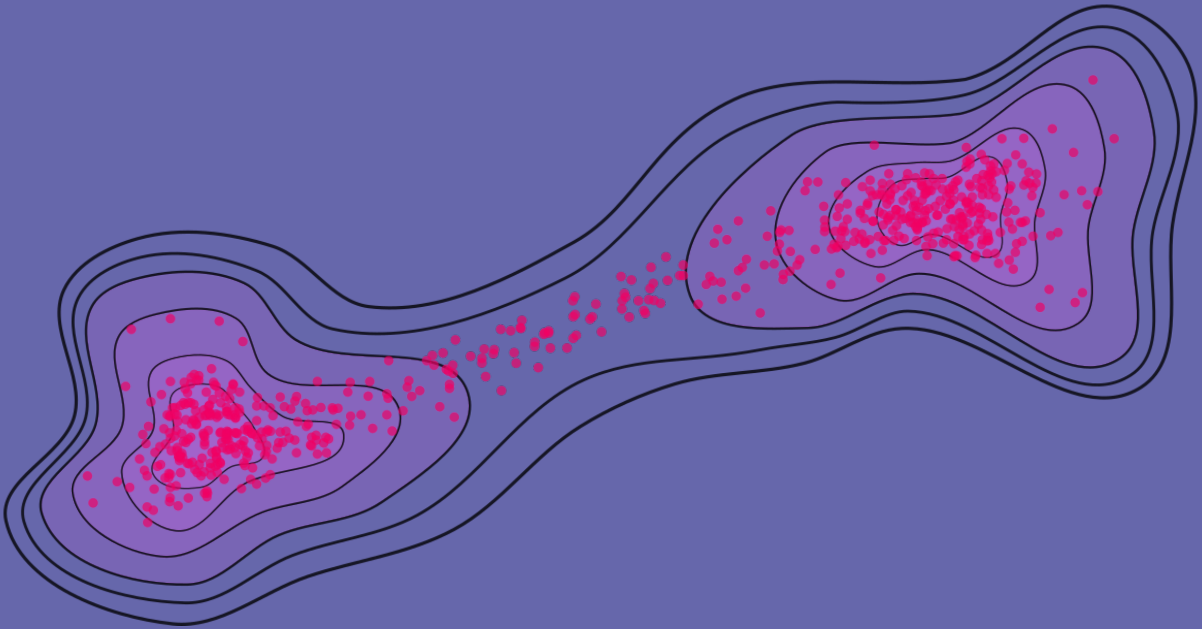
Machine Learning and Molecular Dynamics

From A to B and back



Enrico Trizio

December 28, 2023

Università di Milano-Bicocca

Istituto Italiano di Tecnologia

Department of Materials Science

Atomistic Simulations Group

PhD program in Materials Science and Nanotechnology
· XXXVI Cycle ·

Machine Learning and Molecular Dynamics

Candidate:
Enrico Trizio
mat. 807082

Supervisor:
Prof. Michele Parrinello

Coordinator:
Prof. Francesco Montalenti

Academic Year 2022/2023

Abstract

In this Thesis, we apply a combination of machine learning (ML) and enhanced sampling techniques to extend the scope of molecular dynamics (MD) simulations. One of the main limitations of MD is related to the time scale that standard simulations can cover. Most relevant processes indeed belong to the category of the so-called rare events, as they are characterized by several long-lived metastable states separated by large free energy barriers, which result in kinetic bottlenecks. The purpose of enhanced sampling methods is to alleviate this limitation and reduce the mismatch between real and simulated time scales. This is often done by adding external biasing potentials aimed at accelerating the dynamics of the process. Such potentials are defined as functions of a small set of collective variables (CVs), which are, in turn, functions of the atomic coordinates and should encode the relevant degrees of freedom of the system. The determination of proper CVs is of the utmost importance for these methods to be effective, and in the last years, it has been proposed to apply ML techniques to their design in a data-driven way.

In this regard, we present the Deep Targeted Discriminant Analysis (Deep-TDA) method, in which the CVs are extracted via a classification criterion from information limited to the metastable states. We also explore the option of including information from the transition path ensemble into this framework to further improve the quality of the results. Moreover, these and many other methods from the literature were included in the `mlcolvar` library we created to provide a unified framework for developing and testing data-driven CVs. We also propose a method in which ML tools are used to build a bias potential to stabilize the region around the transition state (TS) to increase its sampling. This is done by approximating the behavior of the committor function of the system with a classifier-like CV of the Deep-TDA type and expressing the bias in terms of the gradient of such a function, thus allowing its localization on the TS region.

Finally, we showcase the impact of the synergy of ML and MD simulations, studying the structures and mechanisms involved in the λ-transition of liquid sulfur and its peculiar chemistry. This liquid-liquid phase transition has attracted much interest in the last century as it is associated with a living polymerization of eight-membered crown-shaped sulfur rings into long linear polymers. However, despite previous studies, a detailed picture of this phenomenon and the underlying processes is still missing. To improve in this sense, we combine enhanced sampling simulations based on data-driven CVs and ML interatomic potentials. This way, we perform fast simulations of quantum mechanical accuracy that allow us to finally shed light on this puzzling process.

