Stochastic Normalizing Flows

Hao Wu [1] **Jonas Köhler** [2] **Frank Noé** [2 3 4]

Abstract

The sampling of probability distributions specified up to a normalization constant is an important problem in machine learning and statistical mechanics. Here we propose Stochastic Normalizing Flows (SNF) which combine trainable invertible networks with stochastic sampling methods such as Markov Chain Monte Carlo (MCMC) or Langevin Dynamics (LD). We show that stochasticity overcomes expressivity limitations of invertible networks, whereas the trainable invertible networks improve sampling efficiency over pure MCMC/LD. By invoking ideas from non-equilibrium statistical mechanics we derive an efficient training procedure by which both the sampler's and the flow's parameters can be optimized end-to-end, and exact importance weights of flow samples can be computed without having to marginalize out the randomness of the stochastic blocks. We illustrate the representational power, sampling efficiency and asymptotic correctness of SNFs on benchmarks and a molecular model.

1. Introduction

A common problem in machine learning and statistics with important applications in physics is the generation of asymptotically unbiased samples from a target distribution defined up to a normalization constant: $\mu_X(\mathbf{x}) \propto \exp(-u(\mathbf{x}))$.

Sampling of such unnormalized distributions is often done with Markov Chain Monte Carlo (MCMC) or other stochastic sampling methods (Frenkel & Smit, 2001). Such methods are asymptotically unbiased, but might get stuck in local energy minima due to inefficient proposals.

Normalizing flows (NFs) (Tabak et al., 2010; Tabak & Turner, 2013; Dinh et al., 2014; Rezende & Mohamed, 2015; Dinh et al., 2016; Papamakarios et al., 2019) combined with importance sampling methods are an alternative approach that enjoys growing interest in molecular and material sciences and nuclear physics (Müller et al., 2018; Li & Wang, 2018; Noé et al., 2019; Köhler et al., 2019; Albergo et al., 2019; Nicoli et al., 2019). NFs are learnable invertible functions pushing forward a probability density over a latent or "prior" space Z towards target space the X. Utilizing the change of variable rule these models provide exact densities of generated samples allowing to train them by either maximizing the likelihood on data (ML) or minimizing the Kullback-Leibler divergence (KL) towards a target distribution.

Let F_{ZX} be such a map and its inverse $F_{XZ} = F_{ZX}^{-1}$. We can consider it as composition of T invertible transformation layers $F_0, ..., F_T$ with intermediate states $\mathbf{y}_t$ given by:

$$\mathbf{y}_{t+1} = F_t(\mathbf{y}_t) \qquad \mathbf{y}_t = F_t^{-1}(\mathbf{y}_{t+1}) \tag{1}$$

We call Z (prior) samples $\mathbf{z} = \mathbf{y_0}$ and X (target) samples $\mathbf{x} = \mathbf{y_T}$. We suppose each transformation layer is differentiable with a Jacobian determinant $|\det \mathbf{J}_t(\mathbf{y})|$. This allows to apply the *change of variable* rule:

$$p_{t+1}(\mathbf{y}_{t+1}) = p_{t+1}\left(F_t(\mathbf{y}_t)\right) = p_t(\mathbf{y}_t)\left|\det \mathbf{J}_t(\mathbf{y}_t)\right|^{-1} \tag{2}$$

As we often work with log-densities, we abbreviate the log Jacobian determinant as: $\Delta S_t = \log|\det \mathbf{J}_t(\mathbf{y})|$. The log Jacobian determinant of the entire flow is defined by $\Delta S_{ZX} = \sum_t \Delta S_t(\mathbf{y}_t)$ and correspondingly ΔS_{XZ} for the inverse flow.

Unbiased sampling with Boltzmann Generators is particularly important for applications in physics and chemistry where unbiased expectation values are required (Li & Wang, 2018; Noé et al., 2019; Albergo et al., 2019; Nicoli et al., 2019). A Boltzmann generator utilizing an NF achieves this by (i) generating one-shot samples $\mathbf{x} \sim p_X(\mathbf{x})$ from the flow and (ii) a reweighing/resampling procedure respecting weights

$$w(\mathbf{x}) \propto \exp\left(-u_Z\left(F_{XZ}(\mathbf{x})\right) - \log \rho_X(\mathbf{x}) + \Delta S_{XZ}(\mathbf{x})\right),$$

[1]Tongji University, School of Mathematical Sciences, Shanghai, China [2]Freie Universität Berlin, Department of Mathematics and Computer Science, Berlin, Germany [3]Freie Universität Berlin, Department of Physics, Berlin, Germany [4]Rice University, Department of Chemistry, Houston, USA. Correspondence to: Frank Noé <frank.noe@fu-berlin.de>.

Second workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020), Virtual Conference

turning these one-shot samples into asymptotically unbiased samples. Reweighing/resampling methods utilized in this context are e.g. *Importance Sampling* (Müller et al., 2018; Noé et al., 2019) or *Neural MCMC* (Li & Wang, 2018; Albergo et al., 2019; Nicoli et al., 2019).

Training NFs are usually trained by minimizing the forward KL divergence $KL\left[p_X \| \mu_X\right]$ (energy minimization) and/or the backward KL divergence $KL\left[\mu_X \| p_X\right]$ (density estimation).

As can be shown[1], this corresponds to maximizing the forward / backward weights of samples drawn from p_X / μ_X.

Topological problems of NFs A major caveat of sampling with exactly invertible functions for physical problems are topological constraints (Falorsi et al., 2018; 2019). Problems arising from these constraints can be reduced by using mixtures of flows (Dinh et al., 2019; Cornish et al., 2019) or augmenting the base space (Dupont et al., 2019) at the cost of limiting expressibility or losing the exact density.

Contributions We show how NFs can be combined with intermediate stochastic sampling blocks into arbitrary sequences and how this sequence can be jointly optimized efficiently. This relaxes topological constraints and improve expressivity of utilized NFs. It further improves sampling efficiency over pure stochastic sampling as the flow's and sampler's parameters can be jointly optimized.

We apply the model to benchmark problems as well as to the recently introduced problem of asymptotically unbiased sampling of molecular structures with flows (Noé et al., 2019) and show that it significantly improves sampling the multi-modal torsion angle distribution.

