BoltzNCE: Learning Likelihoods for Boltzmann Generation with Stochastic Interpolants and Noise Contrastive Estimation

Rishal Aggarwal
CMU-Pitt Computational Biology
Dept. of Computational & Systems Biology
University of Pittsburgh
Pittsburgh, PA 15260
rishal.aggarwal@pitt.edu

Jacky Chen
CMU-Pitt Computational Biology
Dept. of Computational & Systems Biology
University of Pittsburgh
Pittsburgh, PA 15260
jackychen@pitt.edu

Nicholas M. Boffi
Machine Learning Dept.
Dept. of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA 15213
nboffi@andrew.cmu.edu

David Ryan Koes
Dept. of Computational & Systems Biology
University of Pittsburgh
Pittsburgh, PA 15260
dkoes@pitt.edu

Abstract

Efficient sampling from the Boltzmann distribution given its energy function is a key challenge for modeling complex physical systems such as molecules. Boltzmann Generators address this problem by leveraging continuous normalizing flows to transform a simple prior into a distribution that can be reweighted to match the target using sample likelihoods. Despite the elegance of this approach, obtaining these likelihoods requires computing costly Jacobians during integration, which is impractical for large molecular systems. To overcome this difficulty, we train an energy-based model (EBM) to approximate likelihoods using both noise contrastive estimation (NCE) and score matching, which we show outperforms the use of either objective in isolation. On 2D synthetic systems where failure can be easily visualized, NCE improves mode weighting relative to score matching alone. On alanine dipeptide, our method yields free energy profiles and energy distributions that closely match those obtained using exact likelihoods while achieving $100\times$ faster inference. By training on multiple dipeptide systems, we show that our approach also exhibits effective transfer learning, generalizing to new systems at inference time and achieving at least a $6\times$ speedup over standard MD with only a bit of fine-tuning. While many recent efforts in generative modeling have prioritized models with fast *sampling*, our work demonstrates the design of models with accelerated *likelihoods*, enabling the application of reweighting schemes that ensure unbiased Boltzmann statistics at scale. Our code is available at https://github.com/RishalAggarwal/BoltzNCE.

1 Introduction

Obtaining the equilibrium distribution of molecular conformations defined by an energy function is a fundamental problem in the physical sciences [1–3]. The Boltzmann distribution describes the equilibrium probability density and is given by $p(x) \propto \exp(-E(x)/K_B T)$ where $E(x)$ is the energy of molecular conformer x, K_B is the Boltzmann constant, and T is temperature. Sampling from this distribution is particularly difficult for molecular systems due to the non-convex nature of the energy

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

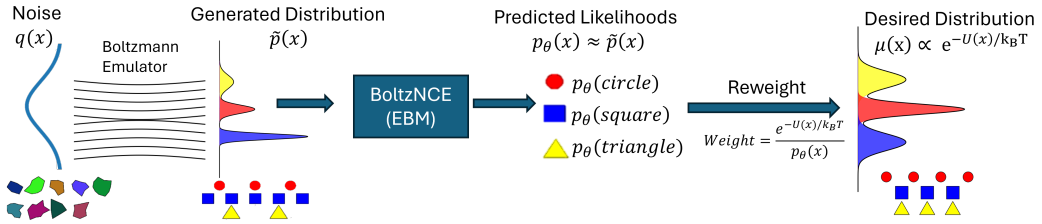

Figure 1: **Overview.** BoltzNCE offers an accelerated alternative to exact model likelihood computation. Samples from a prior are first transformed to a distribution of conformers by a Boltzmann Emulator, which is easy to sample from but difficult to evaluate likelihoods for. The generated samples are then reweighted with likelihoods estimated by an energy-based model (EBM), which we train to approximate the emulator distribution with a hybrid score matching and noise contrastive estimation scheme. This EBM gives access to likelihoods in a single function call, enabling us to reweight to the target Boltzmann distribtion up to $100\times$ faster than exact computation.

landscape, leading to the presence of widespread energy basins, metastability, and slow transitions. Traditional approaches for sampling conformers, such as Markov chain Monte Carlo (MCMC) and molecular dynamics (MD) [4, 5], often get trapped in these energy wells, which necessitates long simulation timescales to produce uncorrelated samples [6]. Consequently, it is particularly inefficient to obtain samples from independent metastable states.

In recent years, several generative deep learning methods have been proposed to address the molecular sampling problem. One prominent class is Boltzmann Generators (BGs) [7–10], which transform a simple prior distribution (e.g., a multivariate Gaussian) into a distribution over molecular conformers that can be reweighted to approximate the Boltzmann distribution. When reweighting is not applied, the model is referred to as a Boltzmann Emulator [8], whose primary aim is to efficiently sample metastable states of the molecular ensemble. While often qualitatively reasonable, Boltzmann Emulators alone fail to recover the true Boltzmann distribution and require the reweighting step for exact recovery of the system's equilibrium statistics.

To compute the likelihoods of generated samples, BGs are constrained to the class of normalizing flows. While earlier methods built these flows using invertible neural networks [7, 11], more recent approaches prefer using continuous normalizing flows (CNFs) [8, 12] due to their enhanced expressitivity and flexibility in model design. Despite these advantages, computing likelihoods for CNF-generated samples requires expensive Jacobian trace evaluations along the integration path [13, 14]. This computational overhead limits their scalability, particularly for large-scale protein systems. In this work, we ask the question:

Can the likelihood of large-scale scientific generative models be efficiently amortized to avoid the prohibitive path integral?

Here, as a means to learn such a likelihood surrogate, we investigate the use of energy-based models (EBMs). EBMs learn the energy function of a synthetic Boltzmann distribution, $p_\theta(x) \propto \exp(E_\theta(x))$ where E_θ is the energy to be learned [15]. Scalable training of EBMs remains a major challenge due to the need for sampling from the model distribution, which often requires simulation during training [16, 17]. As a result, developing efficient training algorithms for EBMs continues to be an active area of research [15, 18–20].

We adopt noise contrastive estimation (NCE) [21], which trains a classifier to distinguish between samples drawn from the target distribution and those from a carefully chosen noise distribution. The key advantage of this approach is that it circumvents the need to compute intractable normalizing constants [22, 23]. Despite this, NCE can suffer from the *density-chasm* problem [24, 25], whereby the optimization landscape becomes flat when the data and noise distributions differ significantly [26]. To address this issue, we introduce an annealing scheme between a simple noise distribution and the data distribution using stochastic interpolants [27]. Annealing mitigates the density chasm problem by introducing intermediate distributions between noise and data, which facilitates the generation of more informative samples. In particular, it ensures that negative samples lie closer to positive samples, improving the effectiveness of NCE optimization [24, 28]. We further enhance the training process by combining an InfoNCE [29] loss with a score matching [30] objective defined over the law of the

stochastic interpolant. Notably, our proposed method for training the EBM is both *simulation-free* and avoids the computation of normalizing constants, making it scalable to large systems.

Contributions. On synthetic 2D systems, we show that training with both losses performs significantly better than either individually. For the alanine dipeptide system, our method recovers the correct semi-empirical free energy surface while achieving a $100\times$ speedup over exact likelihood calculations. On other dipeptide systems, we further demonstrate the method's ability to generalize to previously unseen molecular systems. To summarize, our *main contributions* are:

- **Training.** We develop a scalable, simulation-free framework for training EBMs by combining stochastic interpolants, score matching, and noise contrastive estimation.
- **Fast likelihoods.** We show that learned likelihoods can replace expensive Jacobian computations in the reweighting step, recovering exact Boltzmann statistics.
- **Empirical validation.** We achieve $100\times$ speedup on alanine dipeptide compared to exact likelihoods, and demonstrate $6\times$ speedup over MD on unseen dipeptide systems.

2 Methodological Framework

In this work, we introduce a new class of generative models for sampling from a Boltzmann distribution that features accelerated likelihood computations (Figure 1). To this end, we train a standard Boltzmann Emulator and an EBM, which each enable efficient sampling and likelihood evaluation, respectively. Given access to an EBM approximating the output distribution of the Boltzmann Emulator, we can evaluate sample likelihoods in just a single function call, enabling rapid reweighting for accurate estimation of observables. We train these EBM models with a new hybrid approach that we call *BoltzNCE*. As critical components of our approach, we first provide background on stochastic interpolants and flow matching [27, 31], then show how these can be used to build Boltzmann Emulators and finally, we introduce the innovations underlying the BoltzNCE method.

2.1 Background: stochastic interpolants and Boltzmann emulators

A stochastic interpolant [27, 32] is a stochastic process that smoothly deforms data from a fixed base distribution ρ_0 into data sampled from the target distribution $\rho_1 = p^*$. Under specific choices of the hyperparameters, stochastic interpolants recover standard settings of diffusion models [30, 33], flow matching [31], and rectified flows [34]. They can be used to learn generative models, because they provide access to time-dependent samples along a dynamical process that converts samples from the base into samples from the target. Concretely, given samples $\{x_1^i\}_{i=1}^n$ with $x_1^i \sim \rho_1$ sampled from the target, we may define a stochastic interpolant as the time-dependent process

$$I_t = \alpha_t x_0 + \beta_t x_1 \tag{1}$$

Above, $\alpha : [0,1] \to \mathbb{R}$ and $\beta : [0,1] \to \mathbb{R}$ are continuously differentiable functions satisfying the boundary conditions $\alpha_0 = 1, \alpha_1 = 0, \beta_0 = 0$, and $\beta_1 = 1$. In the absence of domain knowledge, we often take the base distribution to be a standard Gaussian, $\rho_0 = \mathsf{N}(0, I)$.

The probability density $\rho_t = \mathsf{Law}(I_t)$ induced by the interpolant coincides with the density of a probability flow that pushes samples from ρ_0 onto ρ_1,

$$\dot{x}_t = b_t(x_t), \qquad b_t(x) = \mathbb{E}[\dot{I}_t \mid I_t = x], \tag{2}$$

where $b_t(x)$ is given by the conditional expectation of the time derivative of the interpolant at a fixed point in space. The score $s_t(x) = \nabla \log \rho_t(x)$ is further given by the conditional expectation [27]:

$$s_t(x) = \alpha_t^{-1} \mathbb{E}\left[x_0 \mid I_t = x\right], \tag{3}$$

which we will use to train our energy-based model as a likelihood surrogate.

Flow matching. Given a coupling $\rho(x_0, x_1)$ between ρ_0 and ρ_1, the vector field (2) can be approximated with a neural network $\hat{b}$ via the regression problem

$$\mathcal{L}_b(\hat{b}) = \mathbb{E}\left[\|\hat{b}_t(I_t) - \dot{I}_t\|^2\right], \tag{4}$$

where $\mathbb{E}$ denotes an expectation over (t, x_0, x_1). The objective (4) can be further modified so that the model is trained to predict the clean data x_1 instead of the vector field $\hat{b}$ [35, 36]. We refer the reader to Appendix A for more details on the endpoint objective.

Once trained, the learned model can be used to generate samples $\tilde{x}_0$ by solving the differential equation

$$\dot{\hat{x}}_t = \hat{b}_t(\hat{x}_t), \qquad x_0 \sim \rho_0. \tag{5}$$

The log density $\log \hat{\rho}$ associated with the generated samples $\hat{x}_1$ can be calculated with the continuous change of variables formula,

$$\log \hat{\rho}(\hat{x}_1) = \log \rho_0(x_0) - \int_0^1 \nabla \cdot \hat{b}_t(\hat{x}_t) dt. \tag{6}$$

While (6) gives a formula for the exact likelihood, its computation is expensive due to the appearance of the divergence of the estimated flow $\hat{b}$.

