Boltzmann generators: Efficient sampling of equilibrium states of many-body system with flow-based generative models

Wei-Tse Hsu[1,*] and Lenny Fobe[1,*]

[1]*Department of Chemical and Biological Engineering, University of Colorado Boulder, Boulder, CO 80305*

Owing to incredibly small volume occupied by the equilibrium states in the configurational space, generating statistically independent samples of equilibrium states of many-body systems has been regarded as a significant challenge in the field of computational molecular science. This nature of equilibrium states necessitates the utilization of small steps in molecular simulations (e.g. molecular dynamics (MD) simulations or Monte Carlo (MC) simulations), making it nearly impossible to comprehensively sample the region of interest within reasonable amount of time. Combining deep learing models and statistical mechanics, Boltzmann generators circumvent this sampling issue by repacking the high probability regions (i.e. equilibrium states) of configurational space into a concentrated region in latent space. In this study, we employ Boltzmann generators to different systems in attempt to generate statistically independent samples. As a consequence, we show that Boltzmann generators are able to generate Boltzmann-weighted samples and accurately compute the free energy profile along the reaction coordinate of interest. In addition, comparison between Boltzmann generators trained on different loss functions are demonstrated.

I. INTRODUCTION

In the past decades, molecular simulations, including molecular dynamics (MD) simulations and Monte Carlo (MC) simulations, have played a crucial role in various disciplines of science, including biophysics,[1] pharmaceutical chemistry[2] or material science.[3] However, the usefulness of classical molecular simulations is severely restricted by kinetic bottlenecks as a result of systems characterized by a rough free energy surface. For most systems, metastable states separated by numerous energy barriers typically occupy vanishingly small volume in phase space, leading to prohibitive computational cost for generating statistically independent samples from a Boltzmann distribution.

To address this challenge of phase space sampling, over the years, an enormous amount of research has been devoted to the development of advanced sampling methods. While these methods generally can mitigate the problem, they also have their drawbacks. For example, for sampling techniques such as metadynamics,[4] umbrella sampling,[5] adaptive biasing force[6] or their variants,[7,8] it is required to define reasonable reaction coordinates (RC) that can capture all the slow degrees of freedom of the system, which could be particularly challenging for complicated condensed matter systems. Although methods such as replica exchange molecular dynamics (REMD),[9,10] expanded ensemble[11] and their variants[12,13]do not require predefined reaction coordinates, they do require a series of alchemical intermediate states to bridge the gap in probability overlap between different metastable states. Overall, all these methods fail to draw statistically independent samples from Boltzmann-type distributions in one-shot to compute statistical observables of the systems, such as free energy differences.

Boltzmann generators,[14] in contrast, do not require any knowledge of reaction coordinates, nor the intermediate thermodynamic states, successfully integrating the strengths of deep learning and statistical mechanics. In a Boltzmann generator, we train an invertible neural network to learn the transformation of coordinates from configuration space $\mathbf{x}$ to the so-called latent space $\mathbf{z}$. To ensure the adjacency between the low-energy configurations from different equilibrium states in latent space, we adopt a Gaussian distribution $\mu_z(\mathbf{z})$ as the prior distribution such that high-probability configurations are mapped to the center of the probability distribution and therefore can be easily sampled.

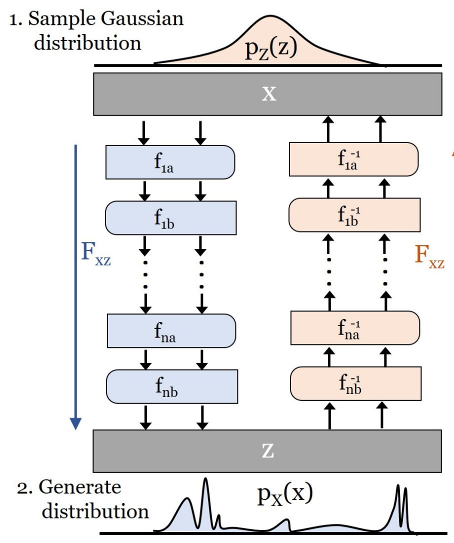

FIG. 1. The architecture of a Boltzmann generator. The figure was adapted from the work[14] by Noé et al.