2. Stochastic normalizing flows

A SNF is a sequence of T stochastic and deterministic transformations. We sample $\mathbf{z} = \mathbf{y}_0$ from the prior μ_Z, and generate a forward path $(\mathbf{y}_1, \ldots, \mathbf{y}_T)$ resulting in a proposal $\mathbf{y}_T$ (Fig. 1). Correspondingly, latent space samples can be generated by starting from a sample $\mathbf{x} = \mathbf{y}_T$ and invoking the backward path $(\mathbf{y}_{T-1}, \ldots, \mathbf{y}_0)$. The conditional forward / backward path probabilities are

$$\mathbb{P}_f(\mathbf{z} = \mathbf{y}_0 \to \mathbf{x} = \mathbf{y}_T) = \prod_{t=0}^{T-1} q_t(\mathbf{y}_t \to \mathbf{y}_{t+1}) \quad (3)$$

$$\mathbb{P}_b(\mathbf{x} = \mathbf{y}_T \to \mathbf{y}_0 = \mathbf{z}) = \prod_{t=0}^{T-1} \tilde{q}_t(\mathbf{y}_{t+1} \to \mathbf{y}_t) \quad (4)$$

where $\mathbf{y}_{t+1}|\mathbf{y}_t \sim q_t(\mathbf{y}_t \to \mathbf{y}_{t+1})$ and $\mathbf{y}_t|\mathbf{y}_{t+1} \sim \tilde{q}_t(\mathbf{y}_{t+1} \to \mathbf{y}_t)$ denote the forward / backward sampling

[1] All derivations can be found in the Suppl. Material.

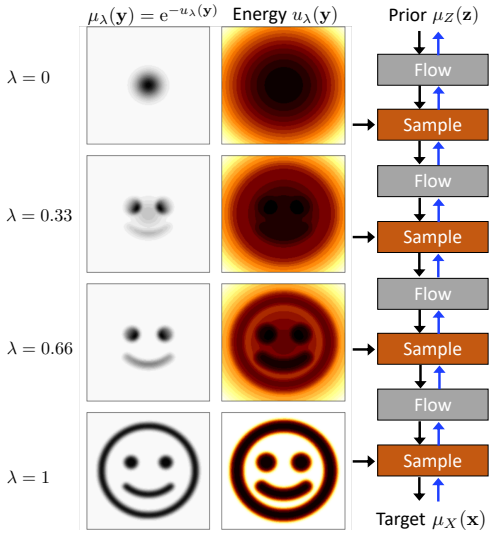

Figure 1. **Schematic for Stochastic Normalizing Flow (SNF).** An SNF transform a tractable prior $\mu_Z(\mathbf{z}) \propto \exp(-u_0(\mathbf{z}))$ to a complicated target distribution $\mu_X(\mathbf{x}) \propto \exp(-u_1(\mathbf{x}))$ by a sequence of deterministic invertible transformations (flows, grey boxes) and stochastic dynamics (sample, ochre) that sample with respect to a guiding potential $u_\lambda(\mathbf{x})$. SNFs can be run in forward mode (black) and reverse mode (blue).

density at step t respectively. If step t is a deterministic transformation F_t this simplifies as

$$\mathbf{y}_{t+1} \sim \delta\left(\mathbf{y}_{t+1} - F_t(\mathbf{y}_t)\right), \quad \mathbf{y}_t \sim \delta\left(\mathbf{y}_t - F_t^{-1}(\mathbf{y}_{t+1})\right).$$

While the probability of generating a sample $\mathbf{x}$ from a NF can be computed by Eq. (2), this is not possible for SNFs. The marginal probability of generating $\mathbf{x}$ involves integrating over all paths that end in $\mathbf{x}$:

$$p_X(\mathbf{x}) = \int \mu_Z(\mathbf{y}_0)\mathbb{P}_f(\mathbf{y}_0 \to \mathbf{y}_T)\, \mathrm{d}\mathbf{y}_0 \cdots \mathrm{d}\mathbf{y}_{T-1}. \quad (5)$$

This integral is generally intractable, thus a feasible training method must avoid Eq. (5).

Following (Nilmeier et al., 2011), we can draw samples $\mathbf{x} \sim \mu_X(\mathbf{x})$ by running Metropolis-Hastings moves in the path-space of $(\mathbf{z} = \mathbf{y}_0, ..., \mathbf{y}_T = \mathbf{x})$ if we select the unconditional backward path probability $\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x} \to \mathbf{z})$ as the target distribution and the unconditional forward path probability $\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z} \to \mathbf{x})$ as the proposal density. Since we sample paths independently, it is simpler to assign an unnormalized importance weight proportional to the acceptance ratio to each sample path from $\mathbf{z} = \mathbf{y}_0$ to $\mathbf{x} = \mathbf{y}_T$:

$$w(\mathbf{z} \to \mathbf{x}) = \mathrm{e}^{u_Z(\mathbf{z}) - u_X(\mathbf{x}) + \sum_t \Delta S_t(\mathbf{y}_t)} \propto \frac{\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x} \to \mathbf{z})}{\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z} \to \mathbf{x})} \quad (6)$$

where

$$\Delta S_t = \log \frac{\tilde{q}_t(\mathbf{y}_{t+1} \to \mathbf{y}_t)}{q_t(\mathbf{y}_t \to \mathbf{y}_{t+1})} \tag{7}$$

denotes the forward-backward probability ratio of step t, and corresponds to the usual change of variable formula in NF for deterministic transformation steps. These weights allow asymptotically unbiased sampling and training of SNFs while avoiding eq. (5). By changing denominator and numerator in (6) we can alternatively obtain the backward weights $w(\mathbf{x} \to \mathbf{z})$.

SNF training As in NFs, the parameters of a SNF can be optimized by minimizing the Kullback-Leibler divergence between the forward and backward path probabilities, or alternatively maximizing forward and backward path weights as long as we can compute ΔS_t:

$$\begin{aligned} J_{\mathrm{KL}} &= \mathbb{E}_{\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z}\to\mathbf{x})}\left[-\log w(\mathbf{z}\to\mathbf{x})\right] \\ &= \mathrm{KL}\left(\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z}\to\mathbf{x})||\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x}\to\mathbf{z})\right) + \text{const.} \end{aligned}$$

In the ideal case of $J_{\mathrm{KL}} = 0$, all paths have the same weight $w(\mathbf{z} \to \mathbf{x}) = 1$ and independent and identically distributed sampling of μ_X can be achieved. Accordingly, we can maximize the likelihood of the generating process on data drawn from μ_X by minimizing:

$$\begin{aligned} J_{\mathrm{ML}} &= \mathbb{E}_{\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x}\to\mathbf{z})}\left[-\log w(\mathbf{x}\to\mathbf{z})\right] \\ &= \mathrm{KL}\left(\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x}\to\mathbf{z})||\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z}\to\mathbf{x})\right) + \text{const.} \end{aligned}$$

Asymptotically unbiased sampling As stated in the theorem below, SNFs are Boltzmann Generators: We can generate asymptotically unbiased samples of $\mathbf{x} \sim \mu_X(\mathbf{x})$ by performing importance sampling or Neural MCMC using the path weight $w(\mathbf{z}_k \to \mathbf{x}_k)$ of each path sample k.