Boltzmann Generators. Boltzmann Generators leverage generative models to sample conformers and to compute their likelihoods, which enables reweighting the generated samples to the Boltzmann distribution. In practice, these models can be built using the stochastic interpolant framework described in (1), (2) and (4) given target data generated via molecular dynamics. We can generate unbiased samples from the target distribution by first sampling $\hat{x}_1 \sim \hat{\rho}_1$ by solving (5) and (6), and then by reweighting with the importance weight $w(\hat{x}_1) = \exp(\frac{-E(\hat{x}_1)}{K_B T})/\hat{\rho}_1(\hat{x}_1)$. With this weight, we can also approximate any observable O under the Boltzmann distribution μ using self-normalized importance sampling [7, 8, 12]:

$$\langle O \rangle_\mu = \mathbb{E}_{\hat{x}_1 \sim \hat{\rho}_1}\left[w(\hat{x}_1) O(\hat{x}_1)\right] \approx \frac{\sum_{i=1}^N w(\hat{x}_1^i) O(\hat{x}_1^i)}{\sum_{i=1}^N w(\hat{x}_1^i)}. \tag{7}$$

While the likelihood integral (6) has been used in prior implementations of Boltzmann generators [8, 12], in general its computational expense prevents it from scaling to large molecular systems. In this work, our aim is to amortize the associated cost by learning a second energy-based model that can estimate the likelihoods $\log \hat{\rho}_1$.

2.2 BoltzNCE

Our method is designed to calculate free energies and to enable the computation of observables via (7) in an efficient and scalable manner. It proceeds in two stages: (i) standard flow training, and (ii) amortization of the likelihood via EBM training. In the first stage, we train a Boltzmann emulator on a dataset $\mathcal{D}$ of conformers using the stochastic interpolant framework described in Section 2.1. The trained emulator is then used to generate samples $\hat{x}_1 \sim \hat{\rho}_1$ via (5), leading to a dataset of generated conformers $\hat{\mathcal{D}}$. In the second stage, we train an EBM on $\hat{\mathcal{D}}$ to approximate the energy U of the generated distribution,

$$\hat{\rho}_1(x) = \exp(U(x))/Z, \qquad Z = \int \exp(U(x)) dx. \tag{8}$$

The generated samples are then reweighted to the Boltzmann distribution using (8) in (7).

Training the emulator. We train the Boltzmann Emulator using stochastic interpolants as described in Section 2.1. Boltzmann Emulator models are trained using either the vector field (4) or the endpoint objectives. The *endpoint parameterization* [35, 36] is where the network predicts the clean endpoint x_1 rather than the velocity field directly. The corresponding velocity field for sampling is given by

$$\hat{b}_t(x) = \alpha_t^{-1}(\dot{\alpha}_t x + (\dot{\beta}_t \alpha_t - \beta_t)\hat{x}_1(t, x)) \tag{9}$$

where $\hat{x}_1(t, x)$ is the network's predicted endpoint at time t given noisy sample x. A derivation of this velocity field is provided in Appendix A. Since $\alpha_1 = 0$, this velocity field diverges at $t = 1$; in practice, we integrate from $t = 0$ to $t = 1 - 1e^{-3}$ to avoid this singularity. While the endpoint parameterization works well in generating samples, it leads to unstable likelihoods due to the singularity at t=1, a drawback that is addressed by the use of EBMs in the BoltzNCE method. Once trained, we generate the dataset $\hat{\mathcal{D}}$ by sampling $\hat{x}_1 \sim \hat{\rho}_1$ via (5).

Training the EBM. Rather than learn a single energy function corresponding to $\hat{\rho}_1$ as in (8), we propose to define a second stochastic interpolant from the base to $\hat{\rho}_1$,

$$\tilde{I}_t = \alpha_t x_0 + \beta_t \hat{x}_1, \tag{10}$$

and estimate the associated *time-dependent* energy function U_t for $\tilde{\rho}_t = \mathsf{Law}(\tilde{I}_t)$,

$$\tilde{\rho}_t(x) = \exp\left(U_t(x)\right)/Z_t, \qquad Z_t = \int \exp(U_t(x))dx. \tag{11}$$

By construction, we have that $\tilde{\rho}_1 = \hat{\rho}_1$, though in general $\tilde{\rho}_t \neq \hat{\rho}_t = \mathsf{Law}(\hat{x}_t)$. Given access to a model $\hat{U}_t$ of U_t, we can evaluate $\log \hat{\rho}_1(x)$ up to the normalization constant $\hat{Z}_1$ with a single function evaluation $\hat{U}_1(x)$, eliminating the need for (6). We train $\hat{U}$ using a combination of denoising score matching and an InfoNCE objective [29]. The score matching objective leverages (3) to yield

$$\mathcal{L}_{\text{SM}}(\hat{U}) = \mathbb{E}\left[\left|\alpha_t \nabla \hat{U}_t(\tilde{I}_t) + x_0\right|^2\right], \tag{12}$$

where the $\mathbb{E}$ is over the draws of $(t, x_0, \hat{x}_1)$ defining $\tilde{I}_t$. To define the InfoNCE objective, we first write down the joint distribution over $(t, \tilde{I}_t)$ given by $\tilde{\rho}(t, x) = p(t)\tilde{\rho}_t(x)$ for $t \sim p(t)$. In practice, we choose time uniformly, so that $p(t) = 1$ and $\tilde{\rho}(t, x) = \tilde{\rho}_t(x)$. Given this joint density, we may define the conditional distribution $\tilde{\rho}(t|x) = \tilde{\rho}(t, x)/\tilde{\rho}(x) = \tilde{\rho}_t(x)/\tilde{\rho}(x)$ where $\tilde{\rho}(x) = \int \tilde{\rho}(t, x)dt = \int \tilde{\rho}_t(x)dt$ is the marginal distribution of x. This conditional distribution describes the probability that a given observation x was a sample at time t. The InfoNCE objective maximizes the conditional likelihood by minimizing its negative log-likelihood. We can write this intractable quantity as

$$\text{NLL} = -\mathbb{E}\left[\log\left(\frac{\tilde{\rho}_t(x)}{\tilde{\rho}(x)}\right)\right] \approx -\mathbb{E}\left[\log\left(\frac{\exp(\hat{U}_t(\tilde{I}_t) - \log \hat{Z}_t)}{\int \exp(\hat{U}_{t'}(\tilde{I}_t) - \log \hat{Z}_{t'})dt'}\right)\right] = \mathcal{L}_{\text{NLL}}(\hat{U}), \tag{13}$$

where the expectation is over the draws of $(t, x_0, \hat{x}_1)$ defining $\tilde{I}_t$. This expression depends on unknown normalization constants $\hat{Z}_t$ and involves an integral over time t'. To avoid explicit normalization, we parameterize $\hat{U}_t$ to directly approximate $U_t(x) - \log Z_t$, absorbing the log-normalization constant as a time-dependent bias. To approximate the integral, InfoNCE leverages a Monte Carlo approximation, leading to a multi-class classification problem: given a sample $\tilde{I}_t$, we aim to distinguish the true time t from a set of negative times $\{\tilde{t}_k\}_{k=1}^K$. This yields the tractable objective:

$$\mathcal{L}_{\text{InfoNCE}}(\hat{U}) = -\mathbb{E}\left[\log\left(\frac{\exp(\hat{U}_t(\tilde{I}_t))}{\sum_{t' \in \{\tilde{t}_k\} \cup \{t\}} \exp(\hat{U}_{t'}(\tilde{I}_t))}\right)\right]. \tag{14}$$

We train $\hat{U}$ by minimizing a weighted combination of both objectives:

$$\mathcal{L}_{\text{BoltzNCE}}(\hat{U}) = \mathcal{L}_{\text{SM}}(\hat{U}) + \mathcal{L}_{\text{InfoNCE}}(\hat{U}) \tag{15}$$

Since the noise level changes continuously with t, only nearby times $t' \approx t$ have non-negligible conditional likelihood $\tilde{\rho}(t'|\tilde{I}_t)$. We therefore sample negatives $\{\tilde{t}_k\}$ from a narrow Gaussian centered at t, providing informative contrast for learning fine-grained temporal discrimination. We note that both objective functions are *simulation-free* after generation of $\hat{\mathcal{D}}$, as they only require sampling the interpolant $\tilde{I}_t$. While, in principle, the EBM could be trained on the original interpolant I_t (1), training on the generated interpolant $\tilde{I}_t$ (10) is preferable because it approximates the emulator's energy function and enhances transferability to new molecular systems where long-timescale MD data is unavailable.

For more details on model training and inference, refer to Appendices B, E.8 and E.10.

3 Related Work

Boltzmann Generators. Boltzmann Generators have become an active line of research since their initial development with invertible neural networks [7], having now been used to sample both molecular systems [10, 11, 37–40] and lattice systems [10, 41–43]. Tan et al. [44] introduce a more stable reweighting scheme that takes advantage of Jarzynski's equality to attain the equilibrium distribution.

However, most of these methods have required input through system-specific featurizations such as internal coordinates, hindering transferability. The emergence of CNFs and equivariant neural network architectures has enabled the development of BGs on Cartesian coordinates [8, 12] thereby not being system specific and enabling transferability. Despite these advancements, transferability has so far only been demonstrated on small systems such as dipeptides, primarily due to the computational limitations associated with likelihood evaluation at scale.

Boltzmann Emulators. Boltzmann Emulators, unlike BGs, are designed solely to produce high-quality samples without reweighting to the Boltzmann distribution. Because they are not required to be invertible, they can typically be applied to much larger systems. This flexibility also enables the use of a wider range of generative approaches, including diffusion models. Boltzmann Emulators have been employed to generate peptide ensembles [45], protein conformer distributions [35, 46–49], small molecules [50–52], and coarse-grained protein structures [53, 54]. However, they are inherently limited by the data distribution they were trained on. As a result, they are generally unsuitable for generating unbiased samples from the Boltzmann distribution or for performing free energy calculations independently. In this work, we aim to leverage the strengths of Boltzmann Emulators and bridge the gap between Emulators and Generators using energy-based models (EBMs). While EBMs may still yield biased estimates, they bring us closer to true Boltzmann statistics at a fraction of the computational cost.

Energy-Based Models. Energy-Based Models (EBMs) are particularly appealing in the physical sciences, as they describe density functions in a manner analogous to the Boltzmann distribution. This similarity enables the use of various techniques from statistical physics to compute thermodynamic properties of interest [39, 55]. Despite their promise, training EBMs remains a challenging task. Recent advancements have introduced training objectives inspired by noise contrastive estimation [18, 20, 24, 25, 55, 56], contrastive learning [29, 57], and score matching [15, 58, 59]. OuYang et al. [60] have also proposed an "energy-matching" objective to train a neural sampler on the Boltzmann distribution; however, more work needs to be done to make this approach practical for molecules.

