Smooth Normalizing Flows

Anonymous Author(s)
Affiliation
Address
email

Abstract

Normalizing flows are a promising tool for modeling probability distributions in physical systems. While state-of-the-art flows accurately approximate distributions and energies, applications in physics additionally require smooth energies to compute forces and higher-order derivatives. Furthermore, such densities are often defined on non-trivial topologies. A recent example are Boltzmann Generators for generating 3D-structures of peptides and small proteins. These generative models leverage the space of internal coordinates (dihedrals, angles, and bonds), which is a product of hypertori and compact intervals. In this work, we introduce a class of smooth mixture transformations working on both compact intervals and hypertori. Mixture transformations employ root-finding methods to invert them in practice, which has so far prevented bi-directional flow training. To this end, we show that parameter gradients and forces of such inverses can be computed from forward evaluations via the inverse function theorem. We demonstrate two advantages of such smooth flows: they allow training by force matching to simulation data and can be used as potentials in molecular dynamics simulations.

1 Introduction

Generative learning using *normalizing flows* (NF) [38, 33, 31] has become a widely applicable tool in the physical sciences which has e.g. been used for sampling lattice models [28, 29, 24, 4, 1], approximating the equilibrium density of molecular systems [30, 22, 43, 45] or estimating free energy differences [42, 10].

Such models approximate a target density μ via diffeomorphic maps $f(\cdot;\boldsymbol{\theta})\colon \Omega \subset \mathbb{R}^d \to \Omega$ by transforming samples $\boldsymbol{z} \sim p_0(\boldsymbol{z})$ of a base density into samples $\boldsymbol{x} = f(\boldsymbol{z};\boldsymbol{\theta})$ such that they follow the push-forward density

$$\boldsymbol{x} \sim p_f(\boldsymbol{x};\boldsymbol{\theta}) := p_0\left(f^{-1}(\boldsymbol{x};\boldsymbol{\theta})\right)\left|\det \partial_{\boldsymbol{x}} f^{-1}(\boldsymbol{x};\boldsymbol{\theta})\right|. \tag{1}$$

Flows can be trained on data by maximizing the likelihood or via minimizing the reverse KL divergence $D_{KL}\left[p_f(\cdot;\boldsymbol{\theta})\|\mu\right]$ if μ is known up to a normalizing constant.

While NFs are usually introduced as *smooth* diffeomorphisms, most applications like density estimation or sampling only require C^1-smooth of the transformation. Higher-order smoothness of flow transformations has not been discussed so far and can become a challenge as soon as multi-modal transformations on other topologies than $\mathbb{R}^d$ are discussed.

C^k-smoothness of NFs with $k > 1$ is especially important for applications in physics. Physical models usually come in the form of differential equations, where the derivatives bear a physical meaning that is often crucial to fit, evaluate, or interpret the model. Thus, the construction of expressive, smooth flow architectures will likely open up new avenues of research in this domain.

Boltzmann Generators A recent application of NFs to a physical problem are Boltzmann Generators (BG) [30]. They are generative models trained to sample conformations of molecules in

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

equilibrium, which follow a Boltzmann-type distribution $\mu(\boldsymbol{x}) \propto \exp(-u(\boldsymbol{x}))$. Here u is the dimensionless potential energy defined by the molecular system and the thermodynamic state (e.g. Canonical ensemble at a certain temperature) it is simulated in. BGs can be trained by a combination of MLE on possibly biased trajectory data and simultaneously minimizing the reverse KL divergence to the target density $\mu(\boldsymbol{x})$. This bi-directional training scheme can achieve sampling of low-energy equilibrium structures. After training, BGs can be used e.g. for importance sampling or for providing efficient proposals when being used in MCMC applications [36].

In the context of BGs, the negative gradient of the log push-forward density with respect to $\boldsymbol{x}$ corresponds to the atomistic forces. Access to well-behaved forces is pivotal in most classical molecular computations: they simplify optimization of molecular models, drive molecular dynamics (MD) simulations, enable computations of macroscopic observables, and facilitate multiscale modeling.

In this work, we will primarily focus on two important implications of smooth flow forces. First, we show that they enable the training of NFs via force matching (FM), i.e. minimizing the force mean-squared error with respect to reference data. In combining FM with density estimation, NFs are pushed to match the distributions and their derivatives, which implicitly leads to favorable regularization. Second, we apply flow forces to drive dynamics simulations. Such simulations are required in many applications to not only sample the target distribution but also compute dynamical observables such as transition rates or diffusivities.

Respecting Topological Constraints Many physical models operate on nontrivial topologies, i.e. the d-dimensional hyper-torus $\mathbb{T}^d$, which can become an obstacle in constructing smooth flows.

An important example are BGs for peptides and small proteins which require an internal coordinate (IC) transformation to achieve low-energy sampling of structures [30]. This non-trainable layer transforms Euclidean coordinates $\boldsymbol{x} \in \mathbb{R}^{n\times 3}$ into a representation given by distances $\boldsymbol{d} \in [a_1, b_1] \times \ldots \times [a_{n-1}, b_{n-1}]$, angles $\boldsymbol{\alpha} \in [0, \pi]^{n-2}$ and dihedral torsion angles $\boldsymbol{\tau} \in \mathbb{T}^{n-3}$. As molecular energies are often invariant to translation and rotation, IC transformations are useful as ICs are bijective with the equivalence class of all-atom coordinates that are identical up to global rotations and translations. The learning problem can then be reduced to modeling the joint distribution $\mu(\boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\tau})$ which is supported on the nontrivial topological space $\mathcal{X}_{\text{IC}} := \mathbb{I}^{2n-3} \times \mathbb{T}^{n-3}$, where $\mathbb{I} = [0, 1]$ denotes the closed unit interval.