Theorem 1. *Let O be a function over $\mathcal{X}$. An asymptotically unbiased estimator is given by*

$$\mathbb{E}_{\mathbf{x}\sim\mu_X}\left[O(\mathbf{x})\right] \approx \frac{\sum_k w(\mathbf{z}_k \to \mathbf{x}_k)\, O(\mathbf{x}_k)}{\sum_k w(\mathbf{z}_k \to \mathbf{x}_k)}. \tag{8}$$

if paths are drawn from the forward path distribution $\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z} \to \mathbf{x})$.

3. Implementing SNFs in practice

We focus on the use of SNFs as samplers of $\mu_X(\mathbf{x})$ for problems where the target energy $u_X(\mathbf{x})$ is known. Stochastic blocks like MCMC / LD make updates of the current state $\mathbf{y}$ with respect to some potential $u_\lambda(\mathbf{y})$ such that they will asymptotically sample from $\mu_\lambda(\mathbf{y}_t) \propto \exp(-u_\lambda(\mathbf{y}_t))$. Here we define these intermediate potentials by interpolating between prior and target potentials using $\lambda \in [0, 1]$ as it is done in *annealed importance sampling* (Neal, 1998).

$$u_\lambda(\mathbf{y}) = (1-\lambda)u_Z(\mathbf{y}) + \lambda u_X(\mathbf{y}), \tag{9}$$

Deterministic Flow layers use trainable invertible networks to approximate the partial density transformation between adjacent λ steps. For training and reweighting, deterministic blocks can be treated the same as stochastic blocks by realizing:

$$\Delta S_t = \log\left|\det \mathbf{J}_t(\mathbf{y}_t)\right|. \tag{10}$$

Langevin dynamics Overdamped Langevin dynamics using an Euler discretization with time step ϵ_t is given by

$$\mathbf{y}_{t+1} = \mathbf{y}_t - \epsilon_t \nabla u_\lambda(\mathbf{y}_t) + \sqrt{2\epsilon_t/\beta}\boldsymbol{\eta}_t \tag{11}$$

where $p(\boldsymbol{\eta}_t) = \mathcal{N}(0, \mathbf{I})$. The backward step $\mathbf{y}_{t+1} \to \mathbf{y}_t$ is realized under these dynamics with the backward noise realization: $\tilde{\boldsymbol{\eta}}_t = \sqrt{\frac{\beta\epsilon_t}{2}}\left[\nabla u_\lambda(\mathbf{y}_t) + \nabla u_\lambda(\mathbf{x}_{t+1})\right] - \boldsymbol{\eta}_t$. The log path probability ratio is, (Nilmeier et al., 2011):

$$\Delta S_t = -\frac{1}{2}\left(\|\tilde{\boldsymbol{\eta}}_t\|^2 - \|\boldsymbol{\eta}_t\|^2\right)$$

Markov Chain Monte Carlo Any MCMC method with a forward and backward proposal densities $q_t = \tilde{q}_t$ that satisfy the detailed balance condition $e^{-u_\lambda(\mathbf{y}_t)}q_t(\mathbf{y}_t \to \mathbf{y}_{t+1}) = e^{-u_\lambda(\mathbf{y}_{t+1})}q_t(\mathbf{y}_{t+1} \to \mathbf{y}_t)$ w.r.t. the interpolated density $\mu_\lambda(\mathbf{y})$ have the log path probability ratio

$$\Delta S_t = u_\lambda(\mathbf{y}_{t+1}) - u_\lambda(\mathbf{y}_t). \tag{12}$$

This includes Metropolis-Hastings and Hamiltonian MC.

4. Results

Representational power versus sampling efficiency We first illustrate that SNFs can improve the representational power of deterministic NFs while having better sampling efficiency than pure MCMC. To this end we use images to define complex two-dimensional densities (Fig. 2a-c, "Exact") as target densities $\mu_X(\mathbf{x})$ to be sampled. We compare three flows: (i) a normalizing flow RealNVP layers, (ii) pure MCMC along the annealing path, (iii) a SNF with RealNVP layers prior to MCMC steps such that number of NF layers / MCMC steps equate to those of (i) and (ii) respectively.

Qualitatively, we can see that pure RealNVP NFs lack representational power to resolve fine details, while pure MCMC suffers from getting stuck in local minima. SNFs achieve high-quality sampling as soon as the former two are combined and jointly optimized (Fig. 2 a-c). Quantitatively, we compare the KL divergence between generated densities $p_X(\mathbf{x})$ and exact densities $\mu_X(\mathbf{x})$. Both normalizing flows and SNFs improve with greater depth, but SNFs achieve significantly lower KL divergence at a fixed network depth. Moreover, SNFs have higher statistical efficiency than pure Metropolis MC flows.

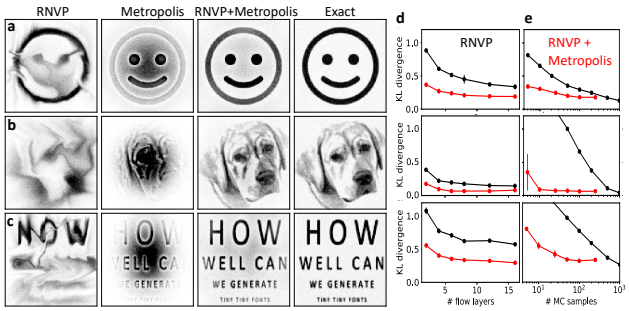

Figure 2. **Sampling of two-dimensional densities**. **a-c)** Sampling of image densities with different methods. Columns: (1) NF with RealNVP layers, (2) Metropolis MC sampling, (3) SNF combining (1+2), (4) Unbiased sample from exact density. **d-e)** Compare expressivity and statistical efficiency by showing KL divergence (mean and standard deviation over 3 training runs) between samples and true density from Fig. 2. **d**) Comparison of NF (black) and SNF (red) as a function of the number of RealNVP transformations. Number of MC steps in SNF is fixed to 50. **e**) Comparison of pure MCMC (black) and SNF (red) as a function of the number of MC steps. Number of RealNVP transformations in SNF is fixed to 10.