4 Experiments

In this section, we validate BoltzNCE on several synthetic low-dimensional systems as well as on molecular conformer generation for dipeptides. Our experiments demonstrate three key results: (i) Combining the denoising score matching (12) and InfoNCE (14) objectives to form (15) significantly improves EBM quality compared to using either alone; (ii) BoltzNCE enables endpoint-parameterized emulators to function as Boltzmann Generators by decoupling likelihood estimation from the sampling vector field, avoiding the numerical instability that makes Jacobian trace computation intractable; (iii) The learned EBM provides accurate and efficient likelihood estimates, enabling the calculation of Boltzmann statistics at speeds up to two orders of magnitude faster than exact likelihood computations.

4.1 Low-dimensional systems

The effectiveness of the score matching and InfoNCE loss functions (12) and (14) are tested on low-dimensional synthetic systems. Specifically, we study an 8-mode Gaussian mixture in two dimensions and the two-dimensional checkerboard distribution. The results for forward evaluation of the trained EBMs are shown in Figure 2. We leverage a simple feedforward neural network that takes the point $x \in \mathbb{R}^2$ as input and outputs a scalar value. KL-divergence values between the predicted and ground truth densities are also reported in Table 1.

Objective Functions	8-Mode Gaussian Mixture	Checkerboard
InfoNCE	0.2395	3.8478
Score Matching	0.2199	0.8384
Score Matching & InfoNCE	**0.0150**	**0.1987**

Table 1: **Low-dimensional Systems: Quantitative Results.** KL-divergence (↓) when using different objective functions to train the EBM. For both systems, using our combined objective (15) significantly outperforms using either (12) or (14) individually.

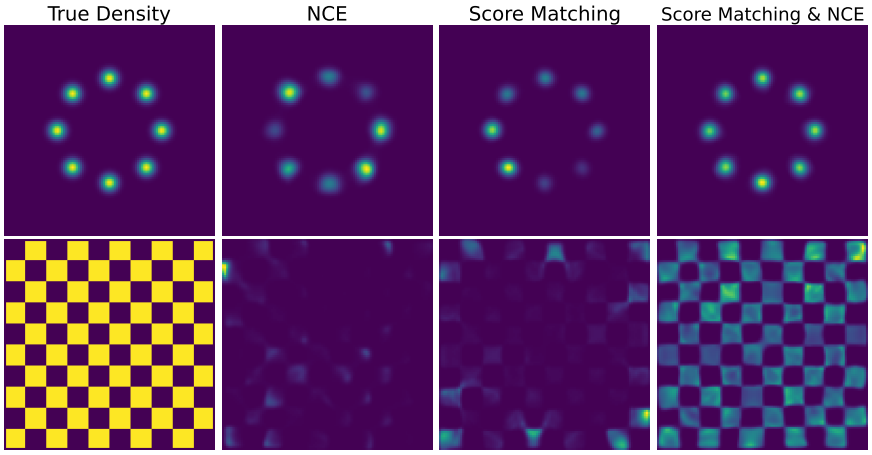

Figure 2: **Low-dimensional Systems: Qualitative Results.** EBM density learned on synthetic two-dimensional systems. (Above) An 8-mode Gaussian mixture. (Below) The checkerboard distribution. In both cases, the true density is shown in the leftmost column, and the results obtained with different methods are shown to the right. Using both objectives (right) provides the best performance.

Our results indicate that combining both objective functions leads to significantly improved performance compared to using either alone. Intuitively, the InfoNCE loss appears to act as a regularizer for the score matching objective, helping to improve the relative weighting of different modes in the learned distribution. This effect is not observed when using the score matching loss on its own. For subsequent experiments on molecules, both objective functions are used to train the EBM.

4.2 Alanine Dipeptide

As a more complex system, we first study the alanine dipeptide molecule. Our aim here is to obtain the equilibrium distribution of the molecule as specified by the semi-empirical GFN-xTB force field [61]. Running a simulation with this force field is computationally intensive, so we use the same setup as Klein and Noé [8]. Conformers are generated using molecular dynamics with the classical Amber ff99SBildn force field and subsequently relaxed using GFN-xTB. We use two dataset variants: *unbiased*, corresponding to the original distribution, and *biased*, in which the positive φ state is oversampled for equal metastable state representation (Figure 6).

We train geometric vector perceptron (GVP)[62, 63] based Boltzmann Emulators using both the vector field objective (4) (GVP-VF) and the endpoint objective (GVP-EP) on the unbiased and biased datasets, and compare their performance to Equivariant Continuous Normalizing Flow (ECNF) models from Klein and Noé [8] trained on the same datasets. Additionally, we use the Graphormer architecture [64] to parameterize the EBM. Further details on data featurization, model architectures, and hyperparameters are provided in Appendices C and E.6.

Table 2: **Methods Overview** Flow matching objectives and likelihood estimation methods for different models tested on Alanine Dipeptide.

Method	FM Objective	Likelihood Estimation
ECNF [8]	Vector Field	Jac-trace integral
GVP Vector field	Vector Field	Jac-trace integral
GVP Endpoint	Endpoint	Jac-trace integral
BoltzNCE Vector field	Vector Field	**EBM forward pass**
BoltzNCE Endpoint	Endpoint	**EBM forward pass**

To evaluate Boltzmann generation, we use models trained on the biased dataset, as these yield more accurate estimates of free energy differences. Free energy differences between the positive and negative φ metastable states are computed, since this transition corresponds to a slow dynamical process (Appendix D.4, Figure 6). Ground-truth free energy values are obtained via umbrella sampling simulations from Klein et al. [12]; further details on umbrella sampling are provided in Appendix E.2. In addition, we compute the energy-based ($\mathcal{E}$–W_2) and torsion-based ($\mathbb{T}$–W_2) Wasserstein-2 distances between the unbiased dataset and the generated (proposal or reweighted) distributions.

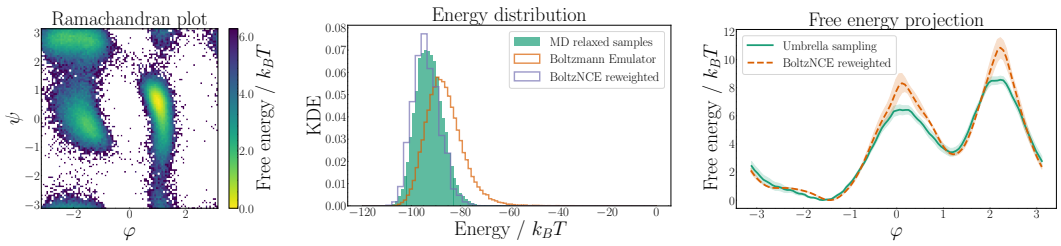

Figure 3: **BoltzNCE: Qualitative Results.** Results for BoltzNCE on alanine dipeptide trained on the biased dataset. We use a GVP vector field as the Boltzmann Emulator. BoltzNCE successfully captures the energy distribution and the free energy projection. (Left) Ramachandran plot of generated samples. (Middle) Energy histogram along with BoltzNCE reweighting. (Right) Calculated free energy surfaces for the angle φ on the right.

Method	$\Delta F/k_BT$	ΔF Err.	Prop. $\mathcal{E}$-W_2	Rew. $\mathcal{E}$-W_2	Prop. $\mathbb{T}$-W_2	Rew. $\mathbb{T}$-W_2	Inf. (h)	Train (h)
Umbrella Sampling	4.10 ± 0.26	–	–	–	–	–	–	–
ECNF [8]	$\mathbf{4.09 \pm 0.05}$	$\mathbf{0.01 \pm 0.05}$	–	–	–	–	9.37	**3.85**
ECNF (reproduced)	4.07 ± 0.23	0.03 ± 0.23	8.08 ± 0.56	0.37 ± 0.02	1.10 ± 0.01	0.59 ± 0.00	9.37	**3.85**
GVP Endpoint	4.89 ± 2.61	0.79 ± 2.61	$\mathbf{6.19 \pm 1.08}$	2.88 ± 0.01	1.12 ± 0.01	0.58 ± 0.01	26.2	4.42
GVP Vector Field	4.38 ± 0.67	0.28 ± 0.67	7.20 ± 0.13	0.46 ± 0.05	$\mathbf{1.09 \pm 0.01}$	0.60 ± 0.00	18.4	4.42
BoltzNCE Endpoint	4.14 ± 0.94	0.04 ± 0.94	6.24 ± 0.52	2.78 ± 0.04	1.12 ± 0.01	0.59 ± 0.01	0.16	12.2
BoltzNCE Vector Field	4.08 ± 0.13	0.02 ± 0.13	7.12 ± 0.15	$\mathbf{0.27 \pm 0.02}$	1.12 ± 0.00	$\mathbf{0.57 \pm 0.00}$	**0.09**	12.2

Table 3: **BoltzNCE: Quantitative Results.** Dimensionless free energy difference, energy $\mathcal{E}$-W_2 and torsion angle $\mathbb{T}$-W_2 Wasserstein-2 distances calculated by different Boltzmann Generator and BoltzNCE models. Standard deviations are shown across 5 runs. Free energy difference values for umbrella sampling and ECNF taken from Klein and Noé [8], which we consider to be ground truth. BoltzNCE Vector Field provides the best performance/inference time tradeoff as compared to all other methods.

The ECNF, GVP-VF, and GVP-EP models estimate likelihoods using the Jacobian trace integral and act as Boltzmann Generators. Energy-based models (EBMs) trained on samples generated by the GVP-based emulators are also evaluated and referred to as BoltzNCE-VF and BoltzNCE-EP. The specific flow-matching objectives and likelihood estimation methods for each model are summarized in Table 2.

GVP models excel as Emulators but fail as Generators due to poor likelihood estimates. The results for the Boltzmann Emulator models trained on the unbiased dataset are given in Appendix F.1. In general, the two GVP models demonstrate better performance than the ECNF. The GVP-EP model performs the best, making it a strong candidate for a Boltzmann Emulator, however, as shown below, it fails as a Boltzmann Generator as its vector field (Eq. 9) diverges as $t \to 1$, which we find corrupts its likelihood calculation.

All Boltzmann generation results are summarized in Table 3. Comparing the GVP and ECNF models, we find that while GVP models match or exceed ECNF performance as emulators, they produce less accurate free energy differences and exhibit higher reweighted $\mathcal{E}$–W_2 and $\mathbb{T}$–W_2 scores. This inaccuracy may stem from unreliable likelihood estimates produced during ODE integration, which requires the divergence of the model with respect its input to be accurate and well-behaved. As a result, Boltzmann Generators face additional design constraints to ensure stability and reliability of their likelihood computation. In the following, we show how BoltzNCE resolves this issue by learning likelihoods separately, decoupling emulator quality from likelihood tractability and enabling greater flexibility in the design of the emulator.

