SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

Machine Learning Techniques for Improved Transition Path Sampling

Michael Plainer

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

Machine Learning Techniques for Improved Transition Path Sampling

Techniken des Maschinellen Lernens für verbessertes Transition Path Sampling

Author:	Michael Plainer
Supervisor:	Prof. Dr. Stephan Günnemann
Advisors:	Hannes Stärk, M.Sc., Dr. Charlotte Bunne
Submission Date:	November 15, 2023

Acknowledgments

I cannot express enough thanks to my advisor, Hannes Stärk, who went above and beyond with his support. His continued guidance and ideas were exceptional, and without his high standards and encouragement, this thesis would not have been possible. Further, I would also like to thank Charlotte Bunne for giving me invaluable feedback and helping me organize my thoughts and ideas. Additionally, I would like to express my gratitude to Stephan Günnemann for his supervision of this thesis and his suggestions and contributions to the resulting paper. I would also like to thank Christoph Dellago, Simon Dobers, Sebastian Falkner, Bowen Jing, Hendrik Jung, Vincent Stimper, and David Swenson for the fruitful discussions and their feedback.

Thank you!

Abstract

The ability to efficiently, and most importantly accurately, simulate the atoms of molecules has opened many opportunities in various disciplines. In areas such as drug discovery or material science, we are interested in the constant small fluctuations captured by molecular simulations but also in finding rare transitions to different states as well. Transition path sampling (TPS) offers a powerful approach to exploring the pathways of rare events in complex systems, providing a comprehensive landscape of the transitional trajectories that traditional methods often miss.

Current algorithms are based on Markov chain Monte Carlo and rely on computationally expensive molecular dynamics simulations. In this thesis, we will propose two new methods to overcome the issues of current approaches. As for the first approach, we demonstrate how we can sample transition paths in a learned latent space of a Boltzmann generator without the need for molecular dynamics simulations. For this, we reformulate the acceptance criterion of Metropolis-Hastings in the latent space to ensure that paths can be sampled with the correct probability. Additionally, we investigate how we can improve the current state of traditional TPS methodology. For this, we introduce a self-attention-based neural network architecture that uses the entire transition path to determine the optimal point to start molecular simulations from.

We demonstrate the capabilities of our approaches on the molecule alanine dipeptide and introduce metrics and evaluation techniques to compare them with existing work. While the introduced latent TPS approach is mathematically correct, the produced results are not convincing and often exhibit unfavorable performance due to low acceptance of paths. Our ideas to improve point selection with context-aware neural networks on the other hand, seem promising and can improve on the state-of-the-art.

Kurzfassung

Die effiziente, und vor allem genaue, Simulation von Atomen in molekularen Systemen hat in vielen Bereichen neue Möglichkeiten eröffnet. Zum Beispiel in der Arzneimittelforschung oder Materialwissenschaft ist es wichtig sowohl die kleinen Abweichungen in stabilen Zuständen zu simulieren, als auch die seltenen Übergänge zwischen verschiedenen Zuständen zu finden. Transition Path Sampling (TPS) ist eine Methode um solche seltenen Übergänge in komplexen Systemen zu erforschen. Dies ermöglicht einen besseren Überblick und die Erfoschung von Bewegungen, die herkömmliche Methoden übersehen.

Aktuelle Algorithmen basieren auf Markowketten Monte Carlo und benötigen rechenintensive Molekulardynamiksimulationen. In dieser Arbeit führen wir zwei neue Methoden ein um die Nachteile aktueller Ansätze zu überwinden. Zuallererst werden wir zeigen, wie wir vielfältige Übergänge in dem gelernten latenten Raum eines Boltzmann Generators finden können, ohne aufwendige Molekulardynamiksimulationen. Dafür formulieren wir das Akzeptanzkriterium von Metropolis-Hastings für den latenten Raum um, damit wir Übergänge mit der richtigen Wahrscheinlichkeit erzeugen können. Zudem untersuchen wir eine andere Methodik, welche den aktuellen Stand von klassischen TPS-Verfahren verbessern kann. Wir führen ein neurales Netz basierend auf Self-Attention ein, welches einen existierenden kompletten Übergang benutzt, um den optimalen Punkt für den Start einer Molekularsimulation vorherzusagen.

Wir zeigen die Möglichkeiten unseres Ansatzes anhand des Moleküls Alanindipeptid und führen neuen Metriken und Evaluierungstechniken ein, um verschiedene TPS-Ansätze zu vergleichen. Obwohl unser simulationsfreier Ansatz in einem latenten Raum mathematisch korrekt ist, sind die Ergebnisse nicht überzeugend. Zudem hat die Methode aufgrund von niedriger Akzeptanzwahrscheinlichkeit eine schlechte Laufzeit. Unsere Ideen um Molekularsimulationen gezielt mit einem neuralen Netz zu verbessern sind hingegen vielversprechend und können in unseren Testfällen den aktuellen Stand der Technik verbessern.

Contents

1. Introduction

Molecules undergo constant change guided by the forces that act on the atoms and their temperature. Such systems often remain in a stable state for prolonged periods of time before experiencing swift transitions to another meta-stable state, where they will again spend a relatively long time. This transition to another state is reminiscent of the sudden spark of a chemical reaction or the precise moment when two molecules intertwine. Such changes can be provoked by environmental variations, such as chemical reactions and molecular binding events, or be caused by an inherently random process. Understanding the underlying mechanisms of such state changes, identifying the distinct states a molecule resides in, and accessing the distribution of such transitions is crucial in many applications [Dellago et al., 1998b, Best and Hummer, 2005, Dellago, 2007], e.g., in protein folding (and thus drug discovery) [Kirmizialtin et al., 2012, 2015].

Molecular dynamics (MD) simulations can be used to model the tiny fluctuations in equilibrium states but cannot consistently find these transitions. This is because these changes happen so quickly that we would need a fine-grained simulation. At the same time, however, these events occur so infrequently that it is not computationally feasible to simulate the underlying system for millions of steps. In this thesis, we explore the task of efficiently finding such transitions—*transition path sampling*—without the need for expensive, unguided molecular dynamics simulations.

While there are a few different approaches for transition path sampling (TPS) [Bolhuis et al., 2002], they are all based on Markov chain Monte Carlo sampling. An existing transition is used to sample a new trajectory, and we repeat this until a representative ensemble of transition paths has been found. Different approaches find new transitions based on the current trajectory in different ways. One of the most commonly used idea in practice is *shooting* [Dellago et al., 1998a], where we change the velocity of the atoms of the current transition and perform MD simulations to see if it still describes a transition. The reasoning behind this is that similar MD simulations of the same molecules will likely also describe a transition. In many cases, a transition cannot be found within a reasonable number of steps, and the simulation has to be discarded.

In this thesis, we employ machine learning to sample such transition paths. Prior machine learning-based TPS techniques [Falkner et al., 2023, Jung et al., 2023] still rely on shooting-based MCMC moves, but the model specifies the location to shoot from.

This can be problematic as it only solves some underlying issues of shooting-based TPS. Namely, a trade-off between efficiency and the diversity of sampled transition paths exists. If the modified velocity is vastly different, most of the simulations will not reach the target state and thus get rejected. When only using a slightly different velocity, the paths will be very similar to each other and are likely to be accepted. Currently, there is no way to solve the problem of making significant changes while still generating valid and realistic trajectories. This makes it challenging to explore multiple reaction channels and quickly sample a diverse ensemble of transition paths.

We propose two approaches to improve the current state-of-the-art TPS methods. The main idea of the first approach is to use a Boltzmann generator [Noé et al., 2019] that was trained to approximate the underlying energy landscape of the molecule. Once trained, we can use it to map the coordinates of each molecule state to a latent space where the samples are distributed according to a standard normal distribution. Since this latent space was trained on the energy, we can perform relatively simple simulation-free modifications to the current trajectory to produce diverse and realistic paths. Especially, high-energy barriers that make it difficult for classical MD simulations can be easier to overcome in a well-conditioned latent space. We apply Metropolis-Hastings [Metropolis et al., 1953, Hastings, 1970] that we adapt to work on latent space trajectories to decide which modifications are meaningful and which will be rejected. We will also explore different ways, so-called *transition kernels*, to modify a path in latent space. This approach does not require molecular simulations, and all steps can be performed in parallel. While the latent sampling technique is novel and promising, at its current state it fails to demonstrate convincing results.

In the second approach, we will explore an extension of the ideas presented by Jung et al. [2023]. For this, we still rely on shooting-based TPS with molecular dynamics simulations but will train a neural network that predicts the optimal point to start our simulation. To achieve this, we introduce an attention-based architecture that uses complete transition paths to predict the probability of a successful shooting. Our initial experiments show that a simpler architecture, only relying on the neighboring frames instead of the whole trajectory, can produce better results as it can learn more efficiently.

To summarize, we make the following **contributions** in this thesis:

1. We propose a latent space approach for transition path sampling and derive how the acceptance criterion of Metropolis-Hastings can be applied to these paths.
2. We introduce multiple latent space proposal kernels that can sample new paths, without the need for expensive molecular dynamics simulations.
3. We present an attention-based neural network to predict optimal shooting points.

2. State of the Art

In this chapter, we will introduce the fundamentals needed for this thesis and discuss the current state of the art for similar approaches. First and foremost, in Section 2.1, we will discuss how a molecule can be simulated and investigate the distribution the atoms converge to when simulating the system for a long time. In Section 2.2, we will introduce normalizing flows, which can learn distributions by observing samples, and will see how they can be trained to learn the converged distribution of atom positions. Section 2.3 introduces the problem of transition path sampling and describes how Markov chain Monte Carlo can solve this problem efficiently. Section 2.4 explores the current state of the art in transition path sampling, most prominently how we can "shoot" samples to produce new trajectories. In this section, we will also explore machine learning-based techniques for transition path sampling and investigate works that perform Markov chain Monte Carlo in a latent space.

2.1. Molecular Dynamics

Molecular dynamics (MD) simulations have become fundamental in fields such as drug research [Ariga et al., 2007, Hou et al., 2010, Durrant and McCammon, 2011, Hollingsworth and Dror, 2018], material design [Kim et al., 2013, Rao et al., 2013, Xu and Wang, 2016, Li et al., 2016], and even astrophysics [Mazevet et al., 2003, Andersson and van Dishoeck, 2008, Pais and Stone, 2012, Bonitz et al., 2020], only to name a few. The main goal of these simulations is to explore how the 3D coordinates of each particle changes over time, allowing us to understand microscopic processes better. They can be applied larger system such as proteins, or only a few atoms. In this thesis, we will work with molecules, and will simulate the positions of their atoms over time. Classical MD models can be used for these simulations which often rely on Newton's equation for motion instead of quantum mechanics to reduce the computational overhead. The movements and positions of the atoms can be simulated by solving the so-called n-body problem, described by

$$\boldsymbol{M}\frac{d^2x}{dt} = -\nabla U(x) = F(x), \tag{2.1}$$

for n atom positions [Leimkuhler and Matthews, 2015]. $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing the atoms' masses, $x \in \mathbb{R}^{n\times 3}$ represents the positions of the atoms,

$U(x) \in \mathbb{R}$ is the interaction potential (i.e., the energy) and its gradient ∇U is the force $F(x) \in \mathbb{R}^{n\times 3}$. Intuitively, this differential equation describes that the velocity of a particle depends on the current force it exhibits (and is slowed down by its mass). This force is usually parameterized via a *force field*, where the concrete forces that act upon the atoms and constants are defined. For example, if the atoms are simulated in water, different coefficients are needed than if the simulation were performed in a vacuum. In practice, there are many different concrete force fields to choose from [Halgren and Damm, 2001, Ponder and Case, 2003, Mackerell, 2004]. Although we could solve this differential equation with suitable initial conditions to simulate the atoms' positions over time, we will first look at a more typical formulation of the problem.

Real-world molecular systems are heavily dependent on the current temperature. The higher the system's temperature, the "jigglier" and random the movement of particles, and the easier it is for molecules to overcome high-energy barriers. Moreover, we often want to investigate molecules in a solvent with a particular friction rather than in a vacuum. To overcome these problems, we can use a stochastic differential equation to formulate the problem by the *Langevin* equation of motion

$$\boldsymbol{M}\frac{d^2x}{dt} = -\nabla U(x) - \gamma \boldsymbol{M}\frac{dx}{dt} + \sqrt{2\boldsymbol{M}\gamma k_B T}\boldsymbol{R}. \tag{2.2}$$

This equation differs to the Newtonian definition such that it introduces a friction coefficient γ that slows the particle down and adds a standard normally distributed random displacement $\boldsymbol{R} \sim \mathcal{N}(0, \mathbb{1}^{n\times 3})$ that depends on temperature T and the Boltzmann constant k_B.

2.1.1. Closed-Form Solution with Verlet Integration

There are different ways to solve (stochastic) differential equations. However, in the field of molecular dynamics, Verlet integration [Verlet, 1967] is commonly used. Solving the stochastic differential equation stated in Equation 2.2 with Verlet integration, gives as a timestep-based integration scheme to propagate the particle positions $x_t \to x_{t+1}$ from time t to $t+1$. It has, for example, been implemented in openMM [Eastman et al., 2017] as

$$\begin{aligned} x_{i+1} &= x_i + \Delta t v_{i+1} \\ v_{i+1} &= \alpha v_i + \frac{1}{\gamma}(1-\alpha)\nabla U(x_i)\boldsymbol{M}^{-1} + \sqrt{k_B T(1-\alpha^2)\boldsymbol{M}^{-1}}\boldsymbol{R}, \end{aligned} \tag{2.3}$$

with $\alpha = \exp(-\gamma\Delta t)$. With specific initial positions x_0 and velocities v_0 this scheme can be iteratively applied to generate time-evolving conformations (i.e., spatial arrangement) of the molecule.

Although a small timestep size Δt would create the most fine-grained and detailed simulations, a small step size requires many simulation steps to produce meaningful results. Hence, one typically has to resort to larger step sizes to speed up the simulation. While this might be suitable for slow-transitioning biomolecular systems, the model and hence the simulation could fall apart and produce no meaningful results at all. Determining a suitable step size can be challenging, because even for very specific tasks—such as simulating protein folding—the timescales can vary from microseconds [Lindorff-Larsen et al., 2011] to seconds [Aronsson et al., 1997].

2.1.2. Boltzmann Distribution

When we continue the simulation of a molecule for $t \to \infty$, we can observe a stable distribution of atom positions. Each conformation x is then distributed according to the Boltzmann distribution [Boltzmann, 1868]

$$p(x) = \frac{\exp(-U(x)/k_B T)}{\int_{x \in X} \exp(-U(x)/k_B T) dx} \propto \exp(-U(x)/k_B T). \tag{2.4}$$

We can see that this closely resembles an exponential distribution, where the probability for observing a certain state x is proportional to the exponential of its negative energy: the higher the energy, the less likely the state. Figure 2.1 shows that in systems with a higher temperature, states with higher energy are observed more often. When imagining the extreme case of an infinite temperature, all states have the same probability because the random part from Equation 2.2 would dominate the position.

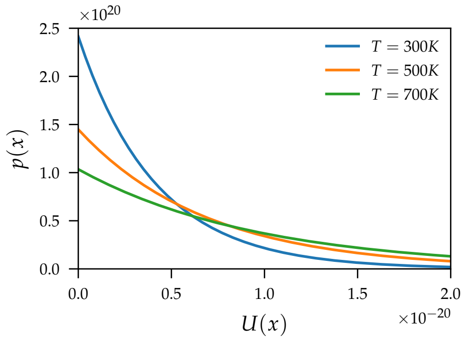

Figure 2.1.: This figure shows the probability of observing states x with a certain energy $U(x)$ in differently tempered heat baths (i.e., constant temperature).

Analytically computing the probability p to observe a conformation x, would require us to compute the normalizing constant $\int_{x \in X} \exp(-U(x)/k_B T) dx$. As this is infeasible in practice, we cannot compute the probability of states and not directly draw samples following this distribution.

However, without computing the normalizing constant, we can still compare the probability of two states as $\frac{p(x_1)}{p(x_2)} = \exp((U(x_1) - U(x_2))/k_B T)$, where the normalizing constant cancels out. To sample conformations from the Boltzmann distribution, we can use this ratio with Markov chain Monte Carlo techniques, or by training a Boltzmann generator: a recent machine learning technique that allow us to learn the Boltzmann distribution with normalizing flows.

2.2. Normalizing Flows

Many real-world tasks aim to learn or approximate an underlying distribution that can only be observed through samples. Normalizing flows [Tabak and Vanden-Eijnden, 2010, Tabak and Turner, 2013] are a class of models that can solve this problem. With a suitable construction, they can be used to efficiently sample new values and calculation the probability of observations. Normalizing flows are versatile and have been used to generate realistic pictures of celebrities [Kingma and Dhariwal, 2018], aid in drug research [Kuznetsov and Polykovskiy, 2021, Garcia Satorras et al., 2021], or to improve sentence embeddings in natural language processing [Li et al., 2020].

At its core, a normalizing flow is a trainable invertible function T that maps the samples of the observed distribution $x \sim X$ from and to a simpler known base distribution Z. Since the mapping T is diffeomorphic[1], each sample $z \sim Z$ corresponds to one sample from the distribution of the data $T(z) = x \sim X$ and vice versa (compare Figure 2.2). To generate new observations, we can draw samples from the simple distribution and map them with T to get a sample in the target space X. Similarly, to evaluate the density of a sample x, we use the inverse of T and evaluate the density of the sample $z = T^{-1}(x)$ in the simpler known distribution.

Typically, Z is a standard normal distribution of the same dimension as the data, hence the name *normalizing* flow. The name flow, on the other hand, stems from the fact that T usually is not a single function but consists of a series of many (simple) diffeomorphic functions $T = T_1 \circ T_2 \circ \cdots \circ T_n$. With this, the samples *flow* through the functions to produce normally distributed samples.

For a fixed $T : \mathbb{R}^D \mapsto \mathbb{R}^D$, the density of the target distribution can be reformulated by

[1]A function is diffeomorphic iff it is invertible, and both the function and its inverse are differentiable.

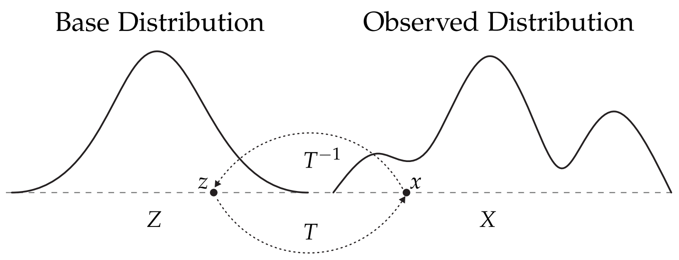

Figure 2.2.: This illustration shows the underlying idea of a normalizing flow. A function T and its inverse T^{-1} map the samples of a base distribution Z (i.e., a standard normal) from and to the samples of the observed distribution X.

the change of variables formula [Papamakarios et al., 2021] such that

$$\begin{aligned} p_X(x) &= p_Z\left(T^{-1}(x)\right)\left|\det J_T(z)\right|^{-1} \\ &= p_Z\left(T^{-1}(x)\right)\left|\det J_{T^{-1}}(x)\right|, \end{aligned} \tag{2.5}$$

where $J_T(z)$ is the $D \times D$ Jacobian of $T(\cdot)$

$$J_T(z) = \begin{bmatrix} \frac{\partial T}{\partial z_1} & \cdots & \frac{\partial T}{\partial z_D} \end{bmatrix} = \begin{bmatrix} \frac{\partial T_1}{\partial z_1} & \cdots & \frac{\partial T_1}{\partial z_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial T_D}{\partial z_1} & \cdots & \frac{\partial T_D}{\partial z_D} \end{bmatrix}, \tag{2.6}$$

that contains the gradient for each parameter. This formula allows us to calculate the probability of samples $x \sim X$ by evaluating the density of $z = T^{-1}(x)$. The additional term $\left|\det J_T(z)\right|^{-1}$ accounts for the change in volume between the two spaces to ensure that p_X is a well-defined density [Papamakarios et al., 2021]. Since p_Z is typically a Gaussian distribution, it can be efficiently evaluated by construction. The only thing missing is that we still need to ensure that the determinant of the Jacobian of T can be efficiently computed as well.

2.2.1. Architecture and Training

Before we can train a normalizing flow, we must decide on a concrete family of transformations to choose the optimal T from. Typically, this involves a trade-off where the functions should be as expressive as possible while allowing fast computation of its inverse and $\det J_T$ [Kobyzev et al., 2021]. One of the simplest choices is an *affine* transformation, where $T(z) = \boldsymbol{A}z + \boldsymbol{b}$ with the optimizable parameters $\boldsymbol{A} \in \mathbb{R}^{D \times D}$ and $\boldsymbol{b} \in \mathbb{R}^D$. The Jacobian is simply the matrix $\boldsymbol{A}$, and the inverse of T can be computed in

$\mathcal{O}(D^3)$. However, this family of transformations is limited in its expressiveness as it already fails to convert a standard normal distribution to a bimodal distribution.

Many other transformations are suitable to construct normalizing flows [Kingma et al., 2016, Kingma and Dhariwal, 2018, Kobyzev et al., 2021]. Especially when multiple transformations are concatenated, relatively simple transformations can sometimes already capture diverse distributions. However, in most cases, more sophisticated approaches are necessary for a good approximation.

More expressive functions, such as neural networks, are non-invertible and thus cannot be used directly. Dinh et al. [2017] introduced a coupling architecture for normalizing flows, enabling us to use arbitrary deterministic functions regardless of their invertibility. Instead of directly mapping the samples with the neural network, they propose that the neural network predicts the parameters of an invertible function $\boldsymbol{\theta} = NN(z)$. The parameters $\boldsymbol{\theta}$ could be $\boldsymbol{A}, \boldsymbol{b}$ for an affine transformation, or any parameters that characterize a diffeomorphic function. Since these parameters are fixed for a given sample, the transformation can still be easily inverted. And because each sample will get its own parameters, simple transformations, such as polynomials, can be used to approximate complex functions [Durkan et al., 2019].

There are different ways to train normalizing flows [Kobyzev et al., 2021]. Most commonly, they are trained by maximizing the log-likelihood of the observed samples. For this, we assume that we have observed samples $x \sim X$ following an unknown density p_X^* that should be learned. p_Z is the base distribution (e.g., a Gaussian) and $p_X(\cdot|\boldsymbol{\theta})$ is the density of X the flow with parameters $\boldsymbol{\theta}$ yields. In a perfect approximation, p_X^* is equal to p_X. To find the optimal parameters $\boldsymbol{\theta}^*$, we minimize the loss

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{ML}}(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} D_{KL}\left[p_X^*(x) \;||\; p_X(x|\boldsymbol{\theta})\right] \\
&= \arg\min_{\boldsymbol{\theta}} -\mathbb{E}_{x\sim X}\left[\log\left(p_Z\left(T_{\boldsymbol{\theta}}^{-1}(x)\right)\right) - \log\left|\det J\left(T_{\boldsymbol{\theta}}^{-1}(x)\right)\right|\right],
\end{aligned} \tag{2.7}$$

where D_{KL} is the Kullback–Leibler divergence [Csiszar, 1975].