Table 1. **Alanine dipeptide**: KL-divergences of RNVP flow and SNF (RNVP+MCMC) between generated and target distributions for all multimodal torsion angles. Mean and standard deviation from 3 independent runs.

KL-DIV.	ϕ	γ_1	ψ	γ_2	γ_3
RNVP	1.69 ±0.03	3.82 ±0.01	0.98 ±0.03	0.79 ±0.03	0.79 ±0.09
SNF	**0.36** ±0.05	**0.21** ±0.01	**0.27** ±0.03	**0.12** ±0.02	**0.15** ±0.04

Alanine dipeptide In a second experiment we evaluate SNFs on density estimation and sampling of molecular structures from a simulation of the alanine dipeptide molecule in vacuum (Fig. 3) - a 66 dimensional molecule with multimodal torsion angle density. For the SNF we combine RealNVP layers with MCMC sampling with an initial internal coordinate transformations as introduced in (Noé et al., 2019). This is compared to a pure normalizing flow based on RealNVP layers which are currently state of the art for this kind of problem. Fig. 3a shows random structures sampled by the trained SNF. Fig. 3b shows marginal densities in all five multimodal torsion angles (backbone angles ϕ, ψ and methyl rotation angles γ_1, γ_2, γ_3). While the RealNVP flow misses many modes, SNFs resolves the multimodal structure and achieves a lower KL divergence between generated and target marginal distributions (Table 1).

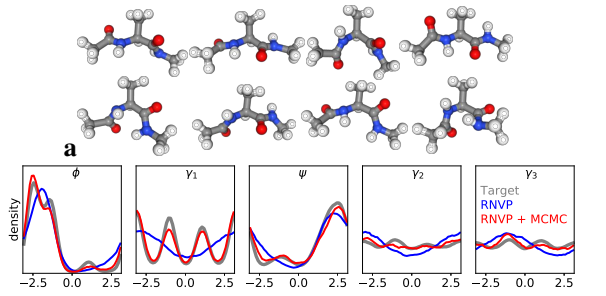

Figure 3. **Alanine dipeptide** sampled with NFs and SNFs. **a**) One-shot SNF samples of alanine dipeptide structures. **b**) neg. log. marginal densities in 5 unimodal torsion angles (top) and all 5 multimodal torsion angles (bottom).

5. Related work

Most importantly to this work is (Nilmeier et al., 2011) which provides the theoretical framework for computing path probability ratios for fixed deterministic and stochastic protocols which we extended to SNFs.

Furthermore related is the work of (Sohl-Dickstein et al., 2015; Chen et al., 2017; Gu et al., 2019; Hodgkinson et al., 2020) which all propose learnable stochastic processes designed for density estimation and/or sampling. Our work differs in so far that it is the only framework providing the combination of arbitrary learnable sampling/transformation blocks together with guarantees for asymptotically unbiased sampling and an efficient and exact training mechanism.

In work like (Salimans et al., 2015; Levy et al., 2017; Song et al., 2017; Hoffman et al., 2019; Hoffman, 2017) it has been shown that learnable proposals combined with stochasticity can improve sampling efficiency. Yet, such approaches do neither provide an exact reweighing scheme nor a way to optimize arbitrary transformation / sampling steps end-to-end efficiently.

Finally, there is work on modeling stochastic processes by combining learnable transformations and intermediate noise (Tzen & Raginsky, 2019; Jia & Benson, 2019; Liu et al., 2019; Li et al., 2020). These methods however are not designed for marginal density estimation / asymptotically unbiased sampling.

6. Conclusions

We have introduced stochastic normalizing flows (SNFs) that combine both stochastic processes and invertible deterministic transformations into a single learning framework. By leveraging nonequilibrium statistical mechanics we show that SNFs can efficiently be trained to sample asymptotically unbiased from target densities. This can be done by utilizing path probability ratios and avoiding intractabe marginaliza-

tion. Besides possible applicability in classical machine learning domains such as variational and Bayesian inference, we believe that the latter property can make SNFs a key component in the efficient sampling of many-body physics systems. In future research we aim to apply SNFs with many stochastic sampling steps to accurate large-scale sampling of molecules.

7. Acknowledgements

This work was funded by the European Research Commission (ERC CoG "ScaleCell"),Deutsche Forschungsgemeinschaft (CRC 1114/A04, NO 825/4–1, RTG DAEDALUS/P4) the MATH+ research center (AA1-6, EF1-2), and the "1000-Talent Program of Young Scientists in China". Part of this research was performed while the author was visiting the Institute for Pure and Applied Mathematics(IPAM), which is supported by the National Science Foundation (Grant No. DMS-1440415).

References

Albergo, M. S., Kanwar, G., and Shanahan, P. E. Flow-based generative models for markov chain monte carlo in lattice field theory. *Phys. Rev. D*, 100:034515, 2019.

Chen, C., Li, C., Chen, L., Wang, W., Pu, Y., and Carin, L. Continuous-time flows for efficient inference and density estimation. *arXiv preprint arXiv:1709.01179*, 2017.

Cornish, R., Caterini, A. L., Deligiannidis, G., and Doucet, A. Localised generative flows. *arXiv preprint arXiv:1909.13833*, 2019.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv:1605.08803*, 2016.

Dinh, L., Sohl-Dickstein, J., Pascanu, R., and Larochelle, H. A rad approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. In *Advances in Neural Information Processing Systems*, pp. 3134–3144, 2019.

Falorsi, L., de Haan, P., Davidson, T. R., De Cao, N., Weiler, M., Forré, P., and Cohen, T. S. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

Falorsi, L., de Haan, P., Davidson, T. R., and Forré, P. Reparameterizing distributions on lie groups. *arXiv preprint arXiv:1903.02958*, 2019.

Frenkel, D. and Smit, B. *Understanding molecular simulation*. Academic Press, 2001.

Gu, M., Sun, S., and Liu, Y. Dynamical sampling with langevin normalization flows. *Entropy*, 21(11):1096, 2019.