BoltzNCE enables fast and accurate likelihood estimation for Emulators. In contrast, the BoltzNCE models yield more accurate estimates of the free energy difference and achieve lower reweighted $\mathcal{E}$-W_2 and $\mathbb{T}$-W_2 scores, with BoltzNCE-VF providing the best performance. This indicates that the likelihoods predicted by BoltzNCE are generally more reliable than those obtained via the Jacobian trace integral. Representative energy histograms and free energy surfaces along the slowest transition (φ dihedral angle) for the BoltzNCE-VF model are shown in Figure 3. For energy histograms and free energy projections of other methods, refer to Appendix F. As demonstrated in

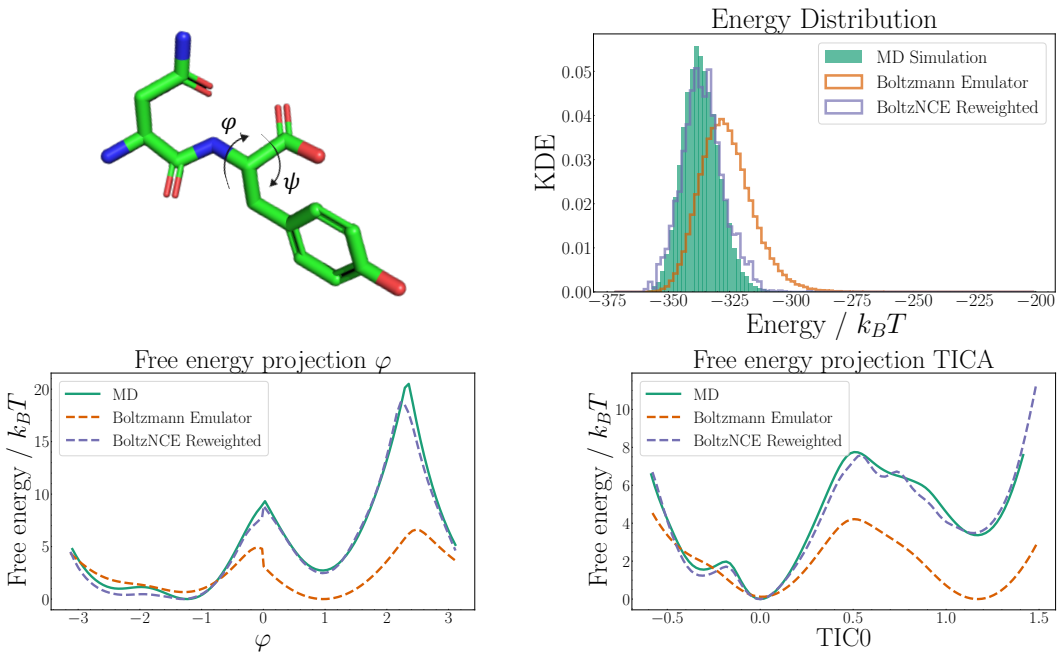

Figure 4: **BoltzNCE Results on NY dipeptide.** BoltzNCE inference results for NY dipeptide (top left) after fine-tuning. Energy distribution (top right), free energy surfaces along the φ angle (bottom left) and the first TICA component (bottom right). BoltzNCE successfully captures the right energy distribution and free energy projections for the dipeptide.

the figure, the BoltzNCE method is able to accurately capture the free energy projection and energy distribution for alanine dipeptide.

We report inference time costs in Table 3, including the time to generate and estimate likelihoods for 10^6 conformers. BoltzNCE provides an overwhelming inference time advantage over the standard Boltzmann Generator by two orders of magnitude while matching accuracy.

To further evaluate the accuracy of EBM likelihoods, we show that it is both more accurate and more computationally efficient than the Hutchinson trace estimator (Appendix F.3). We also demonstrate further flexibility in the design of the EBM training algorithm by comparing OT to independent coupling (Appendix F.4). It is important to note, however, that BoltzNCE has an upfront cost of training the EBM associated with it. In principle, this upfront cost could be reduced by training the EBM in parallel on the original interpolant (1) and then fine-tuning on the generated interpolant (10).

4.3 Generalizability on dipeptides

Finally, we demonstrate BoltzNCE's ability to generalize to unseen molecules using systems of dipeptides. For this experiment, we use the same setup and dataset from Klein and Noé [8], which was originally developed in Klein et al. [65]. The training set consists of classical force field MD simulations of 200 dipeptides, each run for about 50 ns. Since this dataset may not have reached convergence, we bias the dataset in a similar manner to alanine dipeptide to ensure equal representation of the modes along the φ angle. For testing, we utilize 1 μs long simulations of 7 randomly chosen dipeptides not present in the training set.

In this experiment, we benchmark BoltzNCE against independent $1\mu s$ MD runs and the TBG-ECNF model [8] trained on the biased dataset. We use the BoltzNCE-VF method as it achieved the best performance on the alanine dipeptide system. We also compute time-lagged independent components (TICA) [66] from the test MD simulations and plot the free energy projections along the first component. For details on TICA, refer to Appendix E.5.

The inference procedure with BoltzNCE is modified to improve generalizability. Samples generated from the flow-matching model are passed through a conformer matching and chirality checking procedure to check validity of generated samples. For more details, refer to Appendices E.3 and E.4. The EBM, on the other hand, is first pretrained using conformers of training peptides generated by the

Method	φ ΔF Error	$\mathcal{E}$-W_2	$\mathbb{T}$-W_2	Inference Time (h)
MD Baseline	0.18 ± 0.22	$\mathbf{0.17 \pm 0.03}$	$\mathbf{0.22 \pm 0.03}$	24.04
TBG-ECNF* [8]	$\mathbf{0.13 \pm 0.10}$	0.36 ± 0.12	0.34 ± 0.06	123.07
BoltzNCE	0.43 ± 0.21	1.08 ± 0.65	0.44 ± 0.13	**4.005**

Table 4: **Generalizability Results.** Generalizability of the BoltzNCE method in comparison to TBG and MD simulations on systems of 7 dipeptides. *Fewer samples (30,000) used due to high GPU compute time. BoltzNCE provides a significant time advantage over the other methods while achieving good performance.

flow-matching model and then fine-tuned on the dipeptide of interest during inference. In addition, we exclude the top 0.2% of importance weights during reweighting to reduce variance, following the approach introduced in Tan et al. [44].

BoltzNCE yields accurate Boltzmann statistics at a fraction of MD/TBG computational cost. Quantitative evaluations across seven dipeptide systems (Table 4) show that BoltzNCE closely reproduces Boltzmann-weighted energy distributions and free energy surfaces obtained from molecular dynamics (MD), as illustrated for the NY dipeptide in Figure 4. Although the TBG-ECNF method attains the highest accuracy in free energy estimation, it incurs orders-of-magnitude higher computational cost due to its reliance on exact likelihood calculations, thereby serving as an upper bound on BoltzNCE performance. In contrast, BoltzNCE achieves comparable accuracy by approximating likelihoods at substantially lower computational expense.

Visual inspection of all test systems (Figure 10) confirms that this minor performance drop remains acceptable, with BoltzNCE exhibiting excellent agreement with MD-derived energy distributions and free energy landscapes. A small reduction in $\mathcal{E}$-W_2 scores, primarily driven by a single outlier in the NF dipeptide (Appendix G), does not affect overall fidelity. Collectively, these results demonstrate that BoltzNCE provides an efficient, scalable, and generalizable framework for amortized Boltzmann sampling on unseen molecular systems, maintaining high thermodynamic accuracy at a fraction of the computational cost.

5 Discussion

In this work, we introduce a novel, scalable, and simulation-free framework for training energy-based models that integrates stochastic interpolants, InfoNCE, and score matching. We show that InfoNCE and score matching act complementarily to enhance model performance. Our approach learns the density of conformers sampled from a Boltzmann Emulator, eliminating the need for costly Jacobian trace calculations and achieving orders-of-magnitude speedups. On alanine dipeptide, BoltzNCE can even surpass the accuracy of ODE-based divergence integration. Across multiple dipeptide systems, the method generalizes to unseen molecules with minimal fine-tuning, providing substantial computational savings over conventional molecular dynamics. This framework bridges the gap between Boltzmann Emulators and Generators, removing the dependence on invertible architectures and expensive likelihood computations, while enabling high-fidelity, scalable Boltzmann sampling.

6 Limitations and Future Work

The present work is limited to dipeptide molecular systems. While on ADP the method demonstrates excellent accuracy, the performance drops in the generalizability settings. This limitation could likely be addressed through more extensive exploration of model architectures and hyperparameter optimization, which we have only minimally investigated here. The method also has the potential to be scaled to larger molecular systems; however, the accuracy of the method needs to be further tested in higher-dimensional settings.

Training the energy-based model requires applying the score matching loss to its gradients, which increases compute requirements beyond typical levels for neural networks training. Additionally, since the likelihoods estimated by the EBM are approximate, a degree of mismatch between the samples and their predicted likelihoods is inevitable.

Although the current work is limited to a molecular setting, we believe the proposed EBM training framework could be broadly applicable in other domains where energy-based models are useful, such as inverse problems in computer vision and robotics.

7 Acknowledgments

We thank Leon Klein for making the code and data for his original Boltzmann Generator and Equivariant Flow Matching methods readily available.

This work is funded through R35GM140753 from the National Institute of General Medical Sciences. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of General Medical Sciences or the National Institutes of Health.

References

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[2] Haiyang Zheng and Jin Wang. Alphafold3 in drug discovery: A comprehensive assessment of capabilities, limitations, and applications. *bioRxiv*, pages 2025–04, 2025.

[3] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *BioRxiv*, 2022: 500902, 2022.

[4] Benedict Leimkuhler. Molecular dynamics. In *Encyclopedia of Applied and Computational Mathematics*, pages 931–940. Springer, 2015.

[5] David W Borhani and David E Shaw. The future of molecular dynamics simulations in drug discovery. *Journal of computer-aided molecular design*, 26(1):15–26, 2012.

[6] John L Klepeis, Kresten Lindorff-Larsen, Ron O Dror, and David E Shaw. Long-timescale molecular dynamics simulations of protein structure and function. *Current opinion in structural biology*, 19(2):120–127, 2009.

[7] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

[8] Leon Klein and Frank Noé. Transferable boltzmann generators. *arXiv preprint arXiv:2406.14426*, 2024.

[9] Alessandro Coretti, Sebastian Falkner, Jan Weinreich, Christoph Dellago, and O Anatole von Lilienfeld. Boltzmann generators and the new frontier of computational sampling in many-body systems. *arXiv preprint arXiv:2404.16566*, 2024.

[10] Manuel Dibak, Leon Klein, Andreas Krämer, and Frank Noé. Temperature steerable flows and boltzmann generators. *Physical Review Research*, 4(4):L042005, 2022.

[11] Jonas Köhler, Andreas Krämer, and Frank Noé. Smooth normalizing flows. *Advances in Neural Information Processing Systems*, 34:2796–2809, 2021.

[12] Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36:59886–59910, 2023.

[13] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[14] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[15] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[16] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in neural information processing systems*, 32, 2019.

[17] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.

[18] Sumeet Singh, Stephen Tu, and Vikas Sindhwani. Revisiting energy based models as policies: Ranking noise contrastive estimation and interpolating energy models. *arXiv preprint arXiv:2309.05803*, 2023.

[19] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *arXiv preprint arXiv:2003.05033*, 2020.

[20] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022.

[21] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[22] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The journal of machine learning research*, 13(1):307–361, 2012.

[23] C Dyer. Notes on noise contrastive estimation and negative sampling. arxiv. *arXiv preprint arXiv:1410.8251*, 2014.

[24] Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

[25] Bingbin Liu, Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. Analyzing and improving the optimization landscape of noise-contrastive estimation. *arXiv preprint arXiv:2110.11271*, 2021.

[26] Holden Lee, Chirag Pabbaraju, Anish Sevekari, and Andrej Risteski. Pitfalls of gaussians as a noise distribution in nce. *arXiv preprint arXiv:2210.00189*, 2022.

[27] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[28] Omar Chehab, Aapo Hyvarinen, and Andrej Risteski. Provable benefits of annealing for estimating normalizing constants: Importance sampling, noise-contrastive estimation, and beyond. *Advances in Neural Information Processing Systems*, 36:45945–45970, 2023.