2.2.2. Boltzmann Generators

A Boltzmann generator [Noé et al., 2019] is a specific type of normalizing flow that learns to approximate the Boltzmann distribution of a (molecular) system. They have been used in various biochemical contexts [Noé et al., 2019, Liu et al., 2022, Köhler et al., 2023]. Once trained, they can efficiently sample states according to the underlying energy. The generated samples will appear with a probability proportional to their energy, meaning that high-energy states are observed rarely. Further, Boltzmann generators allow us to compute the probability $p(x)$ of each conformation x—in a way, approximating the normalizing constant of the system's energy function.

As Boltzmann generators are normalizing flows, they can be trained identically by maximizing the log-likelihood of samples (compare Equation 2.7). The only difference is that, in this case, the samples represent observations of a system's state. To get samples to train a Boltzmann Generator with maximum likelihood, we could perform a long-running MD simulation. With enough samples, and a sufficiently long-running simulation, a Boltzmann generator trained with maximum likelihood will be a good approximation. However, in most cases this is not possible, and thus Noé et al. [2019] introduce an additional loss term

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} w_{\mathrm{ML}} \mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta}) + w_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}}(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} w_{\mathrm{ML}} D_{KL}\left[p^*_X(x) \mid\mid p_X(x|\boldsymbol{\theta})\right] + w_{\mathrm{KL}} D_{KL}\left[p_Z(z) \mid\mid p_Z(z|\boldsymbol{\theta})\right] \\
&= \arg\min_{\boldsymbol{\theta}} -w_{\mathrm{ML}} \mathbb{E}_{x\sim X}\left[\log\left(p_Z\left(T^{-1}_{\boldsymbol{\theta}}(x)\right)\right) - \log\left|\det J\left(T^{-1}_{\boldsymbol{\theta}}(x)\right)\right|\right] \\
&\qquad - w_{\mathrm{KL}} \mathbb{E}_{z\sim Z}\left[\log\left(U\left(T_{\boldsymbol{\theta}}(z)\right)\right) - \log\left|\det J\left(T_{\boldsymbol{\theta}}(z)\right)\right|\right]
\end{aligned} \tag{2.8}$$

that uses a state's energy to speed up the training. $p_Z(\cdot|\boldsymbol{\theta})$ is the density of the latent space, when sampling according to the true Boltzmann distribution $x \sim p^*_X$ and transporting it with $z = T^{-1}(x)$. $p_Z(z)$ is defined by the base distribution, which—as with normalizing flows—is usually a Gaussian. The two independent terms are often scaled with weights $w_{\mathrm{ML}}, w_{\mathrm{KL}}$ depending on the concrete problem. Intuitively, the additional term $\mathcal{L}_{\mathrm{KL}}(\boldsymbol{\theta})$ is used for energy-based training and ensures that the probability with which states are sampled, aligns with their energy U. Noé et al. [2019] also present a third term in the loss to steer the sampling along specific coordinates, which is not relevant for this thesis.

Both parts of the loss (i.e., $\mathcal{L}_{\mathrm{ML}}$ and $\mathcal{L}_{\mathrm{KL}}$) are computationally expensive to evaluate with an increasing number of dimensions. The maximum likelihood part relies on samples, requiring expensive MD simulations, and evaluating the energy for energy-based training is also demanding for complex systems. Consequently, training Boltzmann generators for larger systems, such as proteins, is a challenging endeavor. In some cases, long-running molecular simulations for well-known systems, such as the protein BPTI, have been computed on supercomputers and are readily available [Shaw et al., 2010] which makes maximum likelihood training easier. There are also recent advances that make the training procedure itself more efficient [Köhler et al., 2021, Midgley et al., 2022, 2023a,b, Felardos et al., 2023] by steering which samples to draw. Despite all those efforts, Boltzmann generators are still only suitable for smaller systems.

2.3. Transition Path Sampling

Under most circumstances, molecular systems stay in a stable equilibrium and only exhibit small fluctuations. Transitions to a different state usually require overcoming high energy barriers and thus only appear rarely. However, when such a transition occurs, either caused by an external factor or pure chance, it will happen quickly. The molecule will then remain in the new state for a relatively long time until again transitioning to a different state. While these events happen so rarely, they do happen and can give insights into the underlying molecule. In many cases, these transitions happen in femtoseconds, or sometimes even faster [Dellago et al., 2006]. While MD simulations are a great tool to capture the typical fluctuations in the stable states, they are not suitable to capture transitions. Capturing a single transition would require 10^{15} steps with a step size of femtoseconds. Due to these vastly different timescales, molecular dynamics simulations cannot be used to produce these transitions reliably.

The goal of transition path sampling (TPS) [Dellago et al., 1998b, Bolhuis et al., 2002] is to efficiently simulate many of these rare transitions. While some works only focus on finding a single pathway that requires the least energy [Sheppard et al., 2008], here we focus on sampling an *ensemble* of transitions. Sampling more paths is a more challenging problem but can give more insights into the underlying system and behavior of the states. As there are an infinite number of transitions, this ensemble should contain many paths with a high probability and few with a low probability, that they would appear in the real-world—as shown in the energy landscape in Figure 2.3.

Transition path sampling, or more specifically an ensemble of transition pathways, can be used for various applications in many different fields such as designing catalysts [Basner and Schwartz, 2005, Crehuet and Field, 2007, Quaytman and Schwartz, 2007, Saen-oon et al., 2008, Bučko et al., 2011] and materials [Xi et al., 2013, Selli et al., 2016, Sharma et al., 2016], drug discovery [Kirmizialtin et al., 2012, 2015, Dickson, 2018], and various other biological and chemical contexts [Escobedo et al., 2009, Varilly and Chandler, 2013]. In the following sections will give an overview of the technical details and methodologies behind TPS. For readers that want to gain a more in-depth view of the topic, we can recommend the works of Dellago et al. [1998b, 2006], Dellago and Bolhuis [2009], Escobedo et al. [2009] or Bolhuis and Swenson [2021] for a more recent view. These works have served as an inspiration for this section.

2.3.1. Formalization of Transition Pathways

A single transition pathway comprises a series of time-equidistant *frames* that fully characterize it. Each frame can be seen as a snapshot of the system's state, which

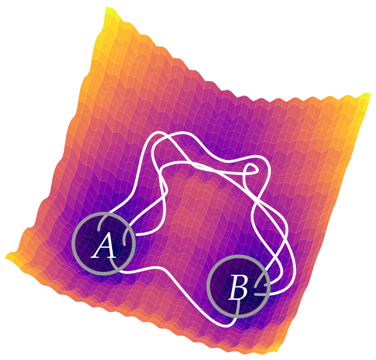

Figure 2.3.: A 2D potential energy surface that has two meta-stable states: A and B. There are two main channels of transitions between the states: Many transitions go around the high energy barrier, and few pass it. These transitions pathways create an ensemble of paths that can be used to analyze the behavior of the system. With these few transitions, we can already learn the two main reaction channels of the system, but typically a larger ensemble with more pathways is necessary to capture rare transition modes.

evolves over time. In molecular systems, a snapshot would contain all the positions and velocities of the atoms at a specific point in time, and a transition path would describe how the atoms change over time. More formally, a transition path $\mathcal{T}$ consists of the positions and velocities $(\boldsymbol{x}, \boldsymbol{y})$ and is an ordered sequence of ℓ frames

$$\mathcal{T} = (\boldsymbol{x}, \boldsymbol{v}) = ((x_1, x_2, \ldots, x_\ell), (v_1, v_2, \ldots, v_\ell)), \tag{2.9}$$

where frame i is observed at time $(i-1)\Delta t$, the atoms are at position x_i and have velocity v_i. Each frame changes with a small timestep Δt according to the underlying dynamics. If all transition paths in the ensemble have the same number of l frames—and thus require the same time $l\Delta t$ for a transition—we refer to it as fixed-length TPS. In flexible-length TPS the paths can have a variable number of frames.

A transition path is thus a series of frames that all depend on each other. If we changed the first position, all subsequent frames would change accordingly. This connectedness is a major difference from what a Boltzmann generator would allow us to produce, as it samples all frames independently from each other.

2.3.2. Transitions between Frames

Since we want to create an ensemble of paths where transitions appear with the same probability as in an infinitely running MD simulation, we will look at how this transition probability can be computed. In this work, we assume that the evolution of frames is a Markov chain, meaning that each snapshot contains the necessary information so that the next steps only depend on the current one. The probability of a given trajectory $\mathcal{T}$ depends on the initial conditions (x_1, v_1), and the underlying dynamics used to evolve the frame $(x_i, v_i) \to (x_{i+1}, v_{i+1})$.

$$p_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y}) = \rho(x_1, v_1) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i) \tag{2.10}$$

describes this probability, where ρ computes the probability of the start states and depends on the initial conditions of the system. Typically, the velocity can be assumed independent from the position such that $\rho(x_1, v_1) = p(x_1) \cdot \mathcal{N}(v_1 \mid 0, \boldsymbol{M}^{-1} k_B T)$, where $p(x_1)$ is the Boltzmann distribution (compare Section 2.1.2), and the velocities are distributed according to a normal distribution [Castellan, 1983, Sec. 4.6].

2.3.3. State Definitions

The probability density function $p_{\mathcal{T}}$ can be used to evaluate the probability of any time evolution of frames, regardless of their properties. In TPS however, we are only interested in so-called *reactive* paths, that connect two regions of interest: A and B. For this, we will slightly adapt the transition probability such that

$$p_{AB}(\boldsymbol{x}, \boldsymbol{v}) \propto \mathbb{1}_A(x_1) \cdot \rho(x_1, v_1) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i) \cdot \mathbb{1}_B(x_\ell), \tag{2.11}$$

with the indicator functions

$$\mathbb{1}_A(x) = \begin{cases} \mathbf{1}, & \text{if } x \in A \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \mathbb{1}_B(x) = \begin{cases} \mathbf{1}, & \text{if } x \in B \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

to only allow paths with a certain start and end position. With the introduction of these indicator functions, the function p_{AB} typically does not integrate to 1 anymore and is thus not a proper density function. Integrating p_{AB} to compute the normalizing constant and thus get exact densities would be computationally infeasible. We will see in the later sections that this does not pose as an issue, as we will be computing the ratio of probabilities where the normalizing constant cancels out.

In most cases, we have an intuitive definition for the states A, B, for example a protein that is folded in two different ways and behaves differently in those. In lower-dimensional cases, we might even be able to visually investigate the energy landscape to come up with concrete state definitions. For example, in Figure 2.4 we see the two-dimensional energy surface of a toy example. The local minima of this function can be good indicators for which states are suitable candidates. However, states are not a single point, but should include all surrounding conformations so that it covers the most typical fluctuations in this equilibrium. When defining the states too broadly, the resulting transition paths could end up in a different state or not converge at all. In this case, the pathways will not be meaningful, as they will not be reactive. The stable states in Figure 2.4 could potentially be defined larger without negatively impacting the sampling procedure.

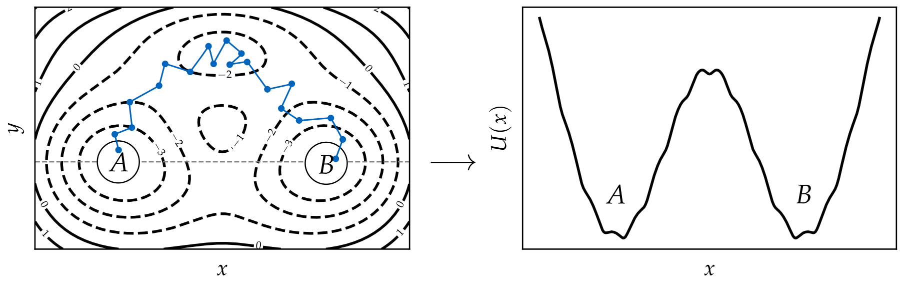

Figure 2.4.: *Left:* Shows the contour plot of a 2D energy surface with multiple local minima. The two lowest minima define the states A and B. An exemplary transition path is shown in blue. *Right:* A plot showing the energy at a specific y, highlighting the high-energy barrier between the states A and B.

This issue becomes more prominent when dealing with higher-dimensional systems that cannot be visualized so easily anymore. In these cases, states are typically not defined via the atom positions directly but rely on lower-dimensional order coordinates [Dellago et al., 2006]. These artificial parameters concisely describe the system, and could for example be based on bond lengths, or angles. The molecule alanine dipeptide, for example, can be described by two dihedral angles, as seen in Figure 2.5. Although good order-parameters make it substantially easier to define states, they themselves are difficult to determine and there are various ways to do this [Molgedey and Schuster, 1994, Gu et al., 2014, Kabelka et al., 2021]. Order parameters are also beneficial in other areas [Liao, 2020], and because states can be defined without these order parameters, there have been works that use TPS to determine them [Hooft et al., 2021, Ray et al., 2023].

2.3.4. Markov Chain Monte Carlo Sampling

Almost all transition path sampling techniques rely on Markov chain Monte Carlo (MCMC), typically the Metropolis-Hastings [Metropolis et al., 1953, Hastings, 1970] algorithm. It is an iterative approach, where a so-called transition *kernel* uses the

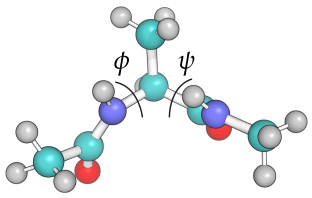

Figure 2.5.: The molecule alanine dipeptide with its main dihedral angles ϕ, ψ is shown.

current path to suggest a new one. This algorithm phrases the underlying process as a Markov chain, where the next transition depends on the current path (and a random probability). The main steps of the algorithm are

1. Use the current path and a transition kernel K to propose a new path,
2. Compute the acceptance probability α,
3. Draw a uniform random variable $u \sim \mathcal{U}_{[0,1]}$,
4. Accept this proposed path if $u \leq \alpha$.

This procedure is then repeated to iteratively sample new paths and create an ensemble. Whenever a path is accepted, we add it to the ensemble and is it as the new current path. If it is discarded, the current path will not be modified. We will now investigate the acceptance probability α and the conditions that need to be fulfilled so that this algorithm samples the true underlying ensemble of paths.

First of all, with Metropolis-Hastings, each proposed trajectory is only accepted with the probability

$$\alpha = \min \left\{ 1, \frac{p_{AB}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}})}{p_{AB}(\boldsymbol{x}, \boldsymbol{v})} \cdot \frac{q(\boldsymbol{x}, \boldsymbol{y} \mid \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}})}{q(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}} \mid \boldsymbol{x}, \boldsymbol{y})} \right\}, \tag{2.13}$$

where q is the transition probability of kernel K, and $q(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}} \mid \boldsymbol{x}, \boldsymbol{y})$ the probability that we sample the new path $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}})$, given the current path $(\boldsymbol{x}, \boldsymbol{v})$[2]. Intuitively, the acceptance rate α compares the probability of the new path to the old. More probable and less sampled paths are more likely to get accepted. Note that p_{AB} will be 0 for non-reactive paths. In these cases, we discard the current path and simply sample a new one. In Section 2.4, we will discuss various different choices for the transition kernel, and similarly will propose new kernels in Chapter 3.

The second requirement for this algorithm to work is of a more technical nature and requires that the Markov chain has a unique stationary distribution [Robert and

[2]As mentioned earlier, this procedure has very lean requirements and can operate on unnormalized densities for p_{AB} and q, since the normalization constants cancel out.

Casella, 2004, Ch. 7]. One sufficient condition for this is that the Markov chain is positive recurrent, meaning all possible states can be reached in an expected finite time [Robert and Casella, 2004, Ch. 6]. This is a property, that the transition kernel must fulfill. While it might be challenging to prove, most kernels that can generate all possible transition paths achieve this.

With these conditions fulfilled, we can sample new transition paths by iteratively changing an initial path and building up an ensemble of paths. The transition paths sampled by this procedure are guaranteed to appear in the ensemble with the probabilities as governed by the underlying Boltzmann distribution. Proposal kernels might have different behaviors regarding runtime or stability, but as long as the conditions are met, they are guaranteed to sample the correct ensemble eventually.

2.3.5. Finding an Initial Trajectory

When sampling transition paths with MCMC, we need a way to provide an initial path to start the procedure. While the quality of this initial path does not impact the technical guarantees of most algorithms, it can significantly impact the runtime. A low-energy path could make it difficult to produce new paths or might limit the algorithm to only explore similar paths instead of exploring less-likely but diverse paths. In practice, people often start with a high-temperature simulation to create an initial path [Rowley and Woo, 2007]. This higher temperature allows the molecule to overcome high-energy barriers more easily, but typically adds significant computational overhead [Dellago et al., 2006, Rowley and Woo, 2007]. To make it more realistic, this path can then be equilibrated by performing short simulations at the target temperature. Other approaches include adding external forces or initiating trajectories from predefined points with different velocities [Rowley and Woo, 2007]. There is no general procedure available that works for all systems [Dellago and Bolhuis, 2007].

2.4. Related Work

Next, we will explore different concrete choices for the transition kernel in Section 2.4.1 and Section 2.4.2. In Section 2.4.3, we will look at works that have explored MCMC that do not operate in product space, but in a space with nice properties.

2.4.1. Classical Transition Path Sampling

One of the most versatile and commonly used proposal kernels is based on shooting moves [Dellago et al., 1998a,b]. They build on the idea that a random frame (x_i, v_i) of an existing trajectory is selected, where the velocity is set to a new random vector

$(x_i, \tilde{v}_i)$. This frame with the new velocity is then simulated forward (and backward) in time to produce a new path, as seen in Figure 2.6. If the simulation does not seem to reach the target within a fixed number of steps, the path is discarded, we repeat the procedure. In molecular terms one can imagine this as taking a random frame of the path and "shooting" it in a random direction and seeing if it "lands" in the target region.

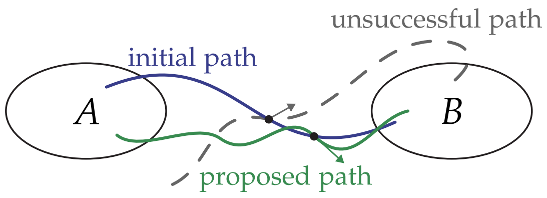

Figure 2.6.: An illustration of the shooting scheme. A random frame is selected from the current path (blue line), and a random velocity is set (arrows). This has been performed twice. The shooting initiated by the gray arrow has been discarded as it only reaches state B, but not A. The green shooting proposal yields a reactive path, which will be accepted or rejected based on the acceptance probability α.

Once a suitable proposal trajectory has been found, it goes through the Metropolis-Hastings procedure as discussed in Section 2.3.4. If we assume microscopic reversibility for the simulation—which many integrators fulfill—the path probability p_{AB} will be the same as p_{BA} with time moving backwards. Under this assumption, the acceptance probability simplifies to

$$\alpha = \frac{\text{\# of frames in proposed path}}{\text{\# of frames in current path}}, \tag{2.14}$$

as for example discussed by Dellago et al. [2006], Jung et al. [2017]. This is a major advantage of shooting moves, because while the paths might take extensive molecular dynamics to simulate, there is almost no overhead involved to accept or reject paths. Especially for fixed-length TPS, where all trajectories have the same number of frames, this ratio is equal to 1, and *all* sampled reactive trajectories are accepted.

Types of Shooting Schemes

Shooting moves are mainly classified whether they are one-way or two-way shooting techniques and if they are aimless. One-way shooting techniques only simulate the trajectory with the new velocity either forward, or backward in time, while keeping the other part of the path fixed. In this setting, the trajectory is only partially modified at each proposal, often resulting in highly correlated paths [Bolhuis and Dellago, 2010].

Two-way shooting schemes require more computational power because they simulate the complete trajectory starting from the given frame forward, and backward. Two-way shooting decreases the probability of successful paths because the simulation now needs to reach states A and B, compared to one-way shooting, where one state is always fixed. If the shooting scheme is aimless, the velocity and the frame to shoot from are chosen at random. Many works bias shooting schemes to produce more accepted paths [Bolhuis et al., 2002, Juraszek and Bolhuis, 2008]. Juraszek and Bolhuis [2008], for example, propose to use a Gaussian bias function instead.

Other Approaches

Shooting moves are widely used because of their simplicity and effectiveness [Bolhuis and Swenson, 2021]. However, they face the trade-off between efficiency and diverse transition paths [Bolhuis and Swenson, 2021]. If the added velocity is very large, the paths will likely not reach the target state and thus get discarded. On the other hand, when only adding a small force, the paths that will likely be accepted but are highly correlated. This can be especially problematic if there are multiple transition modes to explore, where there is slow mode switching. Many different versions of shooting have thus been proposed to mitigate such issues and improve the acceptance probability [Juraszek and Bolhuis, 2008, Borrero and Dellago, 2016, Bolhuis and Swenson, 2021, Jung et al., 2017].

But there are also different transition path sampling techniques, such as the string method [E et al., 2005]. The main idea in this approach is that small molecular simulations are performed for each individual frame, while introducing a potential that keeps the individual frames together and prevents them from converging to the local minima. However, other approaches are rarely used in practice and current methods, such as the one introduced by Jung et al. [2023], rely on variations of shooting moves.

2.4.2. Machine Learning-based Transition Path Sampling

Similarly, most machine learning-based techniques rely on shooting moves and employ neural networks to determine the points to shoot from [Falkner et al., 2023, Jung et al., 2023]. Jung et al. [2023] propose to learn a biasing function that predicts the frame most suitable to shoot from. The idea is that states A and B are separated by a very diverse potential energy. When we simulate points very close to A, they will end up in state A again, and similarly for B. The idea is to make use of the *committor* function $p_B(x)$, which determines the probability that a configuration x will end up in state B if simulated forward in time. For simplicity, we assume that there are only two stable states in the system, and hence $p_A(x) = 1 - p_B(x)$. However, somewhere along the transition path, there is likely an unstable saddle point (compare Figure 2.4 (right)), where the probability of $p_A(x) = p_B(x) = 0.5$, or at least close to this. Jung et al. [2023] use a neural network to learn to predict at which frame this committor probability is closest to 0.5 and use this frame to shoot from. They demonstrate their approach on a variety of different systems and show that this selection procedure increases the probability of accepted trajectories.

Falkner et al. [2023] do not limit themselves to choose frames from the current path, but sample random configurations in parallel to shoot from. For this, they employ a Boltzmann generator to sample conformations from the systems. They then use the Boltzmann generator to re-weight the proposed paths to create a correct ensemble. Since they do not need a current path to sample transition paths, their approach can be efficiently parallelized.

Liu et al. [2022] also employ a Boltzmann generator but only aim to find the minimum energy path between two meta-stable states instead of sampling the path ensemble. They introduce constraints to the normalizing flow with an additional loss term during training so that the linear path in latent space represents the path of least energy. Their approach can only be used to find this single minimum energy path and thus solves a simpler task than sampling a whole ensemble of transition paths.