Recent work [30, 43] suggested to model the density within an open set $\Omega \subset \mathcal{X}_{\text{IC}}$, leverage C^∞-smooth normalizing flows defined on $\mathbb{R}^d$ and then prevent significant mass around singular points using regularizing losses. Such an approach however can lead to bias and ill-behaved training and requires a-priori knowledge of the support of the densities. Later work [9] approached the problem using C^1-smooth splines flows which leads to accurate samples, however results in broken forces.

To overcome these limitations while still benefiting from the merits of prior work, such as bi-directional training and fast forward/inverse transformations we formulate the following desiderata for flow transformations on-top of IC layers: (A) They must have support on $\mathbb{I}^d$ and $\mathbb{T}^d$. (B) They should be C^∞-smooth. (C) They must allow bi-directional training. (D) Forward and inverse direction must be efficient to evaluate.

Satisfying (D) can be achieved using coupling layers [11, 12]. This reduces the problem to finding element-wise conditional transformations satisfying (A-C).

Contributions In this work we propose the following novelties:

- We present a C^∞-smooth transformation with compact support which can be used for expressive transformations on $\mathbb{I}^d$ as well as the hypertorus $\mathbb{T}^d$. This satisfies (A) and (B).
- We present novel algorithm which allows optimizing non-analytic inverse directions of flows that can only be evaluated via black-box root finding methods. This satisfies (C).
- We show that training of smooth flows through combinations of force matching, density estimation, and energy-based training can achieve nearly perfect equilibrium densities.
- We show that forces of such smooth flows can be used in dynamical simulations.

2 Related Work

While related work exist for each of the requirements (A-D) above, none of them provides a framework that addresses all of them. Specifically, multi-modality in element-wise flow transformations has been approached using splines [13, 27], monotonoic polynomials [21, 32], monotonic neural networks which directly approximate an inverse CDF transformation [19, 8], mixtures of unimodal transformations [18, 34] and stochastic transition steps [43, 7]. Flows on non-trivial manifolds have been discussed for hypertori and -spheres [34], Lie groups [15, 4], hyperbolic spaces [3] as well as on general manifolds [14, 16, 26, 25, 5]. Applications of flows for molecular systems include approximation of the equilibrium density [30, 43, 22], generation of molecular conformations [45, 35] and free energy differences [42, 10]. Using forces to train neural network potentials of molecules was done e.g. in [40, 20]. Backpropagation through black-box functions was generally discussed in [17]. More related to our work is Bai et al. [2] who discuss training models whose outputs are obtained through fix-point equations. Finally Shirobokov et al. [37] discuss backpropagation through black-box simulators using surrogate models.

We discuss related work on flows on $\mathbb{I}$ and the unit circle in detail in Section 4.

3 Incorporating Forces into Flow Training

NFs are most commonly trained by minimizing the negative log likelihood (NLL),

$$\mathcal{L}_{\mathrm{NLL}}(\boldsymbol{\theta}) := -\mathbb{E}_{\boldsymbol{x}\sim\mu(\boldsymbol{x})}[\log p_f(\boldsymbol{x};\boldsymbol{\theta})] = D_{\mathrm{KL}}[\mu||p_f(\cdot;\boldsymbol{\theta})] + \mathrm{const}, \tag{2}$$

or by minimizing the reverse KL divergence (KLD)

$$\mathcal{L}_{\mathrm{KLD}}(\boldsymbol{\theta}) := D_{\mathrm{KL}}[p_f(\cdot;\boldsymbol{\theta})||\mu] + \mathrm{const}. \tag{3}$$

BGs as discussed in Noé et al. [30] use a convex combination of the two in order to avoid mode-collapse while still being able to achieve low-energy samples. If reference forces $\mathbf{f}(\boldsymbol{x}) = -\partial_{\boldsymbol{x}} u(\boldsymbol{x})$ corresponding to samples $\boldsymbol{x}$ are available and $f(\cdot;\boldsymbol{\theta})$ is at least C^2-smooth, the optimization can naturally be augmented by the force mean-squared error:

$$\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{x}\sim\mu(\boldsymbol{x})}\left[\|\mathbf{f}(\boldsymbol{x}) + \partial_{\boldsymbol{x}} \log p_f(\boldsymbol{x};\boldsymbol{\theta})\|_2^2\right]. \tag{4}$$

As was shown in Wang et al. [40] such *force-matching* can lead to unbiased potential surfaces even if samples are not sampled from equilibrium. Similarly to Noé et al. [30] we can thus define a loss function for smooth flows as the convex combination

$$\mathcal{L}(\boldsymbol{\theta}) = \omega_n \mathcal{L}_{\mathrm{NLL}}(\boldsymbol{\theta}) + \omega_k \mathcal{L}_{\mathrm{KLD}}(\boldsymbol{\theta}) + \omega_{\mathrm{f}} \mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}). \tag{5}$$

4 Smooth flows on the closed interval and the unit circle

We now discuss how to achieve smooth flows on $\mathbb{I}^d$ and $\mathbb{T}^d$. To unify the discussion we consider the unit circle S^1 to be the quotient space $\mathbb{I}/\sim$ using the relation $x \sim x' \Leftrightarrow (x = x') \vee (x = 0 \wedge x' = 1) \vee (x' = 0 \wedge x = 1)$. The d-dimensional hypertorus is given by the direct product $\mathbb{T}^d = S^1 \times \ldots \times S^1$. Following the usual definition we say that f is C^k-smooth on a compact interval $[a, b]$ iff there exists a C^k-smooth continuation of f on an open set $\Omega \supset [a, b]$.

