table 3 and the p-values for the Fashion MNIST dataset are given in Table 4. A **0** value represents that the p-value is numerically indistinguishable from 0. We use different letters to denote different models: we denote VAE, DReG-IWAE, maxELT $\alpha = 0$, maxELT $\alpha = 1$, maxELT $\alpha = 0$ SKSD , maxELT $\alpha = 1$ SKSD, (Hoffman, 2017), (Ruiz and Titsias, 2019), (Salimans et al., 2015) and (Caterini et al., 2018) with a, b, c, d, e, f, g, h, i and j respectively.

Table 3. p-values for paired t-tests on test log-likelihood values on the MNIST dataset.

	b	c	d	e	f	g	h
a	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	-	**4.93e-18**	**0**	**0**	**0**	**0**	**0**
c	-	-	**0**	**0**	**0**	**0**	**9.84e-252**
d	-	-	-	**1.07e-115**	**1.50e-104**	**2.04e-173**	0.7038
e	-	-	-	-	0.2405	**1.45e-14**	**3.70e-108**
f	-	-	-	-	-	**2.95e-19**	**3.27e-104**
g	-	-	-	-	-	-	**1.25e-179**

Table 4. p-values for paired t-tests on test log-likelihood values on the Fashion MNIST dataset.

	b	c	d	e	f	g	h	i	j
a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	-	**2.51e-22**	**1.25e-94**	**2.12e-205**	**0**	**4.54e-217**	**3.38e-36**	3.33e-1	**1.47e-9**
c	-	-	**6.46e-42**	**6.48e-124**	**0**	**5.90e-129**	**4.32e-114**	**4.87e-13**	**4.75e-5**
d	-	-	-	**8.45e-28**	**1.05e-262**	**2.04e-40**	**8.11e-231**	**1.13e-64**	**2.50e-51**
e	-	-	-	-	**1.35e-135**	**4.81e-4**	**0**	**4.90e-138**	**3.49e-128**
f	-	-	-	-	-	**1.27e-79**	**0**	**0**	**0**
g	-	-	-	-	-	-	**0**	**2.78e-158**	**4.04e-144**
h	-	-	-	-	-	-	-	**1.06e-29**	**7.63e-58**
i	-	-	-	-	-	-	-	-	**1.31e-5**

6. Molecular Configurations Sampling Experiments

6.1. Details about the setup

All the RNVP models we used as initial distributions have the same architecture. They consist of five coupling blocks composed of two alternating coupling layers. The scale and shift within the coupling layers is given by fully connected networks having three layers with 128 hidden units each. Before each coupling block, we applied activation normalization (Kingma and Dhariwal, 2018). The models where trained for 30000 iterations with their respective objectives.

For use in scaling the initial distribution samples, we use an empirical estimate of the normalizing flow sample mean using 10^5 samples. To calculate the SKSD metric, to save computation, we use fixed vectors for what (Gong et al., 2021) refer to as $\boldsymbol{g}_r$ instead of optimizing them. This was because we found $\boldsymbol{g}_r$ ended up very close to one-hot vectors during optimization anyway so this approximation does not make a large difference to the method.

Training the RNVP model with the α-divergence when $\alpha = 0$ was achieved by using the reverse KL-divergence as a training objective (Papamakarios et al., 2021; Hernández-Lobato et al., 2016).

As baselines for tuning HMC parameters we used a grid search as well as training the acceptance probability. For the grid search, 25 different parameter settings were tested for each initial distribution. In each setting the step size and log mass is held constant over all HMC layers but 5 different step size constants are tested in combination with 5 different log mass constants giving 25 total combinations. The metric used to find the best combination was the median KL-divergence over all 60 marginals computed using 10^4 samples against the 10^5 molecular dynamics training data. When training the acceptance probability, the log mass was held at 0 for all layers and the step size was a constant for all layers with this step size constant being adjusted so that the average acceptance probability was 0.65. The constant was updated each training iteration with an update of the form $\epsilon_{t+1} = \epsilon_t - a_t(0.65 - \overline{p}_\text{a})$, where $\overline{p}_\text{a}$ is the average acceptance probability and a_t being a parameter decreasing according to the Robbins-Monro conditions (Robbins and Monro, 1951).

When tuning the HMC hyperparameter with our method, i.e. maxELT and maxELT & SKSD, we did so by performing 1000 iterations with the Adam optimizer.

When testing the models, the KL-divergences of the Ramachandran plots were computed from the histograms and for the one-dimensional marignals they were obtained by first computing a kernel density estimate for each marginal using a Gaussian kernel with bandwidth chosen using Scott's rule and then finding the KL-divergence between the kernel density estimates using numerical integration.