Lelièvre et al. [2023] explore two different ways to sample transition paths. In the first approach, they explore a data-driven approach where they train a variational autoencoder (VAE) [Kingma and Welling, 2013] on existing transition paths. However, they authors show that this approach does not produce meaningful results and we claim that this is not meaningful because they rely on existing transition paths that are difficult to obtain. Their second approach on the other hand, explores reinforcement learning techniques to sample transition paths. They aim to maximize the likelihood of trajectories leaving the initial meta-stable state. In our opinion, their experiments are limited as they are only explored on a smaller two-dimensional toy system, but further exploration of the reinforcement approach could be promising.

2.4.3. Non-Product-Space Markov Chain Monte Carlo

Typically, MCMC operates in the product space, so in the case of transition path sampling this means that we operate on the atom positions of the molecules. However, in this section, we will explore works that use a different space to improve the conditioning of MCMC algorithms for challenging spaces. While performing MCMC in a different space does not change the underlying technical guarantees, it can greatly improve the runtime, stability, and rate for convergence in ill-posed product spaces.

Parno and Marzouk [2018] were the first to introduce the notion of MCMC in a different space. They showed how a fixed non-linear transport map T can be used to map the samples to a non-Gaussian distribution, where they space can be explored more efficiently. They then show how this can be phrased as an optimization problem where they use the previous MCMC samples to train this fixed transport T. They extend their studies in Marzouk et al. [2016] and demonstrate it in a broader context. Titsias [2017] similarly constructs such a transport map with affine transformations, which, as already discussed in Section 2.2.1, lack expressiveness.

Hoffman et al. [2019] build upon these ideas and construct an inverse autoregressive normalizing flow [Kingma et al., 2016] to construct a latent space. In this latent space, they perform Hamiltonian MCMC [Duane et al., 1987, Neal, 2011] where samples are modified according to Hamiltonian dynamics. Doing this in a Gaussian latent space reduces the computational requirements. This is because computing the gradient in product space is the most expensive part of Hamiltonian MCMC, but the gradient of a Gaussian distribution can be easily computed. However, since this latent space allows for faster mixing, they are able to demonstrate that their approach is consistently better than classical Hamiltonian MCMC. In this work, we will also explore how their approach can be extended to fit the TPS framework and will evaluate the results.

To our knowledge, there are no works that use non-product space MCMC for transition path sampling. In the next chapter, we will explore how this framework can be extended to operate in a latent space.

3. Latent Space Transition Path Sampling

In this chapter of the thesis, we will explore a novel approach to sample transition paths in the latent space of a Boltzmann generator. For this, we will first motivate in Section 3.1 the reason why sampling in latent space can be beneficial and which problems we try to solve with it. This is followed by a precise formulation of the framework in Section 3.2, and a variety of different MCMC kernels that operate on latent space representations in Section 3.3. We will then explore some drawbacks of the approach and introduce approximations with further improvements in Section 3.4. While these adaptions influence the theoretical guarantees, they can aid the empirical results in some cases. We will end the chapter in Section 3.5 by showing how we can train a Boltzmann generator on a uniform base distribution while still creating a Gaussian latent space. The results of this framework will be investigated in Chapter 5

3.1. Motivation

Due to its versatility and manageable computational effort, shooting-based TPS has become the standard approach for sampling an ensemble of reactive pathways. However, shooting still has two significant drawbacks. One is that it requires expensive MD simulation to propose new paths. Especially for larger molecules, such as proteins, this can be an obstacle to producing a representative ensemble of paths. The second disadvantage is more subtle and needs further explanation.

Shooting-based TPS faces a trade-off between exploring the energy landscape (i.e., proposing diverse paths) and the probability for a path to be accepted. If the proposed shooting move only changes the velocity by a small amount, the resulting path will be similar. As such, the new path will almost surely be reactive, and the acceptance probability of Metropolis-Hastings will be high. While this results in a computationally efficient sampling algorithm, the produced paths will be highly correlated with each other, and different transition channels might not be sampled. On the other end of the spectrum, when the random velocity is too large, the trajectory will be far away from the target states, not producing a reactive path. In the worst case, this means that no transition paths can be found at all.

Choosing a suitable velocity depends on the underlying properties of the energy function of a particular molecule and has to be reviewed for each system. For some ill-posed energy landscapes, shooting-based algorithms can get stuck even with a suitable velocity offset. Imagine a scenario where we encounter a path with low energy and steep energy barriers surrounding it. A large random velocity would be needed to get out of this well, but with such, the simulation will unlikely reach the target states. With this, the performance of TPS degrades to the one of molecular simulation because the movement is not guided anymore. More sophisticated approaches are needed to overcome these drawbacks.

In this chapter of the thesis, we will introduce a new framework for TPS that allows us to generatively sample complete transition paths with MCMC using machine learning. We will propose relatively simple modifications to the current path in a well-conditioned learned latent space to overcome the unfavorable properties of ill-posed energy landscapes. This approach can increase performance since these operations do not rely on expensive MD simulations. Further, some operations, such as computing the gradient in this latent space, can be done efficiently, speeding up proposal algorithms.

3.2. Latent MCMC TPS Framework

Similarly to existing TPS methods, we will make use of MCMC to sample an ensemble of transition paths. Beginning with an initial path, we iteratively propose new paths based on the current one and accept or reject based on an acceptance probability (compare Section 2.3.4). By repeating this, we sample multiple transition pathways that will eventually follow the true underlying probability defined by the energy landscape. In our concrete case, we will propose the paths based on the latent space representation of frames.

For this to work, we assume access to a Boltzmann generator, trained for the molecule of interest. This gives us a function $T : Z \mapsto X$, and its inverse $T^{-1} : X \mapsto Z$ that can map conformations of molecules $x \in X$ to a latent space representation $z = T^{-1}(x) \in Z$ and vice versa. Each frame of a given transition path $\boldsymbol{x} = (x_1, x_2, \ldots, x_\ell)$ in product space can then be transported to the latent space spanned by the Boltzmann generator, such that $\boldsymbol{z} = (T^{-1}(x_1), T^{-1}(x_2), \ldots, T^{-1}(x_\ell))$. Based on a latent proposal kernel K, we propose a new path $\tilde{\boldsymbol{z}} = K(\boldsymbol{z})$. In Section 3.3, we will introduce different concrete choices for this proposal kernel. The newly proposed latent path $\tilde{\boldsymbol{z}}$ can then be transported back into configuration space X by mapping each frame with T such that $\tilde{\boldsymbol{x}} = (T(\tilde{z}_1), T(\tilde{z}_2), \ldots, T(\tilde{z}_\ell))$. We then evaluate the proposed path $\tilde{\boldsymbol{x}}$ with a modified Metropolis-Hastings acceptance criterion.

This procedure is illustrated in Figure 3.1. We can see that a small modification in the latent space can correspond to a significant change in the product space. With this,

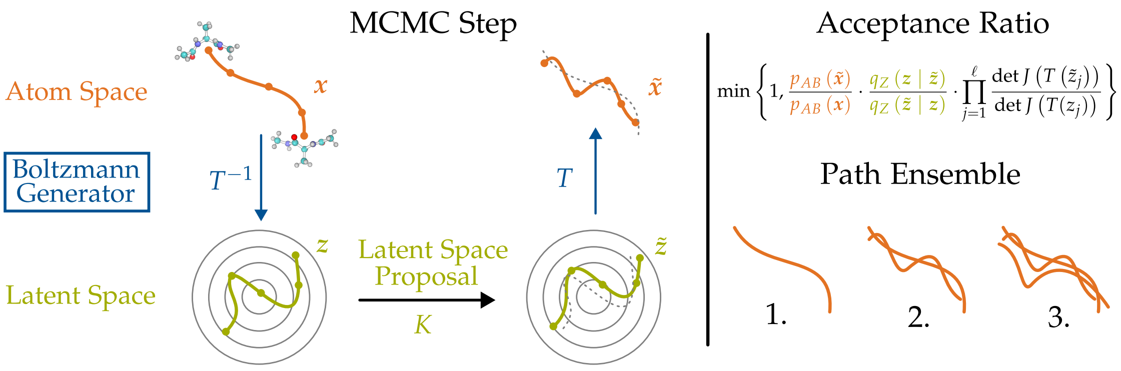
MCMC Step
Atom Space
x
$\tilde{x}$
Boltzmann Generator
T^{-1}
T
Latent Space
z
$\tilde{z}$
Latent Space Proposal
K
Acceptance Ratio
$\min\left\{1, \frac{p_{AB}(\tilde{x})}{p_{AB}(x)} \cdot \frac{q_Z(z \mid \tilde{z})}{q_Z(\tilde{z} \mid z)} \cdot \prod_{j=1}^{\ell} \frac{\det J(T(\tilde{z}_j))}{\det J(T(z_j))}\right\}$
Path Ensemble
1.
2.
3.

Figure 3.1.: A transition path x is moved into the latent space by a Boltzmann generator $T^{-1}(\cdot)$ to produce a path z in latent space. A proposal kernel K uses this latent representation to propose a new path $\tilde{z}$. The Boltzmann generator is then used to bring the path back into configuration space to obtain a new proposal $\tilde{x}$. Whether this new path is accepted is decided by the acceptance criterion. Over time, we sample a path ensemble containing representative transition pathways.

simple/small modifications still allow us, to sample diverse transition paths. Each proposed path is then accepted or rejected, iteratively building up the path ensemble. The specific acceptance probability will be derived in Section 3.2.2.

By limiting ourselves to an analytically known latent space, namely where the variables are distributed according to a standard normal distribution, many calculations can be simplified and easily evaluated. This allows us to exploit the beneficial properties of the latent space—such as fast gradient evaluation or overcoming high energy-barriers more easily—and sample trajectories that would have been difficult with classical TPS.

Over the next sections, we will investigate this setting in more detail. We will first demonstrate how this approach can be used to sample a single path in Section 3.2.1, which can then be used as the initial path for MCMC. In Section 3.2.2, we will derive how the Metropolis-Hastings algorithm can be modified to sample paths in latent space. Afterwards, in Section 3.2.3, we investigate the missing pieces and show how we can compute the probabilities necessary to decide whether to accept or reject a proposed path. We conclude this chapter with a complete algorithm, laying out all the necessary steps in Section 3.2.4.

3.2.1. Constructing an Initial Path in Latent Space

The first step for an MCMC procedure is to get a suitable initial value. As discussed in Section 2.3.5, finding such a trajectory can be difficult, and a wide variety of approaches exist to find one [Dellago et al., 2006, Rowley and Woo, 2007, Dellago and Bolhuis, 2007]. While the quality of the initial path does not impact the theoretical guarantees, it can impact the runtime. For example, paths that are linearly interpolating the states in the product space are unrealistic [Zhu et al., 2019] and, hence, do not make for a good initial path.

However, since a trained Boltzmann generator already has some notion about the energy distribution, we can construct a simple path in latent space that very likely is better suited than other approaches. For this, we propose to transport frames in the initial states $x_1 \in A, x_\ell \in B$ to the latent space such that $z_1 = T^{-1}(x_1)$, and $z_\ell = T^{-1}(x_\ell)$. By linearly interpolating between these states in latent space, we can construct a trajectory $\boldsymbol{z} = (z_1, z_2, \dots, z_\ell)$. Transporting each frame of this path back into product space gives us an initial path $\boldsymbol{x}$, as seen in Figure 3.2. Such a path that is linear in latent space can be constructed efficiently, while it still represents a relatively realistic path. Note however, that there are no guarantees for such paths and neither the spacing between the frames. When constructing a path with MD, the frames will be spaced time-equidistant, with this method the distance between frames is unknown. We will explore such paths in more detail for the molecule alanine dipeptide in Section 5.4.1.

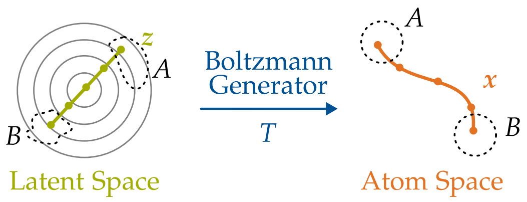

Figure 3.2.: A transition path can be constructed by linearly interpolating between two states in latent space and transporting the frames with a Boltzmann generator. Such paths are non-linear in the product space and are more realistic than linear interpolation in product space.

Similar work has been done by Liu et al. [2022], where they explored a modified Boltzmann generator loss so that paths linear in latent space, have minimum energy.

3.2.2. MCMC Framework for Latent Paths

Once we have access to an initial path, we can begin the MCMC procedure and iteratively propose new paths. However, we need to account for the fact that proposals are made in the latent space instead of the product space. For this, we assume that our kernel K makes proposals in latent space with the probability for the forward path proposal $q_Z(\tilde{\boldsymbol{z}} \mid \boldsymbol{z})$. This probability depends on the concrete kernel K and describes the probability to sample the path $\tilde{\boldsymbol{z}}$ given $\boldsymbol{z}$. Similarly, the backward proposal can be written as $q_Z(\boldsymbol{z} \mid \tilde{\boldsymbol{z}})$. To compute the acceptance probability of a path, we need to evaluate these proposal probabilities in the product space with $q(\cdot \mid \cdot)$ instead of the latent space $q_Z(\cdot \mid \cdot)$. The proposal kernel in product space can be written in terms of the latent space proposal and takes the form of

$$q(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = p(\boldsymbol{z} \mid \boldsymbol{x}) \cdot q_Z(\tilde{\boldsymbol{z}} \mid \boldsymbol{z}) \cdot p(\tilde{\boldsymbol{x}} \mid \tilde{\boldsymbol{z}}). \tag{3.1}$$

$p(\boldsymbol{z} \mid \boldsymbol{x})$ accounts for the change of density when moving the path $\boldsymbol{x}$ with the Boltzmann generator into latent space and $p(\tilde{\boldsymbol{x}} \mid \tilde{\boldsymbol{z}})$ arises from moving the new latent path back into configuration space.

To compute the remaining probabilities that account for the transformations from and to the latent space, we have to consider that the Boltzmann generator processes all frames independently. We can thus write $p(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{i=1}^{\ell} p(z_i \mid x_i)$ multiplying all the terms. In a similar fashion, $p(\tilde{\boldsymbol{z}} \mid \tilde{\boldsymbol{x}})$ can be rewritten by the individual frames of the proposed path as well. Stating the ratio of these proposal probabilities—as needed for the acceptance criterion of the Metropolis-Hastings algorithm stated in Equation 2.13—can be written as

$$\frac{q(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{q(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})} = \frac{q_Z(\boldsymbol{z} \mid \tilde{\boldsymbol{z}})}{q_Z(\tilde{\boldsymbol{z}} \mid \boldsymbol{z})} \cdot \prod_{i=1}^{\ell} \frac{p(\tilde{z}_i \mid \tilde{x}_i) p(x_i \mid z_i)}{p(z_i \mid x_i) p(\tilde{x}_i \mid \tilde{z}_i)}. \tag{3.2}$$

Each term in the product can first be simplified by rewriting the conditional probability with the joint probability

$$\frac{p(\tilde{z} \mid \tilde{x}) p(x \mid z)}{p(z \mid x) p(\tilde{x} \mid \tilde{z})} = \frac{\frac{p(\tilde{x},\tilde{z})}{p(\tilde{x})} \frac{p(x,z)}{p(z)}}{\frac{p(x,z)}{p(x)} \frac{p(\tilde{x},\tilde{z})}{p(\tilde{z})}} = \frac{p(x)p(\tilde{z})}{p(\tilde{x})p(z)}, \tag{3.3}$$

where we write x, z for an individual frame x_i, z_i to simplify the notation. This can further be simplified using the change of variables formula [Bogachev, 2007, pp. 194–197], which allows us to reformulate $p(x) = p(z) \cdot |\det J(T(z))|^{-1}$, and hence

$$\frac{p(x)p(\tilde{z})}{p(\tilde{x})p(z)} = \frac{p(x)p(\tilde{z})}{p(\tilde{x})p(z)} = \frac{p(z)|\det J(T(z))|^{-1} p(\tilde{z})}{p(\tilde{z})|\det J(T(\tilde{z}))|^{-1} p(z)} = \frac{|\det J(T(\tilde{z}))|}{|\det J(T(z))|}. \tag{3.4}$$

With this, we can write the acceptance probability of a transition path with a kernel operating in latent space as

$$\alpha = \min \left\{ 1, \frac{p_{AB}(\tilde{\boldsymbol{x}})}{p_{AB}(\boldsymbol{x})} \cdot \frac{q_Z(\boldsymbol{z} \mid \tilde{\boldsymbol{z}})}{q_Z(\tilde{\boldsymbol{z}} \mid \boldsymbol{z})} \cdot \prod_{i=1}^{\ell} \frac{|\det J(T(\tilde{z}_i))|}{|\det J(T(z_i))|} \right\}. \tag{3.5}$$

With this acceptance probability, we can construct various latent proposal kernels, as long as we can compute q_Z. The complete procedure of how this can be used to sample a representative path ensemble is laid out in Algorithm 1.

With this derivation, we also demonstrate that we are not limited to normalizing flows. The only requirements are that we can compute frames by their latent representation (and vice versa), and the Jacobian of the transformation.

All that is remaining now are a concrete way to compute the probability of a transition path $p_{AB}(\boldsymbol{x})$, and choices for proposal kernels K and their respective proposal probabilities $q_Z(\tilde{\boldsymbol{z}} \mid \boldsymbol{z})$, which we will tackle next.

3.2.3. Calculating the Path Probability

Until now, when we discussed the probability for calculating a path in Equation 2.10 and Equation 2.11, we have relied on a setting where we know the positions and velocities of each atom at every transition frame. When sampling transition paths with molecular dynamics, this is known. However, in our approach, we only sample the atom positions for each frame of the transition path. For access to the velocities, the Boltzmann generator would need to be trained to sample transition paths as well as velocities. Since this poses a challenging problem, we will demonstrate how to compute the probability of a path $p_{AB}(\boldsymbol{x})$ without access to the velocities.

First, we will look at the probability of any trajectory defined by the sequence of atom positions $\boldsymbol{x} = (x_1, x_2, \dots, x_\ell)$, and velocities $\boldsymbol{v} = (v_1, v_2, \dots, v_\ell)$

$$\begin{aligned} p_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{v}) &= \rho(x_1, v_1) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i) \\ &= p(x_1) \cdot \mathcal{N}(v_1 \mid 0, \boldsymbol{M}^{-1} k_B T) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i), \end{aligned} \tag{3.6}$$

where we assume from now on that the velocities are distributed according to a normal that depends on the inverse mass $\boldsymbol{M}$, temperature T, and the Boltzmann constant k_B [Castellan, 1983, Sec. 4.6]. To calculate the probability without velocities, we will marginalize over all possible velocities $v_1, v_2, \dots, v_\ell$ of each individual frame and atom

$$p_{\mathcal{T}}(\boldsymbol{x}) = \int_{v_1} \int_{v_2} \cdots \int_{v_\ell} p_{\mathcal{T}}(\boldsymbol{x}, (v_1, v_2, \dots, v_\ell)) dv_1 dv_2 \dots dv_\ell. \tag{3.7}$$

With an iterative path integration scheme, such as the leap-frog Verlet integrator we discussed to solve Langevin dynamics in Section 2.1.1, with access to the initial velocity v_1, and all intermediate atom positions $(x_1, x_2, \dots, x_\ell)$, we can solve for all remaining velocities $(v_2, v_3, \dots, v_\ell)$ by

$$\boldsymbol{v}_{i+1} = \frac{x_{i+1} - x_i}{\Delta t}. \tag{3.8}$$

Hence, when fixing v_1 all subsequent velocities are deterministic and the probability for observing these velocities is $p(v_2, v_3, \dots, v_\ell \mid v_1, x_1, x_2, \dots, x_\ell) = 1$ iff we chose the correct velocities. In all other cases, this probability will be 0. For calculating the probability of a trajectory, this means that we can simplify Equation 3.7 even further by fixing the initial velocity v_1 and solving for the remaining velocities to write

$$\begin{aligned} p_{\mathcal{T}}(\boldsymbol{x}) &= \int_{v_1} \rho(x_1, v_1) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i) dv_1 \\ &= \int_{v_1} p(x_1) \cdot \mathcal{N}(v_1 \mid 0, \boldsymbol{M}^{-1} k_B T) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i) dv_1 \\ &= \mathbb{E}_{v_1 \sim \mathcal{N}(0, \boldsymbol{M}^{-1} k_B T)} \left[p(x_1) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i) \right]. \end{aligned} \tag{3.9}$$

In the last equality, we relied on the definition of the expected value.

Similarly to before, we are only interested in reactive pathways and hence can compute the probability of a transition path connecting the states A and B with

$$p_{AB}(\boldsymbol{x}) \propto \mathbb{1}_A(x_1) \cdot \mathbb{E}_{v_1 \sim \mathcal{N}(0, \boldsymbol{M}^{-1} k_B T)} \left[p(x_1) \cdot \prod_{i=1}^{\ell-1} p(x_{i+1}, v_{i+1} \mid x_i, v_i) \right] \cdot \mathbb{1}_B(x_\ell). \tag{3.10}$$

In practice, we can approximate this expectation with a finite number of samples, giving us a straightforward way to compute the probability of any (reactive) trajectory by only looking at the atom positions. To compute $p(x_1)$ we use the Boltzmann distribution as defined in Equation 2.4, and we will discuss next how to compute the step probability.

Step Probability

The rules for how the molecular system evolves over time are governed by the underlying system and its settings, such as the force field, and the concrete choice of molecular dynamics. In this thesis, we will always assume Langevin dynamics with a Verlet integration which tells us how the atoms evolve over time based on the forces that act on the atoms. Looking back at Equation 2.3 where we stated the iterative scheme used to compute the values of the next timestep, we can see that a step consists of a deterministic movement and a random displacement based on the current temperature of the system. The atom positions x_{i+1} are thus distributed according to

$$x_{i+1} \sim \mathcal{N}\left(x_i + \Delta t \left(\alpha v_i + \frac{1}{\gamma}(1-\alpha) \nabla U(x_i) \boldsymbol{M}^{-1} \right), (\Delta t)^2 k_B T (1-\alpha^2) \boldsymbol{M}^{-1} \right), \tag{3.11}$$

which can be used to evaluate the density $p(x_{i+1} \mid x_i, v_i)$ without requiring v_{i+1}.

Performance

One of the main goals when designing this framework was to improve the sampling time of transition paths by removing the need for expensive molecular dynamics. What we described over the past few sections to compute the probability of a trajectory, and more importantly to evaluate the probability to move from one frame x_i to x_{i+1} essentially contains molecular dynamic simulations. However, in our approach, all steps can be performed in parallel, especially the expensive computation of the energy function $U(x_i)$, and its derivative ∇U. This has the potential to significantly speed up the algorithm for longer paths compared to path sampling with molecular dynamics, because there all steps need to be performed sequentially.