[29] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[30] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[31] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[32] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024.

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

[34] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[35] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*, 2024.

[36] Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Harmonic self-conditioned flow matching for multi-ligand docking and binding site design. *arXiv preprint arXiv:2310.05764*, 2023.

[37] Peter Wirnsberger, Andrew J Ballard, George Papamakarios, Stuart Abercrombie, Sébastien Racanière, Alexander Pritzel, Danilo Jimenez Rezende, and Charles Blundell. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153(14), 2020.

[38] Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. *arXiv preprint arXiv:2208.01893*, 2022.

[39] Xinqiang Ding and Bin Zhang. Deepbar: a fast and exact method for binding free energy computation. *The journal of physical chemistry letters*, 12(10):2509–2515, 2021.

[40] Joseph C Kim, David Bloore, Karan Kapoor, Jun Feng, Ming-Hong Hao, and Mengdi Wang. Scalable normalizing flows enable boltzmann generators for macromolecules. *arXiv preprint arXiv:2401.04246*, 2024.

[41] Rasool Ahmad and Wei Cai. Free energy calculation of crystalline solids using normalizing flows. *Modelling and Simulation in Materials Science and Engineering*, 30(6):065007, 2022.

[42] Kim A Nicoli, Christopher J Anders, Lena Funcke, Tobias Hartung, Karl Jansen, Pan Kessel, Shinichi Nakajima, and Paolo Stornati. Estimation of thermodynamic observables in lattice field theories with deep generative models. *Physical review letters*, 126(3):032001, 2021.

[43] Maximilian Schebek, Michele Invernizzi, Frank Noé, and Jutta Rogal. Efficient mapping of phase diagrams with conditional boltzmann generators. *Machine Learning: Science and Technology*, 5(4):045045, 2024.

[44] Charlie B. Tan, Avishek Joey Bose, Chen Lin, Leon Klein, Michael M. Bronstein, and Alexander Tong. Scalable equilibrium sampling with sequential boltzmann generators. *arXiv:2502.18462 [cs.LG]*, 2025. URL https://arxiv.org/abs/2502.18462.

[45] Osama Abdin and Philip M Kim. Pepflow: direct conformational sampling from peptide energy landscapes through hypernetwork-conditioned diffusion. *bioRxiv*, pages 2023–06, 2023.

[46] Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, et al. Predicting equilibrium distributions for molecular systems with deep learning. *Nature Machine Intelligence*, 6(5):558–567, 2024.

[47] Mathias Schreiner, Ole Winther, and Simon Olsson. Implicit transfer operator learning: Multiple time-resolution models for molecular dynamics. *Advances in Neural Information Processing Systems*, 36:36449–36462, 2023.

[48] Yan Wang, Lihao Wang, Yuning Shen, Yiqun Wang, Huizhuo Yuan, Yue Wu, and Quanquan Gu. Protein conformation generation via force-guided se (3) diffusion models. *arXiv preprint arXiv:2403.14088*, 2024.

[49] Sarah Lewis, Tim Hempel, José Jiménez-Luna, Michael Gastegger, Yu Xie, Andrew YK Foong, Victor García Satorras, Osama Abdin, Bastiaan S Veeling, Iryna Zaporozhets, et al. Scalable emulation of protein equilibrium ensembles with generative deep learning. *Science*, 389(6761): eadv9817, 2025.

[50] Juan Viguera Diez, Sara Romeo Atance, Ola Engkvist, and Simon Olsson. Generation of conformational ensembles of small molecules via surrogate model-assisted molecular dynamics. *Machine Learning: Science and Technology*, 5(2):025010, 2024.

[51] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *Advances in neural information processing systems*, 35:24240–24253, 2022.

[52] Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant flow matching with hybrid probability transport for 3d molecule generation. *Advances in Neural Information Processing Systems*, 36:549–568, 2023.

[53] Nicholas E Charron, Felix Musil, Andrea Guljas, Yaoyi Chen, Klara Bonneau, Aldo S Pasos-Trejo, Jacopo Venturin, Daria Gusew, Iryna Zaporozhets, Andreas Krämer, et al. Navigating protein landscapes with a machine-learned transferable coarse-grained model. *arXiv preprint arXiv:2310.18278*, 2023.

[54] Jonas Kohler, Yaoyi Chen, Andreas Kramer, Cecilia Clementi, and Frank Noé. Flow-matching: Efficient coarse-graining of molecular dynamics without forces. *Journal of Chemical Theory and Computation*, 19(3):942–952, 2023.

[55] Anish Sevekari, Rishal Aggarwal, Maria Chikina, and David Koes. Accelerating nce convergence with adaptive normalizing constant computation. In *ICML 2024 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2024.

[56] Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pages 2552–2573. PMLR, 2022.

[57] Hankook Lee, Jongheon Jeong, Sejun Park, and Jinwoo Shin. Guiding energy-based models via contrastive latent variables. *arXiv preprint arXiv:2303.03023*, 2023.

[58] Shahar Yadin, Noam Elata, and Tomer Michaeli. Classification diffusion models: Revitalizing density ratio estimation. *arXiv preprint arXiv:2402.10095*, 2024.

[59] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pages 8489–8510. PMLR, 2023.

[60] RuiKang OuYang, Bo Qiang, Zixing Song, and José Miguel Hernández-Lobato. Bnem: A boltzmann sampler based on bootstrapped noised energy matching. *arXiv preprint arXiv:2409.09787*, 2024.

[61] Christoph Bannwarth, Sebastian Ehlert, and Stefan Grimme. Gfn2-xtb—an accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. *Journal of chemical theory and computation*, 15(3):1652–1671, 2019.

[62] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

[63] Bowen Jing, Stephan Eismann, Pratham N Soni, and Ron O Dror. Equivariant graph neural networks for 3d macromolecular structure. *arXiv preprint arXiv:2106.03843*, 2021.

[64] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[65] Leon Klein, Andrew Foong, Tor Fjelde, Bruno Mlodozeniec, Marc Brockschmidt, Sebastian Nowozin, Frank Noé, and Ryota Tomioka. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. *Advances in Neural Information Processing Systems*, 36:52863–52883, 2023.

[66] John D Chodera and Frank Noé. Markov state models of biomolecular conformational dynamics. *Current opinion in structural biology*, 25:135–144, 2014.

[67] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[68] Ian Dunn and David Ryan Koes. Accelerating inference in molecular diffusion models with latent representations of protein structure. *ArXiv*, pages arXiv–2311, 2024.

[69] Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3d de novo molecule generation. *arXiv:2404.19739 [q-bio.BM]*, 2024. URL https://arxiv.org/abs/2404.19739.

[70] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph, 2023. URL https://arxiv.org/abs/2105.02605.

[71] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

[72] Moritz Hoffmann, Martin Scherer, Tim Hempel, Andreas Mardt, Brian de Silva, Brooke E Husic, Stefan Klus, Hao Wu, Nathan Kutz, Steven L Brunton, et al. Deeptime: a python library for machine learning dynamical models from time series data. *Machine Learning: Science and Technology*, 3(1):015009, 2021.

[73] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

[74] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[75] Patrick Kidger, Ricky T. Q. Chen, and Terry J. Lyons. "hey, that's not an ode": Faster ode adjoints via seminorms. *International Conference on Machine Learning*, 2021.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract match experimental results section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations and future work section has been included (section 6)

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the information is present in the experimental section 4 and appendix (section B,C,E.6,E.8). The code and data is also available at https://github.com/RishalAggarwal/BoltzNCE.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer:[Yes]

   Justification: The code and data is available at https://github.com/RishalAggarwal/BoltzNCE.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental details have been specified in section 4 and in the appendix sections CE.6, E.8.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Mean and standard deviations are provided across 5 runs for all metrics (section 4)

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources have been specified in training and inference time sections in the appendix D,E.8.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The paper conforms with all guidelines and rules of the NeurIPS code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The work is more foundational research and does not have any impact directly on society

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

Justification: The paper does not contain any models/methods that could be considered a risk

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The authors of all the data and benchmark models have been credited through citations

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Reproducible code along with license is provided at https://github.com/RishalAggarwal/BoltzNCE.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Endpoint Objective

Stochastic interpolants anneal between $x_0 \sim \mathcal{N}(0, \mathbf{I})$ and $x_1 \sim p_*(x)$ with:

$$I_t = \alpha_t x_0 + \beta_t x_1 \tag{16}$$

solving for x_0:

$$x_0 = \frac{I_t - \beta_t x_1}{\alpha_t} \tag{17}$$

I_t evolves according to the vector field given by the conditional expectation:

$$b_t(x) = \dot{\alpha}_t \mathbb{E}\left[x_0 | I_t = x\right] + \dot{\beta}_t \mathbb{E}\left[x_1 | I_t = x\right] \tag{18}$$

Substituting 17 in 18 we get:

$$b_t(x) = \frac{\dot{\alpha}_t (x - \beta_t \mathbb{E}\left[x_1 | I_t = x\right])}{\alpha_t} + \dot{\beta}_t \mathbb{E}\left[x_1 | I_t = x\right] \tag{19}$$

$$b_t(x) = \alpha_t^{-1} (\dot{\alpha}_t x + (\dot{\beta}_t \alpha_t - \beta_t) \mathbb{E}\left[x_1 | I_t = x\right]) \tag{20}$$

Similarly, the model estimate of the vector field is given by:

$$b_{\theta,t}(x) = \alpha_t^{-1} (\dot{\alpha}_t x + (\dot{\beta}_t \alpha_t - \beta_t) \hat{x}_1(t, x)) \tag{21}$$

Where $\hat{x}_1(t, I_t)$ is the predicted endpoint by the model. The objective is then given by:

$$\mathcal{L}_{EP} = \int_0^T \|b_{\theta,t}(I_t) - b_t(I_t)\|^2 dt \tag{22}$$

$$\mathcal{L}_{EP} = \int_0^T \mathbb{E}\left[\| \frac{\dot{\beta}_t \alpha_t - \beta_t}{\alpha_t} (\hat{x}_1(t, I_t) - x_1)\|^2\right] dt \tag{23}$$

B EBM training algorithm

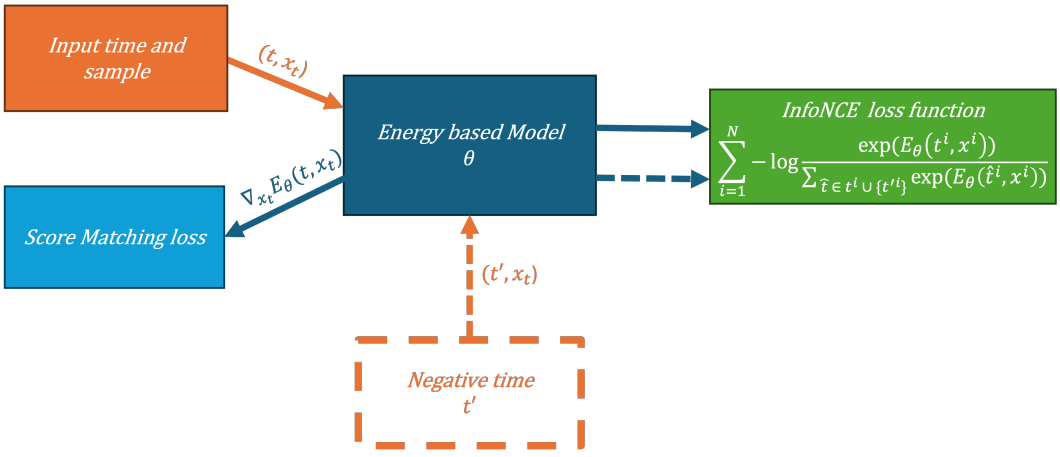

Figure 5: Energy Based Model training workflow

A diagrammatic representation of the method used for training the energy-based model is shown in Figure 5. The model takes a sample x and time point t as input and outputs predicted energy $E_\theta(t, x)$. The gradient of the output with respect to the sample $\nabla_x E_\theta(t, x)$ is used for the score matching loss. The same sample is also passed with negative time points $\{t'\}$, and the predicted energies $E_\theta(t', x)$ are used along with the previously output energies $E_\theta(t, x)$ for the InfoNCE loss.