6.2. Coordinate groups and model comparison

The coordinate transformation, introduced in (Noé et al., 2019), which we used, splits the feature dimensions in four different groups: 17 bond angles, 17 bond lengths, 17 dihedral angles, and 9 Cartesian coordinates, adding up to 60 dimensions in total. The different coordinate types are illustrated and explained in Figure 2. Due to their differing physical meaning and relevance, they follow different distributions. For each group, three sample marginals are shown in Figure 3,

Table 5. Mean of the KL-divergences of the one-dimensional marginals of the four models with respect to the ground truth for each of the proposals. **Bold** values indicate that this model has the lowest average KL-divergence for the given proposal.

	maxELT	maxELT & SKSD	Grid search	$\overline{p}_\text{a} = 0.65$
$\alpha = 0$	0.0494	**0.0486**	0.0498	0.0496
$\alpha = 1$	0.0513	0.0460	0.0350	**0.0345**
ML	**0.00520**	0.00657	0.0121	0.0391

Table 6. Median of the KL-divergences of the one-dimensional marginals of the four models with respect to the ground truth for each of the proposals. **Bold** values indicate that this model has the lowest median KL-divergence for the given proposal.

	maxELT	maxELT & SKSD	Grid search	$\overline{p}_\text{a} = 0.65$
$\alpha = 0$	**0.00156**	0.00159	0.00166	0.00163
$\alpha = 1$	0.00320	**0.00152**	0.00205	0.00197
ML	0.00136	**0.00134**	0.00139	0.00142

Figure 4, and Figure 5. We used the MacCallum lab's implementation for our coordinate transform which can be found at `https://github.com/maccallumlab/BoltzmannGenerator`.

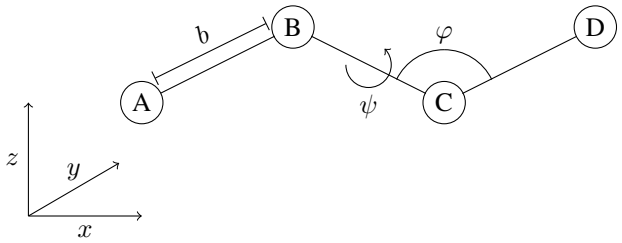


Figure 2. Illustration of molecular coordinates. The state of the molecule can be described through the Cartesian coordinates, i.e. x, y, and z, of each of the four atoms A, B, C, and D. Alternatively, internal coordinates, i.e. bond lengths, bond angles, and dihedral angles, can be used. Here, the bond length b is the distance between atom A and B, the bond angle φ is the angle between the bonds between B and C as well as C and D, and the dihedral angle ψ is the angle between the plans spanned by A, B, and C as well as B, C, and D. Combining Cartesian and internal coordinates is also possible, as (Noé et al., 2019) did.

The bond lengths and angles follow mostly unimodal, almost Gaussian, distributions which is due to the regular vibrations of the atoms within the molecule. The dihedral angles can have multiple modes while the Cartesian coordinates are quite irregular.

As evaluating the KL-divergences of low-dimensional distributions is much easier than approximating it for higher dimensional distributions, we compare our models using the KL-divergences of the Ramachandran plots and all the one-dimensional marginals. Ramachandran plots are an important tool to visualize how proteins fold locally and, hence, their KL-divergence with respect to the ground truth, i.e. the test set determined through a MD simulation, is an important performance measure. While the KL-divergences are given in the main paper, the respective plots are shown in Figure 6–8.

For the one-dimensional marginals, we computed the KL-divergences with respect to the ground truth. The result are 60 values which we compare to each other in two ways. First, we computed the mean and the median of the KL-divergences, which are given in Table 5 and 6. In most cases, our methods, i.e. maxELT and maxELT & SKSD clearly outperform the baselines. Second, we performed a paired one-sided Wilcoxon rank test for pairs of models to assess the statistical significance of our results. The alternative hypothesis for this test is that one model has consistently lower KL-divergences with respect to the ground truth than the other. The resulting p-values are listed in Table 7, 8, and 9. Overall, we see again that our method of tuning HMC parameters, especially maxELT & SKSD, is competitive or outperforms the baselines, i.e. the grid search and the training to get an average acceptance probability of 0.65.