3.2.4. Algorithm

Our approach is summarized in detail in Algorithm 1. It is important to highlight that this algorithm needs the inverse transform of the Boltzmann generator T^{-1} only once. This is beneficial since some flow architectures, such as inverse autoregressive normalizing flows [Kingma et al., 2016], can only be efficiently evaluated in one direction. With this approach, a wide variety of different architectures is thus suitable.

Algorithm 1: Fixed-length latent space transition path sampling.

```
Input: Initial path $\boldsymbol{x}^{(0)}$ with $\ell$ frames, a trained Boltzmann generator consisting
       of the map $T$ and its inverse $T^{-1}$, the number of paths to sample $N$, and a
       latent proposal kernel $K$ with proposal probability $q_Z(\cdot \mid \cdot)$.
Output: MCMC samples following target distribution $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right\}$.
1  Calculate latent space representation of initial path
       $$\boldsymbol{z}^{(0)} = \left\{T^{-1}\left(x_1^{(0)}\right), \ldots, T^{-1}\left(x_\ell^{(0)}\right)\right\}.$$
2  for $i \leftarrow 1 \ldots N$ do
3      repeat
4          Propose new path in latent space $\tilde{\boldsymbol{z}} = K\left(\boldsymbol{z}^{(i-1)}\right)$.
5          Compute the proposed path in configuration space
               $$\tilde{\boldsymbol{x}} = \left\{T\left(\tilde{z}_1\right), \ldots, T\left(\tilde{z}_\ell\right)\right\}.$$
6          Compute acceptance probability
               $$\alpha = \min\left\{1, \frac{p_{AB}\left(\tilde{\boldsymbol{x}}\right)}{p_{AB}\left(\boldsymbol{x}^{(i-1)}\right)} \cdot \frac{q_Z\left(\boldsymbol{z}^{(i-1)} \mid \tilde{\boldsymbol{z}}\right)}{q_Z\left(\tilde{\boldsymbol{z}} \mid \boldsymbol{z}^{(i-1)}\right)} \cdot \prod_{j=1}^{\ell} \frac{|\det J\left(T\left(\tilde{z}_j\right)\right)|}{|\det J\left(T(z_j^{(i-1)})\right)|}\right\}.$$
7          Draw a uniformly distributed random number $u \sim \mathcal{U}_{[0,1]}$.
8      until proposed path $\tilde{\boldsymbol{x}}$ is reactive and $u \leq \alpha$;
9      Accept proposed path $\boldsymbol{z}^{(i)} = \tilde{\boldsymbol{z}}, \boldsymbol{x}^{(i)} = \tilde{\boldsymbol{x}}$.
10 end
```

3.3. Latent Proposal Kernels

Now with the framework and setup defined, all that is left to do is to explore different choices for latent space path proposal kernels K. In this section, we will look at three specific kernels that propose paths without the need for any simulation.

3.3.1. Gaussian Proposal

We will begin by discussing a simple latent space kernel, the *Gaussian Proposal Kernel*. The idea is that we simply add iid. Gaussian noise to each frame in latent space such that

$$\tilde{z} = K_{\mathcal{N}}(z) = (z_1 + \varepsilon_1, z_2 + \varepsilon_2, \ldots, z_\ell + \varepsilon_\ell)\,, \tag{3.12}$$

where $z_i \sim \mathcal{N}(0, \Sigma)$. If the noise in every dimension of the latent space is identical, $\Sigma = \sigma^2 \mathbb{1}^D$, where D is the dimension of the latent space and σ the standard deviation. This idea is illustrated in Figure 3.3.

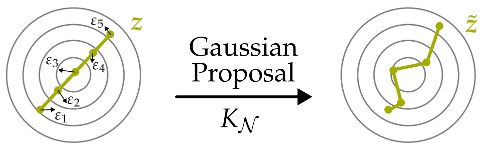

Figure 3.3.: Visualization of the Gaussian proposal kernel that adds independent Gaussian noise to each frame in latent space.

Paths proposed by this kernel, will only in very rare instances move all the frames of the path in coherent and meaningful way. Thus, most paths proposed by this kernel will be rejected. However, since this operation can be performed in parallel and is in itself highly efficient, it is still a useful kernel and—with a large enough σ—can explore the whole latent space.

Selecting the variance of the noise we add in latent space exhibits a trade-off similar to what we discussed for shooting moves. A large variance will produce diverse paths, but they will likely not be physical and thus rejected. When only adding a small noise, most paths will be accepted because they are very similar; however, it might be that the variance is too small to explore any other transition channel. In practice, one has to test out different variance scales depending on the system and the trained Boltzmann generator.

When examining the acceptance criterion of latent space approaches stated in Equation 3.5), we can see that we have to correct each frame with a factor depending on the change in density between the latent and product space. Here, we propose to use a dynamic way to select the variance of the Gaussian transition kernel, so that this correction term cancels out. When choosing a constant variance, the term $q_Z(\cdot \mid \cdot)$ will be symmetric and the ratio cancels out. However, when we choose a variance for each frame, we will get

$$\frac{q_Z\left(z_i \mid \tilde{z}_i\right)}{q_Z\left(\tilde{z}_i \mid z_i\right)} = \frac{\mathcal{N}\left(z_i \mid \tilde{z}_i, \tilde{\sigma}_i\right)}{\mathcal{N}\left(\tilde{z}_i \mid z_i, \sigma_i\right)} = \frac{\sigma_i}{\tilde{\sigma}_i} \cdot \frac{\exp\left(-\frac{1}{2\tilde{\sigma}_i^2}\left(z_i - \tilde{z}_i\right)^2\right)}{\exp\left(-\frac{1}{2\sigma_i^2}\left(\tilde{z}_i - z_i\right)^2\right)}, \tag{3.13}$$

when assuming a 1D Gaussian without loss of generality, and a frame of the transition path z_i and the proposal $\tilde{z}_i$. When we choose $\sigma_i = |\det J(T(z_i))|$, and $\tilde{\sigma}_i = |\det J(T(\tilde{z}_i))|$, the acceptance criterion simplifies to

$$\alpha = \min\left\{1, \frac{p_{AB}(\tilde{\boldsymbol{x}})}{p_{AB}(\boldsymbol{x})} \cdot \prod_{i=1}^{\ell} \frac{\exp\left(-\frac{1}{2|\det J(T(\tilde{z}_i))|^2}\left(z_i - \tilde{z}_i\right)^2\right)}{\exp\left(-\frac{1}{2|\det J(T(z_i))|^2}\left(\tilde{z}_i - z_i\right)^2\right)}\right\}. \tag{3.14}$$

With this way, we choose the variance based on the underlying properties of the latent space itself. In our experiments, this automatic selection of the noise was difficult to achieve because the determined variances were too small for our test system to produce diverse paths. Still, it might be convenient for some well-conditioned systems.

3.3.2. Latent Hamiltonian Dynamics

To move the frames more coherently and not completely independent from each other, we propose a more sophisticated kernel. Instead of performing molecular dynamics simulation in the product space, we will move each frame individually in the direction of the gradient in the latent space and add a random momentum. To achieve this, we extend the ideas presented by Hoffman et al. [2019] to perform Hamiltonian MCMC [Duane et al., 1987, Neal, 2011] in the latent space. In contrast to their work, we use the latent space of a Boltzmann generator and extend their approach from MCMC sampling to transition path sampling.

For this, our *Hamiltonian proposal kernel* K_∇ simulates independent Markov chains in parallel for each latent frame z_i, as illustrated in Figure 3.4. We perform L leapfrog steps with a step size of ϵ, by first initializing the approach

$$\begin{aligned} m_i^{(0)} &= m_i + (\epsilon/2)\nabla p_Z(z_i), \\ z_i^{(0)} &= z_i, \end{aligned} \tag{3.15}$$

where m_i is sampled from a standard normal $\mathcal{N}(0, \mathbb{1}^D)$. Then we continue for L leapfrog iterations to compute $z_i^{(1)}, \dots, z_i^{(L)}$ and $m_i^{(1)}, \dots, m_i^{(L)}$ for $t \in 0, 1, \dots, L-1$

$$\begin{aligned} z_i^{(t+1)} &= z_i^{(t)} + \epsilon m_i^{(t)}, \\ m_i^{(t+1)} &= m_i^{(t)} + \epsilon \nabla p_Z(z_i^{(t+1)}). \end{aligned} \tag{3.16}$$

The final values of the procedure are computed by

$$\begin{aligned} z_i^* &= z_i^{(L)}, \\ m_i^* &= m_i^{(L)} - (\epsilon/2) \nabla p_Z(z_i^{(L)}). \end{aligned} \tag{3.17}$$

Because we can efficiently evaluate the gradient of the latent space ∇p_Z, this approach can be performed more efficiently than a simulation in product space where ∇U would be time-consuming to evaluate.

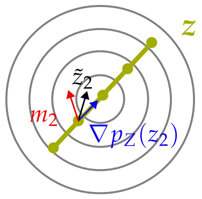

Figure 3.4.: An intuitive explanation of the Hamiltonian proposal kernel K_∇. For each frame, we evaluate the gradient of the latent space ∇p_Z. Together with an initially random momentum m, each frame is simulated to produce a new trajectory $\tilde{z}$.

However, K_∇ is not symmetric anymore and the ratio of the forward and backward proposal can be computed as

$$\frac{q_Z(z \mid \tilde{z})}{q_Z(\tilde{z} \mid z)} = \prod_{i=1}^{\ell} \frac{p_Z(m_i^*)}{p_Z(m_i)} = \prod_{i=1}^{\ell} \frac{\mathcal{N}(m_i^* \mid 0, \mathbb{1})}{\mathcal{N}(m_i \mid 0, \mathbb{1})}. \tag{3.18}$$

This ratio is a straightforward extension from Hoffman et al. [2019] since we simulate each frame independently.

3.3.3. Gaussian Process-based Proposals

All previous methods did not include any learning after the latent space has been constructed. Since we do not impose any restrictions on the latent space other than the distribution of samples, a kernel that learns the underlying properties of the proposed paths might be beneficial. In this section, we propose different kernels that are based on fitting a *Gaussian process* (GP) to the previously sampled transition paths so that we can sample new trajectories.

In a Gaussian process, a function f applied to each point $t \in \mathbb{R}$ is a random variable that is distributed according to a Gaussian distribution $f(t) \sim \mathcal{N}(\mu(t), k(t, t'))$, where $\mu(t)$ specifies the mean at that point, and k is a kernel applied to t and observations t'. Note that this kernel k must not be confused with the proposal kernel K. A proposal kernel K samples a new path, and the kernel k is a positive semi-definite function[1] that compares the similarity of points.

For this work, it is sufficient to know that a Gaussian process is fully characterized by the kernel k, of which the parameters can be optimized to fit the observed data. Contrary to other models, this kernel is based on the similarity of points t, instead of the observations $f(t)$. The idea is to predict a similar variance for similar points, where our kernel defines similarity. Once fit, the Gaussian process $\mathcal{GP}$ can be used to estimate the mean and the variance at each point t. The variance will be lower at points that have been observed, and higher at sparse parts of the distribution. For a more in-depth view, we recommend the work of Rasmussen and Williams [2006], which has served as a foundation for this summary.

Setup for a Gaussian Process on Latent Paths

Since the frames in our underlying latent space are normally distributed, a Gaussian process—with the correct choice of a kernel—is a suitable way to model the underlying distribution of frames. We propose to model the Gaussian process $\mathcal{GP}$ such that $f : \mathbb{R} \mapsto \mathbb{R}^D$ maps the time $t \in [1, \ell]$ where a frame was sampled. In other words, t represents the index of a frame of the transition path in latent space $\boldsymbol{z} = (z_1, z_2, \ldots, z_\ell)$, which means that generally it holds that $t \in \mathbb{N}$. The Gaussian process is then fitted on a set of the s paths in latent space $\{\boldsymbol{z}^{(i)}\}_{i \in \{1,2,\ldots,s\}}$, as seen in Figure 3.5.

For this kernel, we differentiate between a *fixed* and an *adaptive* variant. In the fixed scenario, $\mathcal{GP}$ is fitted once to a set of latent space paths $\{\boldsymbol{z}^{(i)}\}_{i \in \{1,2,\ldots,s\}}$. This set of paths could for example be a representative set of transition paths produced by a slower method, where the aim is to speed up sampling. For the fixed kernel, we instead propose to sample arbitrary points $a \in A$, $b \in B$ from the two meta-stable states and linearly interpolate between them in latent space. While the trajectories will very likely not be physical enough, they serve as a diverse set of possible transition paths between these states, and the MCMC approach ensures that the distribution of paths is correct. The performance of this kernel is highly influenced by the quality of the initial paths.

[1] A function k is positive semi-definite, and thus a kernel, iff the matrix of all possible pairs of points $M = k(t, t')$ is positive semi-definite [Steinwart and Christmann, 2008]. This matrix is called Gram matrix or kernel matrix [Schölkopf et al., 2001].

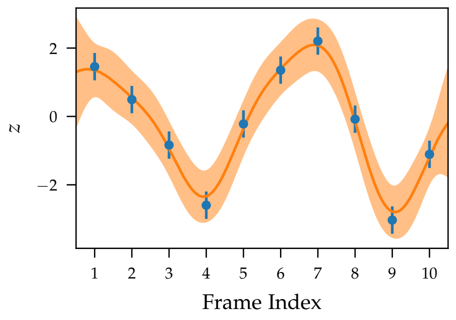

Figure 3.5.: This figure shows a Gaussian process that was fitted on 10 frames. In our case, the x-axis will always be the index of the frame number in the current trajectory. The y-axis illustrates the observations and are the higher-dimensional latent coordinates. A Gaussian process captures the mean, and the variance—here the 95% confidence interval symmetric around the mean is illustrated.

In the adaptive Gaussian process, the set $\{z^{(i)}\}_{i\in\{1,2,\dots,s\}}$ will only contain the initial path $z^{(0)}$ in the beginning, and each accepted path will be added to this set and a new Gaussian process fit. This, however, means that a different Gaussian process and thus a different kernel is used for each step. For this to be correct, the kernel must fulfill additional requirements for correct path sampling: The proposal kernel must satisfy vanishing adaption [Andrieu and Thoms, 2008] and the kernel needs to converge. Intuitively, this holds for our Gaussian process as the set $\{z^{(i)}\}_{i\in\{1,2,\dots,s\}}$ will converge to the true path ensemble and, in the limit, will not change anymore.

This retraining poses a second problem that is of a more technical nature: we need to solve a minimization problem to determine suitable parameters for the $\mathcal{GP}$. Especially for high-dimensional problems, such as our transition paths, this can take time. However, to overcome this challenge we use the parameters of the $\mathcal{GP}_{\{1,2,\dots,s\}}$ as the basis when searching for the parameters of $\mathcal{GP}_{\{1,2,\dots,s,s+1\}}$. In most cases, a single path will not change much of the underlying approximation, and the new Gaussian process will have a similar optimal solution. This procedure significantly speeds up the MCMC algorithm and is necessary for it to work on a large number of paths.

Sampling Latent Paths with a Gaussian Process

A trivial way to sample paths given a Gaussian process $\mathcal{GP}$ is to estimate the mean and the variance at the points $t \in \{1, 2, \dots, \ell\}$ for a transition path with length ℓ and frames $(x_1, x_2, \dots, x_\ell)$. With these values, we can produce a new transition path by sampling $\tilde{z}_t \sim \mathcal{N}(\mu(t), k(t, t'))$ for each frame t. This procedure does not use the current path z in any way, because a path is only added after it has been accepted. We refer to this phrasing of the sampling procedure as *unconditional*, since the kernel that samples new paths is not conditioned on the current path $\boldsymbol{z}$ at all. Although we believe that this is an interesting setting as well, since transition paths can be sampled without a given path, it is vastly different from the other kernels.

To overcome this issue, we alternatively propose a *conditional* sampling where we use the current path as the mean. More formally, the sampling is similar as in the unconditional setting with the only difference that we draw samples from the distribution $\tilde{z}_t \sim \mathcal{N}(z_t, k(t, t'))$. In a more technical setting, we implemented this such that we still sample from the Gaussian process unconditionally but subtract the mean $\mu(t)$ and add z_t. Similarly, as before, we then transport the latent space frames into the product space with the trained Boltzmann generator to evaluate the acceptance criterion.

One interesting notion we explored when sampling new paths is that we defined the Gaussian process $\mathcal{GP}$ on the continuous space $\mathbb{R}$, but only evaluated it at the fixed frame indices in $\mathbb{N}$. However, due to the definition of Gaussian processes, the least variance will be at the positions t, where frames are observed. To increase the diversity of sampled transition paths, we can use different sampling positions. For example, for a *uniform* Gaussian process we draw $1, 2, \dots, \ell$ different points t from $\mathcal{U}_{[0.5, l+0.5]}$ such that the positions where we evaluate the paths is random. With this, the individual frames can shift further apart or closer together, as with transition paths in product space. Similar ideas can be constructed by defining ℓ normal distributions centered at $1, 2, \dots, \ell$ and drawing one sample from each to get an ordered list of points t to sample from.

Choice of Kernel for the Gaussian Process

As discussed previously, the underlying choice of kernel k and its parameters $\boldsymbol{\theta}$ are the main characteristics that define a Gaussian process. The parameters $\boldsymbol{\theta}$ are fitted such that the Gaussian process best matches the observations, and depending on the choice k, a different class of functions defines the transition between points t.

In our experiments, we found out that an adaption of the RBF-Kernel worked best for a variety of different experiments. Since this kernel uses the distance between two points, we believe this is a suitable choice for most transition paths, as the variance and mean of the resulting GP smoothly transition between individual frames. Our adaptation contains an additional white kernel that can capture variance of the individual points. The kernel k can be formulated as

$$k(x, x') = c \cdot \exp\left(-\frac{\|x - x'\|_2^2}{2l^2}\right) + n \cdot \mathbb{1}_{x \neq x'}, \tag{3.19}$$

with learnable parameters l, c, n. We believe it is worth to explore different kernels depending on the underlying properties of the system and latent space.

There has also been research conducted that uses neural networks to parameterize kernels, such as by Wilson et al. [2016a] and Wilson et al. [2016b]. This could be an interesting addition to our framework to improve the empirical results and should be explored in future research.

3.3.4. Theoretical Guarantees

The requirements for a proposal kernel to be suitable for MCMC are very lenient and have been discussed in Section 2.3.4. Mainly, the assumption is that each possible transition path can occur in a finite amount of time. Since we make use of relatively simple operations, that mostly rely on Gaussian distributions this is intuitively fulfilled for all our proposal kernels.

However, what takes a bit more care is the computation of the proposal probability $q_Z(\cdot \mid \cdot)$ of the concrete proposals. For both, the Gaussian proposal kernel and the Hamiltonian proposal kernel, we have already derived these probabilities in the respective sections. We will now show the acceptance probability for the conditional Gaussian Process transition kernel as well. For simplicity, we assume a 1D Gaussian for an individual frame $z = z_i$ and the proposal $\tilde{z} = \tilde{z}_i$

$$\frac{q_Z(z \mid \tilde{z})}{q_Z(\tilde{z} \mid z)} = \frac{\mathcal{N}(z|\mu = \tilde{z}, \sigma = GP(t(\tilde{z})))}{\mathcal{N}(\tilde{z}|\mu = z, \sigma = GP(t(z)))} = \frac{\frac{1}{\sigma\sqrt{2\pi}} \cdot \exp(-\frac{1}{2}(\frac{(z-\tilde{z})^2}{\sigma}))}{\frac{1}{\sigma\sqrt{2\pi}} \cdot \exp(-\frac{1}{2}(\frac{(\tilde{z}-z)^2}{\sigma}))} = 1 \tag{3.20}$$

Which means that the acceptance probability simplifies to

$$\alpha = \min\left\{1, \frac{p_{AB}\left(\tilde{\boldsymbol{x}}\right)}{p_{AB}\left(\boldsymbol{x}\right)} \cdot \prod_{j=1}^{\ell} \frac{\det J\left(T\left(\tilde{z}_j\right)\right)}{\det J\left(T(z_j)\right)}\right\}. \tag{3.21}$$

The same holds for all other Gaussian process-based proposal kernels as well, since they only rely on symmetric operations.

3.4. Exploring Approximations and Improvements

So far, we have only explored approaches that are mathematically correct, regardless of their performance and practicality. For instance, while we can parallelize the computation of the path probability, computing the derivative can still pose a problem, especially for more sophisticated systems. In this section of the thesis, we will explore approaches that might improve the empirical results of our framework but can only be included when sacrificing the theoretical guarantees. While these ideas can work well in some experimental settings, they will likely be lacking in different scenarios. Hence, these approximations will not be explored in-depth in the experimental evaluation. Still, we find the ideas interesting enough to introduce them in this section.

3.4.1. Independent Derivative-Free Path Probability

When computing the probability of a transition path p_{AB}, we need access to the underlying forces ∇U of the system. Computing the energy U itself can already be computationally challenging, and by needing access to the gradient as well, we essentially perform molecular dynamics simulations. As discussed previously, the big difference is that here these molecular simulations can be performed in parallel because we only have to compute one step at a time. However, speeding up this part of the algorithm can yield significant performance gains for larger systems.

We thus propose to use the product of the so-called *Boltzmann factors* to approximate the probability of the path such that

$$\frac{p_{AB}(\tilde{\boldsymbol{x}})}{p_{AB}(\boldsymbol{x})} \approx \prod_{i=1}^{\ell} \frac{p(\tilde{x}_i)}{p(x_i)} = \prod_{i=1}^{\ell} \exp\left(\frac{U(x_i) - U(\tilde{x}_i)}{k_B T}\right), \tag{3.22}$$

with the energy U and the Boltzmann distribution p at temperature T. This ratio describes the relative probability between the individual frames and can be computed without the normalizing constant of the Boltzmann distribution because it cancels out. When using this formula to approximate the probability of a trajectory, no notion of the time and distance between the frames is included. In a practical setting, this means that a transition path where the frames only occur in the low-energy meta-stable states A and B has high probability. With this path, the transition between the states would be instantaneously. Although with no notion of time, this path would have the highest probability, we are interested in observing the actual transition between the states. Hence, there is no reason to believe that this approximation in itself accurately allows us to accurately compute the probability of a path.

3.4.2. Making Paths Equidistant

One way to make this approximation more useful is to prevent the problem of frames converging to local minima. To achieve this, we propose to make the paths space-equidistant in the product space. For this, we can increase the number of frames of a transition path ℓ and then select a subset of the frames in product space such that the paths are (almost) space-equidistant. The other approach, and the one we have opted for in our implementation, is to linearly interpolate between frames in latent space to produce intermediate frames. We then select again the points which make the path equidistant. This allows us to use the same number of frames in the latent space as for the final transition path in product space. The drawback of this, however, is that by linearly interpolating between frames in latent space, we need to assume that the kernels operate in a similar fashion. A Gaussian process-based transition kernel, for example, may smoothly interpolate in latent space to produce paths, meaning that it would be more beneficial—but also more computationally expensive—to operate with more points in latent space.