Smooth flows on $\mathbb{I}$ Any smooth diffeomorphism on $\mathbb{R}$ or an open interval $(a, b) \supset \mathbb{I}$ can be restricted to $\mathbb{I}$ and re-scaled to satisfy this requirement. Let $f\colon \Omega \to \Omega$ be C^k-smooth for some open set $\mathbb{I} \subset \Omega \subset \mathbb{R}$. Then $\tilde{f}(x) = (f(x) - f(0))/(f(1) - f(0))$ defines a C^k-smooth diffeomorphism on $\mathbb{I}$. Simple C^∞ transformations on $\mathbb{R}$ are given by affine transformations [12]. After re-scaling any affine map will result in the same identity mapping and thus will not be able to model complex densities. Powerful and computationally efficient multi-modal transformations on $\mathbb{I}$ can be achieved using rational-quadratic splines [13]. Those are C^1-smooth and thus will result in discontinuous forces which can be disadvantageous for physical applications. Possible other C^∞-smooth candidate transformations with analytic forward evaluation are given by mixtures of logistic transformations [18] or deep sigmoid/deep dense sigmoid flows [19]. Those are only continuous on $(0, 1)$ and thus special care has to be taken to avoid problematic behavior on the tails. Finally, multi-modal C^∞ transformations on $\mathbb{R}$ can be achieved using non-analytic methods [6, 41]. Yet, computational costs and numerical accuracy limits their applicability e.g. when used in deep coupling layers which are required to capture multivariate correlations.

Smooth flows on S^1 Flows on the hypertorus were discussed in Rezende et al. [34] who introduced necessary conditions for the transformations to define C^0-continuous densities, such that they can be used in density estimation tasks. In their work Rezende et al. [34] introduced three candidates satisfying this condition: mixtures of non-compact projections (NCP), rational-quadratic circular spline flows (CSF) and mixtures of Moebius transformations (MoMT). While NCPs and CSFs satisfy C^1-continuity and thus render forces discontinuous, only MoMTs define smooth densities. While MoMTs trivially also define C^∞-flows on $\mathbb{I}$ their periodicity would be limiting when applied to non-periodic densities.

Smooth compact bump functions In addition to this previous work we leverage a third way to construct smooth transformations which works for both $\mathbb{I}$ and S^1. Here we follow a general construction principle for smooth bump functions e.g. as explained in Tu [39]. All proofs can be found in the supplementary material.

First, define a C^k-smooth and strictly increasing ramp function $\rho\colon \mathbb{I} \to \mathbb{I}$ with $\rho(0) = 0$ and $\rho(1) = 1$. Second, define the generalized sigmoid

$$\sigma[\rho](x) := \frac{\rho(x)}{\rho(x) + \rho(1-x)}. \tag{6}$$

Then $\sigma[\rho]$ will be a C^k-diffeomorphsim on $\mathbb{I}$ and furthermore all derivatives up to order k vanish at the boundary. The first derivative defines a non-negative smooth bump function with support $[0, 1]$ and maximum at 0.5. We can introduce a concentration parameter $a \in \mathbb{R}_{>0}$ and a location parameter $b \in \mathbb{R}$, which makes $g(x) := \sigma[\rho]\left(a \cdot (x - b) + \frac{1}{2}\right)$ a smooth unimodal bijection with all higher-order derivatives having support $[b - \frac{1}{a}, b + \frac{1}{a}]$. Furthermore, by introducing $c \in (0, 1]$ and setting

$$f(x) := (1 - c) \cdot \left(\frac{g(x) - g(0)}{g(1) - g(0)}\right) + c \cdot x, \tag{7}$$

we can define flexible unimodal C^k-diffeomorphisms on $[0, 1]$. When used within coupling layers, we let a, b, c be the output of some not further constrained neural network.

C^k and C^∞ ramp functions There are many possible choices for ramp functions satisfying above mentioned properties. A simple C^k ramp function is given by the k-th order monomial $\rho(x) = x^k$. More interestingly, C^∞ smoothness on $[0, 1]$ can be achieved using the ramp $\rho(x) = \exp\left(-\frac{1}{\alpha \cdot x^\beta}\right)$ for $x > 0$ and $\rho(x) = 0$ for $x \leq 0$. Here we set $\alpha > 0$ and $\beta > 1$. We make α a trainable parameter. Optimizing β is possible in principle, yet fixing $\beta \in \{1, 2\}$ and treating it as a hyper-parameter stabilized training and led to better results.

Smooth and efficient circular wrapping As discussed in Rezende et al. [34] we can turn any C^k-smooth density $p(x)$ with support on $\mathbb{R}$ into a C^k-smooth density on S^1 using the marginalization $\tilde{p}(x) = \sum_{k \in \mathbb{Z}} p(x + k)$. This construction is generally problematic due to two reasons: (i) non-vanishing tails (e.g. if $p(x)$ is Gaussian) require evaluating an infinite series which in most cases has no analytic expression and can be numerically challenging (ii) smooth densities with compact support in some interval $[a, b]$, for example sigmoidal transforms, can introduce non-smooth behavior at the interval boundaries.

Smooth and compactly supported transformations as introduced above do not suffer from this: (i) due to compact support the series will always have a finite number of non-vanishing contributions, (ii) as the transformations are C^k on all of $\mathbb{R}$ and all derivatives are compactly supported wrapping will always result in a C^k-smooth density on S^1.