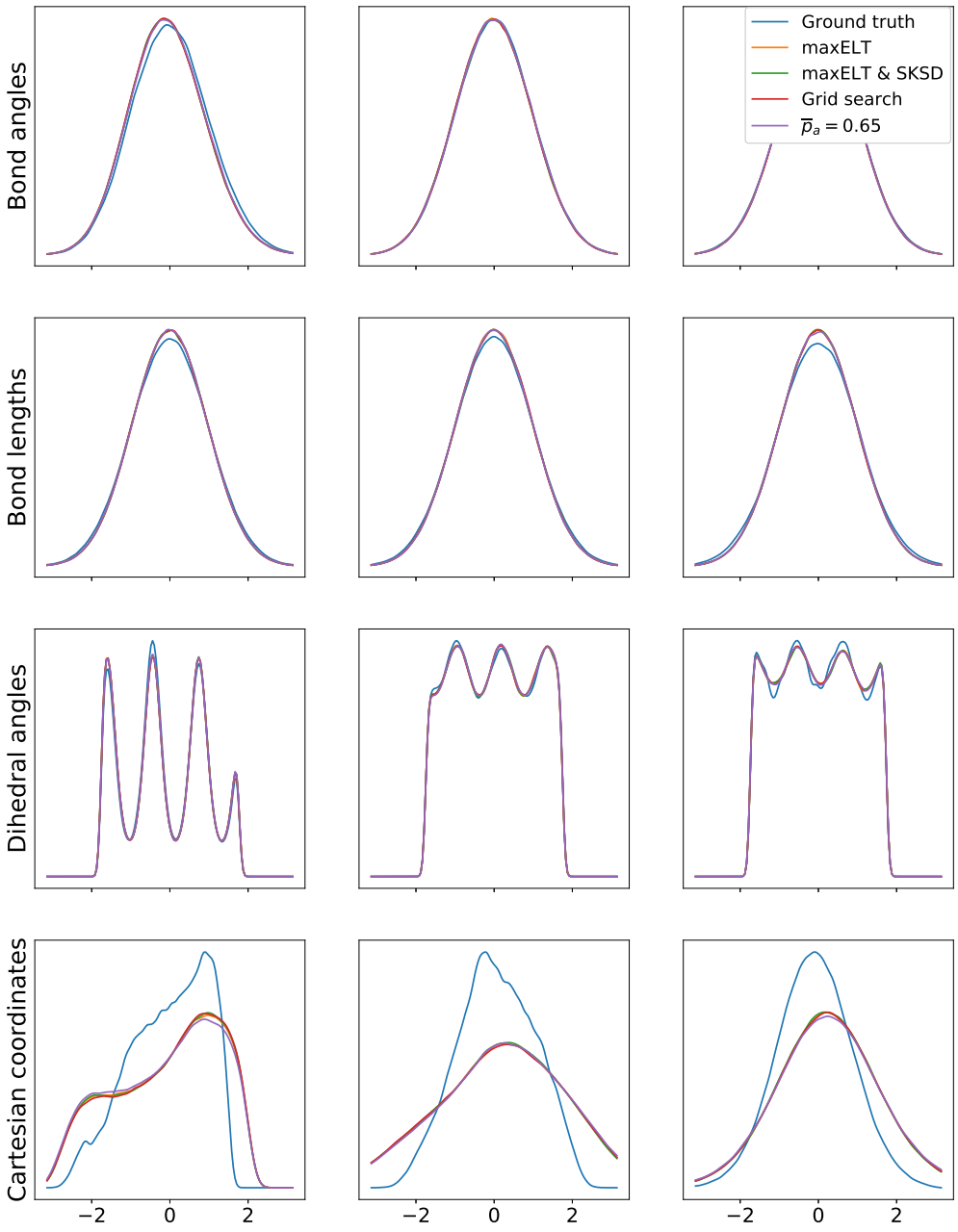


Figure 3. Sample distributions of marginals from the four coordinate groups. The graphs compare the ground truth with models having a RNVP as initial distribution followed by 50 HMC steps. The RNVP was trained with the $\alpha = 0$-divergence and the HMC parameters were tuned with the indicated methods.