We also provide a pseudocode block for training the energy-based model with stochastic interpolants, InfoNCE, and score matching in algorithm block 1.

Algorithm 1: Training EBM with stochastic interpolants, InfoNCE, and score matching

Input: Energy-Based model θ, samples from prior X_0, generated samples $\tilde{X}_1$, interpolant functions α_t, β_t, negative time sampling variance σ

for *epoch* $\leftarrow 1$ **to** $epoch_{\max}$ **do**

  for *batch* (x_0, x_1) *in* $(X_0, \tilde{X}_1)$ **do**

    $(x_0, \tilde{x}_1) \leftarrow$ `coupling function`$(x_0, \tilde{x}_1)$

    sample $t \sim \mathcal{U}(0, 1)$

    $I_t \leftarrow \alpha_t x_0 + \beta_t \tilde{x}_1$

    $\mathcal{L}_{SM} \leftarrow \frac{1}{N} \sum_{n=1}^{N} |\alpha_t \nabla E_\theta(t^n, I_t^n) + x_0^n|^2$

    sample $t' \sim \mathcal{N}(t, \sigma^2)$

    $\mathcal{L}_{\text{InfoNCE}} \leftarrow \frac{1}{N} \sum_{n=1}^{N} -\log \frac{\exp(E_\theta(t^n, I_t^n))}{\exp(E_\theta(t^n, I_t^n)) + \exp(E_\theta(t'^n, I_t^n))}$

    $\mathcal{L} \leftarrow \mathcal{L}_{\text{SM}} + \mathcal{L}_{\text{InfoNCE}}$

    $\theta \leftarrow$ `Update`$(\theta, \nabla_\theta \mathcal{L})$

Output: Updated model parameters θ

C Data featurization and Model Architecture

C.1 Data featurization

The data is featurized such that all atom types are defined according to the peptide topology. For transferable models, amino acid identity and positional information are included along with atomic features. Molecular structures are represented as fully connected graphs, and both models operate directly on the Cartesian coordinates of the atoms.

C.2 Geometric Vector Perceptrons

Boltzmann Emulators are parameterized with an SE(3)-equivariant graph neural network that leverages geometric vector perceptrons (GVPs) [62]. Briefly, a GVP maintains a set of equivariant vector and scalar features per node that is updated in an SE(3)-equivariant/invariant manner via graph convolutions. We utilize this architecture as it has been shown to have improved performance over equivariant graph neural networks (EGNNs) [67] in molecular design tasks [68].

For our models, we use a modified version of the GVP which has been shown to increase performance as described in Dunn and Koes [69]. The message passing step is constructed by applying the GVP message passing operation defined in Jing et al. [63].

$$(m_{i\to j}^{(s)}, m_{i\to j}^{(v)}) = \psi_M\Big([h_i^{(l)} : d_{ij}^{(l)}],\ v_i : \left[\frac{x_i^{(l)} - x_j^{(l)}}{d_{ij}^{(l)}}\right]\Big) \tag{24}$$

Here $m_{i\to j}^{(v)}$ and $m_{i\to j}^{(s)}$ are the vector and scalar messages between nodes i, j. h_i, d_{ij} are the scalar features, edge features, and a radial basis embedding respectively, while x represents the coordinates of the node. For the detailed Node Position Update and Node Feature Update operations, refer to Appendix C of Dunn and Koes [69].

C.3 Graphormer Operations

Our EBMs are implemented using the graphormer [64] architecture, which has demonstrated state-of-the-art performance in molecular property prediction tasks. Graphormers function similarly to standard transformers, with the key difference being the incorporation of an attention bias derived from graph-specific features. In 3D-graphormers, this attention bias is computed by passing a Euclidean distance matrix through a multi-layer perceptron (MLP).

Graphformers are neural network architectures where layer-wise GNN components are nested alongside typical transformer blocks [70]. For our EBMs, we follow the implementation of the Graphformer with one minor modification. For the original Graphformer, each attention head is calculated as:

$$\text{head} = \text{softmax}\Big(\frac{QK^{\mathsf{T}}}{\sqrt{d}} + B\Big)V \tag{25}$$

where B is a learnable bias matrix. In our implementation, B is calculated by passing the graph's euclidean distance matrix through an MLP.

D Metrics

D.1 NLL

To calculate the NLL of the holdout conformers, we take (6) and evaluate the ODE in the reverse direction for a given sample. This provides the NLL of the sample. NLL values are reported over batches of $1 * 10^3$ samples.

D.2 Energy - W2

In order to quantify the difference in energy distributions between generated molecules and MD relaxed samples, we calculate the Wasserstein-2 distance between the two distributions. This can be intuitively thought of as the cost of transforming one distribution to another using optimal transport. Mathematically, we solve the optimization process with the loss:

$$\mathcal{E}\text{-}W_2 = \left(\inf_{\pi} \int c(x,y)^2 \, d\pi(x,y)\right)^{\frac{1}{2}} \tag{26}$$

where $\pi(x,y)$ represents a coupling between two pairs (x,y) and $c(x,y)$ is the euclidean distance. We use the Python Optimal Transport package in our implementation [71]. $\mathcal{E}$-W_2 values are reported over batches of $1 * 10^5$ samples.

D.3 Angle - W2

Similar to the $\mathcal{E}$-W_2 metric, we seek to quantify the differences in the distributions of dihedral angles generated and those from MD relaxed samples. Here, following the convention defined in Tan et al. [44], we define the optimal transport in torsional angle space as:

$$\mathbb{T}\text{-}W_2 = \left(\inf_{\pi} \int c(x,y)^2 \, d\pi(x,y)\right)^{\frac{1}{2}} \tag{27}$$

where $\pi(x,y)$ represents a coupling between two pairs (x,y). The cost metric on torsional space is defined as:

$$c(x,y) = \left(\sum_{i=1}^{2s} \big((x_i - y_i)\%\pi\big)^2\right)^{\frac{1}{2}} \tag{28}$$

where $(x,y) \in [-\pi,\pi)^{2s}$

Similar to Energy-W2 calculations, we use the Python Optimal Transport package for implementation [71]. $\mathbb{T}$-W_2 values are computed in radians over batches of 1×10^5 samples.

D.4 Free energy difference

We believe the free energy projection of a system is a relevant baseline for the following two reasons:

- **It represents a high dimensional integral of probability:** The equation of a free energy projection along a reaction coordinate is given by:

$$F(r') = -K_B T \ln \rho(r') \tag{29}$$

  where $\rho(r')$ is the probability density of observing the system at position r' along the coordinate,

$$\rho(r') = \frac{1}{Z} \int_x \delta(r(x) - r') e^{(-U(x)/K_B T)} \, dx \tag{30}$$

  In principle, if we solve the above integral along all points of the reaction coordinate, we solve a $(D-1)$ dimensional integral, where $x \in R^D$. Thus, this serves as a good metric on how well the model matches the ground truth R^D distribution.

  For dipeptides, the process of going from the negative φ angle to the positive φ angle is considered a slow process as there are regions of high energy/low probability in between the two. Therefore, this serves as an ideal reaction coordinate to study dipeptide systems.
- **Domain relevance:** Applied studies in the biophysical/biochemical/structural biology domain work on elucidating the free energy projection along a **reaction coordinate**. This helps the researchers identify the relative stability of different modes along the coordinate as well as the rate of reaction along the coordinate.

Free energy differences are computed between the positive and negative metastable states of the φ dihedral angle. The positive state is defined as the region between 0 and 2, while the negative state encompasses the remaining range. The free energy associated with each state is estimated by taking the negative logarithm of the reweighted population count within that state.

The code for calculating the free energy difference is as follows:

```
left = 0.
right = 2

hist, edges = np.histogram(phi, bins=100, density=True,weights=weights)
centers = 0.5*(edges[1:] + edges[:-1])
centers_pos = (centers > left) & (centers < right)

free_energy_difference = -np.log(hist[centers_pos].sum()/
hist[~centers_pos].sum())
```

Where *phi* is a numpy array containing the φ angles of the generated dataset ($\varphi \in (-\pi, \pi]$) and *weights* is an array containing the importance weight associated with it.

D.5 Inference times

Inference time for free energy estimation is measured over 1×10^6 samples. Specifically, we use a batch size of 500 and generate 200 batches of conformers. During sample generation, Boltzmann Generators also compute the Jacobian trace. All run times are recorded on NVIDIA L40 GPUs. Reported values represent the mean of five independent runs for alanine dipeptide and the average across individual runs on the seven test-system dipeptides in the generalization setting.

E Technical Details

E.1 Dataset Biasing

Since transitioning between the negative and positive φ is the slowest process, with the positive φ state being less probable, we follow the convention of Klein and Noé [8], Klein et al. [12] and use a

Experiment	Model Type	Architecture	Parameters
Alanine Dipeptide	Flow Matching	ECNF	147,599
Alanine Dipeptide	Flow Matching	GVP	108,933
Alanine Dipeptide	Energy-Based Model	Graphormer	4,879,949
Dipeptides (2AA)	Flow Matching	ECNF	1,044,239
Dipeptides (2AA)	Flow Matching	GVP	735,109
Dipeptides (2AA)	Energy-Based Model	Graphormer	6,102,153

Table 5: Model size of different neural network architectures used in this work.

version of the dataset with bias to achieve nearly equal density in both states, which helps in obtaining a more accurate estimation of free energy. To achieve the biased distribution, weights based on the von Mises distribution, f_{vM}, are incorporated and computed along the φ dihedral angle as

$$\omega(\varphi) = r \cdot f_{\mathrm{vM}}\big(\varphi \mid \mu = 1, \kappa = 10\big) + 1 \tag{31}$$

Where r is the ratio of positive and negative φ states in the dataset. To achieve dataset biasing, samples are drawn based on this weighted distribution.

E.2 Umbrella sampling

Umbrella sampling is a physics-based method used to estimate the free energy profile along a reaction coordinate. It involves selecting a set of points along the coordinate, performing separate simulations around each point using a biasing potential, typically a harmonic restraint, and then combining the resulting data to reconstruct an unbiased free energy landscape. The biasing potential keeps the system near the target point while also promoting sampling of regions that are otherwise rarely visited due to energy barriers.

For alanine dipeptide, Klein et. al[12] ran umbrella sampling simulations with the GFN2-xtb force-field. We utilize the data from the same simulation and treat it as the ground truth value of the free energy projection.

E.3 Conformer matching

For the generalizability experiments, the bonded graph of a generated sample is inferred using empirical bond distances and atom types in a similar manner to Klein and Noé [8]. The inferred graph is then compared with a reference graph of the molecule and the sample is discarded if the two graphs are not isomorphic.