Hodgkinson, L., van der Heide, C., Roosta, F., and Mahoney, M. W. Stochastic normalizing flows. *arXiv preprint arXiv:2002.09547*, 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. *International Conference on Machine Learning*, pp. 1510–1519, 2017.

Jia, J. and Benson, A. R. Neural jumpstochastic differential equations. *arXiv preprint arXiv:1905.10403*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

Levy, D., Hoffman, M. D., and Sohl-Dickstein, J. Generalizing hamiltonian monte carlo with neural networks. *arXiv preprint arXiv:1711.09268*, 2017.

Li, S.-H. and Wang, L. Neural network renormalization group. *Phys. Rev. Lett.*, 121:260601, 2018.

Li, X., Wong, T.-K. L., Chen, R. T. Q., and Duvenaud, D. Scalable gradients for stochastic differential equations. *arXiv preprint arXiv:2001.01328*, 2020.

Liu, X., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. Neural importance sampling. *arXiv preprint arXiv:1808.03856*, 2018.

Neal, R. M. Annealed importance sampling. *arXiv preprint arXiv:physics/9803008*, 1998.

Nicoli, K. A., Nakajima, S., Strodthoff, N., Samek, W., Müller, K.-R., and Kessel, P. Asymptotically unbiased estimation of physical observables with neural samplers. *arXiv:1910.13496*, 2019.

Nilmeier, J. P., Crooks, G. E., Minh, D. D. L., and Chodera, J. D. Nonequilibrium candidate Monte Carlo is an efficient tool for equilibrium simulation. *Proc. Natl. Acad. Sci. USA*, 108:E1009–E1018, 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):eaaw1147, 2019.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Pearlman, D. A., Case, D. A., Caldwell, J. W., Ross, W. R., Iii, C. T. E., Debolt, S., Ferguson, D., Seibel, G., and Kollman, P. AMBER, a computer program for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to elucidate the structures and energies of molecules. *Comp. Phys. Commun.*, 91:1–41, 1995.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Salimans, T., Kingma, D., and Welling, M. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

Song, J., Zhao, S., and Ermon, S. A-nice-mc: Adversarial training for mcmc. In *Advances in Neural Information Processing Systems*, pp. 5140–5150, 2017.

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

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

Tzen, B. and Raginsky, M. Neural stochastic differential equations: Deep latent gausian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

Supplementary Material

Training normalizing flows

Energy-based training and forward weight maximization If the target density μ_X is known up to a constant Z_X, we minimize the forward KL divergence between the generated and the target distribution.

$$\begin{aligned} &\mathrm{KL}(p_X \parallel \mu_X) \\ &= \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}\left[\log p_X(\mathbf{x}) - \log \mu_X(\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim \mu_Z(\mathbf{z})}\left[u_X(F_{ZX}(\mathbf{z})) - \Delta S_{ZX}(\mathbf{z})\right] + \mathrm{const.} \end{aligned} \tag{13}$$

The importance weights wrt the target distribution can be computed as:

$$w_X(\mathbf{x}) = \exp\left(-u_X\left(F_{ZX}(\mathbf{z})\right) + u_Z(\mathbf{z}) + \Delta S_{ZX}(\mathbf{z})\right) \propto \frac{\mu_X(\mathbf{x})}{p_X(\mathbf{x})}. \tag{14}$$

As $\mathbb{E}_{\mathbf{z}\sim p_Z(\mathbf{z})}\left[u_Z(\mathbf{z})\right]$ is a constant, we can equivalently minimize KL or maximize log weights:

$$\max \mathbb{E}_{\mathbf{z}\sim p_Z(\mathbf{z})}\left[\log w_X(\mathbf{x})\right] = \min \mathrm{KL}(p_X \parallel \mu_X), \tag{15}$$

Maximum likelihood and backward weight maximization The backward KL divergence $\mathrm{KL}(\mu_X \parallel p_X)$ is not always tractable as $\mu_X(\mathbf{x})$ can be difficult to sample from. Replacing $\mu_X(\mathbf{x})$ by the empirical data distribution $\rho_X(\mathbf{x})$, the KL becomes a negative log-likelihood:

$$\begin{aligned} &\mathrm{NLL}(\rho_X \parallel p_X) \\ &= \mathbb{E}_{\mathbf{x}\sim \rho_X(\mathbf{x})}\left[u_Z(F_{XZ}(\mathbf{x})) - \Delta S_{XZ}(\mathbf{x})\right] + \mathrm{const.} \end{aligned} \tag{16}$$

Using $\mathbb{E}_{\mathbf{x}\sim\rho_X(\mathbf{x})}\left[-\log \rho_X(\mathbf{x})\right] = \mathrm{const}$ and the weights:

$$w_Z(\mathbf{z}) = \exp\left(-u_Z\left(F_{XZ}(\mathbf{x})\right) - \log \rho_X(\mathbf{x}) + \Delta S_{XZ}(\mathbf{x})\right) \propto \frac{\mu_Z(\mathbf{z})}{p_Z(\mathbf{z})},$$

maximum likelihood equals log weight maximization:

$$\max \mathbb{E}_{\mathbf{x}\sim\rho_X(\mathbf{x})}\left[\log w_Z(\mathbf{z})\right] = \min \mathrm{NLL}(\rho_X \parallel p_X). \tag{17}$$

Proof of theorem 1 (unbiased sampling with SNF importance weights)

Considering

$$\begin{aligned} \mathbb{E}_{\mu_X}[O] &= \int \mu_X(\mathbf{x}) O(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &= \iint \mu_X(\mathbf{x}) \mathbb{P}_b(\mathbf{x}\to\mathbf{z}) O(\mathbf{x}) \mathrm{d}\mathbf{z}\mathrm{d}\mathbf{x} \\ &= \iint \mu_Z(\mathbf{z}) \mathbb{P}_f(\mathbf{z}\to\mathbf{x}) \\ &\qquad \cdot \left[\frac{\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x}\to\mathbf{z})}{\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z}\to\mathbf{x})} O(\mathbf{x})\right] \mathrm{d}\mathbf{z}\mathrm{d}\mathbf{x} \\ &= \mathbb{E}_f\left[\frac{\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x}\to\mathbf{z})}{\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z}\to\mathbf{x})} O(\mathbf{x})\right], \end{aligned}$$

where $\mathbb{E}_f$ denotes the expectation over forward path realizations. In practice, we do not know the normalization constant of μ_X and we therefore replace $\frac{\mu_X(\mathbf{x})\mathbb{P}_b(\mathbf{x}\to\mathbf{z})}{\mu_Z(\mathbf{z})\mathbb{P}_f(\mathbf{z}\to\mathbf{x})}$ by the unnormalized path weights in Eq. (6). Then we must normalize the estimator for expectation values, obtaining:

$$\frac{\sum_{k=1}^N \mathbf{w}(\mathbf{z}_k\to\mathbf{x}_k) O(\mathbf{x}_k)}{\sum_{k=1}^N \mathbf{w}(\mathbf{z}_k\to\mathbf{x}_k)} \xrightarrow{p} \mathbb{E}_\mu[O]$$

which converges towards $\mathbb{E}_\mu[O]$ with $N\to\infty$ according to the law of large numbers.