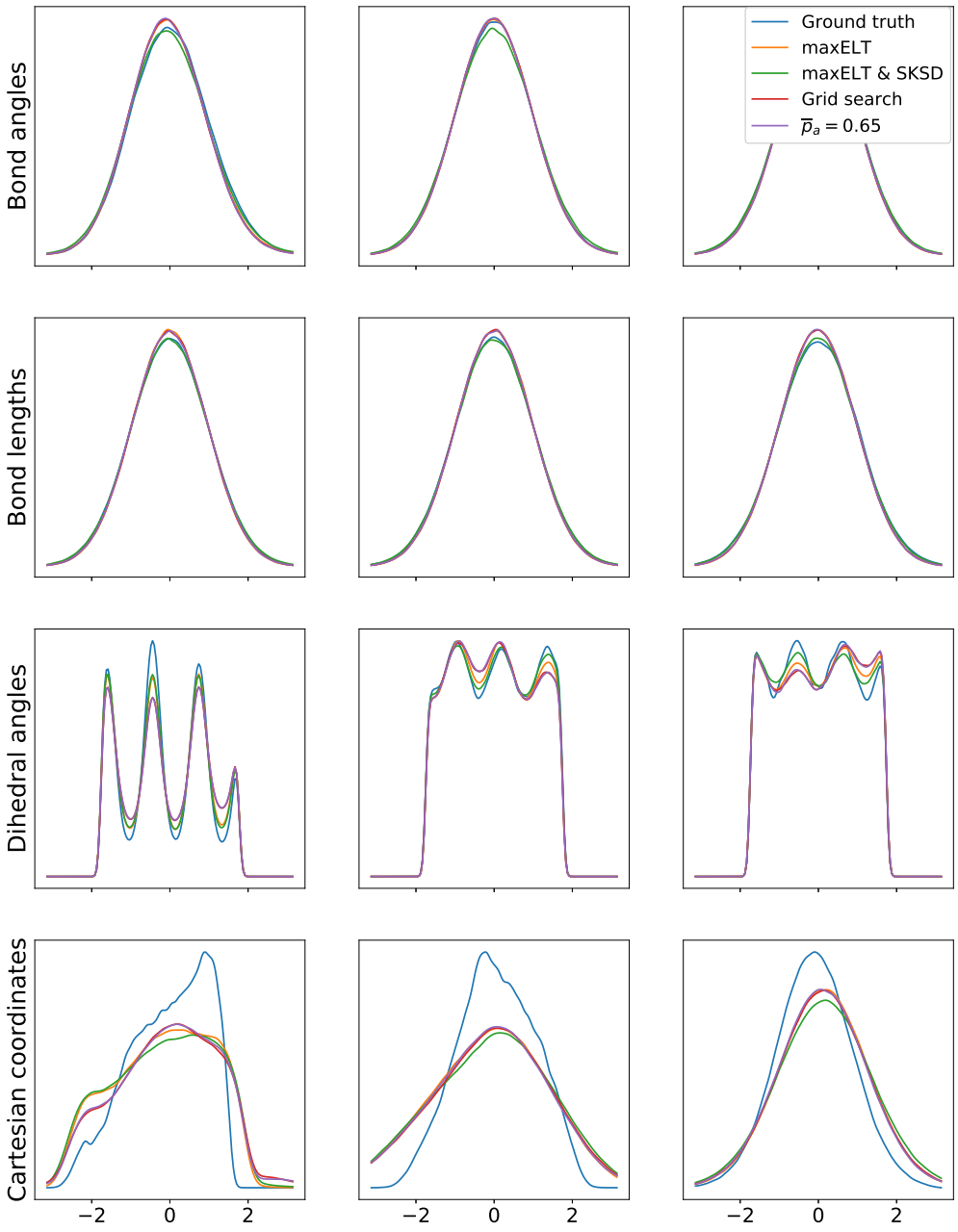


Figure 4. Sample distributions of marginals from the four coordinate groups. The graphs compare the ground truth with models having a RNVP as initial distribution followed by 50 HMC steps. The RNVP was trained with the $\alpha = 1$-divergence and the HMC parameters were tuned with the indicated methods.