The concrete method to equidistance our transition paths depends on the underlying system that is used. For molecular systems, we recommend using the root-mean-square deviation (RMSD) as the distance metric such that

$$\mathrm{RMSD}\left(x, x'\right) = \sqrt{\frac{1}{N}\sum_{a=1}^{N} \|x_a - x'_a\|_2^2}, \tag{3.23}$$

for N atoms in the molecule, where we exclude all hydrogen atoms. $\|q\|_2^2$ is the squared Euclidian distance and defined as $\sum_{j=1}^{3} q_j^2$ for atoms in 3D space. Note that for this formulation we have used a slightly different notation than in the rest of the thesis. x and x' are two individual frames of the transition path that are next to each other such as x_i, x_{i+1}. In this formulation, x_a and x'_a represent the position of the a-th atom, not the index of the frame in the path.

However, with this formulation, the frames still do not have any notion of time anymore. In some cases, we might not be interested in the transition time itself but rather the channels. We believe that especially in these settings our approximation of the path probability could be useful, when also ensuring that the frames are space-equidistant in product space.

3.4.3. Path Relaxation

Transition paths that simple proposal kernels have produced might be unrealistic and not represent meaningful paths. Non-physical paths are less likely to get accepted in the Metropolis-Hastings algorithm because they will have a low probability. Of course, one solution is to rely on more sophisticated kernels that can propose realistic paths. However, with sufficiently complex transition kernels, there is no performance to be gained over classical two-way shooting approaches.

Instead, after proposing a latent path, we can perform a few steps of molecular dynamics simulations on each frame in product space, with a small step size such that the frames converge to more favorable positions. This idea is inspired by the string method [E et al., 2005] that can be used to find minimum energy paths. Similarly to the problem we faced when approximating the probability of a path with the Boltzmann factor, the frames can again converge to local minima if the step size is too large, or the simulation time is too long. For this, we can again make the path space-equidistant in product space, or we can rely on a more sophisticated approach. In nudged elastic band (NEB) [Henkelman et al., 2000, Henkelman and Jónsson, 2000], the idea is to simulate the individual frames while still trying to pull the frames together. This results in a more realistic path and can be used to post-process sampled trajectories. Once we perform these small molecular dynamics simulations, computing the path proposal probability $q_Z(\cdot \mid \cdot)$ is not straightforward anymore. Thus, it can only be done if approximations suit the task at hand. We believe that future work in the direction of nudged elastic band could make for a great relaxation for simulation-free proposal kernels.

3.5. Transforming the Latent Space of a Boltzmann Generator

For most latent transition kernels that we have introduced, we assumed that the molecules are distributed according to a normal distribution in the latent space. However, Noé et al. [2019] advocate to use an internal coordinate representation for molecular systems, where the molecule is not described by the atom positions in 3D space but by features such as bond angles and lengths. They motivate their reasoning because of translational and rotational invariance and because these internal coordinates are already distributed similarly to an easy distribution. In this representation, some of the variables are periodic, such as torsional angles that are within $[0, 2\pi]$ which has to be handled with care. Previous works [Midgley et al., 2023b] have hence often relied on a mixture between a Gaussian and a uniform distribution as the base space of the Boltzmann generator. This Gaussian-uniform mixture is not directly compatible with our proposal kernels. However, for many distributions there is an easy way to convert between these.

In this section, we want to show how one can transform between a random variable that is uniformly distributed and a standard normal. For this, we use the result that the inverse cumulative distribution function of any continuous random variable is uniformly distributed on $[0, 1]$, as for example presented by Embrechts and Hofert [2013]. For the standard normal, one usually writes the CDF as $\Phi(z) = p(Z \leq z)$, where $Z = \mathcal{N}(0, 1)$. If now we have a variable $z \sim Z$, $x = \Phi^{-1}(z)$ will be distributed according to $\mathcal{U}[0, 1]$. In a similar fashion we can transport from a uniform to a Gaussian by using the non-inverted Φ. This allows us to train a Boltzmann generator with a mixture of Gaussian and uniform distributions, and after training we can pre-process the variables to convert the uniformly distributed values from and to a Gaussian.

4. Learning Optimal Shooting Positions

In this chapter of the thesis, we will investigate a completely different approach to transition path sampling that builds on the ideas from Jung et al. [2023]. It is based on improving two-way shooting TPS by learning the optimal frame to shoot from. This has the potential to significantly improve the performance of TPS, as it reduces the number of simulations that get discarded because they do not form a proper transition path. In Section 4.1, we will explain the main training routine and approach introduced by Jung et al. [2023]. We will then introduce two new architectures, one of which is based on transformers [Vaswani et al., 2017], in Section 4.2. We will evaluate the approach in Chapter 5.

4.1. Methodology

The ideas presented by Jung et al. [2023] revolve around the concept of dynamically learning the committor function for a specific energy landscape. The committor $p_B(x)$ describes for each configuration $x \in X$ the probability that a molecular dynamics simulation initialized from x will reach B as the first state. As by its definition, the committor function depends on the concrete molecule, temperature, and forces. In the case of transition path sampling, we have two states $A \leftrightarrow B$ and each simulation is guaranteed to either reach state A or B first, such that $p_A(x) + p_B(x) = 1$.

The committor p_B will change along a transition trajectory: configurations that are close or maybe even within state A will almost surely reach state A first and thus $p_B \approx 0$. Along the trajectory, the probability will change until eventually it will be close to 1 when reaching state B. Configurations where $p_B \approx 0.5$ are on the other hand unstable and a small change in the velocity can decide whether the trajectory will converge to A or B. This configuration will likely be a saddle point of the energy function.

If we have access to the committor function of the underlying system, we can evaluate p_B for every point $x \in \mathcal{T}$ on our transition path. The point where the committor is closest to 0.5 will then be the best point to shoot from, as a transition can end up in both states. Note that a trajectory will only reach either state A or B first. If we say that it ends up in both states we refer to the case where the simulation forward in time ends up in B and backwards in time in A or vice versa.

However, the committor function is not known and hence Jung et al. [2023] proposed to use a neural network to approximate this committor function. In the next section, we will discuss how such a neural network can be trained.

4.1.1. Training

In this thesis, we focus on transition path sampling with only two states and can thus interpret the committor similarly to a Bernoulli experiment. Whenever we choose a shooting point and simulate the molecule's movement, it can be seen as a coin toss with a certain probability that it actually reaches the desired state. The aim for the neural network is to model the function $p_B(\cdot \mid \boldsymbol{\theta})$ with the weights $\boldsymbol{\theta}$. To ensure that the output of the neural network f_θ describes a proper density function, we map it with the sigmoid function such that $p_B(\cdot \mid \theta) = 1/(1+\exp(-f(\cdot \mid \boldsymbol{\theta})))$.

As for the concrete training procedure, Jung et al. [2023] propose to maximize the likelihood of all performed shootings. Meaning that they define the problem for one state (either A or B) and then store a binary value that indicates whether the target state was reached. With this, the neural network can be trained by minimizing the loss

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{k} \log\left(1 + \exp(s_i) f(x_i \mid \boldsymbol{\theta})\right), \tag{4.1}$$

where we have performed k shooting moves at conformations $x_1, x_2, \ldots, x_k \in X$. s_i is either -1 if the trajectory enters B first and $+1$ if it enters A first.

For this training procedure, we need access to whether the shooting from specific conformations was successful. As this is not possible in the beginning, the approach works with a randomly initialized neural network, that predicts shooting points. If the predictions are too far off, the network will be trained for a few epochs on all the available shooting results until now. This makes it so that the neural network is only trained when needed, which avoids the problem of overfitting.

4.1.2. Sampling

Although we now have specific a way to approximate the committor function, we still have to define a procedure that can be used to select the frames to choose from. If we only relied on picking the point x, where the committor probability is closest to 0.5, then our approach might get stuck when the prediction of the neural network is incorrect (or unlikely). To alleviate this problem, the points are sampled according to a Lorentzian distribution such that

$$p_{sel}(x \mid \mathcal{T}, \boldsymbol{\theta}) = 1 \;/ \sum_{x' \in \mathcal{T}} \frac{f(x \mid \boldsymbol{\theta})^2 + \gamma^2}{f(x' \mid \boldsymbol{\theta})^2 + \gamma^2}, \tag{4.2}$$

which describes the relative probability of the frame x compared to the other frames from the transition path $\mathcal{T}$ with an additional hyperparameter γ. For larger values of γ, the exploration is higher and less likely frames are selected more often. At the limit for $\gamma \to \infty$, any frame would have the same probability to be selected.

4.2. Neural Network Architecture

In this section, we will propose two new architectures for neural networks which can improve the performance of ML-guided transition path sampling.

4.2.1. Contextual Neural Network

This architecture is a straightforward extension of the neural network architecture presented by Jung et al. [2023]. Instead of only using the current frame as the input, we provide the previous and next frame as an input as well. With this, we can reformulate the architecture as

$$f(x_{i-1}, x_i, x_{i+1} \mid \boldsymbol{\theta}), \tag{4.3}$$

where we use $\mathbf{0}$ for each neighboring frame that is out of bounds. By introducing this simple modification, the neural network can learn contextual awareness and infer parameters such as the step size, and the velocity.

4.2.2. Self-Attention-based Encoder

However, a context that only includes the neighboring frames might be too small and could hinder learning performance. Furthermore, this increases the dimension, which can make it challenging for the neural network to efficiently use this input as the data is sparse. We instead propose a more sophisticated architecture that relies on self-attention [Vaswani et al., 2017].

Self-attention has been widely used in natural language processing in recent years [Devlin et al., 2019], but has also shown promising results in image recognition [Dosovitskiy et al., 2021], and protein structure prediction [Lin et al., 2022]. With this architecture, often referred to as transformers, we can process a sequence of embeddings and determine for each input the relation to all other inputs. In the case of natural language, we could input a sentence word by word (i.e., token by token), and then the self-attention mechanism learns to weigh for each word the importance relative to all other words. This overcomes some problems with previous contextual and autoregressive architectures, for example long contexts are easier to learn.

Here instead, we propose to use each frame of the trajectory as one token and use self-attention in the form of an encoder architecture so that the predictions can make

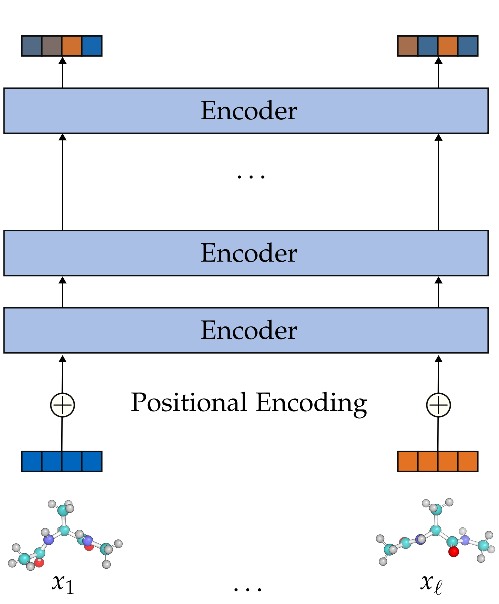

Figure 4.1.: The encoding architecture of a transformer consists of a stack of multiple encoder layers. At the end, one embedding with an arbitrary pre-defined dimension is produced for each frame of the trajectory.

use of the complete trajectory. This architecture has been visualized in Figure 4.1. Each encoder in itself consists of a self-attention layer that performs the weighing for each of the inputs, which are then normalized and mapped with a neural network, as illustrated in Figure 4.2. The weights of this neural network are shared over all inputs, allowing the architecture to be used for variable-length sequences.

We use this transformer encoder as a base to compute a contextual-aware embedding for each frame of the transition path. Instead of the original representation for each frame, we use this embedding as the input for another neural network that can be applied in parallel to all of the embeddings. With this, the predictor can be written as

$$f(x \mid \mathcal{T}, \boldsymbol{\theta}). \tag{4.4}$$

4.2.3. Embeddings and Positional Encoding

When only using an encoder without special embeddings, it cannot learn different features depending on the position of the frame. For example, it can be meaningful for the neural network to learn to never shoot from the first frame, as this is already in a meta-stable state and will not reach the other state. To circumvent this problem, we need to add a positional encoding to the representation of the molecules. While there

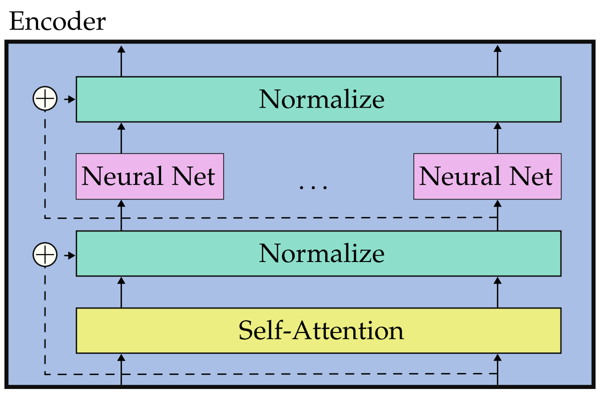

Figure 4.2.: A depiction of the inner architecture of an encoder layer. It relies on self-attention, a neural network, normalization layers, and skip connections.

are many choices for this, we have decided to use the classical sinusoidal positional encoding [Vaswani et al., 2017]. For this, we set a maximum number of frames for our trajectory $\ell_{\max}$, and rewrite each embedding x_i as

$$x_i + e(x_i) = x_i + \begin{cases} \sin\left(i \cdot 10000^{-\frac{i}{\ell_{\max}}}\right), & \text{if } i \text{ is even} \\ \cos\left((i-1) \cdot 10000^{-\frac{i-1}{\ell_{\max}}}\right), & \text{otherwise} \end{cases} . \tag{4.5}$$

With this, we alternatively add a sine or a cosine with the relative position of the frame and the network can learn to incorporate this information. There are different types of positional encodings such as rotary positional encoding [Su et al., 2021], which has been successfully applied in biological contexts [Lin et al., 2022]. We believe that exploring different positional encodings could improve the performance of our approach.

4.2.4. Training

Typically, when training a transformer, we can compute a loss for each representation. However, since we perform shooting from individual frames $x \in X$, we only have access to results for these specific points. To still be able to train, we apply the encoder to the complete trajectory $\mathcal{T}$ and then apply a neural network with shared weights to each embedding. This allows us to propagate the gradient back only for those frames where we actually have the results of a shooting. Since the parameters are shared between the frames, they can be updated accordingly. However, this requires us to store the whole trajectories for training instead of only the individual frames, which increases the memory footprint substantially.

5. Experiments and Evaluation

In this chapter of the thesis, we will explore the approaches discussed in Chapter 3 and Chapter 4 respectively. In Section 5.1, we will explain the setup and properties of the underlying system (alanine dipeptide) that we will be using throughout this chapter. Then, we will show how we trained a Boltzmann generator and document our findings and results in Section 5.2. Beginning with Section 5.3, we will look at the TPS problem and discuss different ways of generating the ground truth data that will be used to compare and evaluate our approaches. Section 5.4 and Section 5.5 are the main part of this chapter, where we evaluate and investigate the two different approaches of this thesis: latent space path sampling and accelerating shooting with machine learning.

Note that the presented approaches (unless otherwise noted) fulfill the underlying theoretical requirements so that they will eventually sample the true ensemble of paths. This means, that when we evaluate the different techniques in this chapter, it often boils down to how quickly the Markov chains produce realistic results and how efficient the approaches are. In practice, the theoretical guarantees are not meaningful if the algorithm takes too long before the distribution is approximately correct.

5.1. Setup

In the following, we note the underlying system, software, and configuration we used.

5.1.1. Implementation

As for the concrete software stack, we have made use of the openMM MD engine [Eastman et al., 2017] for simulation. For machine learning related operations, we relied on pytorch [Paszke et al., 2019], and for the normalizing flow/Boltzmann generator we used normflows [Stimper et al., 2023]. For the two-way shooting technique, we relied on the python library OpenPathSampling [Swenson et al., 2019a,b]. To use the learned approximated committor function to guide the shooting points, we adapted the code from Jung et al. [2023].

In all instances where we present runtime numbers, the results were generated on a single NVIDIA RTX A6000 GPU. However, most of the results were produced with a single NVIDIA GeForce RTX 2080 Ti GPU.

5.1.2. Molecular System

To evaluate our approaches, we have decided to use the molecule alanine dipeptide (ALDP), which has already been introduced in Figure 2.5. This molecule is small enough, that we can efficiently simulate the molecular dynamics and transition path ensembles with existing approaches, while it is still complex enough so that we can make educated comparisons. Although the molecule has 22 atoms, it can mostly be described by its two dihedral angles ϕ, ψ. As for the simulation, we have relied on the implicit test system that is provided with openMM. It uses the amber ff96 force field in an implicit solvent where the forces are guided by OBC GBSA. For the concrete simulation, we have used a time step of $1fs$ at a temperature of $300K$.

5.1.3. State Definitions

Defining the meta-stable states of the system is crucial for transition path sampling, as was we discussed in Section 2.3.3. Finding these states can be challenging in large systems, especially when there is no knowledge about the reactive coordinates. Depending on the specific force field, solvent, and temperature, different meta-stable states occur. Since the molecule ALDP is regularly used a test system, it has been explored in detail and thus these meta-states are known. Typically, these states are defined with the help of the collective variables, in this case the dihedral angles ϕ, ψ. Table 5.1 notes the exact definition of the meta-stable states, and Figure 5.1 shows them visually. This type of histogram is called *Ramachandran plot* [Ramachandran et al., 1963].

Table 5.1.: **Meta-stable states of alanine dipeptide.** This table illustrates the state definitions used for transition path sampling. All states are defined spherical, meaning that all conformations where the dihedral angles are within the radius of this sphere are considered to be in the same state.

State	Center (ϕ, ψ) in Degrees	Radius in Degrees
C_5	$(-150, +150)$	20
P_{II}	$(-70, +135)$	20
α_P	$(-150 - 65)$	20
α_R	$(-70, -50)$	20
C_7	$(+50, -100)$	20
α_L	$(+40, +65)$	20

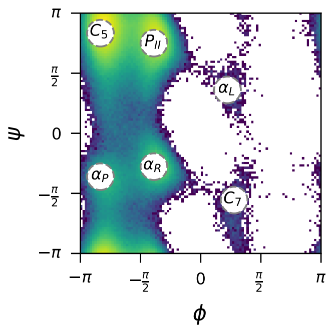

Figure 5.1.: This figure shows a histogram of a long running MD simulation. For each simulated conformation, the periodic dihedral angels ϕ, ψ are computed and then visualized to approximate the observed density. Further, the meta-stable states are highlighted with circles. If a frame is within one of these meta-stable states, the simulation will stay in this area for a relatively long time. Note that the states have been defined with degrees, but this plot shows the angles in radians.

5.2. Training a Boltzmann Generator

Access to a trained Boltzmann generator is crucial for our latent transition path sampling approach, and hence we will discuss in this section the specifics of our training, issues that we encountered, and how they can be addressed.

5.2.1. Architecture

The Boltzmann generator we use parametrizes a mapping between the internal coordinate representation of the molecule and a mixture between uniform and Gaussian variables, as was done by Noé et al. [2019]. Internal coordinates use bond angles, bond lengths, and torsional angles to describe the offset to a base frame. They are rotation and translation independent, making it easier to operate for classical neural networks. Additionally, using internal coordinates of a molecule has the major advantage that the distribution of samples is already well-conditioned, as most of the coordinates have a mean close to 0. Since some of the internal parameters, such as angles, are periodic, the base distribution will capture those with a uniform distribution. We have discussed in Section 3.5 a way to transport between a uniform distribution and a Gaussian efficiently, so that the samples in latent space can again be distributed according to a standard normal distribution.

As for the specific architecture, we have found that neural spline layers [Durkan et al., 2019] are well suited to capture the energy landscape. We use 12 coupling layers where a neural network approximates a spline with 8 knots by a quadratic rational spline function. The output of the splines is then corrected so that the coordinates again represent a periodic space. As for the neural network, we use two layers with a residual block [He et al., 2015] with 256 hidden units each. We rely on a random binary mask between the layers to decide which parameters are passed through the coupling architecture and which ones are used as an input to the neural network.

5.2.2. Training

To train a Boltzmann generator, we have simulated the molecule ALDP for $10ns$ to produce 10 million frames with a step size of $1fs$. We then maximized the likelihood of these samples in the base distribution, as stated in Equation 2.8 when setting $w_{KL} = 0$. We have also experimented training the Boltzmann generator by including the reverse KL divergence as well. However, in the distribution of ALDP the gradients can become quite large and point away from the modes, which makes training challenging. In our case, this made training with the reverse KL unstable when using single float precision and required switching to double float precision to resolve this issue. The resulting flow exhibited slower performance due to the additional computational overhead and we did not notice any significant accuracy gains in our experiments. Hence we decided to evaluate our approach on a flow that has only been trained with maximum likelihood. In general, we advise to use both loss terms, as this can speed up the training, especially for larger systems [Noé et al., 2019].

5.2.3. Results

Once trained, the base distribution can be changed from a Gaussian-Uniform mixture to a Gaussian, and we can sample standard normal distributed random variables. These samples can then be mapped with the forward transform $T : \mathbb{R}^{60} \mapsto \mathbb{R}^{60}$ into the internal coordinate space of the molecule. Given the base frame that was used to define this internal coordinate representation, we can then transport these samples into the full product space $\mathbb{R}^{22\times3}$, where each of the 22 atoms has a 3D position in space. A histogram of the main dihedral angles when drawing 1 million samples is shown in Figure 5.2. Such a Ramachandran plot serves as an essential way to judge the performance of a trained Boltzmann generator [Midgley et al., 2023b].

With this Ramachandran plot, we can see that the trained Boltzmann generator approximates the underlying distribution well. However, since the main objective of this thesis is to sample transition paths between meta-stable states, we are especially interested in how well the Boltzmann generator can capture these states. Figure 5.3 compares how often the meta-stable states are observed in a long-running MD simulation (i.e., the ground truth), and when sampling states with a Boltzmann generator.

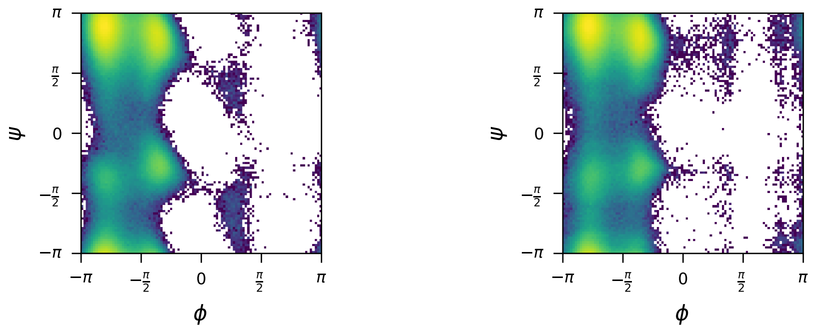

Figure 5.2.: The ground truth Ramachandran plot produced by a long-running MD simulation (*left*), and the reconstructed histogram constructed by sampling independent Gaussian and transporting them with the Boltzmann generator to product space (*right*).