As shown in Figure 1., in a Boltzmann generator, the configuration variable $\mathbf{x}$ is transformed into the latent

* These two authors contributed equally to this work

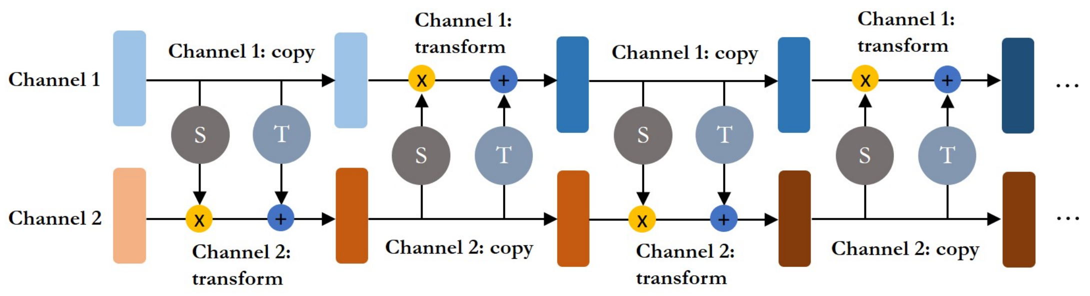

FIG. 2. The structure of affine coupling layers in RealNVP blocks.

variable $\mathbf{z}$ by a deep neural network F_{xz} (the so-called inverse generator) composed of a series of invertible transformation blocks f_1, ..., f_n, which are known as non volume-preserving (NVP) blocks. Conversely, to generate samples in configuration space that approximates the Boltzmann distribution, the network F_{zx} (generator) maps the samples $\mathbf{z}$ drawn from the Gaussian prior distribution $\mu_z(\mathbf{z})$back to the configuration space.

In this study, we demonstrate the usage of Boltzmann generators in generating statistically independent samples in different many-body systems, including 2-dimensional systems such as the double well potential and the Mueller potential, and a higher-dimensional system, which is a bistable dimer immersed in a Lennard-Jones fluid. Incorporated with the consideration of a certain reaction coordinates, these generated samples enable accurate computations of free energy profile. In addition, with a trained Boltzmann generator, we show that realistic transition pathways can also be predicted by simple linear interpolations in latent space.

II. THEORY

A. Real-valued non volume-preserving transformation

As an application of flow-based generative models, a Boltzmann generator implements real-valued non volume-preserving (RealNVP) transformations[15] in a deep neural network and its inverse, which transform the probability density from configuration space to latent space in a way analogous to flows of a fluid. Given the trainable parameters θ, Boltzmann-distributed random variables $\mathbf{x} = F_{zx}(\mathbf{z};\theta)$ and Gaussian random variables $\mathbf{z} = F_{xz}(\mathbf{x};\theta)$, the Jacobian matrices of the transformation F_{zx} and its inverse (F_{xz}) can be expressed as:

$$\mathbf{J}_{zx}(\mathbf{z};\theta) = \left[\frac{\partial F_{zx}(\mathbf{z};\theta)}{\partial z_1}, \ ..., \ \frac{\partial F_{zx}(\mathbf{z};\theta)}{\partial z_n}\right] \tag{1}$$

$$\mathbf{J}_{xz}(\mathbf{x};\theta) = \left[\frac{\partial F_{xz}(\mathbf{x};\theta)}{\partial x_1}, \ ..., \ \frac{\partial F_{xz}(\mathbf{x};\theta)}{\partial x_n}\right] \tag{2}$$

Since the absolute value of the Jacobian's determinant (e.g. $|\det \mathbf{J}_{zx}(\mathbf{z};\theta)|$) measures how much a volume element at the original space is scaled by the transformation, by the change of variables theorem and the inverse function theorem, we can write:

$$\begin{aligned} p_X(\mathbf{x}) &= p_z\left(F_{xz}(\mathbf{x})\right) R_{xz}(\mathbf{x}) \\ &\Rightarrow \log p_X(\mathbf{x}) = \log p_z\left(F_{xz}(\mathbf{x})\right) + \log R_{xz}(\mathbf{x}) \end{aligned} \tag{3}$$

$$\begin{aligned} p_Z(\mathbf{z}) &= p_x\left(F_{zx}(\mathbf{z})\right) R_{zx}(\mathbf{z}) \\ &\Rightarrow \log p_Z(\mathbf{z}) = \log p_x\left(F_{zx}(\mathbf{z})\right) + \log R_{zx}(\mathbf{z}) \end{aligned} \tag{4}$$