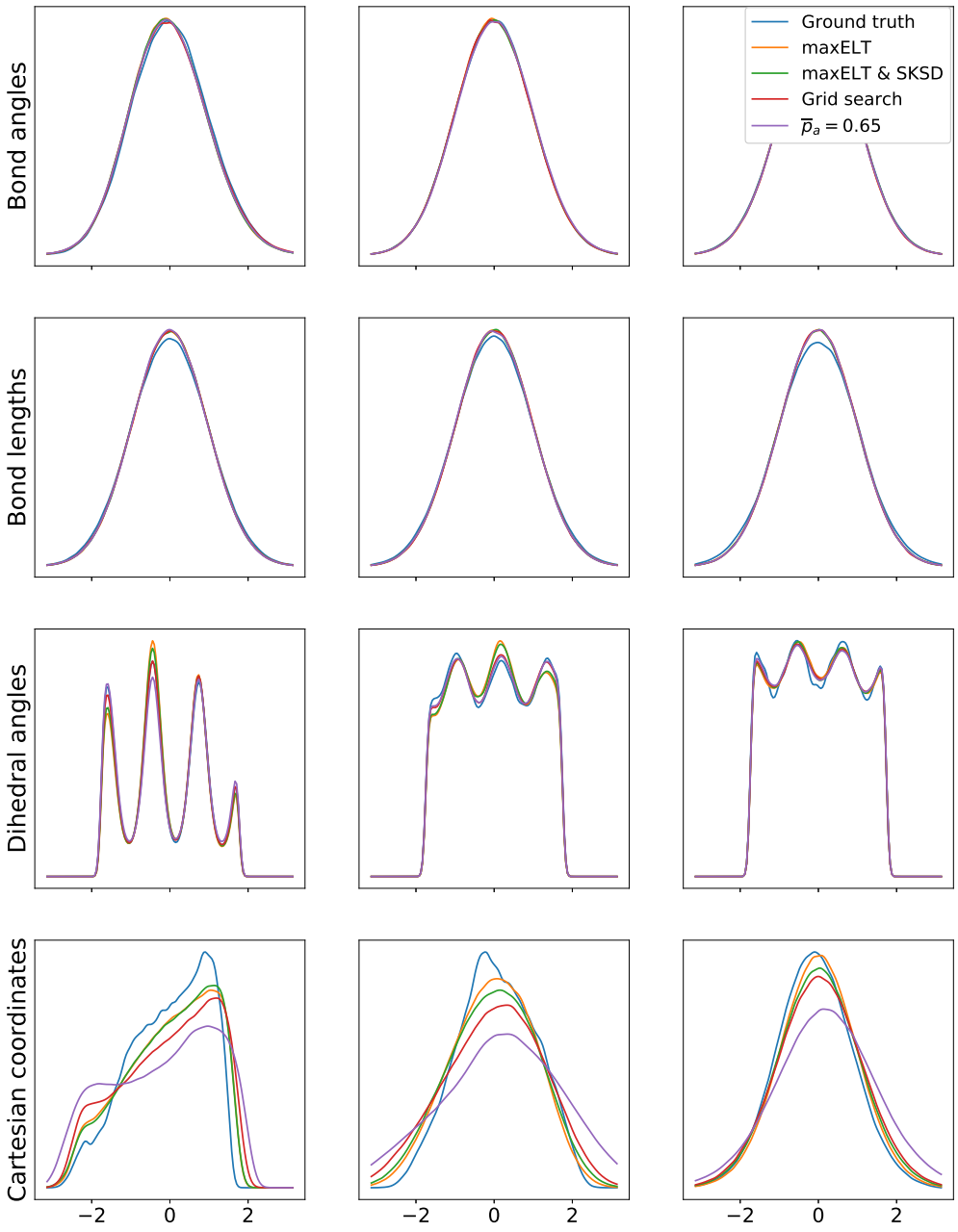
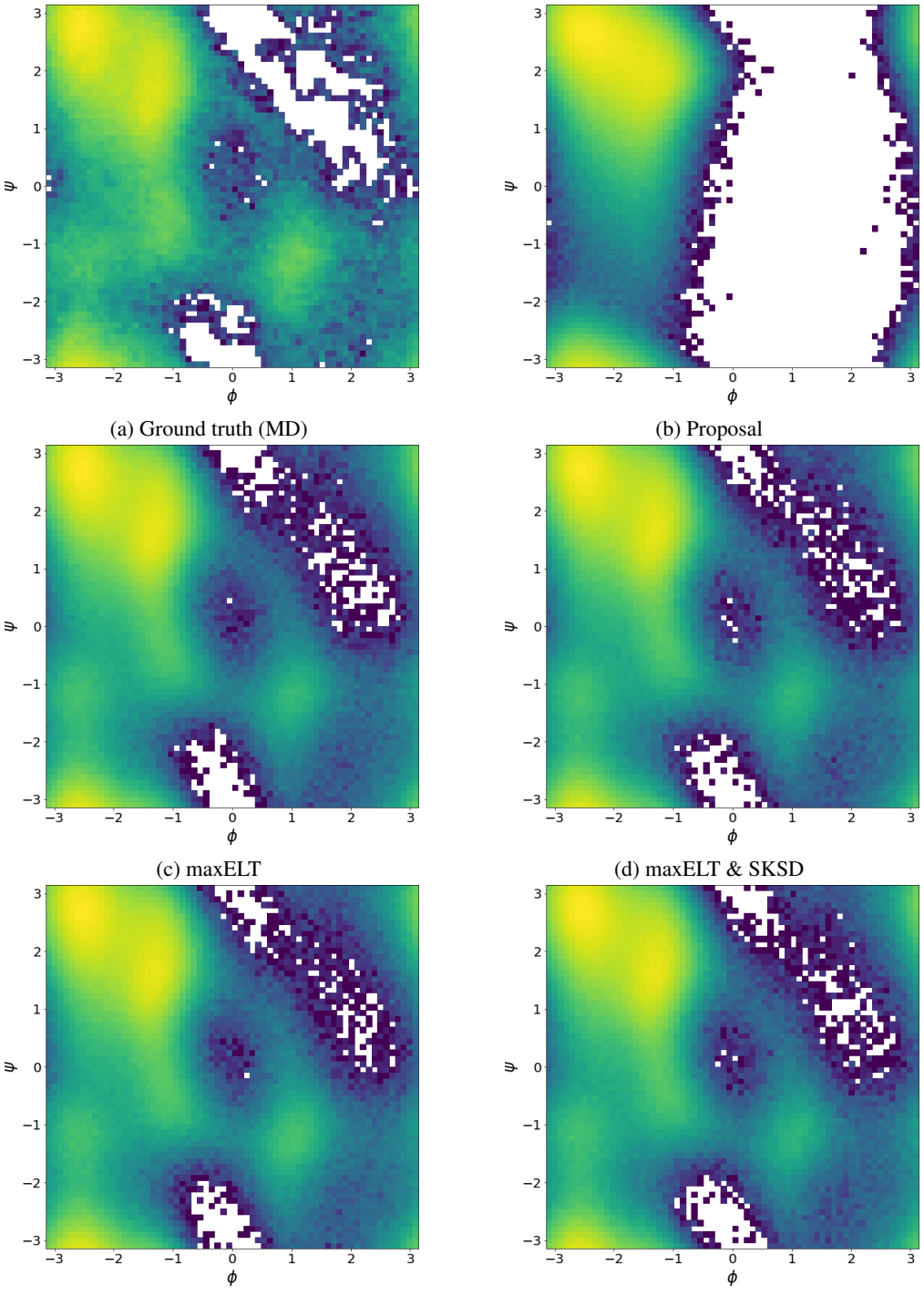