Derivation of the deterministic layer probability ratio

In order to work with delta distributions, we define $\delta^\sigma(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma\mathbf{I})$, i.e. a Gaussian normal distribution with mean $\mathbf{0}$ and variance σ and then consider the limit $\sigma \to 0^+$. In the case where $\sigma > 0$, by defining

$$q_t^\sigma(\mathbf{y}_t \to \mathbf{y}_{t+1}) = \delta^\sigma(\mathbf{y}_{t+1} - F_t(\mathbf{y}_t)),$$

and

$$\begin{aligned}\tilde{q}_t^\sigma(\mathbf{y}_{t+1} \to \mathbf{y}_t) &= \frac{p_t(\mathbf{y}_t) q_t^\sigma(\mathbf{y}_t \to \mathbf{y}_{t+1})}{\int p_t(\mathbf{y}) q_t^\sigma(\mathbf{y} \to \mathbf{y}_{t+1}) \mathrm{d}\mathbf{y}} \\ &= \frac{p_t(\mathbf{y}_t)\delta^\sigma(\mathbf{y}_{t+1} - F_t(\mathbf{y}_t))}{\int p_t(\mathbf{y})\delta^\sigma(\mathbf{y}_{t+1} - F_t(\mathbf{y}))\mathrm{d}\mathbf{y}},\end{aligned}$$

we have

$$\frac{\tilde{q}_t^\sigma(\mathbf{y}_{t+1} \to \mathbf{y}_t)}{q_t^\sigma(\mathbf{y}_t \to \mathbf{y}_{t+1})} = \frac{p_t(\mathbf{y}_t)}{\int p_t(\mathbf{y})\delta^\sigma(\mathbf{y}_{t+1} - F(\mathbf{y}))\mathrm{d}\mathbf{y}},$$

where $p_t(y_t)$ denotes the marginal distribution of $\mathbf{y}_t$. By considering

$$\begin{aligned}&\lim_{\sigma\to 0^+} \int p_t(\mathbf{y})\delta^\sigma(\mathbf{y}_{t+1} - F_t(\mathbf{y}))\mathrm{d}\mathbf{y} \\ &= \lim_{\sigma\to 0^+} \int p_t(F_t^{-1}(\mathbf{y}'))\delta^\sigma(\mathbf{y}_{t+1} - \mathbf{y}') \left|\det\left(\frac{\partial F_t^{-1}(\mathbf{y}')}{\partial \mathbf{y}'}\right)\right| \mathrm{d}\mathbf{y}' \\ &= p_t(F_t^{-1}(\mathbf{y}_{t+1})) \left|\det\left(\frac{\partial F_t^{-1}(\mathbf{y}_{t+1})}{\partial \mathbf{y}_{t+1}}\right)\right| \\ &= p_t(\mathbf{y}_t) \left|\det \mathbf{J}_t(\mathbf{y}_t)\right|^{-1}\end{aligned}$$

and using the definition of ΔS_t in terms of path probability rations, we obtain:

$$\begin{aligned}&\exp\left(\Delta S_t\right) \\ &= \frac{\tilde{q}_t(\mathbf{y}_{t+1} \to \mathbf{y}_t)}{q_t(\mathbf{y}_t \to \mathbf{y}_{t+1})} = \lim_{\sigma\to 0^+} \frac{\tilde{q}_t^\sigma(\mathbf{y}_{t+1} \to \mathbf{y}_t)}{q_t^\sigma(\mathbf{y}_t \to \mathbf{y}_{t+1})} \\ &= \lim_{\sigma\to 0^+} \frac{p_t(\mathbf{y}_t)}{\int p_t(\mathbf{y})\delta^\sigma(\mathbf{y}_{t+1} - F(\mathbf{y}))\mathrm{d}\mathbf{y}} \\ &= \left|\det \mathbf{J}_t(\mathbf{y}_t)\right|\end{aligned}$$

and thus

$$\Delta S_t = \log\left|\det \mathbf{J}_t(\mathbf{y}_t)\right|.$$

Derivation of the overdamped Langevin path probability ratio

These results follow (Nilmeier et al., 2011). The backward step is realized by

$$\mathbf{y}_t = \mathbf{y}_{t+1} - \epsilon_t \nabla u_\lambda(\mathbf{y}_{t+1}) + \sqrt{\frac{2\epsilon}{\beta}}\tilde{\boldsymbol{\eta}}_t. \tag{18}$$

Combining Equations (11) and (18):

$$-\epsilon_t \nabla u_\lambda(\mathbf{y}_t) + \sqrt{\frac{2\epsilon_t}{\beta}}\boldsymbol{\eta}_t = \epsilon_t \nabla u_\lambda(\mathbf{y}_{t+1}) - \sqrt{\frac{2\epsilon_t}{\beta}}\tilde{\boldsymbol{\eta}}_t.$$

and thus

$$\tilde{\boldsymbol{\eta}}_t = \sqrt{\frac{\epsilon_t\beta}{2}}\left[\nabla u_\lambda(\mathbf{y}_t) + \nabla u_\lambda(\mathbf{y}_{t+1})\right] - \boldsymbol{\eta}_t.$$

Resulting in the path probability ratio:

$$\exp\left(\Delta S_t\right) = \frac{q_t(\mathbf{y}_{t+1} \to \mathbf{y}_t)}{q_t(\mathbf{y}_t \to \mathbf{y}_{t+1})} = \frac{p(\tilde{\boldsymbol{\eta}}_t)\left|\frac{\partial \mathbf{y}_t}{\partial \tilde{\boldsymbol{\eta}}_t}\right|}{p(\boldsymbol{\eta}_t)\left|\frac{\partial \mathbf{y}_{t+1}}{\partial \boldsymbol{\eta}_t}\right|}$$
$$= \frac{p(\tilde{\boldsymbol{\eta}}_t)}{p(\boldsymbol{\eta}_t)} = \mathrm{e}^{-\frac{1}{2}\left(\|\tilde{\boldsymbol{\eta}}_t\|^2 - \|\boldsymbol{\eta}_t\|^2\right)}.$$

and thus

$$-\Delta S_t = \frac{1}{2}\left(\|\tilde{\boldsymbol{\eta}}_t\|^2 - \|\boldsymbol{\eta}_t\|^2\right)$$