Equation 3. and 4. indicate that to ensure a low computational cost for training a Boltzmann generator, the computation of the transformations (F_{xz} and F_{zx}) and their determinants of the Jacobian matrices (R_{xz} and R_{zx}) must be efficient. In a RealNVP model, this is achieved by the two affine coupling layers (f_{ia} ad f_{ib} for the block f_i in Figure 1) that comprise one RealNVP block. Specifically, in each affine coupling layer, the input dimensions are split into two channels ($\mathbf{x}_1 = \mathbf{x}_{1:d}$ and $\mathbf{x}_2 = \mathbf{x}_{d+1:D}$), where the first d dimensions remain the same in the first channel, and the remaining dimensions (from $d+1$ to D) undergo an affine transformation (scale-and-shift transformation) accomplished by the scaling (S) and translating (T) functions:

$$f_{xz}(\mathbf{x}_1, \mathbf{x}_2): \begin{cases} \mathbf{z}_1 = \mathbf{x}_1 \\ \mathbf{z}_2 = \mathbf{x}_2 \bigodot \exp(S(\mathbf{x}_1;\theta)) + T(\mathbf{x}_1;\theta) \end{cases} \tag{5}$$

$$f_{zx}(\mathbf{z}_1, \mathbf{z}_2): \begin{cases} \mathbf{x}_1 = \mathbf{z}_1 \\ \mathbf{x}_2 = (\mathbf{z}_2 - T(\mathbf{x}_1;\theta)) \bigodot \exp(-S(\mathbf{z}_1;\theta)) \end{cases} \tag{6}$$

As shown in Equation 5. and 6., this design of affine coupling layers have two advantages:

- The transformations (f_{xz} and f_{zx}) and their corresponding inverse (f_{zx} and $_{xz}$) are both straightforward. In addition, the computation of R_{xz} and

R_{zx} are computationally cheap. For example,

$$\mathbf{J}_{xz} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d\times(D-d)} \\ \frac{\partial \mathbf{z}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \text{diag}(\exp(S(\mathbf{x}_{1:d}))) \end{bmatrix}$$
$$\Rightarrow \log R_{xz} = \sum_{j=1}^{D-d} S(\mathbf{x}_{1:d})_j \tag{7}$$

- Since f_{xz} and f_{zx} do not require computing the inverse of S or T and computing R_{zx} and R_{xz} do not involve computing the Jacobian of S or T, the functions S and T can as complicated as needed; i.e. both S and T can be modeled by deep neural networks.

As shown in Figure 2, in each RealNVP block, the model alternates the duplication and the affine transformation to ensure that the data in both channels are transformed.

B. Training a Boltzmann generator

As Noé et al. suggest in the original paper,[14] Boltzmann generators are trained by combining two modes: training by examples and training by energy, where the former utilizes samples drawn from the configuration space by Monte Carlo or molecular dynamic simulations to train the neural networks and the latter uses the samples drawn from the prior Gaussian distribution in latent space.

Specifically, during the training, the parameters of the neural networks are adjusted as the distance between the exact distribution (μ) and the generated distribution (q) is being minimized, in either latent space and configuration space. This distance can be measured by the Kullback-Leibler (KL) divergence. For example, KL divergence in latent space measuring the distance from the prior Gaussian distribution $\mu_Z(\mathbf{z})$ to the generated distribution in latent space $q_Z(\mathbf{z})$ can be written as:

$$KL_\theta(\mu_Z \| q_Z) = \int \mu_Z(\mathbf{z}) \log\left(\frac{\mu_Z z(\mathbf{z})}{q_Z(\mathbf{z})}\right) d\mathbf{z} \tag{8}$$