E.4 Correcting for chirality

Since SE(3) equivariant neural networks are invariant to mirroring, the Emulator models tend to generate samples from both chiral states. To account for this, we fix chirality *post-hoc* following the convention set by Klein and Noé [8], Klein et al. [12].

E.5 Time-lagged Independent Component Analysis (TICA)

TICA is a dimensionality reduction technique introduced for analyzing time-series data. In general, it is used to identify directions that maximize the autocorrelation at a chosen lag time. Projecting data onto TICA components yields lower dimensional representations that preserve the system's slowest timescales. Similar to Klein and Noé [8], we construct TICA components using the deeptime library [72] at a lag time of 0.5 ns.

E.6 Model hyperparameters

Each GVP-Boltzmann Emulator model consists of one message-passing GVP layer and one update GVP layer. The alanine dipeptide (ADP) emulators use 5 hidden layers with vector gating, whereas the dipeptide emulators use 9. ADP emulators are configured with 64 scalar features and 16 vector features, while the dipeptide emulators use 128 scalar and 32 vector features.

The Graphormer-based potential models are instantiated with 256-dimensional node embeddings and matching 256-unit feed-forward layers within each transformer block, with a total of 8 layers for ADP and 10 layers for dipeptides. Self-attention is applied with 32 heads over these embeddings, and interatomic distances are encoded using 50 Gaussian basis kernels. The total parameter counts for each model used in this work are reported in Table 5.

E.7 Endpoint training weights

The Endpoint loss function for training the Boltzmann Emulator is given by:

$$\mathcal{L}_{EP} = \mathbb{E}_{t\sim\mathcal{U}(0,1),\,(x_1,x_0)\sim C(x_1,x_0)}\left[\|\frac{\dot{\beta}_t\alpha_t - \beta_t}{\alpha_t}(\hat{x}_1(t,I_t) - x_1)\|^2\right] \tag{32}$$

Note that, the coefficients $\frac{\dot{\alpha}_t\beta_t - \alpha_t}{\beta_t}$ become divergent near $t \to 1$ as $\beta_1 = 0$. Therefore, in practice, we threshold the min and the max value of these coefficients as follows:

$$t_w = \min(\max(0.005, |\frac{\dot{\beta}_t\alpha_t - \beta_t}{\alpha_t}|), 100) \tag{33}$$

And optimize the following objective:

$$\mathcal{L}_{EPmod} = \mathbb{E}_{t\sim\mathcal{U}(0,1),\,(x_0,x_1)\sim C(x_0,x_1)}\left[t_w\|\hat{x}_1(t,I_t) - x_1\|^2\right] \tag{34}$$

E.8 Training protocols

Emulator models for ADP were trained for 1,000 epochs, while those for dipeptides were trained for 12 epochs. Both were trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 512. A learning rate scheduler was employed to reduce the rate by a factor of 2 after 20 consecutive epochs without improvement, down to a minimum of $1e^{-5}$. An Exponential Moving Average (EMA) with β = 0.999 was applied to the models and updated every 10 iterations. For ADP, batches were coupled using mini-batch optimal transport[73], while for dipeptides independent coupling with rotational alignment was employed. Mini-batch optimal transport was computed using the SciPy `linear_sum_assignment` function [74]. All models were trained on NVIDIA L40 GPUs with a batch size of 512.

The EBMs in both settings are trained with independent coupling. For ADP, the training set consists of 100,000 conformers generated by the emulator, while for dipeptides the training set includes 50,000 conformers generated by the emulator across the 200 dipeptides in the dataset. The ADP EBM is trained for 1,000 epochs, whereas the dipeptide EBM is trained for 10 epochs.

For ADP, the negative time point is sampled from a Gaussian distribution with a standard deviation of 0.025, while for dipeptides it is sampled from a Gaussian with a standard deviation of 0.0125. Both models are optimized using Adam with a learning rate of 0.001. The learning rate is reduced by half after 30 consecutive evaluations/epochs without improvement, down to a minimum of $1e^{-5}$. Training is performed with a batch size of 512.

E.9 Interpolant Formulation

We specify the interpolant process following the design choices explored in Ma et al. [32]. The Emulator models are trained with linear interpolants while the energy-based models use trigonometric interpolants, both of which satisfy the constraints to generate an unbiased interpolation process.

$$Linear: \alpha_t = 1 - t, \qquad \beta_t = t \tag{35}$$

$$\text{Trigonometric: } \alpha_t = cos(\frac{1}{2}\pi t), \qquad \beta_t = sin(\frac{1}{2}\pi t) \tag{36}$$

Trigonometric interpolants are called general vector preserving interpolants (GVP) in Ma et al. [32]. However, we change the naming of this notation to avoid confusion with geometric vector perceptrons (GVP), which are repeatedly discussed in our paper.

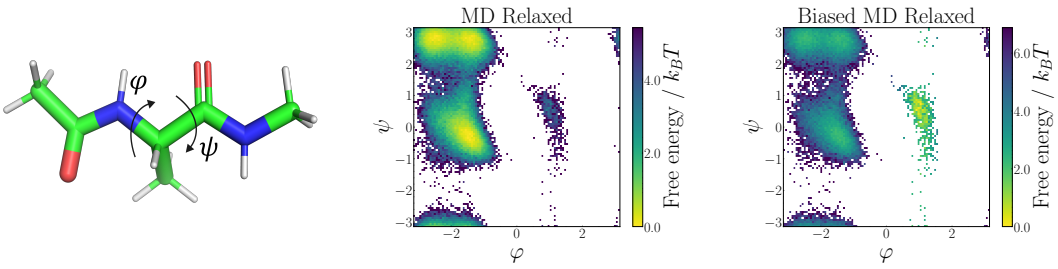

Figure 6: **Alanine Dipeptide System** A visualization of the alanine dipeptide system. Cartoon representation of the alanine dipeptide (left) with its rotatable dihedral angles labeled, Ramachandran plots of unbiased (center) and biased (right) datasets. The biased MD upweights the low frequency mode along the φ dihedral.

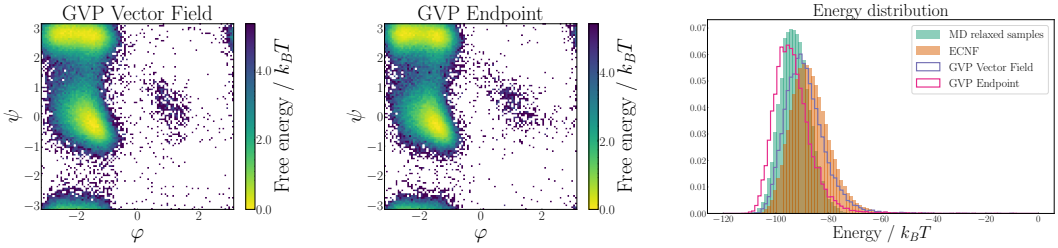

Figure 7: **Evaluation of different emulator models.** The Ramachandran plots of dihedral angles of both endpoint and vector field models are displayed on the left, and center. We see that the energy distributions (right) of both endpoint and vector field emulators as well as a previous method, ECNF, deviate from the distribution of the MD data. However, the Endpoint model distribution deviates the least.

E.10 Integration scheme

All models were integrated with the adaptive step size DOPRI5 solver implemented in the Torchdiffeq package [75]. The tolerance `atol` and `rtol` values were set to $1e^{-5}$ for alanine dipeptide and $1e^{-4}$ for systems of dipeptides. Vector field model integrals are evaluated from 0 to 1, while endpoint models are evaluated from 0 to $1 - 1e^{-3}$ in order to avoid the numerical instability that occurs with endpoint parametrization at time $t = 1$.

F Additional Results

F.1 Vector Field Vs Endpoint Objectives

Inference results for the Boltzmann Emulators are presented in Table 6 and Figure 7. In this section, we aim to quantify what training objective makes the best emulator and, surprisingly, whether a better emulator will always make a better generator.

Method	$\mathcal{E}$-W_2	$\mathbb{T}$-W_2	NLL	NLL std
ECNF	6.22 ± 0.12	0.27 ± 0.01	$\mathbf{-125.53 \pm 0.10}$	$\mathbf{5.09 \pm 0.09}$
GVP Vector Field	4.99 ± 0.50	0.27 ± 0.02	-125.42 ± 0.15	6.92 ± 0.62
GVP Endpoint	$\mathbf{3.11 \pm 0.70}$	$\mathbf{0.26 \pm 0.02}$	-92.04 ± 3.24	175.12 ± 35.51

Table 6: **Boltzmann Emulator results.** Comparison of NLL and W_2 metrics of Boltzmann Emulators across 5 runs ($\pm$ indicates standard deviation). GVP Endpoint emulator captures the energy and torsional target distribution the best. The ECNF model provides the best NLL values despite having the worst W_2 metric values, indicating likelihood integration errors for the GVP models. This is also demonstrated with the higher intra-run NLL std deviation values for the GVP models.

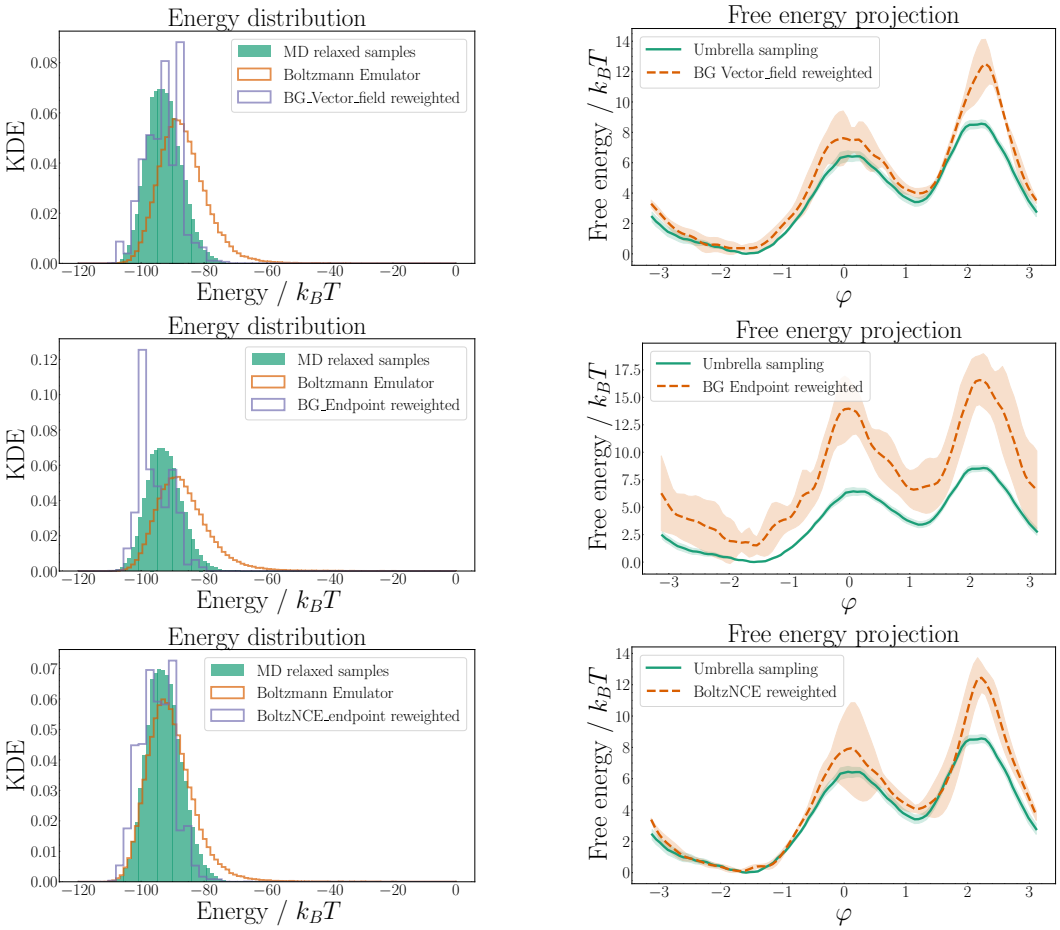

Figure 8: Energy histograms and free energy projections with confidence intervals for the GVP-Vector Field (**top**), GVP-Endpoint (**center**) and BoltzNCE-Endpoint (**bottom**) models.