Multi-modality via mixtures Similarly, as discussed in prior work [18, 34] any convex sum of C^k-diffeomorphisms on $\mathbb{I}$ defines a C^k-diffeomorphism on $\mathbb{I}$. Thus we can combine multiple of those unimodal transformations defined in Eq. (7) to obtain arbitrarily complex multi-modal transformations on $\mathbb{I}$ and S^1, see Fig. 1.

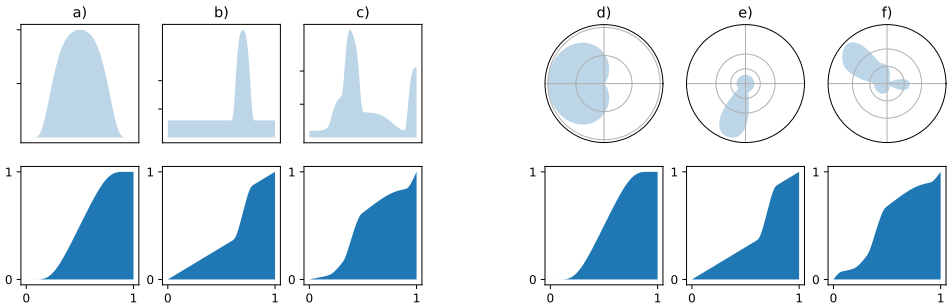

Figure 1: Construction of mixture transformations on compact intervals (left) and hypertori (right). The upper and lower row depict probability densities and cumulative distribution functions, respectively. Multiple unimodal bump functions [a) and d)] are added to a small but finite density [b) and e)] and combined to yield bijective multimodal transformations [c) and f)].

5 Optimizing non-analytic inverse flows

A drawback of mixture-based flows is their lack of an analytic inverse. Prior work such as [18, 34] suggested to rely on black-box inversion methods like a bisection search to sample from trained models. While this leads to accurate samples, the discrete nature of such black-box oracles does not allow to minimize $\mathcal{L}(\boldsymbol{\theta})$ as defined in Eq. (5).

A possible remedy to this can be derived using the *inverse function theorem*. Let $f(\cdot;\boldsymbol{\theta})\colon \Omega \subset \mathbb{R} \to \Omega$ be a scalar diffeomorphism and let furthermore $x = f^{-1}(y;\boldsymbol{\theta})$ be obtained via a black-box inversion algorithm. To minimize losses depending on $x(y;\boldsymbol{\theta})$ or $\log|\partial_y x(y;\boldsymbol{\theta})|$ we need to compute gradients with respect to y and $\boldsymbol{\theta}$. Here we derive the following relations for the derivatives (all details in supplementary material):

$$\partial_y x(y;\boldsymbol{\theta}) = \left(\partial_x f(x;\boldsymbol{\theta})\right)^{-1} \tag{8}$$

$$\partial_{\boldsymbol{\theta}} x(y;\boldsymbol{\theta}) = -\left(\partial_x f(x;\boldsymbol{\theta})\right)^{-1} \partial_{\boldsymbol{\theta}} f(x;\boldsymbol{\theta}) \tag{9}$$

$$\partial_y \log|\partial_y x(y;\boldsymbol{\theta})| = -\left(\partial_x f(x;\boldsymbol{\theta})\right)^{-1} \log|\partial_x f(x;\boldsymbol{\theta})| \tag{10}$$

$$\partial_{\boldsymbol{\theta}} \log|\partial_y x(y;\boldsymbol{\theta})| = -\left(\partial_x f(x;\boldsymbol{\theta})\right)^{-1} \left(\log|\partial_x f(x;\boldsymbol{\theta})|\, \partial_{\boldsymbol{\theta}} f(x;\boldsymbol{\theta}) - \partial_{\boldsymbol{\theta}}\partial_x f(x;\boldsymbol{\theta})\right) \tag{11}$$

Using these rules, we can first compute x via a black-box oracle for any input y and then compute all necessary gradients using forward evaluations and derivatives of $f(\cdot;\boldsymbol{\theta})$. Derivatives of $f(\cdot;\boldsymbol{\theta})$ are usually accessible for computing the log jacobian of the transformation. We can obtain higher-order derivatives using automatic differentiation. It is easy to see that this construction extends to multivariate diffeomorphisms with diagonal Jacobian as used in coupling layers. It can be generalized to arbitrary higher order derivatives using the Faà di Bruno formula, e.g. when aiming to do force matching through a black-box inversion algorithm. As our experiments only require losses involving second order derivatives for the inverse direction we leave this for future work. Note that Equations (8) - (11) can be applied to any other element-wise flow transformation that lacks an analytic inverse and is not tied to mixture models.

Another problem of bisection is its sequential execution which can become prohibitively slow during training. E.g. achieving an error of `1e-6` requires around 20 iterations. To obtain speedup on GPUs we suggest to generalize the classic binary search to searching in a grid of K bins simultaneoulsy (see details in Supp. Mat.). For $K = 2$ it results in the usual bisection method. However, by leveraging vectorized execution we can reduce the number of iterations by a factor $O\left(1/\log K\right)$ at the expense of increasing memory by a factor $O\left(K\right)$. Practical speedup depends on the actual number of parallel workers and thus depending on dimensionality, batch size and number of mixture components the optimal choice of K varies.

6 Experiments

The numerical experiments in this work are tailored to show the benefits of smooth flows over non-smooth flows. Therefore, we mainly compare mixtures of bump functions to neural spline flows [13], which exhibit state-of-the-art generative performance but whose densities are only first-order continuously differentiable.