Given that

$$\mu_Z(\mathbf{z}) = \frac{\mathrm{e}^{-u_z(\mathbf{z})}}{Z_Z} = \frac{\mathrm{e}^{-\frac{1}{2}(\frac{\mathbf{z}}{\sigma})^2}}{\sigma\sqrt{2\pi}} \tag{9}$$

and

$$\mu_X(\mathbf{x}) = \frac{\mathrm{e}^{-u_x(\mathbf{x})}}{Z_X} \tag{10}$$

Equation 8. can be derived to

$$\begin{aligned} &KL_\theta(\mu_Z\|q_Z) \\ &= -H_z + \log Z_X + \mathbb{E}_{\mathbf{z}\sim\mu_Z(\mathbf{z})}\left[u_X(F_{xz}(\mathbf{z}) - \log R_{zx}(\mathbf{z})\right] \end{aligned} \tag{11}$$

Because H_Z and Z_X are constant in θ, we define the KL loss J_{KL} as follows:

$$J_{KL} = \mathbb{E}_{\mathbf{z}\sim\mu_Z(\mathbf{z})}\left[u_X(F_{zx}(\mathbf{z}) - \log R_{zx}(\mathbf{z})\right] \tag{12}$$

Accordingly, when the Boltzmann generator is trained by energy, it adjusts the parameters in F_{zx} (hence the parameters of the inverse of F_{xz}, F_{zx}) to minimize the KL loss J_{KL}. Interestingly, the first term of the KL loss, which is the internal energy of the system, counterplays with the second term (an effective entropic contribution to the free energy) such that the model tries to sample low-energy configurations to minimize $u(F_{zx}(\mathbf{z}))$ and simultaneously penalize the system for collapsing to a single metastable state to maximize the entropy of the generated distribution.

However, it was validated[14] that the entropy term in the KL loss is not sufficient to prevent the so-called mode collapse, which necessitates the introduction of the maximum likelihood (ML) loss function (J_ML), i.e. training by examples. Mathematically, J_ML can be derived by either maximizing the probability of configuration samples $\mathbf{x}$ in the Gaussian distribution or minimizing the distance between the generated distribution in configuration space $q_X(\mathbf{x})$ and the Boltzmann distribution $\mu_X(\mathbf{x})$, i.e. the KL divergence in the configuration space. J_{ML} can be expressed as:

$$J_{ML} = \mathbb{E}_{\mathbf{x}\sim\rho(\mathbf{x})}\left[\frac{1}{2\sigma^2}\|F_{xz}(\mathbf{x})\|^2 - \log R_{xz}(\mathbf{x})\right] \tag{13}$$

Note that since we are not able to sample from the Boltzmann distribution $\mu(\mathbf{x})$ a prior, we instead approximate it by a sample $\rho(\mathbf{x})$.

While training a Boltzmann generator on the KL loss and ML loss enables sampling of high-probability states, sometimes we want to sample low -probability (high-energy) states such as a transition state along a predefined reaction coordinate. In this situation, we have to include the reaction coordinate (RC) loss, which can be defined as:

$$\begin{aligned} J_{RC} &= \int p\left(r(\mathbf{x})\right) \log p\left(r(\mathbf{x})\right) dr(\mathbf{x}) \\ &= \mathbb{E}_{\mathbf{x}\sim q_X(\mathbf{x})} \log p\left(r(\mathbf{x})\right) \end{aligned} \tag{14}$$

where $r(\mathbf{x})$ is the reaction coordinate. Practically, to accurately compute $p\left(r(\mathbf{x})\right)$, we employed batchwise kernel density estimation (KDE)[16,17] with K-fold cross validation optimizing the bandwidth of the Gaussian kernel functions.

Finally, considering all kinds of loss functions, we define the total loss function J_tot:

$$J_{tot} = w_{ML}J_{ML} + w_{KL}J_{KL} + w_{RC}J_{RC} \tag{15}$$

where the w's are the weights of the loss functions.

C. Sampling

Samples of the Boltzmann distribution were generated using Metropolis Monte Carlo (MC) simulations. As described in Noé *et al.*, the use of a small local step size is meant to emulate molecular dynamic simulations, ensuring that individual simulations stay trapped in the meta stable states of interest. This method of generating "MD" simulation samples generates configuration space samples $\mathbf{x}$, which can be directly used to train Boltzmann generators.

In each MC step, the system of interest is perturbed with isotropic normal distribution scaled by a system dependent σ_{Metro}. The proposed configuration is accepted or rejected using the standard Metropolis criterion, shown in equation 16.

$$\alpha(x^*|x_{i-1}) = \min\left[1, \exp\left[\frac{u(\mathbf{x}^*) - u(\mathbf{x_{i-1}})}{k_B T}\right]\right] \tag{16}$$

Where, $u(\mathbf{x})$ represents the energy of a given configuration and $k_B T$ represents the reduced temperature of the system.