We can see that the Boltzmann generator successfully captures this distribution as well, and see that the rare states, namely C_7 and α_L are almost never observed in practice. Capturing transitions from and two these states with classical molecular dynamics simulations is already challenging for this rather small molecule.

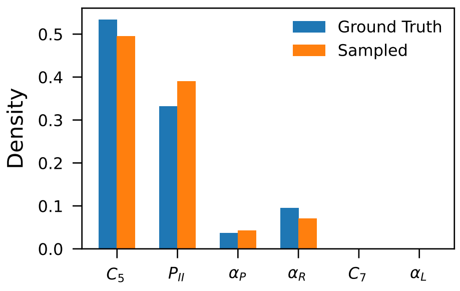

Figure 5.3.: A histogram showing the distribution of samples between the meta-stable states. It compares the frequency observed in the MD simulation, with samples from the trained Boltzmann generator. In both cases, C_5 and P_{II} are the most probable states.

5.3. Generating Ground Truth Path Ensembles

In order to evaluate the quality of our transition path sampling approaches, we first have to produce some ground truth data that we can use to benchmark and compare. In this section, we will discuss two different approaches to generate an ensemble of paths: extracting transitions from a molecular simulation and using existing TPS techniques. We will also explore a way to visualize the path ensembles more efficiently.

5.3.1. Constructing Paths from a Simulation

One–albeit slow—approach to construct a transition path ensemble is to perform a long-running MD simulation and extract the trajectories from there. Since this construction technique does not use any guidance at all, it is not suitable for larger systems. The main idea is that we compute for each frame of the simulation the respective state, and then extract the shortest trajectories that transition from state A to B or vice versa, as illustrated in Figure 5.4. With this approach, the paths in the ensemble will have a variable length.

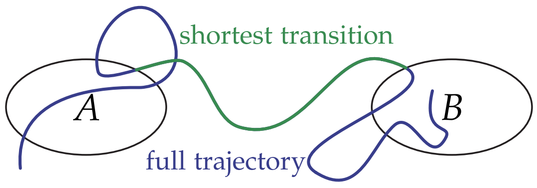

Figure 5.4.: An illustration showing how we can extract the shortest possible transition from a longer trajectory.

However, since this technique can be used to create ensembles for certain state transitions of the molecule alanine dipeptide, we have decided to use it whenever possible. Later we will see that even a long-running MD simulation can only find a few transitions between some states, such as $\alpha_R \leftrightarrow C_7$. This shows the advantages and the necessity for more efficient TPS sampling techniques.

5.3.2. Fixed-Length Ensembles with Two-Way Shooting

This variable-length "sampling" approach that extracts trajectories from a MD simulation, allows us to determine a suitable transition time between states. In Figure 5.5, the distribution of transition times between two meta-stables is illustrated. This density was approximated by the path ensemble that could be constructed from the long-running MD simulation. With this, we have decided that a suitable transition time between the states $C_5 \leftrightarrow \alpha_R$ is $1.6ps$ as many transitions occur at this duration. To determine the transition time between hard modes, we computed a variable-length transition path ensemble with two-way shooting. We than have performed similar studies to determine a transition time of $360fs$ between the modes $\alpha_R \leftrightarrow C_7$. Once we have determined a suitable transition time between states, we can perform TPS with two-way shooting to create a fixed-length path ensemble.

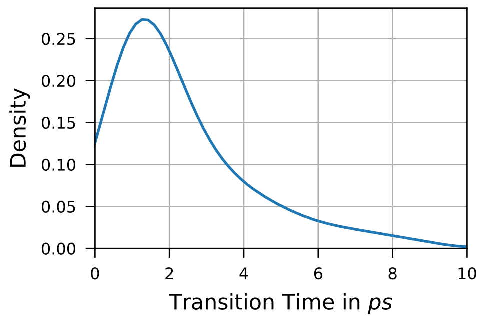

Figure 5.5.: This figure shows the density of transition times between the states $C_5 \leftrightarrow \alpha_R$.

5.3.3. Path Histograms

Visualizing the transition path ensemble can be an effective way to get some first insights of the underlying molecular system and to assess the quality of the sampling method. For higher-dimensional systems, as even small molecules are, some form of dimensionality reduction is needed. If the reactive coordinates are known, such as for alanine dipeptide, we can visualize the transition paths in the space of those. However, transition path sampling is often used to determine reactive coordinates [Hooft et al., 2021, Ray et al., 2023]. Common dimensionality reduction techniques such as tSNE [van der Maaten and Hinton, 2008] or PCA [Hotelling, 1933] can be used, but are often unfit because the samples represent an evolution of the system over time. More generally, one can rely on a lower dimensional embedding found with tICA [Molgedey and Schuster, 1994] to represent the transition path ensemble.

Once a suitable lower-dimensional representation has been found, in our case the dihedral angles ϕ, ψ, we can discuss visualization methods. A straightforward approach to illustrate the ensemble is by using a Ramachandran plot. However, with this technique it is difficult to make out individual paths or patterns of transitions, especially if the ensemble is small. The reason for this being, that the density is sparse and there is no indication on which points belong together.

A different, and in our opinion more suitable approach, is to visualize the individual paths in the form of a *path histogram*. Instead of plotting the frames of a path independently, we draw a line between all pairs of frames to connect them. This gives us one continuous line in the plot per trajectory. If we repeat this for all observed paths, we can construct a histogram where each path increases each bin it passes by 1, even if the path overlaps with itself.

Although a straight line might not be the best interpolation technique, it allows us to see which points are connected. Already with a few transition paths, this allows us to see the overall reaction channels and behavior of the system well.

5.4. TPS with Boltzmann Generator-based MCMC Moves

Now that we have an understanding on how to compare path ensembles, we will continue by assessing the quality of our latent space TPS approach. For this, we will first explore properties of the latent space in Section 5.4.1, and will continue with qualitative evaluation of different kernels in Section 5.4.2 and Section 5.4.3. We will discuss options for an empirical evaluation in Section 5.4.4 and will then explore the Gaussian kernel in more detail in Section 5.4.5. Continuing this in Section 5.4.6, we conclude this chapter by investigating the impact of the proposed approximations that result in a non-correct MCMC approach but can speed up the sampling.

5.4.1. Latent Space Analysis

For our latent TPS approach to work, we need a Boltzmann generator that produces a suitable latent space. To assess the quality, we can investigate the separability of the meta-stable states in latent space. For this, we have transported conformations classified to be in the meta-stable states C_5 and α_R to the latent space. In Figure 5.6, we have visualized this transformation with two different embeddings: PCA, and tSNE. Already with PCA, a linear dimensionality reduction technique, the conformations can be distinguished based on the first principal component. This demonstrates that our Boltzmann generator is able to learn a meaningful representation of the states and can distinguish between different conformations.

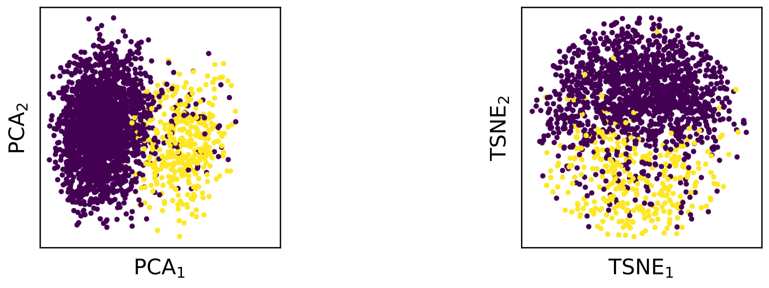

Figure 5.6.: These plots show conformations of the molecule ALDP that have been classified as C_5 and α_R, and their latent space representations. Since the latent space is in $\mathbb{R}^{60}$, we have only visualized the first two dimensions of PCA (*left*) and tSNE (*right*). The two different colors indicate the different states.

This serves as a good theoretical foundation for our latent approach but does not completely motivate that simple transformations describe a realistic behavior. Previously, we have argued that a linear interpolation in latent space can be used as an efficient way of producing a (relatively) realistic pathway. Figure 5.7 (*center*) shows the main dihedral angles of such a linear interpolation in latent space. This simple operation, already produces a non-linear transition path that, as we will see in later section, is close to what can be observed with molecular simulations. When repeating this experiment, and linearly interpolation between random conformations $a \in C_5$ and $b \in \alpha_R$, we can see that this already reveals two reactive channels, as seen in Figure 5.7 (*right*). Similarly, Figure 5.7 (*Left*) shows also a linear interpolation but of the atom coordinates directly without a latent space. Jointly, those two illustrations motivate that the latent space is a suitable representation to capture transition paths. Hence, in our future experiments, we will construct initial paths by linearly interpolating between the target conformations in latent space. Compared to other approaches, such as high-temperature MD simulations, this approach is highly efficient (given a trained Boltzmann generator).

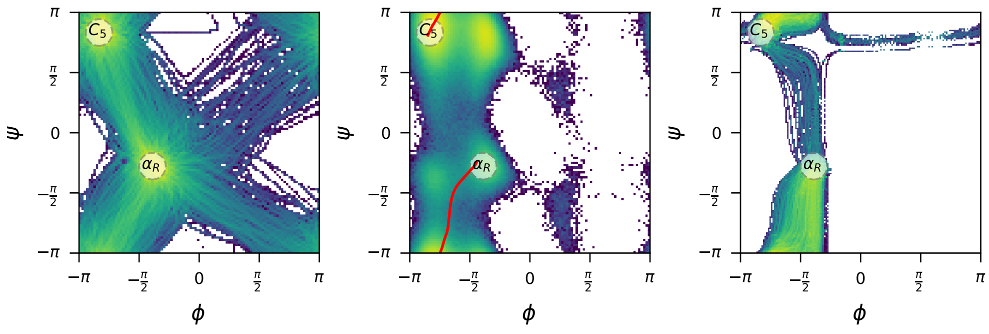

Figure 5.7.: *Left:* The Ramachandran plot when interpolating atom coordinates linearly in product space to find transitions between $C_5 \leftrightarrow \alpha_R$. *Center:* The ground truth Ramachandran plot with a linear interpolation in latent space highlighted in red. *Right:* The histogram that can be observed when randomly picking states from C_5 and α_R and linearly interpolating between them in latent space.

5.4.2. Evaluation with Marginal Densities

A first approach to compare the quality of path ensembles is by visualizing the marginal densities of ϕ and ψ. In Figure 5.8, these have been depicted for the ground-truth path ensemble we could compute from fixed-length TPS and two of our transition kernels: Gaussian noise and Hamiltonian. We can see that the density of ϕ is similar for all different methods. However, we can observe stark differences in the distribution of ψ of the different methods. While our latent kernels perform similarly, they sample significantly more paths with a large angle. Hence, the view is limited when only looking at a portion of the reactive coordinates, and a more extensive analysis is needed.

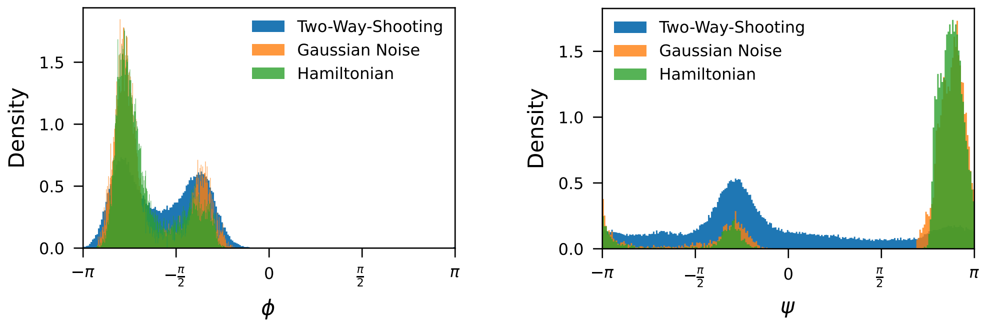

Figure 5.8.: Histograms of the marginal densities of the path ensembles sampled by different transition path approaches. It shows the densities for the dihedral angles ϕ and ψ in the *left* and *right* figure respectively.

5.4.3. Qualitative Comparison with Path Histograms

For a more in-depth and detailed comparison of the path ensembles, we will make use of the aforementioned path histograms. Figure 5.9 shows the main evaluation of our approach and compares the different path ensembles sampled by various methods and kernels. *MD Simulation* shows the variable-length path ensemble that can be extracted from a long-running MD simulation, with *MD Simulation 25%* showing the 25 percent of paths with the highest probability. For *MCMC Shooting*, we used two-way shooting with uniform point selection. All other approaches are produced with our kernels (compare Section 3.3): *Gaussian Noise* adds iid. noise to the path in latent space, *Hamiltonian* performs Hamiltonian dynamics in latent space, *Unconditional GP* fits a Gaussian process in latent space and samples from the mean, *Conditional GP* uses the current path as mean instead, *Unconditional GP Uni* samples not at the frame indices but first draws uniform points and then evaluates the Gaussian process at this position, and *Unconditional GP Fixed* fits a Gaussian process once to paths linear in latent space.

We can see that the ensembles produced by MD simulations are different compared to the others, because this approach uses variable-length TPS. Moreover, we can see that MD simulations struggle to sample difficult transitions already for this small molecule, which is the major motivation for TPS. The two-way shooting approach serves as the ground truth and reveals up to four different transition channels for transitions from $C_5 \leftrightarrow \alpha_R$. Although the Gaussian noise proposal kernel is the simplest of all, it performs surprisingly well. We attribute this to the fact that while only few of the proposed paths are realistic, it can be implemented efficiently and with a sufficiently large noise explores the energy landscapes well. Similarly, the Hamiltonian kernel samples realistic transitions, although it fails to explore all reaction channels. According to this evaluation, the Hamiltonian kernel performs better than other latent approaches.

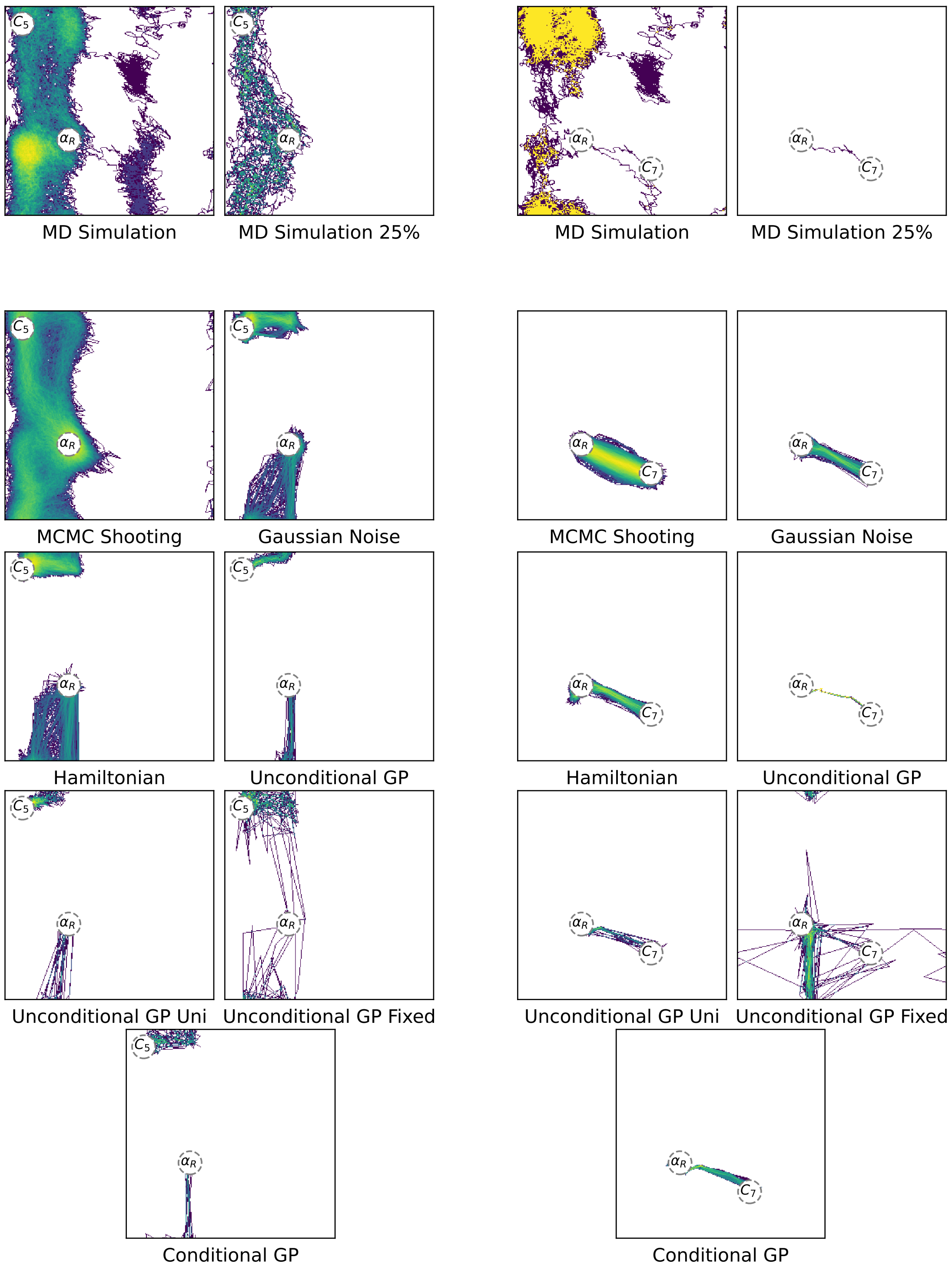

Figure 5.9.: Comparison of the transition path ensembles for transitions between two different states: $C_5 \leftrightarrow \alpha_R$ (*left*) and $\alpha_R \leftrightarrow C_7$ (*right*), and different methods.

As for the Gaussian process kernels, we can see that the conditional sampling technique slightly increases the diversity of sampled paths, similarly to the uniform sampling of the points t. We can further see that the fixed Gaussian process captures more reaction channels, as it is already fitted on a diverse set of paths. While this is advantageous for the transition $C_5 \leftrightarrow \alpha_R$, it poses a problem for the transition $\alpha_R \leftrightarrow C_7$ which includes rare modes. The reason for this is that for the rare conformations in C_7, the Boltzmann generator is less accurate and thus some transitions do not reflect realistic pathways. With this, we learn two channels, whereas only one is accurate.

Furthermore, all Gaussian process approaches exhibit a low path acceptance probability, which is worsened by the computational overhead introduced by fitting the process. Hence, the presented path ensembles are substantially smaller compared to what we could produce with other kernels.

5.4.4. Empirical Evaluation with Approximated KL-Divergence

Visually comparing path histograms can be a cumbersome and error-prone task. To overcome this, we have tried multiple approaches that allow us to empirically assess the quality of the transition path ensemble. For all the metrics we are going to introduce, we relied on some form of distance to the ground truth fixed-length path ensemble we computed with two-way shooting.

The KL-Divergence [Csiszar, 1975] is an asymmetric function that allows us to compute the distance between two distributions. While it is strictly non-negative when analytically computing it, we have to rely on approximations since the underlying densities are unknown. For the metric *KL-NN*, we compute the KL-Divergence between the observed ϕ, ψ by approximating the density with a nearest neighbor approach Perez-Cruz [2008]. In *KL-Hist*, we create a path histogram of the transition paths and compare the resulting discretized densities. For the *Mean-Dist* metric, we compute the average absolute distance between the bins of the path histograms. To account for the fact that we observe a different number of transition paths for different methods, for *KL-Hist* and *Mean-Dist*, we normalize the bins so that they sum up to 1.

In Table 5.2, we have computed the aforementioned metrics on our transition kernels. None of the observed metrics fully align with what we can assess visually. To this end, we have yet to find a suitable formula that is able to accurately describe the quality of a path ensemble.

5.4.5. Exploring Noise Scales for the Gaussian Proposal Kernel

In this section, we will investigate the impact of the variance for the Gaussian noise kernel in more detail. This should serve as an example, as all other kernels have

Table 5.2.: **Empirical evaluation of path ensembles.** For all kernels evaluated in Figure 5.9, we evaluated three different metrics. For all metrics, a lower score is better.

	$C_5 \leftrightarrow \alpha_R$			$\alpha_R \leftrightarrow C_7$		
Proposal Kernel	KL-NN	KL-Hist	Mean-Dist	KL-NN	KL-Hist	Mean-Dist
Gaussian Noise	3.26	$+2.48 \times 10^{-3}$	$\mathbf{2.21 \times 10^{-5}}$	5.27	$\mathbf{-2.92 \times 10^{-3}}$	2.71×10^{-5}
Hamiltonian	3.58	$+5.75 \times 10^{-3}$	2.35×10^{-5}	4.89	-2.56×10^{-3}	2.67×10^{-5}
Unconditional GP	4.33	-2.73×10^{-3}	2.93×10^{-5}	6.10	-1.08×10^{-3}	3.15×10^{-5}
Unconditional GP Uni	3.85	-3.83×10^{-3}	2.89×10^{-5}	5.13	-2.88×10^{-3}	2.96×10^{-5}
Unconditional GP Fixed	**1.68**	$\mathbf{-5.34 \times 10^{-3}}$	2.51×10^{-5}	**3.06**	-2.48×10^{-3}	$\mathbf{2.62 \times 10^{-5}}$
Conditional GP	3.19	-4.02×10^{-3}	2.89×10^{-5}	6.80	-2.22×10^{-3}	3.04×10^{-5}

hyperparameters as well that guide the path sampling in a similar way. Figure 5.10 illustrates the result of the same kernel with different choices for σ^2, which specifies the variance of the noise that will be added to the latent space representation. We can notice, that a larger noise scale increases the diversity of the sampled transitions. However, this kernel faces a tradeoff between the acceptance probability of paths and their diversity: A higher noise results in more diverse paths, but since they will be very different the acceptance probability will be low. This has major implications for the runtime, when using a variance of $\sigma^2 = 0.01$ 200 transition paths can be sampled within 12 minutes, with a variance of $\sigma^2 = 0.05$ it increases to 16 minutes, but with $\sigma^2 = 0.1$ it exponentially grows to 15 hours and 43 minutes.

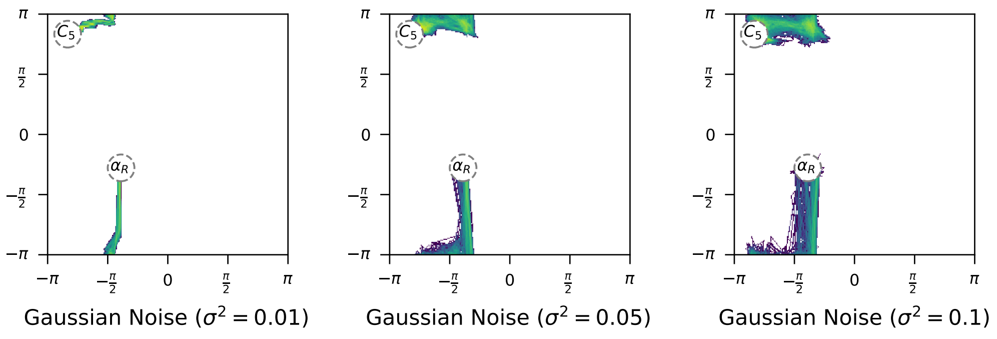

Figure 5.10.: Comparison of three path ensembles, all created with a Gaussian noise proposal kernel but with a different variance σ^2 of the noise sampled from $\mathcal{N}(0, \sigma^2)$.