6.1 Illustrative Toy Example

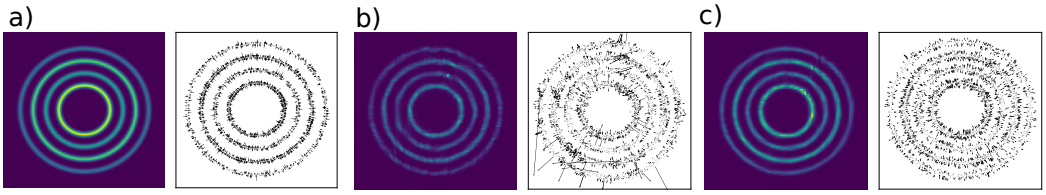

Figure 2: a) Reference density and corresponding force field as approximated by b) a C^1-smooth NSF and c) a C^∞-smooth flow using the mixture of bump functions introduced in Sec. 4.

To highlight the difference in terms of smoothness, the two flow architectures were applied to a two-dimensional toy example. Fig. 2 a) depicts the reference energy and forces on samples from the ground-truth distribution. Both flows were trained through density estimations on those samples (see SI for details). The smooth flows and spline flows matched the density well, which demonstrates their expressivity. However, the force field of the spline flow contained dramatic outliers, while the forces of the smooth flow matched the regular behavior of the reference forces.

6.2 Runtime Comparison

While the rational-quadratic splines are analytically invertible, mixture transformations obviously increase computational cost of the inversion due to the iterative root-finding procedure. To quantify this gap in performance, the inverse evaluation of a spline flow was compared with a modified version, where the analytic inverse was replaced by the multi-bin bisection from Section 5. The performance of the bisection was evaluated for different numbers of bins $K = 2^m,\ m = 1, \ldots, 8$, and the most efficient K was picked for each input size ("dim"). Both transformations were coupled to a conditioner with the same input size (dim) and 64 hidden neurons.

For small tensor dimensions, the optimal multi-bin bisection employed 128 bins on the GPU, which resulted in only a factor of 2-3 slowdown compared to analytic inversion. For larger dimensions, the parallelization over multiple bins became less effective, leading to one order of magnitude difference in computational cost. The compute times and optimal bin sizes were comparable for the inverse network pass and its backward-pass. More details on these runtime comparisons can be found in the supplementary material.

6.3 Smooth Flows Enable Boltzmann Generator Training through Force Matching

To demonstrate the advantages of smooth flows, we train a Boltzmann generator (BG) for a small molecule, alanine dipeptide, which is described in the supplementary material. This is a common test case for molecular computations and was previously considered for sampling with normalizing flows in [43, 9, 23]. The molecular system has 60 degrees of freedom (global rotation and translation excluded). Its potential energy surface is highly sensitive to small displacements [23] and contains singularities.

In previous work [43, 9, 23], BGs for alanine dipeptide used stochastic augmentation of the base space. The introduction of noise variables facilitates creating expressive normalizing flows but prevents computation of deterministic forces. Some have also used spline flows to sample molecular configurations [42, 10].

As a proof-of-concept for the force matching loss, we trained smooth flows for alanine dipeptide using a 1000:1 weighting between the force matching residual and the negative log likelihood, see

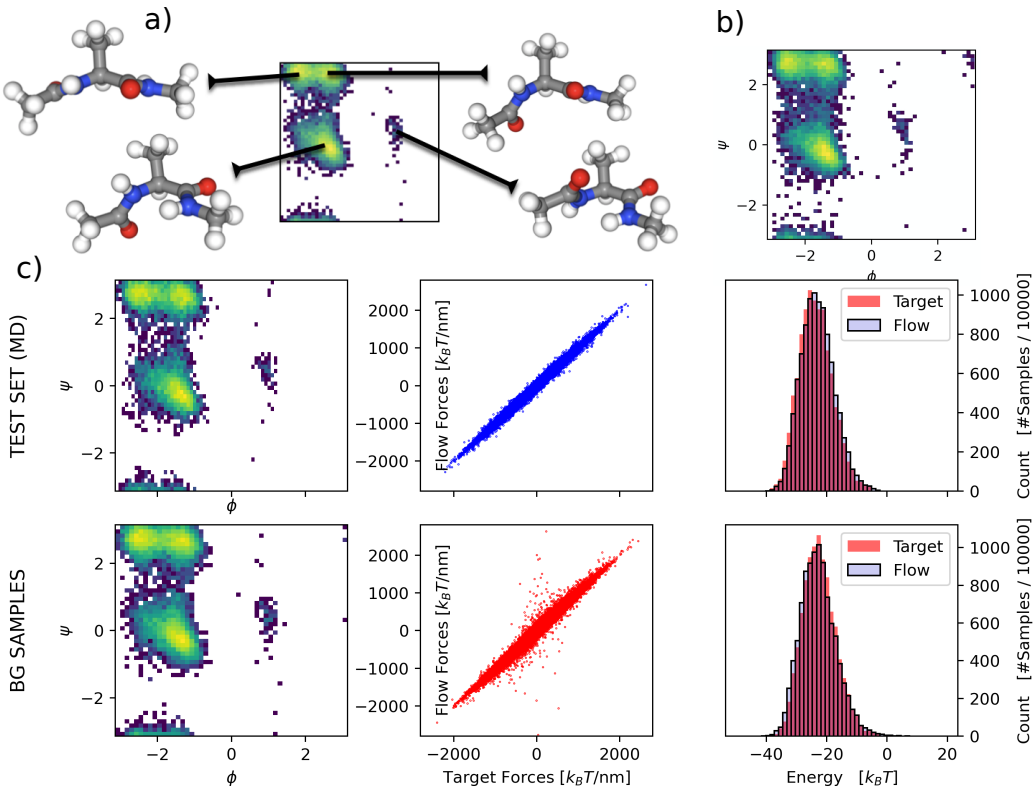

Figure 3: a) Generated sample structures. b) Torsion distribution after training through root-finding. c) Smooth normalizing flow trained on alanine dipeptide through a combination of force matching and density estimation. The top and bottom row show the performance on 10,000 samples each from the holdout test set and from the flow, respectively. Left: joint distribution of backbone torsion angles. Center: scatter plot of flow vs. target force components. Right: energy histograms for the flow energy and the target energy. (Flow energies were shifted by a constant so that the minimum energy matched with the minimum energy from the molecular potential).

supplementary material for details on the flow and training setup. No stochastic augmentation or energy-based training was used and no reweighting was conducted for postprocessing.