Figure 5. Sample distributions of marginals from the four coordinate groups. The graphs compare the ground truth with models having a RNVP as initial distribution followed by 50 HMC steps. The RNVP was trained via maximum likelihood and the HMC parameters were tuned with the indicated methods.



(a) Ground truth (MD)

(b) Proposal

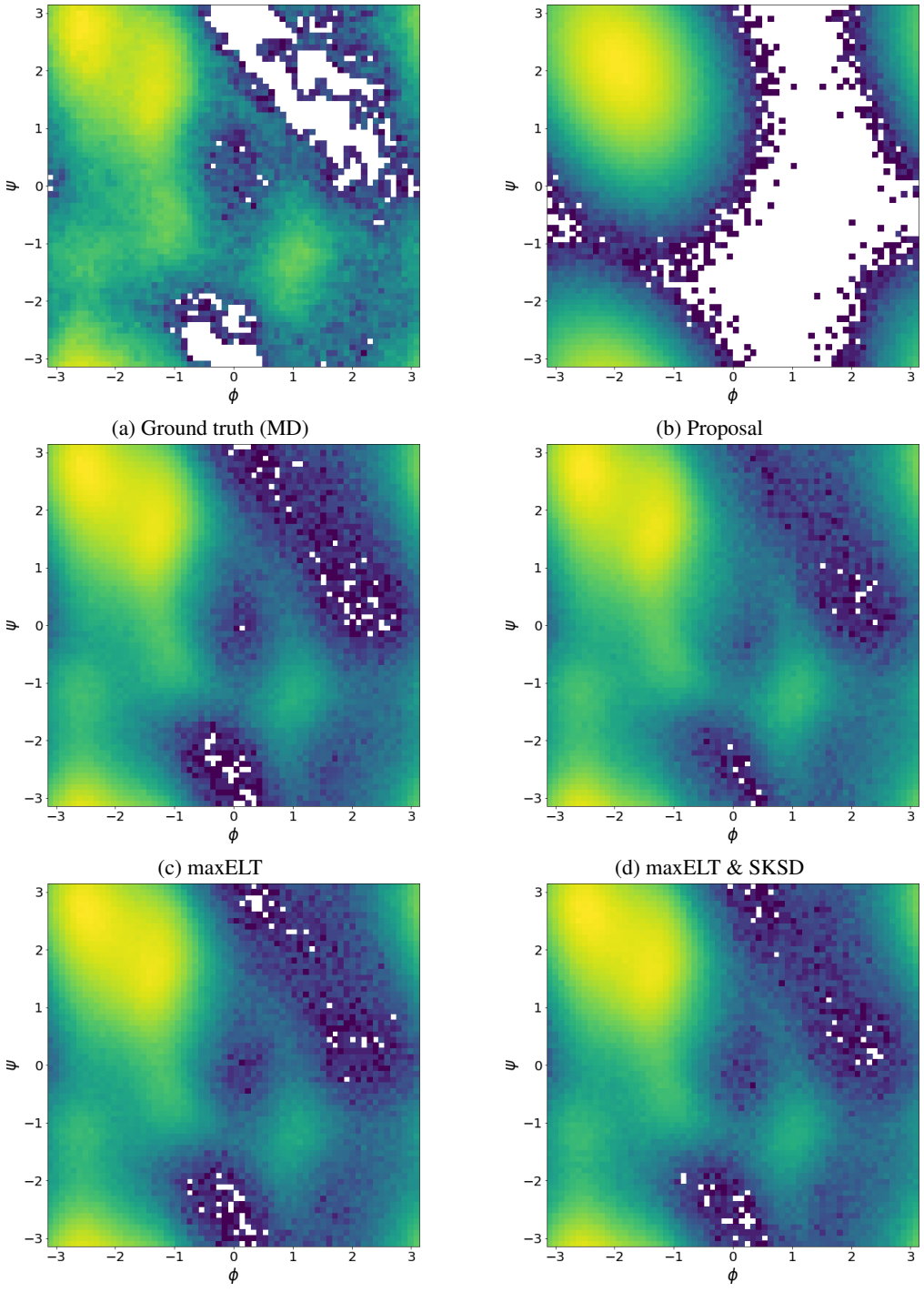
(c) maxELT

(d) maxELT & SKSD

(e) Acceptance probability

(f) Grid search

Figure 6. Ramachandran plots of Alanine Dipeptide. (a) shows the ground truth determined with a MD simulation, (b) the proposal trained with the $\alpha = 0$-divergence, and (c)–(f) HMC models using the proposal from (b) which were tuned with different procedures.