Although the time for the algorithm to sample paths degrades, paths with a higher probability are sampled. We can see in Figure 5.11, that a larger noise scale leads to accepted paths to have a higher probability. This is because a variety of different paths is explored, and more likely paths are more likely to be accepted based on the Metropolis-Hastings' acceptance criterion. Moreover, we can see that the path probability increases for all instances, which is because the initial path used is simply a linear interpolation in latent space and does not necessarily have a high probability when evaluated under Langevin dynamics.

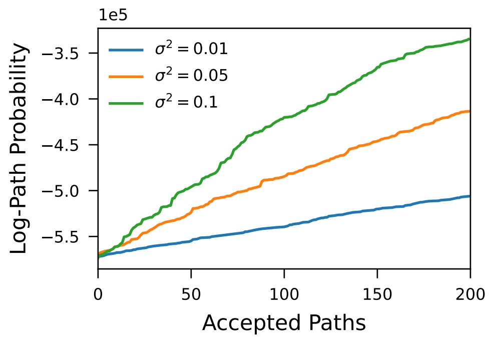

Figure 5.11.: We sampled 200 paths with the Gaussian noise proposal kernel with varying noise scales and illustrated the probability of each accepted path.

5.4.6. Impact of Approximative MCMC Sampling

We saw that while increasing the noise scale for the latent proposal kernel brings benefits in terms of paths diversity and the quality of transition paths, it also significantly impacts the performance. Thus, we strive for ideas to make the inference faster by either employing approximation techniques, or by improving the quality of proposed paths to increase the acceptance criterion. In Section 3.4, we have explored two different ideas that we will investigate in this section. Note that these approximations are not compatible with the theoretical requirements of the MCMC procedure and approaches that rely on them are not guaranteed to sample paths with the correct probability.

Energy-Based Path Probability Approximation

The first approximation we are going to explore is to estimate the path probability of the trajectory by computing the probability of the independent states. With this, we do not need to compute the gradient of the energy function, which can speed up the performance benefits. Previously, we have already discussed that this can potentially lead to degradation of the frames to (the same) local minima, and thus we have also explored this idea in combination with path equidistancing. In Figure 5.12, we compare three different options for the Gaussian kernel with the same noise level but different path processing/probability calculation.

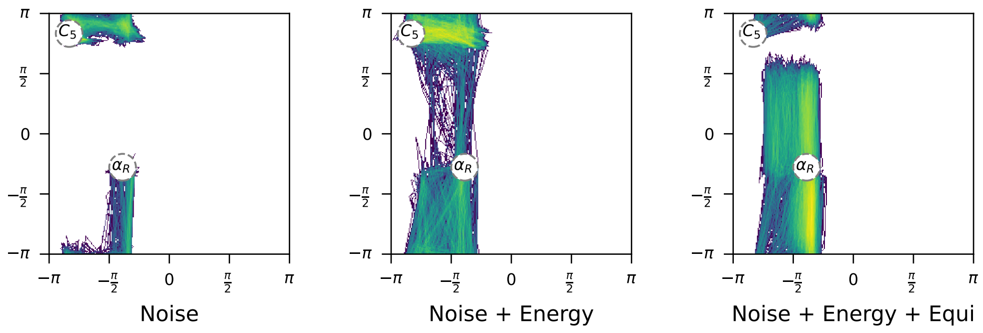

Figure 5.12.: The path histograms that can be created with a Gaussian noise kernel when using the correct path probability (*left*), approximating the probability with the energy of the frames (*center*), and when also equidistancing the frames in product space (*right*).

We can see, that the ensemble found by approximating the path probability with the energy of the states is more diverse as when solving the Langevin dynamics equations. This can be attributed to the fact that the energy-computed path probability is generally higher, as the frames are only evaluated independently and not in sequence. With this, changes only have to improve the path locally, instead of improving the quality of the whole trajectory. However, with this, the underlying paths might be less physical. Making the paths equidistant in product space seems to degrade the quality of the paths and makes the transitions seem uniform and we thus advice against using it.

Using the energy of the paths significantly reduces the runtime from more than 15 hours to only 1 hour 36 minutes for 200 paths when using the energy, and 1 minute when using the energy and making the points equidistant.

Path Relaxation with Molecular Simulations

Since all latent TPS kernels operate without molecular dynamics simulations, many of the proposed paths are unrealistic and will thus get rejected. The aim of the idea presented in this section is to improve the quality of trajectories by performing a short MD simulation of all frames in parallel to relax the path. While the movement of the individual frames is not connected, the spacing between the frames and the overall properties might be improved.

In Figure 5.13, we explored this idea when using the correct Langevin path probability, but also in combination with the approximations from above. We can see that, indeed, these small molecular simulations improve the quality of paths (i.e., paths get more noisy and less linear). Especially for the Langevin path probability, we believe that this can be a useful addition to our algorithm. However, as soon as we stack multiple approximations on top of each other, the performance worsens. Especially when using all discussed approximations together, no meaningful path ensemble can be sampled.

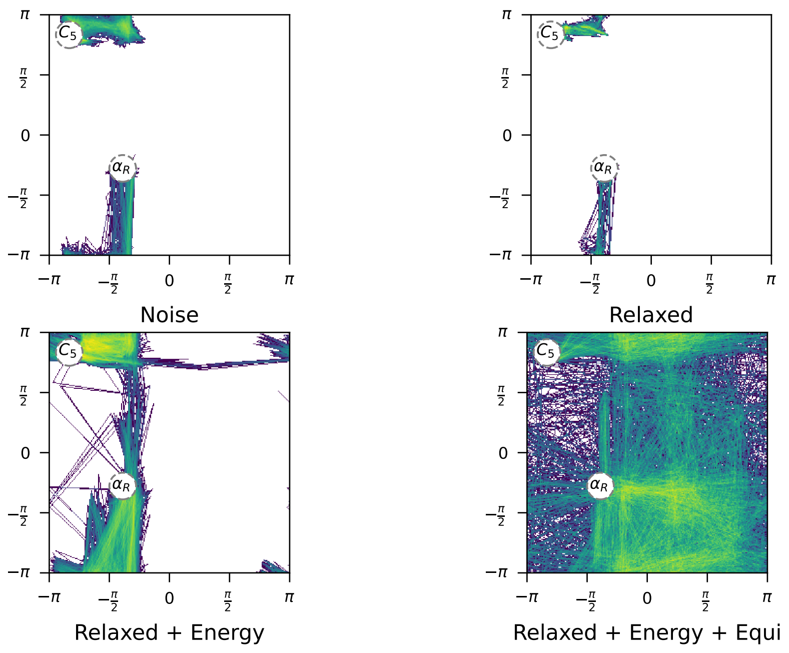

Figure 5.13.: The path histogram that can be created with a Gaussian noise kernel without and with small MD simulations (*top left* and *top right*), the path histogram of a Gaussian kernel with frame-independent energy-based probability (*bottom left*) and additionally with path equidistancing (*bottom right*).

5.5. ML-Guided Two-Way Shooting

In this section, we will investigate whether machine-guided shooting can be improved when access to the complete trajectory or neighboring frames is given.

5.5.1. Architecture

We will compare four different shooting methods: random point selection (baseline), network architecture from Jung et al. [2023], contextual network with neighboring frames, and the self-attention-based encoder with a neural network. The guided shooting architecture we will compare against consists of layers with a decreasing number of weights, followed by residual layers [He et al., 2015]. As an input, we will use an internal representation of the frames, similarly to what we did for the Boltzmann generator in Section 5.2.

To make the results as comparable as possible, we will use the same network architecture as the base model and only deviate from it when absolutely necessary. In the case of the contextual network, this means that we just change the weights in the first layer (so that we can input the previous and next frame). For the transformer-based architecture, we apply this neural network to embeddings that were learned by multiple encoder layers.

5.5.2. Path Ensembles

As a first step, we compare the ensemble of transition paths that can be found when performing 1000 MCMC steps. All the ensembles have thus a slightly different number of transition paths, and hence for some methods the data looks sparser than for others. The results are illustrated in Figure 5.14, which shows that regardless of the point we shoot from, the ensemble will be similar. With this, we can also see why it is uncommon to compare path ensembles in literature.

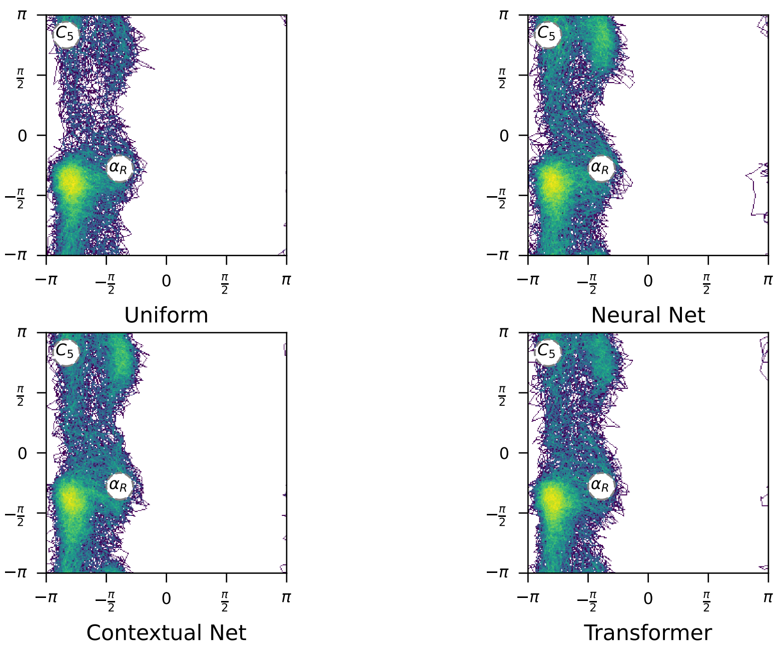

Figure 5.14.: A comparison of the path ensembles that have been created with two-way shooting with different shooting point selection methods.

Note that in this chapter we now explore variable-length shooting, and thus the ensembles slightly deviate from the ones we have seen in Figure 5.9. Most notably, the ensemble will be more similar to the one extracted from the long running MD simulation. This can be especially seen at the state α_P in the lower left, which has a higher density since variable-length transition can "spend more time" in other states.

5.5.3. Acceptance Probability

For large molecules and systems, the dominating part of the runtime will be the number of MD simulations that need to be performed. We believe that as such, the best comparison for different shooting-point selectors is to investigate the number of accepted transition paths over time. In Figure 5.15, we have illustrated this for the four different options, whereas the uniform shooting point selection is the baseline.

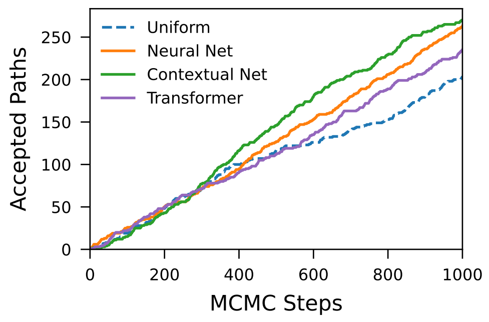

Figure 5.15.: We compare the cumulative number of accepted paths at a given number of MCMC steps. Paths are rejected if either the MD simulation does not reach the target states, or if the acceptance ratio is too low.

With this evaluation, we can see that all approaches outperform the baseline of uniform selection, and the best approach is the contextual neural network. The transformer architecture performs the worst of the guided approaches. Since transformers are more expressive than the contextual network, we believe that the major reason for its lacking performance is due to training time. As the number of parameters is larger, we can see that transformer only outperforms the uniform selection after 500 steps, whereas the simple neural network already does so at 400.

Since molecular dynamics simulations are inherently non-deterministic, and there is no way to fully seed the openMM MD engine, the concrete numbers can vary. We performed multiple runs for each, and in most cases these trends seemed to be consistent. Further investigations are needed to see how they compare on larger systems, over a longer period of time, or to evaluate the impact when transitioning between different states.

6. Conclusion

In this thesis, we have explored two different approaches to improve transition path sampling. We have introduced a novel MCMC-based framework for our primary approach, where transition paths can be sampled in a latent space with various proposal kernels. For this, we employ a Boltzmann generator to map the individual frames of a transition path to a latent space representation. In this well-conditioned latent space, high-energy barriers are more straightforward to overcome, and simple modifications reflect meaningful alterations to the original path. The specific latent proposal kernels have lean requirements, and we have presented a variety of different choices.

Currently, the literature lacks impactful ways to compare the quality of multiple path ensembles sampled by different TPS approaches. We believe that this is due to the theoretical guarantees that almost all current TPS approaches fulfill. As all approaches eventually converge to sample the true ensemble of paths, all comparisons are merely of a practical nature and can vary across the same system. Although our approach fulfills these guarantees as well, we used path histograms, marginal densities, and derived metrics to evaluate the real-world performance of our approach.

In the simulation-free latent approach the observed performance varied significantly depending on the concrete states and kernel. We believe that these findings also translate to other approaches and that there should be more research on the real-world performance of TPS methods. For our latent approach, the paths were accurate but in some cases the kernels failed to produce a diverse ensemble within a reasonable time. With this, our kernel could not match the performance of simulation-based approaches. We attribute this to the fact that all of our kernels operate on individual frames instead of the complete path. As such, many of the proposed paths will have a low probability an thus get rejected, resulting in a low acceptance ratio. If the proposed paths were more physical, we could introduce more variance without impacting performance.

While the introduced approximations and improvements (such as parallel MD simulations on the individual frames) can improve the quality of paths, we still need to fully solve our algorithm's current problems. However, since the introduced latent MCMC TPS framework is mathematically sound and works with different latent proposal kernels, research into kernels operating simultaneously on all the frames can be impactful. Kernels that use reinforcement learning or adaptively change the latent space might be a promising research direction.

Another area for improvement of our approach is that we need access to a trained Boltzmann generator. While there is active research on how to improve the training [Köhler et al., 2021, Midgley et al., 2022, 2023a,b, Felardos et al., 2023], Boltzmann generators still have to be retrained for each individual system. However, when deriving our MCMC framework, we did not assume any technical requirements of Boltzmann generators other than that they are a bijection and that we can compute the Jacobian of the forward transformation. This opens up new possibilities to use more suitable latent spaces, as the one created by variational autoencoders (VAEs) [Kingma and Welling, 2013] or latent diffusion models [Rombach et al., 2021, Park et al., 2023]. With other latent spaces, or by modifying the latent space of a Boltzmann generator, problems such as "How to produce a suitable latent space consistently?" might be easier to overcome. Using a latent space with an arbitrary dimension, as with VAEs, would also allow us to train a single, shared latent space for multiple molecules. This latent space would only need to be fine-tuned for individual molecules or may even work out of the box.

As for the second approach, we have investigated how a neural network can be trained to select the optimal shooting frame for transition path sampling. We have introduced two different architectures to extend the context and, with it, improve on the state-of-the-art. For this, we proposed a contextual neural network and a transformer-based architecture that can use the whole trajectory. Our evaluation shows that both architectures outperform a uniform selection, but only the contextual neural network yields benefits over a standard neural network. We believe that the additional complexity introduced by a transformer will only pay off for larger systems where it is worth to invest more time into training.

We also see potential in improving the current simulation-based approaches by pre-training a molecule-independent transformer architecture, which can then be fine-tuned for individual systems. This could speed up the sampling of transition paths, primarily when we rely on an expressive description of the molecules, for example, produced with graph neural networks.

Overall, transition path sampling is an exciting and crucial problem that could greatly benefit from advances in machine learning. Especially in the setting of latent space TPS, further research might be able to alleviate current drawbacks.

A. Hyperparameters

In the following, we will list for all models and kernels the hyperparameters that were used for training and evaluation respectively.

Parameter	**Search Space**
w_{ML}	[0, 0.25, 0.5, 0.75, **1**]
w_{KL}	[**0**, 0.25, 0.5, 0.75, 1]
Batch Size	[64, 128, 256, 512, **1024**]
Representation	[**Internal**, External]
Learning Rate	[1×10^{-3}, 1×10^{-4}, 5×10^{-4},1×10^{-5}]
Base Distribution	[Gaussian, **Gaussian-Uniform**]
Learning Rate Schedule	[None, **Cosine**]
Warmup Duration	[0, 500, **1000**

Table A.1.: **Hyperparameter of Boltzmann generator.**

Parameter	$C_5 \leftrightarrow \alpha_R$	$\alpha_R \leftrightarrow C_7$
Base Distribution	[**Gaussian**, Gaussian-Uniform]	[**Gaussian**, Gaussian-Uniform]
σ^2	[0.01, 0.05, **0.1**, 0.15, 0.2]	[0.01, **0.05**, 0.1, 0.15, 0.2]

Table A.2.: **Hyperparameter for Gaussian noise kernel.**

Parameter	$C_5 \leftrightarrow \alpha_R$	$\alpha_R \leftrightarrow C_7$
Base Distribution	[**Gaussian**]	[**Gaussian**]
L Steps	[2, 5, **10**, 20]	[2, 5, **10**, 20]
Step Size ϵ	[0.001, 0.005, **0.01**, 0.05, 0.1]	[0.001, **0.005**, 0.01, 0.05, 0.1]

Table A.3.: **Hyperparameter for Hamiltonian kernel.**

Parameter	$C_5 \leftrightarrow \alpha_R$	$\alpha_R \leftrightarrow C_7$
Base Distribution	[**Gaussian**]	[**Gaussian**]
α	$[1 \times 10^{-5}, 1 \times 10^{-3}, \mathbf{1 \times 10^{-2}}]$	$[1 \times 10^{-5}, 1 \times 10^{-3}, \mathbf{1 \times 10^{-2}}]$
Initialize with n paths *	[10, 15, **25**, 50]	[10, 15, **25**, 50]

Table A.4.: **Hyperparameter for Gaussian process kernel.** Values marked with * have only been used where applicable.

Parameter	**Values**
Weights per layer	55, 31, 17, 10
ResNet block	2
Output Dimension	1
Activation	ELU
Dropout	0.1

Table A.5.: **Hyperparameter for base neural network.** These numbers are used to compare the three different approaches for the neural network, the contextual neural network, and the self-attention-based approach.

Parameter	**Search Space**
Layers	[1, 2, 3, 4, 5, **6**, 7, 8]
Heads	[**1**, 5]
Dimension Feed Forward	[16, 32, **64**, 128, 256, 512, 1024, 2048]
Dropout	[0.01, 0.05, 0.10, 0.15, 0.20, **0.25**, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55]
Activation	[**ReLU**, GELU]

Table A.6.: **Hyperparameter of transformer-based approach.**

B. Visualization of Transitions

We visualize transitions between $C_5 \leftrightarrow \alpha_R$ with an overall transition time of $1.6ps$.

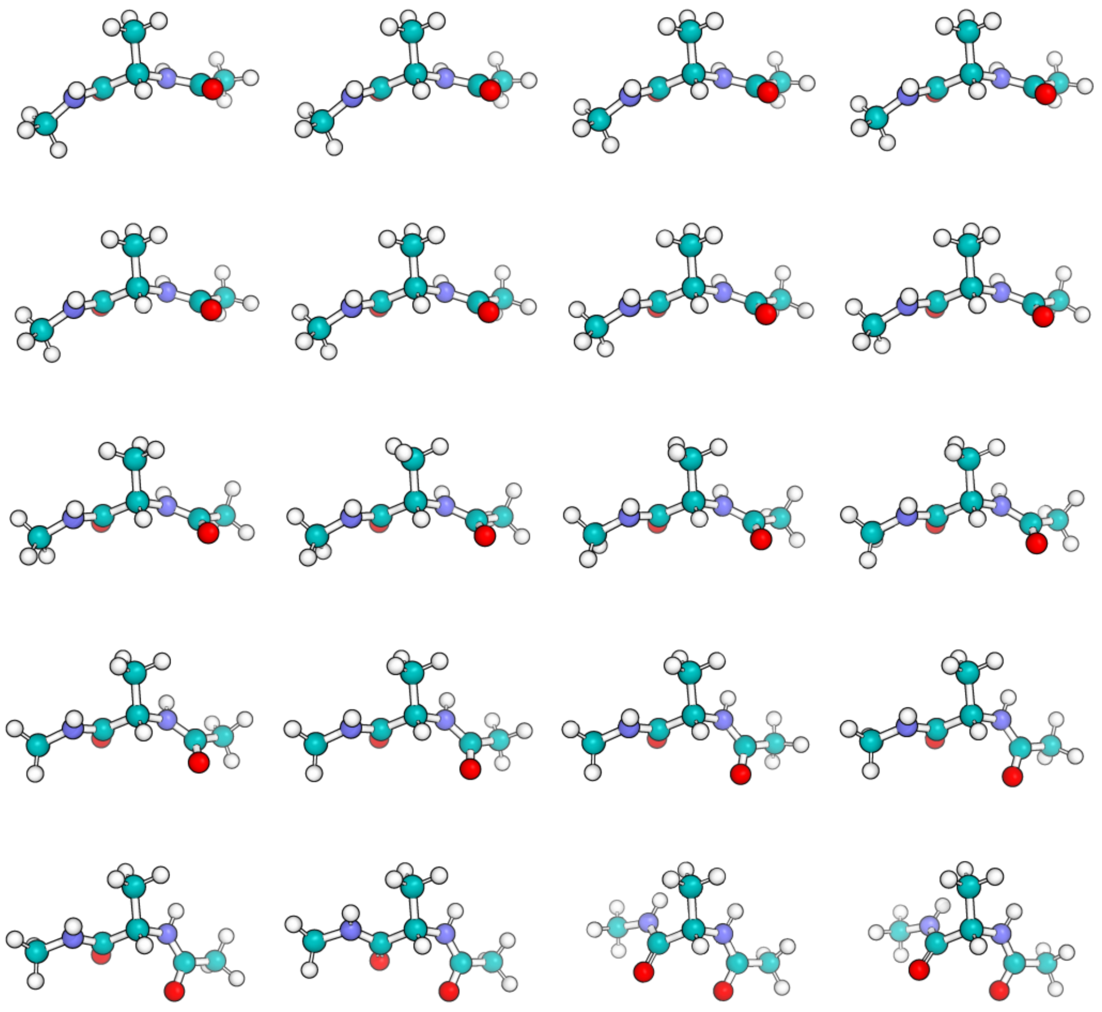

Figure B.1.: A linear translation in latent space between the states $C_5 \leftrightarrow \alpha_R$.