D. Free energy calculations

In view of the fact that the generated samples might be more or less biased by the neural network, to obtain unbiased samples to compute Boltzmann-weighted averages (like the free energy of the system), we have to reweight the generated distribution $q_x(\mathbf{x})$ to the Boltzmann distribution. Therefore, we define a statistical weight $w_x(\mathbf{x})$ such that:

$$w_x(\mathbf{x}) = \frac{\mu_x(\mathbf{x})}{q_x(\mathbf{x})} = \frac{q_z(\mathbf{z})}{\mu_z(\mathbf{z})} \tag{17}$$

and

$$w_x(\mathbf{x}) \propto e^{-u_x(F_{xz}(\mathbf{z}))+u_z(\mathbf{z})+\log R_{zx}(\mathbf{z};\theta)} \tag{18}$$

Then, we apply this statistical weight in the kernel density estimation mentioned in the last section or in a weight histograms such that the probability of i-th bin p_i can be estimated by:

$$p_i = \frac{\sum_{k=1}^{n_i} w_{i,k} \cdot x_{i,k}}{\sum_{i=1}^{m} n_i} \tag{19}$$

where m and n_i are the number of bins and the number of events at bin i, respectively, $x_{i,k}$ is the k-the sample in the i-the bin, and $w_{i,k}$ is its corresponding statistical weight. Using $p\left(r(\mathbf{x})\right)$ calculated by either weighted histograms or KDE, we can calculate the reduced free energy as follows:

$$f\left(r(\mathbf{x})\right) = -\log p\left(r(\mathbf{x})\right) \tag{20}$$

Since we don't know the aboslute free energy value, when plotting a free energy profile, we subtract $f\left(r(\mathbf{x})\right)$ by its minimum to take $f = 0$ as the reference.

III. APPLICATIONS

A. System 1: Double-well Potential (Wei-Tse)

In this study, instead of using Keras[18] and TensorFlow[19] as the original paper,[14] we implemented Boltzmann generators in PyTorch[20] and the codes are maintain in the GitHub repository for this project. To ensure the efficacy of our implementation, we start with the double well potential model, which can be expressed as:

$$u(\mathbf{x}) = u(x_1, x_2) = ax_1^4 - bx_1^2 + cx_1 + dx_2^2 \tag{21}$$

As shown in Figure 3, we adopted $(a,\ b,\ c,\ d) = (1,\ 6,\ 1,\ 1)$ and the two-dimensional potential model exhibits two metastable states separated by a high energy barrier. x_2 is used as the reaction coordinate since it is the slow degree of freedom of the system. To acquire the dataset for training the Boltzmann generator, we performed a 5000-step Monte Carlo simulation for each well starting from its local minimum with the reduced temperature set as 1. To preserve the Boltzmann weights in the samples, we include the configuration at every Monte Carlo step no matter the MC move is accepted. As a result, there are 10002 configuration samples in the training dataset.

For this simple test system, we build a Boltzmann generator composed of 3 RealNVP blocks where there are two affine coupling layers. In an affine coupling layer, there is a neural network composed of three layers for each of the transformation functions (S and T). In the neural network, there are 100 nodes in each layer and the activation functions include the ReLU function and the hyperbolic tangent function. In addition, we used Adam optimizer in the gradient descent method with the learning rate being 0.001. In the end, to train the Boltzmann generator, we include 2048 samples for each batch of the data and we train the model for 40 epochs (200 iterations).