Figure 3 c) compares the flow distribution with the target Boltzmann distribution on the test data from MD and on flow samples. The left column shows that the flow has almost perfectly learned the nontrivial joint distribution between the two backbone torsion angles ϕ and ψ, including the sparsely populated metastable region around $\phi \approx 1$. In contrast to previous work that used affine coupling layers [43], the modes are cleanly separated. The center column compares flows forces with target forces. Those forces are highly sensitive to small perturbations of the molecular configurations. Nevertheless, the forces matched up to a root-mean square deviation of 25 and 46 k_BT/nm in the target ensemble and the generated ensemble, respectively. Note that the training was stopped after 10 epochs, where the validation force matching residual had not yet fully converged, so that even further improvements may be possible with longer training or extensive hyperparameter optimization.

Finally, the right column depicts the distribution of flow and target energies evaluated on the same samples. The flow energies were shifted so that the minimum energy matched with the molecular mechanics energy. It has not escaped our notice that this constant offset (189 k_BT on the test set and 190 k_BT on the flow samples) corresponds to the log partition function, a quantity whose intractability has complicated research in statistical mechanics for decades.

The energy distribution of the flow tracked the ground truth exponential distribution almost perfectly even though no target energies or forces were evaluated on the flow samples during training. This close-to-perfect match demonstrates that including force information into the training process presents

an efficient means for regularization. Figure 3 a) shows representative conformations generated by the BG, which confirm the high quality of the molecular structures.

6.4 Boltzmann Generator Training through Black-Box Root-Finding

Table 1: Test set metrics of alanine dipeptide training with different losses: negative log likelihood (NLL), force matching error (FME), and reverse Kullback-Leibler divergence (KLD). Weighted loss functions were used as defined in Eq. (5) with weight factors ω_k, ω_f and $\omega_n = 1 - \omega_k - \omega_\mathrm{f}$. Metrics were recorded after 10 training epochs. Statistics are means and $2\times$ standard errors over 10 replicas for each experiment. The lowest value with respect to each metric is highlighted in bold type.

Metric	Method	$\omega_k = 0$ $\omega_\mathrm{f} = 0$	$\omega_k = 0$ $\omega_\mathrm{f} = 0.001$	$\omega_k = 0.1$ $\omega_\mathrm{f} = 0$	$\omega_k = 0.1$ $\omega_\mathrm{f} = 0.001$
NLL	spline	-210.32 (±0.16)	-	-196.40 (± 1.53)	48.13 (± 95.37)
	smooth	-211.04 (± 0.09)	**-211.40** (± 0.05)	-206.33 (±2.04)	-208.70 (± 1.93)
FME $\times 10^4$	spline	12.13 (± 4.94)	-	313.64 (± 107.92)	1498.98 (± 1935.72)
	smooth	1.04 (± 0.08)	**0.32** (± 0.02)	5.80 (± 2.30)	0.48 (± 0.05)
KLD	spline	263.77 (± 3.21)	-	230.08 (± 12.38)	1205.69 (± 36.25)
	smooth	**193.91** (± 0.39)	195.17 (± 0.43)	219.03 (± 22.16)	207.81 (± 11.64)

Two experiments were conducted to demonstrate that flows can be trained through a black-box root-finding algorithm.

First, the BG from section 6.3 was trained with all smooth transformations operating in the inverse direction. Consequently, inverse problems had to be solved when computing the negative log likelihood with respect to data, while the sampling occurred analytically. Figure 3 b) shows the joint marginal distribution of the backbone torsions on the BG samples. The distribution matches the data and the smooth flows that were trained in forward mode (see Fig. 3). This result shows that training with gradients computed from the inverse function theorem is feasible.

Second, flows were trained by different combinations of density estimation, force matching, and energy-based training, where computing $\partial_{\boldsymbol{\theta}} \mathcal{L}_\mathrm{KLD}$ requires the gradients of the inverse flow. The training of spline flows with inclusion of the force matching error ($\omega_\mathrm{f} > 0$) was notoriously unstable. This led to 10/10 and 7/10 failed runs for $\omega_k = 0$ and $\omega_k = 0.1$, respectively. This is understandable from the discontinuity of the spline forces. Even the combination of NLL and KLD led to instabilities for 2/10 runs. (Gradients were not clipped during training to enable a mostly unbiased comparison). In contrast, all training runs with our smooth flows concluded successfully.

Table 1 shows the metrics on the test set after 10 training epochs using spline flows and smooth flows (see SI for specifics about the flow architecture and training). With pure density estimation ($\omega_k = \omega_\mathrm{f} = 0$), both architectures achieved similar NLL. However, the smooth flow achieved much lower FME and KLD. The NLL and FME were further improved when force data was presented to the flow during training. Including reverse KLD in the loss yielded reasonable metrics for the smooth flows, indicating that the backpropagation through root-finding was numerically stable. However, the metrics were consistently worse than with pure NLL training for both types of flows (spline and smooth) and were subject to large fluctuations. In contrast, including the FME proved to be a more stable approach to include information about the target energy into the training.