Derivation of the Langevin probability ratio

These results follow (Nilmeier et al., 2011). We define constants:

$$c_1 = \frac{\Delta t}{2m}$$
$$c_2 = \sqrt{\frac{4\gamma m}{\Delta t \beta}}$$
$$c_3 = 1 + \frac{\gamma \Delta t}{2}$$

Then, the forward step of Brooks-Brünger-Karplus (BBK, leap-frog) Langevin dynamics is defined as:

$$\mathbf{v}' = \mathbf{v}_t + c_1\left[-\nabla u_\lambda(\mathbf{x}_t) - \gamma m \mathbf{v}_t + c_2 \boldsymbol{\eta}_t\right] \tag{19}$$
$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t \mathbf{v}' \tag{20}$$
$$\mathbf{v}_{t+1} = \frac{1}{c_3}\left[\mathbf{v}' + c_1\left(-\nabla u_\lambda(\mathbf{x}_{t+1}) + c_2 \boldsymbol{\eta}'_t\right)\right] \tag{21}$$

Note that the factor 4 in sqrt is different from (Nilmeier et al., 2011) – this factor is needed as we employ $\Delta t/2$ in both half-steps. The backward step with reversed momenta, $(\mathbf{x}_{t+1}, -\mathbf{v}_{t+1}) \to (\mathbf{x}_t, -\mathbf{v}_t)$ is then defined by:

$$\mathbf{v}'' = -\mathbf{v}_{t+1} + c_1\left[-\nabla u_\lambda(\mathbf{x}_{t+1}) + \gamma m \mathbf{v}_{t+1} + c_2 \tilde{\boldsymbol{\eta}}_t\right] \tag{22}$$
$$\mathbf{x}_t = \mathbf{x}_{t+1} + \Delta t \mathbf{v}'' \tag{23}$$
$$-\mathbf{v}_t = \frac{1}{c_3}\left[\mathbf{v}'' + c_1\left(-\nabla u_\lambda(\mathbf{x}_t) + c_2 \tilde{\boldsymbol{\eta}}'_t\right)\right] \tag{24}$$

To compute the momenta $\tilde{\boldsymbol{\eta}}_t, \tilde{\boldsymbol{\eta}}'_t$ that realize the reverse step, we first combine Eqs. (20-23) to obtain:

$$\mathbf{v}' = -\mathbf{v}'' \tag{25}$$

Combining Eqs. (21), (22) and (25), we obtain:

$$\left(1 + \frac{\gamma \Delta t}{2}\right)\mathbf{v}_{t+1} = \mathbf{v}' + c_1\left(-\nabla u_\lambda(\mathbf{x}_{t+1}) + c_2 \boldsymbol{\eta}'_t\right)$$
$$\left(1 - \frac{\gamma \Delta t}{2}\right)\mathbf{v}_{t+1} = \mathbf{v}' + c_1\left(-\nabla u_\lambda(\mathbf{x}_{t+1}) + c_2 \tilde{\boldsymbol{\eta}}_t\right),$$

and:

$$\tilde{\boldsymbol{\eta}}_t = \boldsymbol{\eta}'_t - \sqrt{\gamma \Delta t m \beta}\mathbf{v}_{t+1}$$

Combining Eqs. (19), (24) and (25), we obtain:

$$-\mathbf{v}_t\left(1 - \frac{\gamma \Delta t}{2}\right) = \mathbf{v}'' + c_1\left(-\nabla u_\lambda(\mathbf{x}_t) + c_2 \boldsymbol{\eta}_t\right)$$
$$-\mathbf{v}_t\left(1 + \frac{\gamma \Delta t}{2}\right) = \mathbf{v}'' + c_1\left(-\nabla u_\lambda(\mathbf{x}_t) + c_2 \tilde{\boldsymbol{\eta}}'_t\right),$$

and:

$$-\mathbf{v}_t\left(1-\frac{\gamma\Delta t}{2}\right)-c_2\boldsymbol{\eta}_t=-\mathbf{v}_t\left(1+\frac{\gamma\Delta t}{2}\right)-c_2\tilde{\boldsymbol{\eta}}_t'$$
$$\tilde{\boldsymbol{\eta}}_t'=\boldsymbol{\eta}_t-\sqrt{\gamma\Delta t m\beta}\mathbf{v}_t$$

To compute the path probability ratio we introduce the Jacobian

$$J(\boldsymbol{\eta}_t,\boldsymbol{\eta}_t')=\det\begin{bmatrix}\frac{\partial\mathbf{x}_{t+1}}{\partial\boldsymbol{\eta}_t} & \frac{\partial\mathbf{v}_{t+1}}{\partial\boldsymbol{\eta}_t}\\ \frac{\partial\mathbf{x}_{t+1}}{\partial\boldsymbol{\eta}_t'} & \frac{\partial\mathbf{v}_{t+1}}{\partial\boldsymbol{\eta}_t'}\end{bmatrix}$$

and find:

$$\begin{aligned}\exp\left(\Delta S_t\right)&=\frac{\tilde{q}_t\left((\mathbf{x}_{t+1},-\mathbf{v}_{t+1})\to(\mathbf{x}_t,\mathbf{v}_t)\right)}{q_t\left((\mathbf{x}_t,\mathbf{v}_t)\to(\mathbf{x}_{t+1},-\mathbf{v}_{t+1})\right)}\\&=\frac{p(\tilde{\boldsymbol{\eta}}_t)p(\tilde{\boldsymbol{\eta}}_t')J(\tilde{\boldsymbol{\eta}}_t,\tilde{\boldsymbol{\eta}}_t')}{p(\boldsymbol{\eta}_t)p(\boldsymbol{\eta}_t')J(\boldsymbol{\eta}_t,\boldsymbol{\eta}_t')}\\-\Delta S_t&=\frac{1}{2}\left(\left(\|\tilde{\boldsymbol{\eta}}_t\|^2+\|\tilde{\boldsymbol{\eta}}_t'\|^2\right)-\left(\|\boldsymbol{\eta}_t\|^2+\|\boldsymbol{\eta}_t'\|^2\right)\right)\end{aligned}$$

where the Jacobian ratio cancels as the Jacobians are independent of the noise variables.