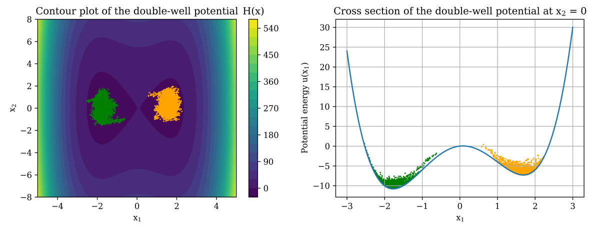

FIG. 3. The samples extracted from Monte Carlo simulations of the double well potential

B. Muller Brown potential

The next system of interest we explore in this paper is a particle on the Muller-Brown (MB) potential energy surface. This system is well-studied and is often used to study an algorithm's ability to find a reaction coordinates over the complex landscape. The MB potential surface is given by:

$$V_{MB}(x,y) = \sum_{i=1}^{4} A_i \exp\left[a_i(x-\bar{x_j})^2 + b_i(x-\bar{x_j})(y-\bar{y_j}) + c_i(y-\hat{y_j})^2\right. \tag{22}$$

The surface can be described as the sum of 4 Gaussians with centers given by the pairs $(\bar{x_j}, \bar{y_j})$. Where the first 3 Gaussians describe the energy minima on the surface and the last Gaussian creates near infinite walls around the surface, such that the only low energy regions are described by the first 3 Gaussians. The MB potential parameters are presented below:

$$\begin{aligned} A &= (-200, -100, -170, 15) \\ a &= (-1, -1, -6.5, 0.7) \\ b &= (0, 0, 11, 0.6) \\ c &= (-10, -10, -6.5, 0.7) \\ \bar{x} &= (1, 0, -0.5, -1) \\ \bar{y} &= (0, 0.5, 1.5, 1) \end{aligned} \tag{23}$$

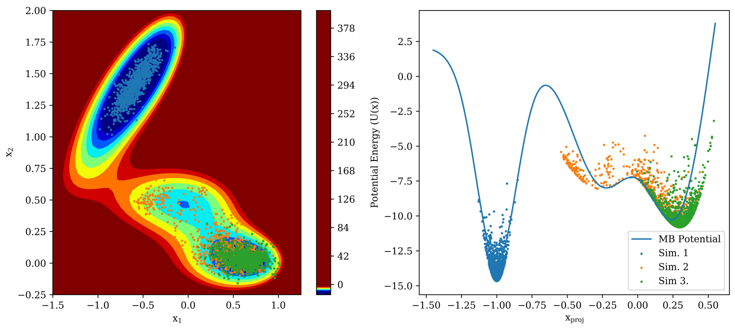

FIG. 4. Visualization of the sampled points on the MB potential. The left figure shows the the samples in the x-y plane. The right figure shows the samples projected onto the $(1, -1)$ vector vs their energy, giving a better idea of the energetic barriers present in the MB potential.

The MB potential represents an increase in complexity from the DW potential as the potential wells of interest are no longer defined along 1 dimension of the simulation. Three simulations were carried out on the surface of the MB potential, each starting at the centers of the first 3 Gaussians. The first 3 Gaussians represent the meta-stable states the particle can visit during a simulation. We will refer to the meta stable states on this surface as state 1, state 2 and state 3, corresponding to the potentials from left to right, along x_1. All simulations on the MB potential were performed with a $\sigma_{metro} = 0.02$. These simulations were run at a reduced temperature of $k_B T = 1.0$ for 10000 steps with a stride of 10 steps.

Figure 4 shows the samples from the 3 simulations and their corresponding energies projected along the $(1, -1)$ vector. We see that both simulations starting in state 1 and state 3 stay in their respective wells, while the simulation starting in state 2 transitions over the energetic barrier to state 3, resulting in a combination of state 2 and state 3 samples from that simulation. Despite seeing the the transition from state 2 to state 3 in our simulation dataset, the MB potential dataset does not have transitions between state 1 and state 2, which will be a region of interest when training Boltzmann generators on this model.

C. System 2: Dimer in the Lennard-Jones bath (Lenny)

The final system we will explore in this project is the bi-stable dimer in a Lennard-Jones (LJ) bath. The dimer system consists of a 2 particle dimer connected by a 1-D DW potential, shown as U_{bond} in 24, surrounded by 36 solvent particles with repulsive LJ potentials ($U_{solvent}$ in **??**). The system is held together with a harmonic-box central potential (U_{box} in **??**), where particle experience a harmonic increase in energy when they leave the bounds of the box. This model system has 78 dimensions for the x-y coordinates of each of the particles. With the increase of dimensions of this system, equilibrium samples are much harder to generate for two reasons. Firstly, the number of available configurations in a 78D is exponentially larger than the 2D systems we've explored previously. Secondly, many more of the configurations in the 78D space are energetically unfavorable. Small changes in the position of particles can result in extremely large energies as particles clash.