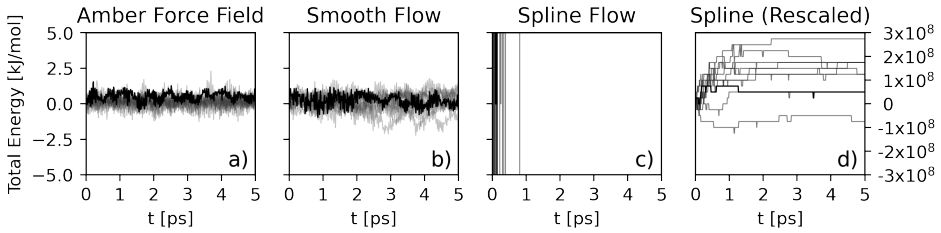

Figure 4: Total energy fluctuation in simulations with a classical force field and with two flows that employed smooth transforms and neural spline transforms, respectively. For each potential, 10 simulations were run starting from 10 different initial configurations (grey color). One run each is highlighted in black. Subfigures c) and d) show the same data on different scales.

6.5 Smooth Flows as Potentials in Molecular Dynamics

A challenging test for the smoothness of a flow is its use as a potential energy in a dynamics simulation. Especially simulations in the microcanonical ensemble are extremely sensitive to numerical errors as the symplectic integration does not impose any additional stabilizing mechanism on the total energy. This means that any inconsistencies between energies and forces on the scale of one integration step easily cause drifting, strongly fluctuating, or exploding energies.

Therefore, we ran MD simulations using the flow energy $u_\theta = -\log p(\cdot, \boldsymbol{\theta})$, from Section 6.3 as the potential. For comparison, spline flows were trained purely by density estimation. Simulations started from stable structures of an MD simulation and were equilibrated in each potential for 1 ps using a Langevin thermostat with 10/ps friction coefficient.

Figure 4 depicts the evolution of the total energy over 5 ps simulations in the microcanonical ensemble. As expected, the classical simulation kept the total energy roughly constant within a standard deviation of 6×10^{-4} kJ/mol per degree of freedom. Astoundingly, the smooth flow potentials also maintained the energies within 8×10^{-4} kJ/mol per degree of freedom using the same 1 fs time step. In contrast, the simulations with spline flow potentials quickly fell apart with potential and kinetic energies growing out of bounds. Those instabilities persisted even with an order-of-magnitude smaller time step (not shown).

While the infeasibility of spline flows for this task was expected, the competitive behavior of smooth flows with molecular mechanics force fields that were tailored for dynamics highlights their regularity and the consistency of their forces.

7 Discussion

In this work we introduced a range of contributions which can improve upcoming work on applying flows to physical problems.

The proposed approach to backpropagation through black-box root-finders can help to obviate the search for analytically invertible transformations if bi-directional training is necessary. As we show the method works especially well for relatively low-dimensional problems. Scaling the approach, generalizing it to non-diagonal Jacobians and improving its numerics are interesting questions for future work.

The MD simulation example shows that C^∞-smooth flows on non-trivial topologies can open new avenues of research as well as new applications for flows. By carefully respecting the topological domain as well as the smoothness of the target potential we can approximate the equilibrium density of a small peptide nearly perfectly. Finally, we showed that incorporating force information together with MLE outperforms the state-of-the-art approach of combining MLE with minimizing the reverse KL divergence. It does not suffer from mode-collapse, yet includes information of the target potential which is required to achieve low-energy samples.

Combined, these results could improve methods used for learning the mean potential surface of coarse-grained molecules, pave the way to multi-scale flows for modeling large protein systems and eventually help to accelerate simulations and sampling.

References

[1] M. Albergo, G. Kanwar, and P. Shanahan. Flow-based generative models for markov chain monte carlo in lattice field theory. *Physical Review D*, 100(3):034515, 2019.

[2] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32:690–701, 2019.

[3] J. Bose, A. Smofsky, R. Liao, P. Panangaden, and W. Hamilton. Latent variable modelling with hyperbolic normalizing flows. In *International Conference on Machine Learning*, pages 1045–1055. PMLR, 2020.

[4] D. Boyda, G. Kanwar, S. Racanière, D. J. Rezende, M. S. Albergo, K. Cranmer, D. C. Hackett, and P. E. Shanahan. Sampling using su (n) gauge equivariant flows. *Physical Review D*, 103(7): 074504, 2021.

[5] J. Brehmer and K. Cranmer. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33, 2020.

[6] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[7] R. Cornish, A. Caterini, G. Deligiannidis, and A. Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International Conference on Machine Learning*, pages 2133–2143. PMLR, 2020.

[8] N. De Cao, W. Aziz, and I. Titov. Block neural autoregressive flow. In *Uncertainty in Artificial Intelligence*, pages 1263–1273. PMLR, 2020.

[9] M. Dibak, L. Klein, and F. Noé. Temperature-steerable flows. *arXiv preprint arXiv:2012.00429*, 2020.

[10] X. Ding and B. Zhang. Deepbar: A fast and exact method for binding free energy computation. *The Journal of Physical Chemistry Letters*, 12(10):2509–2515, 2021.

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

[12] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=HkpbnH9lx`.

[13] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7509–7520, 2019.

[14] L. Falorsi. Continuous normalizing flows on manifolds. *arXiv preprint arXiv:2104.14959*, 2021.