(a) Ground truth (MD)

(b) Proposal

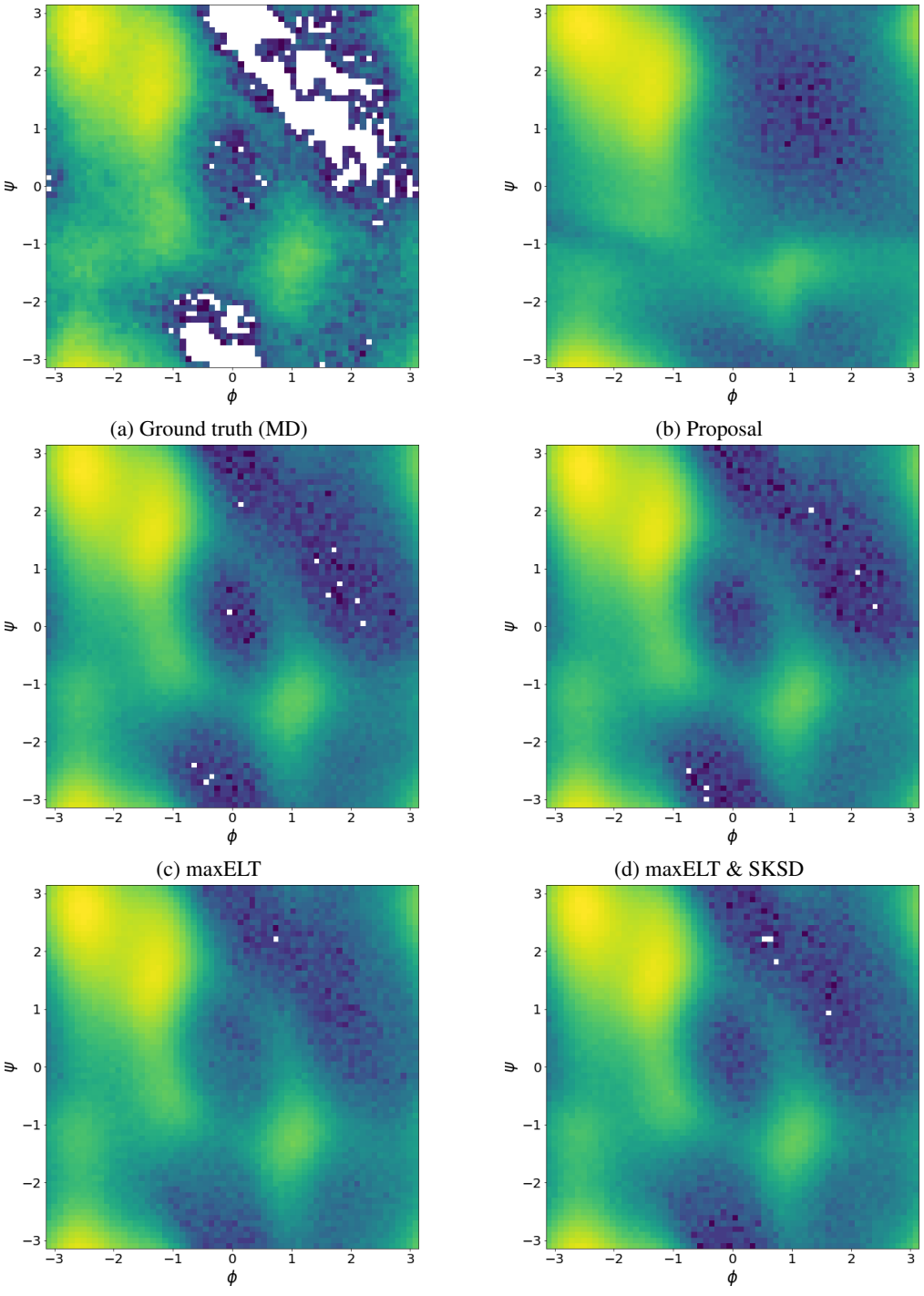
(c) maxELT

(d) maxELT & SKSD

(e) Acceptance probability

(f) Grid search

Figure 7. Ramachandran plots of Alanine Dipeptide. (a) shows the ground truth determined with a MD simulation, (b) the proposal trained with the $\alpha = 1$-divergence, and (c)–(f) HMC models using the proposal from (b) which were tuned with different procedures.