$$\begin{aligned} U(\mathbf{x}) =& U_{bond} + U_{solvent} + U_{box} + U_{center} \\ U_{bond} =& \frac{1}{4}a(d-d_0)^4 - \frac{1}{2}b(d-d_0)^2 + c(d-d_0) \\ U_{solvent} =& \epsilon \sum_{i=1}^{n+1} \sum_{j=i+1, j\neq 2} \left(\frac{\sigma}{||\mathbf{x}_i - \mathbf{x}_j||}\right)^{12} \\ U_{box} =& \sum_{i=1}^{n+2} h(|\mathbf{x}_{ix}| - l_{box}) k_{box} (|\mathbf{x}_{ix}| - l_{box})^2 + \\ & \sum_{i=1}^{n+2} h(|\mathbf{x}_{iy}| - l_{box}) k_{box} (|\mathbf{x}_{iy}| - l_{box})^2 \\ U_{center} =& k_d(\mathbf{x}_{1x} + \mathbf{x}_{2x})^2 + k_d \mathbf{x}_{1y}^2 + k_d \mathbf{x}_{2y}^2 \end{aligned} \tag{24}$$

Another consideration of the dimer system is the solvent being explicitly represented. Solvent particles should be indistinguishable from each other. If the identity of each solvent molecule was preserved over the course of the MC simulation, the configuration space of each solvent particle would include the entirety of the box, due to the diffusion and exchange of particles throughout the system. In the short simulations we perform of these systems, the particles do not have enough time to visit the entirety of their configuration space. To address the permutational invariance of the system, we applied the Hungarian algorithm to reassign particles based on their distance to a reference equilibrium sample. This ensures that the x-y coordinates for each particle stays in a local configurational space, despite particles actually exchanging positions throughout the box during the simulation.

dimer_visual.jpg

FIG. 5.

The dimer system has two meta-stable states, which we define as the extended and a compact configurations, shown in figure 5. Transitions between the extended and compact configurations requires significant solvent rearrangement and generally are not observed in the short simulations we performed. To sample this system, we start two simulations in both the extended and compact configuration. Both simulations were run for 50000, with a reduced temperature of $k_B T = 1.0$. In figure 6, we see that both the compact and extended configurations were well sampled according to the potential energy between the two dimer particles.

IV. RESULTS AND DISCUSSIONS

A. System 1: Double well Potential (Wei-Tse)

In the double well potential model, we first train the Boltzmann generator by examples (on the ML loss) for 200 iterations,

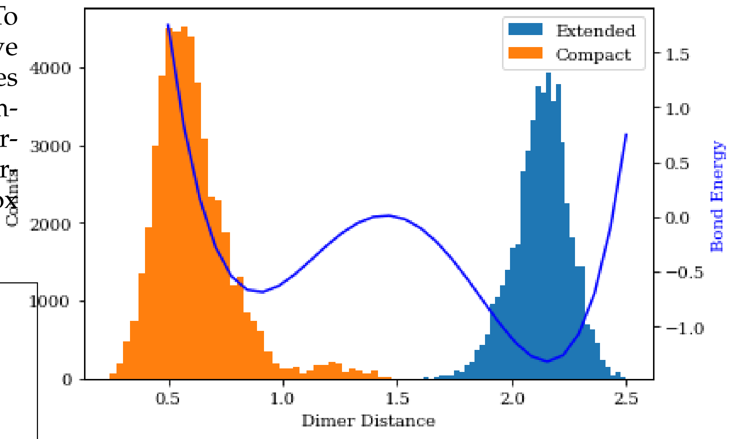

FIG. 6. Distribution of the dimer bond distance from both the extended and compact simulations. The potential energy curve is drawn in blue to show where the compact and extended configurations are energetically favorable. Note, no transitions between the extended and compact configurations were sampled in these simulations.

- Corresponds to page 12 to page 19 of the presentation slides
- Show the results of ML, KL, ML + KL model
  - Distribution of generated samples
  - Free energy profile as a function of reaction coordinate

B. Muller-Brown potential (Lenny)

- Similar to double well potential content-wise

C. System 2: Dimer in the Lennard-Jones bath (Lenny)

- Similar to double well potential content-wise

V. CONCLUSION

VI. FUTURE APPLICATIONS

A. Exploration of the configurational space of flexible molecules

B. Foldamer project (Lenny)

REFERENCES

[1] David B Wells and Aleksei Aksimentiev. "Mechanical properties of a complete microtubule revealed through molecular dynamics simulation". In: *Biophysical journal* 99.2 (2010), pp. 629–637. DOI: 10.1016/j.bpj.2010.04.038..