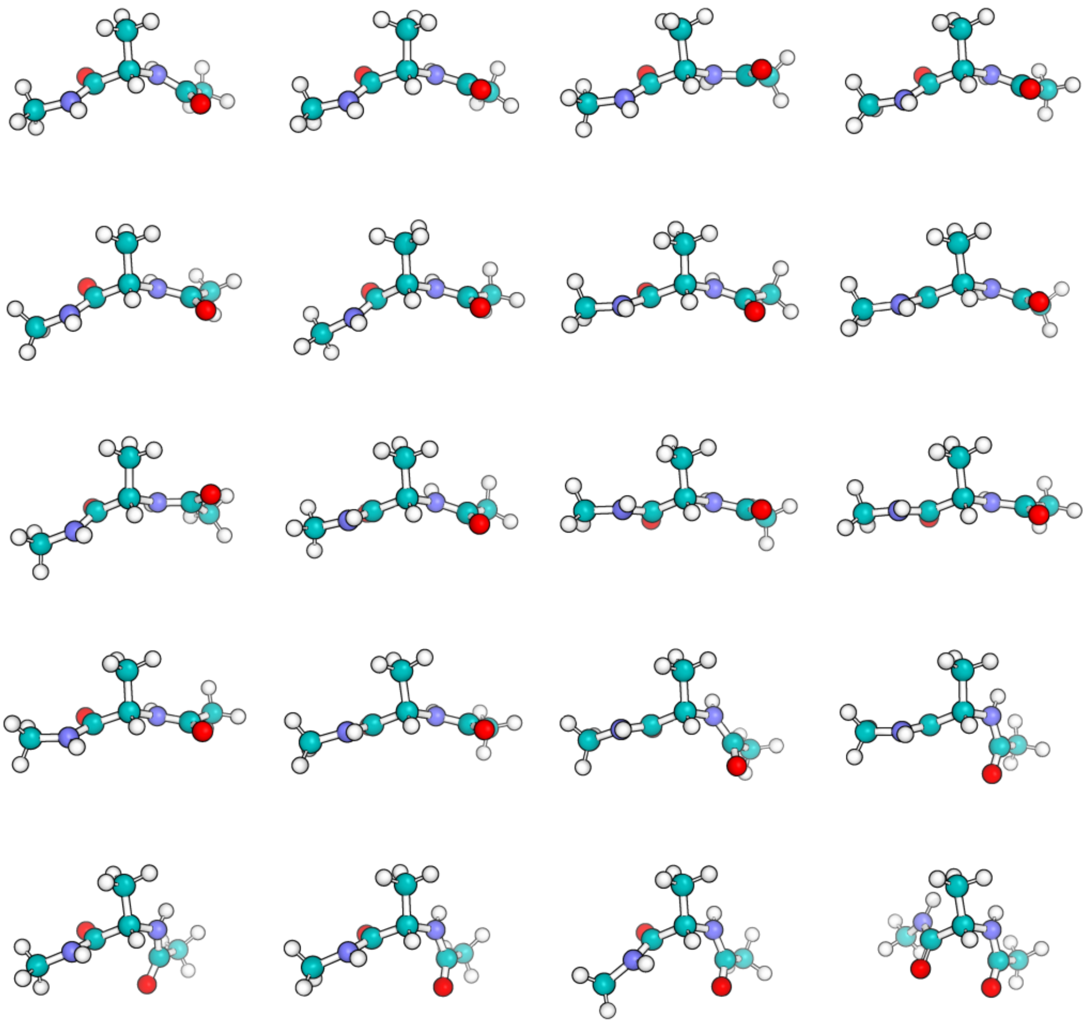

Figure B.2.: A transition path between $C_5 \leftrightarrow \alpha_R$ sampled with the Gaussian noise latent space proposal kernel.

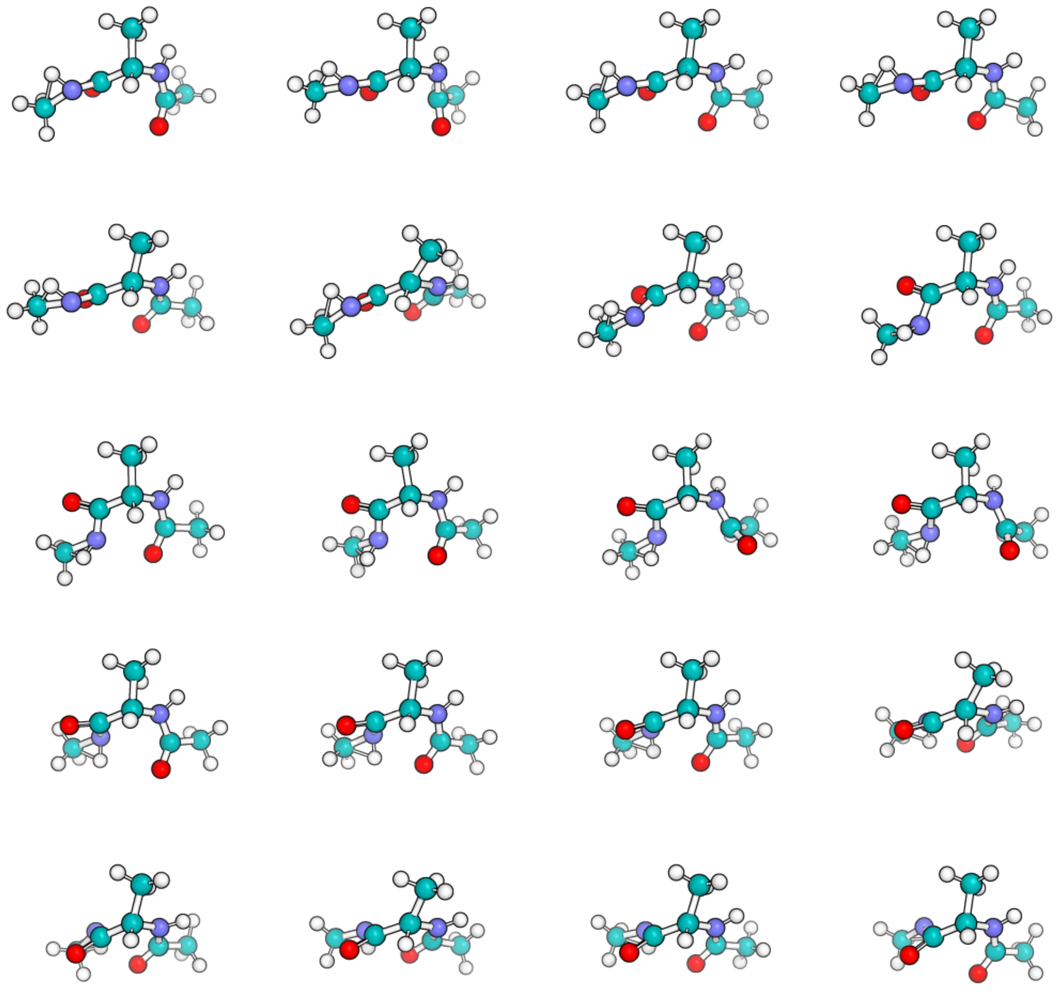
Figure B.3.: A transition path between $C_5 \leftrightarrow \alpha_R$ sampled with two-way shooting.

Bibliography

S. Andersson and E. F. van Dishoeck. Photodesorption of water ice. *Astronomy &; Astrophysics*, 491(3):907–916, October 2008.

Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and Computing*, 18(4):343–373, Dec 2008.

Katsuhiko Ariga, Jonathan P. Hill, and Qingmin Ji. Layer-by-layer assembly as a versatile bottom-up nanofabrication technique for exploratory research and realistic application. *Physical Chemistry Chemical Physics*, 9(19):2319, 2007.

Göran Aronsson, Ann-Christin Brorsson, Lena Sahlman, and Bengt-Harald Jonsson. Remarkably slow folding of a small protein. *FEBS Letters*, 411(2-3):359–364, July 1997.

Jodi E. Basner and Steven D. Schwartz. How enzyme dynamics helps catalyze a reaction in atomic detail: a transition path sampling study. *Journal of the American Chemical Society*, 127(40):13822–13831, September 2005.

Robert B. Best and Gerhard Hummer. Reaction coordinates and rates from transition paths. *Proceedings of the National Academy of Sciences*, 102(19):6732–6737, 2005.

Vladimir I. Bogachev. *Measure theory*. Springer, Berlin, 2007.

Peter G. Bolhuis and Christoph Dellago. Trajectory-based rare event simulations, September 2010.

Peter G. Bolhuis and David W. H. Swenson. Transition path sampling as markov chain monte carlo of trajectories: Recent algorithms, software, applications, and future outlook. *Advanced Theory and Simulations*, 4(4), March 2021.

Peter G. Bolhuis, David Chandler, Christoph Dellago, and Phillip L. Geissler. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annual Review of Physical Chemistry*, 53(1):291–318, October 2002.

Ludwig Boltzmann. *Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten*. 1868.

M. Bonitz, T. Dornheim, Zh. A. Moldabekov, S. Zhang, P. Hamann, H. Kählert, A. Filinov, K. Ramakrishna, and J. Vorberger. Ab initio simulation of warm dense matter. *Physics of Plasmas*, 27(4), April 2020.

E.E. Borrero and Christoph Dellago. Avoiding traps in trajectory space: Metadynamics enhanced transition path sampling. *The European Physical Journal Special Topics*, 225 (8-9):1609–1620, July 2016.

Tomáš Bučko, Lubomir Benco, Jürgen Hafner, and János G. Ángyán. Monomolecular cracking of propane over acidic chabazite: An ab initio molecular dynamics and transition path sampling study. *Journal of Catalysis*, 279(1):220–228, April 2011.

Gilbert William Castellan. *Physical Chemistry*. Addison-Wesley, Reading, Mass, 3rd ed edition, 1983.

Ramon Crehuet and Martin J. Field. A transition path sampling study of the reaction catalyzed by the enzyme chorismate mutase. *The Journal of Physical Chemistry B*, 111 (20):5708–5718, May 2007.

I. Csiszar. *I*-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, 3(1):146 – 158, 1975.

Christoph Dellago. *Transition Path Sampling and the Calculation of Free Energies*, pages 249–276. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

Christoph Dellago and Peter G. Bolhuis. *Transition Path Sampling Simulations of Biological Systems*, pages 291–317. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

Christoph Dellago and Peter G. Bolhuis. Transition path sampling and other advanced simulation techniques for rare events. *Advances in Polymer Science*, 221(1):167 – 233, 2009.

Christoph Dellago, Peter G. Bolhuis, and David Chandler. Efficient transition path sampling: Application to lennard-jones cluster rearrangements. *The Journal of Chemical Physics*, 108(22):9236–9245, June 1998a.

Christoph Dellago, Peter G. Bolhuis, Félix S. Csajka, and David Chandler. Transition path sampling and the calculation of rate constants. *The Journal of Chemical Physics*, 108(5):1964–1977, 02 1998b.

Christoph Dellago, Peter G. Bolhuis, and Phillip L. Geissler. *Transition Path Sampling Methods*, pages 349–391. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Alex Dickson. Mapping the ligand binding landscape. *Biophysical Journal*, 115(9): 1707–1719, November 2018.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jacob D Durrant and J Andrew McCammon. Molecular dynamics simulations and drug discovery. *BMC Biology*, 9(1), October 2011.

Weinan E, Weiqing Ren, and Eric Vanden-Eijnden. Finite temperature string method for the study of rare events. *The Journal of Physical Chemistry B*, 109(14):6688–6693, 2005.

Peter Eastman, Jason Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Chaya D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, 13(7):e1005659, July 2017.

Paul Embrechts and Marius Hofert. A note on generalized inverses. *Mathematical Methods of Operations Research*, 77(3):423–432, 2013.

Fernando A Escobedo, Ernesto E Borrero, and Juan C Araque. Transition path sampling and forward flux sampling. applications to biological systems. *Journal of Physics: Condensed Matter*, 21(33):333101, July 2009.

Sebastian Falkner, Alessandro Coretti, Salvatore Romano, Phillip Geissler, and Christoph Dellago. Conditioning normalizing flows for rare event sampling, 2023.

Loris Felardos, Jérôme Hénin, and Guillaume Charpiat. Designing losses for data-free training of normalizing flows on boltzmann distributions, 2023.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4181–4192. Curran Associates, Inc., 2021.

Yina Gu, Da-Wei Li, and Rafael Brüschweiler. NMR order parameter determination from long molecular dynamics trajectories for objective comparison with experiment. *Journal of Chemical Theory and Computation*, 10(6):2599–2607, May 2014.

Thomas A Halgren and Wolfgang Damm. Polarizable force fields. *Current Opinion in Structural Biology*, 11(2):236–242, April 2001.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

Graeme Henkelman and Hannes Jónsson. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *The Journal of Chemical Physics*, 113(22):9978–9985, 12 2000.

Graeme Henkelman, Blas P. Uberuaga, and Hannes Jónsson. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *The Journal of Chemical Physics*, 113(22):9901–9904, 12 2000.

Matthew Hoffman, Pavel Sountsov, Joshua V. Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport, 2019.

Scott A. Hollingsworth and Ron O. Dror. Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143, September 2018.

Ferry Hooft, Alberto Pérez de Alba Ortíz, and Bernd Ensing. Discovering collective variables of molecular transitions via genetic algorithms and neural networks. *Journal of Chemical Theory and Computation*, 17(4):2294–2306, March 2021.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, September 1933.

Tingjun Hou, Junmei Wang, Youyong Li, and Wei Wang. Assessing the performance of the MM/PBSA and MM/GBSA methods. 1. the accuracy of binding free energy calculations based on molecular dynamics simulations. *Journal of Chemical Information and Modeling*, 51(1):69–82, November 2010.

Hendrik Jung, Kei ichi Okazaki, and Gerhard Hummer. Transition path sampling of rare events by shooting from the top. *The Journal of Chemical Physics*, 147(15), August 2017.

Hendrik Jung, Roberto Covino, A. Arjun, Christian Leitold, Christoph Dellago, Peter G. Bolhuis, and Gerhard Hummer. Machine-guided path sampling to discover mechanisms of molecular self-organization. *Nature Computational Science*, 3(4):334–345, April 2023.

Jarek Juraszek and Peter G. Bolhuis. Rate constant and reaction coordinate of trp-cage folding in explicit water. *Biophysical Journal*, 95(9):4246–4257, November 2008.

Ivo Kabelka, Radim Brožek, and Robert Vácha. Selecting collective variables and free-energy methods for peptide translocation across membranes. *Journal of Chemical Information and Modeling*, 61(2):819–830, February 2021.

Youbin Kim, Jinsup Lee, Min Sun Yeom, Jae Won Shin, Hyungjun Kim, Yi Cui, Jeffrey W. Kysar, James Hone, Yousung Jung, Seokwoo Jeon, and Seung Min Han. Strengthening effect of single-atomic-layer graphene in metal–graphene nanolayered composites. *Nature Communications*, 4(1), July 2013.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4743–4751, Red Hook, NY, USA, 2016. Curran Associates Inc.

Serdal Kirmizialtin, Virginia Nguyen, Kenneth A. Johnson, and Ron Elber. How conformational dynamics of DNA polymerase select correct substrates: Experiments and simulations. *Structure*, 20(4):618–627, April 2012.

Serdal Kirmizialtin, Kenneth A. Johnson, and Ron Elber. Enzyme selectivity of HIV reverse transcriptase: Conformations, ligands, and free energy partition. *The Journal of Physical Chemistry B*, 119(35):11513–11526, August 2015.

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2021.

Jonas Köhler, Andreas Krämer, and Frank Noe. Smooth normalizing flows. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Jonas Köhler, Michele Invernizzi, Pim De Haan, and Frank Noe. Rigid body flows for sampling molecular crystal structures. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17301–17326. PMLR, 23–29 Jul 2023.

Maksim Kuznetsov and Daniil Polykovskiy. MolGrow: A graph normalizing flow for hierarchical molecular generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):8226–8234, May 2021.

Ben Leimkuhler and Charles Matthews. *Molecular Dynamics*. Springer International Publishing, 2015.

Tony Lelièvre, Geneviève Robin, Innas Sekkat, Gabriel Stoltz, and Gabriel Victorino Cardoso. Generative methods for sampling transition paths in molecular dynamics. *ESAIM: Proceedings and Surveys*, 73:238–256, 2023.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, page 9119 – 9130, 2020.

Ying Li, Shan Tang, Martin Kröger, and Wing Kam Liu. Molecular simulation guided constitutive modeling on finite strain viscoelasticity of elastomers. *Journal of the Mechanics and Physics of Solids*, 88:204–226, March 2016.

Qinghua Liao. Enhanced sampling and free energy calculations for protein simulations. In *Computational Approaches for Understanding Dynamical Systems: Protein Folding and Assembly*, pages 177–213. Elsevier, 2020.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, 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.

K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, October 2011.

Tianyi Liu, Weihao Gao, Zhirui Wang, and Chong Wang. Pathflow: A normalizing flow generator that finds transition paths. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 1232–1242. PMLR, 01–05 Aug 2022.

Alexander D. Mackerell. Empirical force fields for biological macromolecules: Overview and issues. *Journal of Computational Chemistry*, 25(13):1584–1604, July 2004.

Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. Sampling via measure transport: An introduction. In *Handbook of Uncertainty Quantification*, pages 1–41. Springer International Publishing, 2016.

S. Mazevet, L. A. Collins, N. H. Magee, J. D. Kress, and J. J. Keady. Quantum molecular dynamics calculations of radiative opacities. *Astronomy &; Astrophysics*, 405(1):L5–L9, June 2003.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

Laurence I. Midgley, Vincent Stimper, Javier Antorán, Emile Mathieu, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Se(3) equivariant augmented coupling flows, 2023a.

Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, and José Miguel Hernández-Lobato. Bootstrap your flow. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2022.

Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. In *International Conference on Learning Representations*, 2023b.

L. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters*, 72(23):3634–3637, June 1994.

Radford M. Neal. Mcmc using hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.

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

Helena Pais and Jirina R. Stone. Exploring the nuclear pasta phase in core-collapse supernova matter. *Physical Review Letters*, 109(15), October 2012.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Yong-Hyun Park, Mingi Kwon, Jaewoong Choi, Junghyo Jo, and Youngjung Uh. Understanding the latent space of diffusion models through the lens of riemannian geometry, 2023.

Matthew D. Parno and Youssef M. Marzouk. Transport map accelerated markov chain monte carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Fernando Perez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE International Symposium on Information Theory*, pages 1666–1670, 2008.

Jay W. Ponder and David A. Case. Force fields for protein simulations. In *Protein Simulations*, pages 27–85. Elsevier, 2003.

Sara L. Quaytman and Steven D. Schwartz. Reaction coordinate of an enzymatic reaction revealed by transition path sampling. *Proceedings of the National Academy of Sciences*, 104(30):12253–12258, July 2007.

G.N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7(1):95–99, July 1963.

Zhonghao Rao, Shuangfeng Wang, and Feifei Peng. Molecular dynamics simulations of nano-encapsulated and nanoparticle-enhanced thermal energy storage phase change materials. *International Journal of Heat and Mass Transfer*, 66:575–584, November 2013.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

Dhiman Ray, Enrico Trizio, and Michele Parrinello. Deep learning collective variables from transition path ensemble. *The Journal of Chemical Physics*, 158(20), may 2023.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, New York, NY, 2. ed edition, 2004. ISBN 978-0-387-21239-5.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Christopher N. Rowley and Tom K. Woo. Generation of initial trajectories for transition path sampling of chemical reactions with ab initio molecular dynamics. *The Journal of Chemical Physics*, 126(2), January 2007.

Suwipa Saen-oon, Sara Quaytman-Machleder, Vern L. Schramm, and Steven D. Schwartz. Atomic detail of chemical transformation at the transition state of an enzymatic reaction. *Proceedings of the National Academy of Sciences*, 105(43):16543–16548, October 2008.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A Generalized Representer Theorem. In G. Goos, J. Hartmanis, J. van Leeuwen, David Helmbold, and Bob Williamson, editors, *Computational Learning Theory*, volume 2111, pages 416–426. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

Daniele Selli, Salah Eddine Boulfelfel, Philipp Schapotschnikow, Davide Donadio, and Stefano Leoni. Hierarchical thermoelectrics: crystal grain boundaries as scalable phonon scatterers. *Nanoscale*, 8(6):3729–3738, 2016.

Nita Dilawar Sharma, Jasveer Singh, Aditi Vijay, K. Samanta, S. Dogra, and A. K. Bandyopadhyay. Pressure-induced structural transition trends in nanocrystalline rare-earth sesquioxides: A raman investigation. *The Journal of Physical Chemistry C*, 120(21):11679–11689, May 2016.

David E. Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror, Michael P. Eastwood, Joseph A. Bank, John M. Jumper, John K. Salmon, Yibing Shan, and Willy Wriggers. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, October 2010.

Daniel Sheppard, Rye Terrell, and Graeme Henkelman. Optimization methods for finding minimum energy paths. *The Journal of Chemical Physics*, 128(13), April 2008.

Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Information Science and Statistics. Springer, New York, 1st ed edition, 2008.

Vincent Stimper, David Liu, Andrew Campbell, Vincent Berenz, Lukas Ryll, Bernhard Schölkopf, and José Miguel Hernández-Lobato. normflows: A pytorch package for normalizing flows. *Journal of Open Source Software*, 8(86):5361, 2023.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding, 2021.

David W. H. Swenson, Jan-Hendrik Prinz, Frank Noe, John D. Chodera, and Peter G. Bolhuis. OpenPathSampling: A Python framework for path sampling simulations. 1. Basics. *Journal of Chemical Theory and Computation*, 15(2):813–836, 2019a.

David W. H. Swenson, Jan-Hendrik Prinz, Frank Noe, John D. Chodera, and Peter G. Bolhuis. OpenPathSampling: A Python framework for path sampling simulations. 2. Building and customizing path ensembles and sample schemes. *Journal of Chemical Theory and Computation*, 15(2):837–856, 2019b.

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

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

Michalis K. Titsias. Learning model reparametrizations: Implicit variational inference by fitting mcmc distributions, 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Patrick Varilly and David Chandler. Water evaporation: A transition path sampling study. *The Journal of Physical Chemistry B*, 117(5):1419–1428, January 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon,

U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Loup Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98–103, Jul 1967.

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 09–11 May 2016b. PMLR.

Li Xi, Manas Shah, and Bernhardt L. Trout. Hopping of water in a glassy polymer studied via transition path sampling and likelihood maximization. *The Journal of Physical Chemistry B*, 117(13):3634–3647, March 2013.

Guangji Xu and Hao Wang. Study of cohesion and adhesion properties of asphalt concrete with molecular dynamics simulation. *Computational Materials Science*, 112: 161–169, February 2016.

Xiaolei Zhu, Keiran C. Thompson, and Todd J. Martínez. Geodesic interpolation for reaction pathways. *The Journal of Chemical Physics*, 150(16):164103, 04 2019.