[15] L. Falorsi, P. de Haan, T. R. Davidson, and P. Forré. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3244–3253. PMLR, 2019.

[16] M. C. Gemici, D. Rezende, and S. Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

[17] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. 6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings. OpenReview.net, 2018. URL `https://openreview.net/group?id=ICLR.cc/2018/Conference`.

[18] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[19] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

[20] B. E. Husic, N. E. Charron, D. Lemm, J. Wang, A. Pérez, M. Majewski, A. Krämer, Y. Chen, S. Olsson, G. de Fabritiis, et al. Coarse graining molecular dynamics with graph neural networks. *The Journal of Chemical Physics*, 153(19):194101, 2020.

[21] P. Jaini, K. A. Selby, and Y. Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pages 3009–3018. PMLR, 2019.

[22] J. Köhler, L. Klein, and F. Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pages 5361–5370. PMLR, 2020.

[23] A. Krämer, J. Köhler, and F. Noé. Training neural networks with property-preserving parameter perturbations. *arXiv preprint arXiv:2010.07033*, 2020.

[24] S.-H. Li and L. Wang. Neural network renormalization group. *Physical review letters*, 121(26): 260601, 2018.

[25] A. Lou, D. Lim, I. Katsman, L. Huang, Q. Jiang, S. N. Lim, and C. M. De Sa. Neural manifold ordinary differential equations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17548–17558. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/cbf8710b43df3f2c1553e649403426df-Paper.pdf`.

[26] E. Mathieu and M. Nickel. Riemannian continuous normalizing flows. *Advances in Neural Information Processing Systems*, 33, 2020.

[27] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák. Neural importance sampling. *ACM Transactions on Graphics (TOG)*, 38(5):1–19, 2019.

[28] K. A. Nicoli, S. Nakajima, N. Strodthoff, W. Samek, K.-R. Müller, and P. Kessel. Asymptotically unbiased estimation of physical observables with neural samplers. *Physical Review E*, 101(2): 023304, 2020.

[29] K. A. Nicoli, C. J. Anders, L. Funcke, T. Hartung, K. Jansen, P. Kessel, S. Nakajima, and P. Stornati. Estimation of thermodynamic observables in lattice field theories with deep generative models. *Physical review letters*, 126(3):032001, 2021.

[30] F. Noé, S. Olsson, J. Köhler, and H. Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

[31] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL `http://jmlr.org/papers/v22/19-1028.html`.

[32] S. Ramasinghe, K. Fernando, S. Khan, and N. Barnes. Robust normalizing flows using bernstein-type polynomials. *arXiv preprint arXiv:2102.03509*, 2021.

[33] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[34] D. J. Rezende, G. Papamakarios, S. Racaniere, M. Albergo, G. Kanwar, P. Shanahan, and K. Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pages 8083–8092. PMLR, 2020.

[35] V. G. Satorras, E. Hoogeboom, F. B. Fuchs, I. Posner, and M. Welling. E (n) equivariant normalizing flows for molecule generation in 3d. *arXiv preprint arXiv:2105.09016*, 2021.

[36] L. Sbailò, M. Dibak, and F. Noé. Neural mode jump monte carlo. *The Journal of Chemical Physics*, 154(7):074101, 2021.

[37] S. Shirobokov, V. Belavin, M. Kagan, A. Ustyuzhanin, and A. G. Baydin. Black-box optimization with local generative surrogates. *Advances in Neural Information Processing Systems*, 33, 2020.

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

[39] L. W. Tu. *Bump Functions and Partitions of Unity*, pages 127–134. Springer New York, New York, NY, 2008. ISBN 978-0-387-48101-2. doi: 10.1007/978-0-387-48101-2_13. URL `https://doi.org/10.1007/978-0-387-48101-2_13`.

[40] J. Wang, S. Olsson, C. Wehmeyer, A. Pérez, N. E. Charron, G. De Fabritiis, F. Noé, and C. Clementi. Machine learning of coarse-grained molecular dynamics force fields. *ACS central science*, 5(5):755–767, 2019.

[41] A. Wehenkel and G. Louppe. Unconstrained monotonic neural networks. *Advances in Neural Information Processing Systems 32 (2019)*, 2019.

[42] P. Wirnsberger, A. Ballard, G. Papamakarios, S. Abercrombie, S. Racanière, A. Pritzel, C. Blundell, et al. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153(14):144112–144112, 2020.

[43] H. Wu, J. Köhler, and F. Noe. Stochastic normalizing flows. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5933–5944. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/41d80bfc327ef980528426fc810a6d7a-Paper.pdf`.

[44] C. Xenophontos. A formula for the nth derivative of the quotient of two functions. `http://www.mas.ucy.ac.cy/~xenophon/pubs/capsule.pdf`. Accessed: 2021-05-28.

[45] M. Xu, S. Luo, Y. Bengio, J. Peng, and J. Tang. Learning neural generative dynamics for molecular conformation generation. *arXiv preprint arXiv:2102.10240*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This work is merely technical and studies smooth normalizing flows applied to physics research.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All mathematical details can be found in the supplementary material.
   (b) Did you include complete proofs of all theoretical results? [Yes] All mathematical details can be found in the supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] They can be found in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Training details are partly explained in the experimental part. More details can be found in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Details regarding compute requirements can be found in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We report used libraries and used code of prior works in the supplementary material.
   (b) Did you mention the license of the assets? [No] All used code that is not written by us has been published under non-restrictive open source licenses
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No personal data are used in this work.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No crowdsourcing / human subject research was conducted in this work.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No crowdsourcing / human subject research was conducted in this work.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No crowdsourcing / human subject research was conducted in this work.