[2] Veronica Salmaso and Stefano Moro. “Bridging molecular docking to molecular dynamics in exploring ligand-protein recognition process: An overview”. In: *Frontiers in pharmacology* 9 (2018), p. 923. DOI: 10.3389/fphar.2018.00923.

[3] Wasinee Khuntawee et al. “Molecular dynamics study of natural rubber–fullerene composites: connecting microscopic properties to macroscopic behavior”. In: *Physical Chemistry Chemical Physics* 21.35 (2019), pp. 19403–19413. DOI: 10.1039/C9CP03155C.

[4] Alessandro Laio and Michele Parrinello. “Escaping free-energy minima”. In: *Proceedings of the National Academy of Sciences* 99.20 (2002), pp. 12562–12566. DOI: 10.1073/pnas.202427399.

[5] Glenn M Torrie and John P Valleau. “Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling”. In: *Journal of Computational Physics* 23.2 (1977), pp. 187–199. DOI: 10.1016/0021-9991(77)90121-8.

[6] Eric Darve, David Rodrıguez-Gómez, and Andrew Pohorille. “Adaptive biasing force method for scalar and vector free energy calculations”. In: *The Journal of chemical physics* 128.14 (2008), p. 144120. DOI: 10.1063/1.2829861.

[7] Yuji Sugita, Akio Kitao, and Yuko Okamoto. “Multidimensional replica-exchange method for free-energy calculations”. In: *The Journal of Chemical Physics* 113.15 (2000), pp. 6042–6051. DOI: 10.1063/1.1308516.

[8] Vittorio Limongelli, Massimiliano Bonomi, and Michele Parrinello. “Funnel metadynamics as accurate binding free-energy method”. In: *Proceedings of the National Academy of Sciences* 110.16 (2013), pp. 6358–6363. DOI: 10.1073/pnas.1303186110.

[9] Robert Swendsen and Jian-Sheng Wang. “Replica Monte Carlo Simulation of Spin-Glasses”. In: *Physical review letters* 57 (1986), pp. 2607–2609. DOI: 10.1103/PhysRevLett.57.2607.

[10] Yuji Sugita and Yuko Okamoto. “Replica-exchange molecular dynamics method for protein folding”. In: *Chemical physics letters* 314.1-2 (1999), pp. 141–151. DOI: 10.1016/S0009-2614(99)01123-9.

[11] AP Lyubartsev et al. “New approach to Monte Carlo calculation of the free energy: Method of expanded ensembles”. In: *The Journal of chemical physics* 96.3 (1992), pp. 1776–1783. DOI: 10.1063/1.462133.

[12] Alejandro Gil-Ley and Giovanni Bussi. “Enhanced conformational sampling using replica exchange with collective-variable tempering”. In: *Journal of chemical theory and computation* 11.3 (2015), pp. 1077–1085. DOI: 10.1021/ct5009087.

[13] Hiraku Oshima, Suyong Re, and Yuji Sugita. “Replica-exchange umbrella sampling combined with Gaussian accelerated molecular dynamics for free-energy calculation of biomolecules”. In: *Journal of chemical theory and computation* (2019). DOI: 10.1021/acs.jctc.9b00761.

[14] Frank Noé et al. “Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning”. In: *Science* 365.6457 (Sept. 6, 2019). DOI: 10.1126/science.aaw1147.

[15] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using real NVP”. In: *arXiv preprint arXiv:1605.08803* (2016).

[16] Emanuel Parzen. “On estimation of a probability density function and mode”. In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[17] Richard A Davis, Keh-Shin Lii, and Dimitris N Politis. “Remarks on some nonparametric estimates of a density function”. In: *Selected Works of Murray Rosenblatt*. Springer, 2011, pp. 95–100.

[18] P.W.D. Charles. *Project Title*. https://github.com/charlespwd/project-title. 2013.

[19] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[20] Adam Paszke et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.