Derivation of the probability ratio for Markov Chain Monte Carlo

For MCMC, q_t satisfies the detailed balance condition

$$\exp(-u_\lambda(\mathbf{y}_t))\cdot q_t(\mathbf{y}_t\to\mathbf{y}_{t+1})=\exp(-u_\lambda(\mathbf{y}_{t+1}))\cdot\tilde{q}_t(\mathbf{y}_{t+1}\to\mathbf{y}_t)$$

with respect to the potential function u_λ. We have

$$\begin{aligned}\Delta S_t&=\log\frac{\tilde{q}_t(\mathbf{y}_{t+1}\to\mathbf{y}_t)}{q_t(\mathbf{y}_t\to\mathbf{y}_{t+1})}\\&=\log\frac{\exp(-u_\lambda(\mathbf{y}_t))}{\exp(-u_\lambda(\mathbf{y}_{t+1}))}\\&=u_\lambda(\mathbf{y}_{t+1})-u_\lambda(\mathbf{y}_t)\end{aligned}$$

Derivation of the probability ratio for Hamiltonian MC with Metropolis acceptance

Hamiltonian MC with Metropolis acceptance defines a forward path density

$$q_t\left((\mathbf{y}_t,\mathbf{v})\to(\mathbf{y}_{t+1},\mathbf{v}^K)\right)$$

which satisfies the joint detailed balance condition

$$\begin{aligned}&\exp(-u_\lambda(\mathbf{y}_t))\mathcal{N}(\mathbf{v}|\mathbf{0},\mathbf{I})\cdot q_t\left((\mathbf{y}_t,\mathbf{v})\to(\mathbf{y}_{t+1},\mathbf{v}^K)\right)\\&\quad=\exp(-u_\lambda(\mathbf{y}_{t+1}))\mathcal{N}(\mathbf{v}^K|\mathbf{0},\mathbf{I})\cdot\tilde{q}_t\left((\mathbf{y}_{t+1},\mathbf{v}^K)\to(\mathbf{y}_t,\mathbf{v})\right).\end{aligned}\tag{26}$$

Considering the velocity $\mathbf{v}$ is independently drawn from $\mathcal{N}(\mathbf{v}|\mathbf{0},\mathbf{I})$, the "marginal" forward path density of $\mathbf{y}_t\to\mathbf{y}_{t+1}$ is

$$q_t\left(\mathbf{y}_t\to\mathbf{y}_{t+1}\right)=\iint\mathcal{N}(\mathbf{v}|\mathbf{0},\mathbf{I})\cdot q_t\left((\mathbf{y}_t,\mathbf{v})\to(\mathbf{y}_{t+1},\mathbf{v}^K)\right)\mathrm{d}\mathbf{v}\mathrm{d}\mathbf{v}^K.$$

Then, it can be obtained from (26) that

$$\begin{aligned}
&\exp(-u_\lambda(\mathbf{y}_t))q_t\left(\mathbf{y}_t \to \mathbf{y}_{t+1}\right) \\
=&\iint \exp(-u_\lambda(\mathbf{y}_t))\mathcal{N}(\mathbf{v}|\mathbf{0},\mathbf{I}) \\
&\quad \cdot q_t\left((\mathbf{y}_t,\mathbf{v}) \to (\mathbf{y}_{t+1},\mathbf{v}^K)\right)\mathrm{d}\mathbf{v}\mathrm{d}\mathbf{v}^K \\
=&\iint \exp(-u_\lambda(\mathbf{y}_{t+1}))\mathcal{N}(\mathbf{v}^K|\mathbf{0},\mathbf{I}) \\
&\quad \cdot \tilde{q}_t\left((\mathbf{y}_{t+1},\mathbf{v}^K) \to (\mathbf{y}_t,\mathbf{v})\right)\mathrm{d}\mathbf{v}\mathrm{d}\mathbf{v}^K \\
=&\exp(-u_\lambda(\mathbf{y}_{t+1}))\tilde{q}_t\left(\mathbf{y}_{t+1} \to \mathbf{y}_t\right),
\end{aligned}$$

and

$$\begin{aligned}
\Delta S_t &= \log \frac{\tilde{q}_t\left(\mathbf{y}_{t+1} \to \mathbf{y}_t\right)}{q_t\left(\mathbf{y}_t \to \mathbf{y}_{t+1}\right)} \\
&= u_\lambda(\mathbf{y}_{t+1}) - u_\lambda(\mathbf{y}_t)
\end{aligned}$$

Hyper-parameters and other benchmark details

All experiments were run using PyTorch 1.2 and on GTX1080Ti cards. Optimization uses Adam (Kingma & Ba, 2014) with step-size 0.001 and otherwise default parameters. All deterministic flow transformations use RealNVP (Dinh et al., 2016). A RealNVP block is defined by two subsequent RealNVP layers that are swapped such that each channel gets transformed once as a function of the other channel. The affine transformation of each RealNVP layer is given by a fully connected ReLU network.

Alanine dipeptide in Fig. 3

- Normalizing flow uses 3 RealNVP blocks with 3 hidden layers and $[128, 128, 128]$ nodes in their transformers. Training was done by minimizing J_{ML} for 1000 iterations with batch-size 256.
- SNF uses the same architecture and training parameters, but additionally 20 Metropolis MC steps each using a Gaussian proposal density with standard deviation 0.1.
- As a last flow layer before $\mathbf{x}$, we used an invertible transformation between Cartesian coordinates and internal coordinates (bond lengths, angles, torsion angles) following the procedure described in (Noé et al., 2019). The internal coordinates were normalized by removing the mean and dividing by the standard deviation of their values in the training data.
- Training data: We set up Alanine dipeptide in vacuum using OpenMMTools. Parameters are defined by the force field ff96 of the AMBER program (Pearlman et al., 1995). Simulations are run at standard OpenMMTools parameters with no bond constraints, 1 femtosecond time-step for 10^6 time-steps (1 nanosecond) at a temperature of 1000 K, which results in rapid exploration of the ϕ/ψ torsion angles and a few hundred transitions between metastable states. 10^5 atom positions were saved as training data.