The energy ($\mathcal{E}$-W_2) and torsion angle ($\mathbb{T}$-W_2) Wasserstein-2 distances quantify the discrepancy between the distributions of energies and torsional angles of generated conformers and those in the dataset. The results show that while the $\mathbb{T}$-W_2 distance remains relatively consistent across all methods, the GVP models capture the dataset's energy distribution better, with the Endpoint model showing the best performance (Figure 7) indicating that it is a very good Boltzmann Emulator on this dataset.

The ECNF and GVP-VF models are comparable on the Negative Log Likelihood (NLL) metric, whereas the GVP-EP model yields the worst values. It is important to note, however, that the endpoint vector field (Eq. 9) diverges at time-point 1. Consequently, the likelihoods for the GVP-EP model were evaluated starting from a later time point $t = 1 - 1e^{-3}$. Furthermore, the divergence at $t \to 1$ can lead to inaccurate likelihood estimates due to instability in the ODE integration. The standard deviation of NLL values within each run is also reported, and the large variance observed for the GVP-EP model further highlights the potential unreliability of its likelihood computations.

F.2 Boltzmann Generator Results

Energy histograms and free energy projections for GVP Vector Field, GVP Endpoint, and BoltzNCE Endpoint methods are shown in Figure 8. The free energy values and energy histograms match up best with the BoltzNCE Endpoint method.

F.3 EBM in comparison to the Hutchinson trace estimator

Likelihoods predicted by the EBM are directly compared to the ground truth likelihoods obtained from applying the change of variable equation on the flow matching generated samples. Note

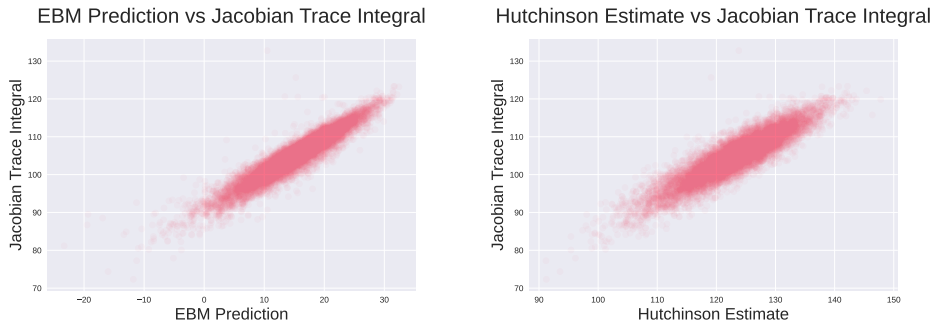

Figure 9: **Likelihood Calculation scatter plots**: log likelihoods estimated by the EBM and Hutchinson estimator vs ground truth estimates from the continuous change of variable equation. EBM predicts likelihoods at high level of accuracy.

Metric	EBM	Hutchinson 1 call	Hutchinson 2 calls	Hutchinson 4 calls	Hutchinson 8 calls
Spearman (↑)	**0.93**	0.87	0.88	0.88	0.87
Time (↓)	**0.5s**	8 min	28 min	23 min	23 min

Table 7: **Likelihood estimation results**: Correlation between likelihood estimation methods and exact likelihoods. EBM performs best.

that the likelihoods estimated by the EBM only estimate the exact likelihoods up to a constant. The Hutchinson trace estimator is also implemented as a comparative benchmark. The Spearman correlation and time required to estimate likelihoods for 10,000 samples is reported in table 7.

The EBM output exhibits strong agreement with the exact likelihoods. Furthermore, the EBM outperforms the Hutchinson estimator, achieving better correlation while being over two orders of magnitude faster at inference on 10,000 samples. Interestingly, increasing the number of estimator calls in the Hutchinson method does not improve its correlation. Due to the use of an adaptive step-size ODE solver, 4 and 8 estimator calls are actually faster than 2 calls.

F.4 Coupling Function Benchmark

To evaluate the effect of different coupling functions on EBM training, we compare independent coupling to mini-batch OT coupling on ADP. The results are presented in Table 8. The results indicate that both coupling functions can be used to train the EBM and both achieve similar performance.

Method	ΔF Error	$\mathcal{E}$-W_2	$\mathbb{T}$-W_2
Independent Coupling	0.02 ± 0.13	0.27 ± 0.02	0.57 ± 0.00
OT Coupling	0.03 ± 0.12	0.23 ± 0.04	0.56 ± 0.005

Table 8: **Coupling Benchmark.** Coupling functions test to train the EBM model on ADP. Both coupling functions provide similar performance, indicating that the training algorithms are independent of coupling functions.

G Dipeptides generalizability results

Quantitative results on the 7 test systems of dipeptides are reported in Table 9. Representative energy histograms and free energy surfaces for the dipeptides are shown in Figure 10. In general, BoltzNCE is able to approximate the right distribution within acceptable error limits at 6x compute time improvement over MD simulations.

Method	Dipeptide	ΔF Error	$\mathcal{E}$-W_2	$\mathbb{T}$-W_2
MD	AC	0.048	0.192	0.213
TBG-ECNF	AC	0.009	0.202	0.290
BoltzNCE	AC	0.356	0.431	0.318
MD	ET	0.069	0.152	0.182
TBG-ECNF	ET	0.187	0.492	0.341
BoltzNCE	ET	0.222	1.329	0.280
MD	GN	0.545	0.122	0.171
TBG-ECNF	GN	0.355	0.198	0.267
BoltzNCE	GN	0.502	1.374	0.296
MD	IM	0.504	0.142	0.269
TBG-ECNF	IM	0.026	0.430	0.362
BoltzNCE	IM	0.688	0.459	0.454
MD	KS	0.044	0.138	0.232
TBG-ECNF	KS	0.133	0.474	0.434
BoltzNCE	KS	0.477	0.419	0.626
MD	NY	0.0003	0.198	0.229
TBG-ECNF	NY	0.034	0.491	0.425
BoltzNCE	NY	0.090	1.293	0.591
MD	NF	0.072	0.221	0.214
TBG-ECNF	NF	0.102	0.275	0.301
BoltzNCE	NF	0.701	2.318	0.536

Table 9: **Dipeptides results.** Quantitative results of different methods on all 7 dipeptide systems. BoltzNCE delivers acceptable performance while offering a substantial time advantage.

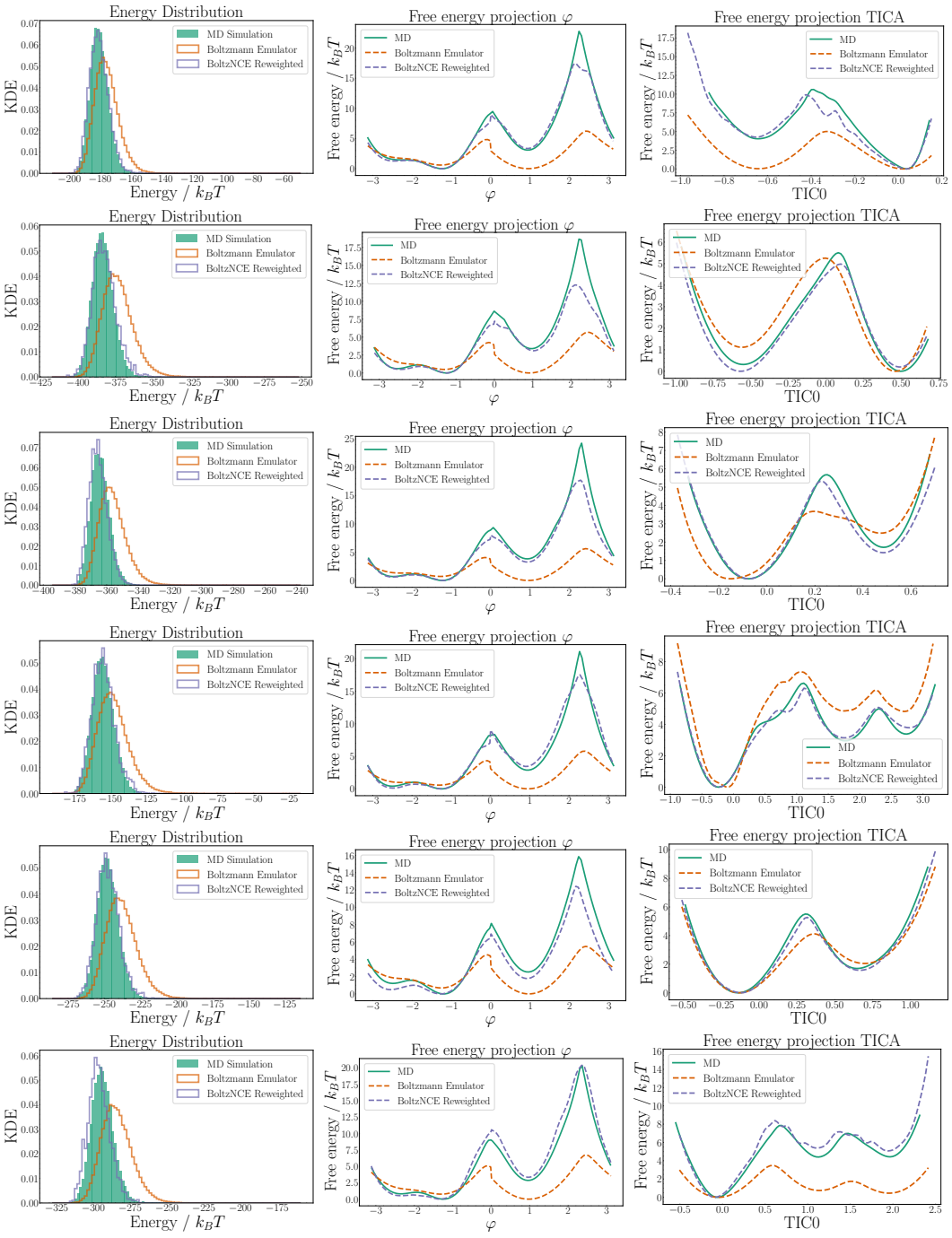

Figure 10: **Qualitative results on dipeptides.** Energy histograms and free energy projections along the φ dihedral angle and the first TICA component for test dipeptides. In order from top to bottom, the figures represent results on the following dipeptides: AC, ET, GN, IM, KS, NF. In all cases, BoltzNCE achieves good approximations of the energy distribution and free energy surfaces.