(a) Ground truth (MD)

(b) Proposal

(c) maxELT

(d) maxELT & SKSD

(e) Acceptance probability

(f) Grid search

Figure 8. Ramachandran plots of Alanine Dipeptide. (a) shows the ground truth determined with a MD simulation, (b) the proposal trained with maximum likelihood, and (c)–(f) HMC models using the proposal from (b) which were tuned with different procedures.

Table 7. P-values of the Wilcoxon test comparing models with initial distribution trained with $\alpha = 0$-divergence. The alternative hypothesis is that the model in the row has smaller KL-divergences of the marginals than the model in the column. The smaller the p-value is, the more likely the alternative hypothesis is true.

	maxELT	maxELT & SKSD	Grid search	$\overline{p}_{\text{a}} = 0.65$
maxELT	-	0.83	0.58	0.61
maxELT & SKSD	0.17	-	0.0030	0.39
Grid search	0.42	1.0	-	0.68
$\overline{p}_{\text{a}} = 0.65$	0.39	0.61	0.32	-

Table 8. P-values of the Wilcoxon test comparing models with initial distribution trained with $\alpha = 1$-divergence. The alternative hypothesis is that the model in the row has smaller KL-divergences of the marginals than the model in the column. The smaller the p-value is, the more likely the alternative hypothesis is true.

	maxELT	maxELT & SKSD	Grid search	$\overline{p}_{\text{a}} = 0.65$
maxELT	-	1.0	1.0	0.61
maxELT & SKSD	4.5e-4	-	0.0030	0.0040
Grid search	0.0041	1.0	-	0.72
$\overline{p}_{\text{a}} = 0.65$	0.0040	1.0	0.28	-

6.3. Scale progression during training

To improve the overlap of the initial distribution with the target, we scale the former and learn the scaling parameter through the SKSD. Figure 9 shows the progression of the scale parameter during training for different initial distributions. The distributions trained using the α-divergence with $\alpha = 0$ tend to be mode seeking. Hence, we expect the scale to be larger than 1 so the proposal covers the whole target distribution, and this is indeed the case. With $\alpha = 2$ the distribution tends to be mode covering, so it needs to be shrunken. The model trained with maximum likelihood already has a low KL-divergence (see e.g. Figure 6 in the main text) so there is not much modification needed, i.e. the scale is close to 1.

Table 9. P-values of the Wilcoxon test comparing models with initial distribution trained with maximum likelihood. The alternative hypothesis is that the model in the row has smaller KL-divergences of the marginals than the model in the column. The smaller the p-value is, the more likely the alternative hypothesis is true.

	maxELT	maxELT & SKSD	Grid search	$\overline{p}_{\text{a}} = 0.65$
maxELT	-	0.94	0.41	0.30
maxELT & SKSD	0.058	-	0.010	5.1e-5
Grid search	0.59	0.99	-	0.0074
$\overline{p}_{\text{a}} = 0.65$	0.70	1.0	0.99	-

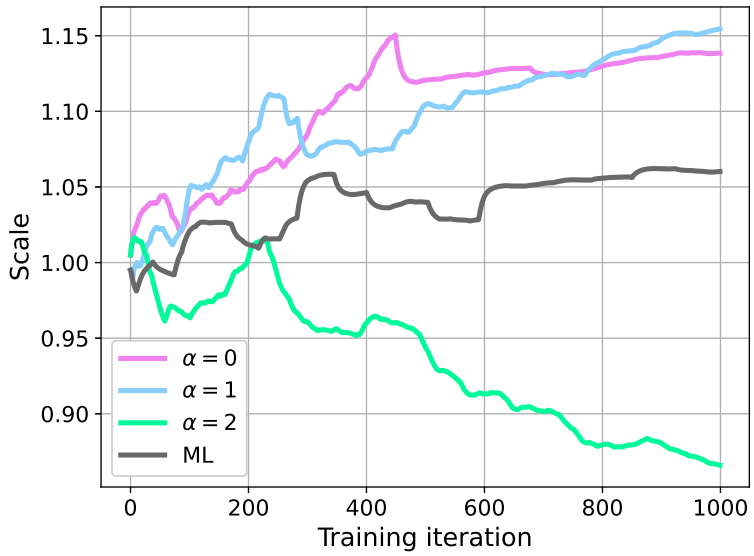


Figure 9. Progression of scale factor for maxELT & SKSD models during training. RNVP models trained by maximum likelihood and the α-divergence with $\alpha = 0, 1, 2$ was used as initial distribution.

References

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

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

J. M. Hernández-Lobato. Alpha-divergence minimization for Bayesian deep learning. *Bayesian Deep Learning Workshop NIPS 2016, Centre Convencions Internacional Barcelona, Barcelona, Spain*, 2016.

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

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

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

D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

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

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

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research (JMLR)*, 2021.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 33(3):400–407, 1951.

G. Roeder, Y. Wu, and D. Duvenaud. Sticking the landing: An asymptotically zero variance gradient estimator for variational inference. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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

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

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

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