Riassunto

In questa Tesi, abbiamo applicato una combinazione di metodi di machine learning (ML) e tecniche di campionamento potenziato per estendere la portata delle simulazioni di dinamica molecolare (MD). Una delle principali limitazioni della MD è legata alle scale di tempo che possono essere coperte con questo tipo di simulazioni. Infatti, la maggior parte dei processi rilevanti in natura appartiene alla categoria dei cosiddetti eventi rari, essendo caratterizzati da una serie di stati metastabili separati da alte barriere energetiche che impediscono transizioni spontanee tra di essi. Lo scopo dei metodi di campionamento potenziato è quello di alleviare questa limitazione e ridurre così la discrepanza tra le scale di tempo dei processi reali e quelle raggiungibili nelle simulazioni. In molti casi, questo risultato viene ottenuto applicando al sistema un potenziale esterno con la finalità di accelerarne la dinamica. Tali potenziali, solitamente, sono definiti come funzioni di poche variabili collettive (CV), le quali sono invece funzioni delle coordinate atomiche e devono codificare le informazioni relative ai principali gradi di libertà del sistema. L'identificazione di CV adeguate è fondamentale per poter applicare in maniera efficace questi metodi di campionamento potenziato e negli ultimi anni è stato proposto di impiegare tecniche di ML per poterle determinare in maniera semi-automatica partendo dai dati ottenuti nelle simulazioni.

A questo proposito, presentiamo il metodo Deep Targeted Discriminant Analysis (Deep-TDA) in cui le CVs sono estratte sulla base di un principio di classificazione a partire da informazioni limitate ai soli stati metastabili. Esploriamo anche la possibilità di includere in questo approccio informazioni relative alla regione nei dintorni dello stato di transizione, al fine di migliorare le qualità del risultato finale. Questi e diversi altri metodi sono parte della libreria `mlcolvar` che abbiamo creato per fornire una piattaforma comune in modo da promuoverne l'utilizzo e ulteriori sviluppi. Inoltre, presentiamo anche un metodo in cui gli strumenti del ML sono impiegati per costruire un potenziale esterno per stabilizzare la regione attorno allo stato di transizione in modo da favorirne il campionamento. Per far questo, approssimiamo il comportamento della committor function del sistema con una CV simile ad un classificatore, nello stesso spirito di Deep-TDA, ed esprimiamo il potenziale esterno in termini del gradiente di tale funzione, permettendo così di localizzarne l'effetto nella zona di transizione.

Per concludere, mostriamo l'impatto della sinergia tra ML e simulazioni MD studiando le strutture e i meccanismi coinvolti nella transizione λ osservata nello zolfo liquido e la sua particolare chimica. Questa transizione di fase liquido-liquido ha infatti attratto molta attenzione negli ultimi anni, essendo associata con la polimerizzazione di anelli ad otto membri di zolfo per formare lunghe

catene polimeriche lineari. Tuttavia, nonostante studi teorici precedenti, manca ancora una descrizione dettagliata di questo fenomeno e dei processi coinvolti. Per contribuire in tal senso, abbiamo combinato simulazioni basate su metodi di campionamento potenziato con CV e potenziali per le interazioni atomiche ottenuti con tecniche di ML. In questo modo, è stato possibile ottenere simulazioni veloci di un'accuratezza paragonabile a quella dei metodi basati su una descrizione quantistica delle interazioni interatomiche, permettendoci di gettare finalmente luce su questo misterioso processo.

Contents

List of Abbreviations

AE	Auto Encoder
AIMD	*Ab initio* molecular dynamics
CV	Collective variable
DFT	Density functional theory
FES	Free energy surface
LDA	Linear discriminant analysis
ML	Machine learning
MPL	Machine learning (interatomic) potential
MetaD	Metadynamics
MD	Molecular dynamics
NN	Neural network
OPES	On-the-fly probability enhaced sampling
PES	Potential energy surface
TDA	Targeted discriminant analysis
TPE	Transition path ensemble
TPI	Transition path informed
TS	Transition state

Introduction

Atomistic simulations, and notably Molecular Dynamics[1,2] (MD), are powerful tools that act as computational microscopes capable of shedding light on the mechanisms of many physical-chemical processes. For this reason, their applications in modern science have been constantly increasing.

One of MD's well-known limitations is the time scale that standard simulations can cover[3]. Indeed, despite much algorithmic and hardware progress, many processes of relevance, like crystallization, chemical reactions, or protein folding, remain out of the reach of present-day simulation capabilities. Almost all the interesting processes in nature indeed belong to the category of the so-called *rare events* and are characterized by timescales that are completely out of the practical reach of standard simulation techniques. This mismatch between real and simulated time scales is related to the fact that many relevant dynamical processes are characterized by several long-lived metastable states separated by large energetic barriers, which result in kinetic bottlenecks that limit our simulations[4].

This has motivated the development of enhanced sampling (ES) methods aimed at accelerating the simulated dynamics and speeding up the sampling[5]. In the last decades, such methods have gained increasing popularity because of their ability to alleviate the limitations of standard MD simulations and thus extend their scope to real life processes taking place on otherwise computationally unaffordable timescales. To do so, many ES methods, such as the time-honored Umbrella sampling[6] and Metadynamics[7,8], are based on the addition of external *biasing* potentials to the system, which are meant to smooth the underlying energetic landscape to remove the undesired and detrimental kinetic bottlenecks. Such bias potentials are typically defined as functions of a small set of *collective variables* (CVs), which are, in turn, functions of the atomic coordinates. If the CVs are appropriately chosen, the bias added will favor transitions between one metastable state and another, eliminating kinetic bottlenecks and speeding up sampling[5]. Moreover, besides offering a powerful computational tool, the CVs also provide a concise representation that is precious for an understanding of the physical process. Given these assumptions, it is easy to understand that a great effort has been devoted to developing methods for designing efficient bias potentials and collective variables.

In most recent years, it has been proposed to apply to this purpose modern *data-driven* techniques from the field of *machine learning*[9] (ML). Generally speaking, ML is a field of study oriented to developing techniques that are aimed at automatically learning from data and making predictions without having been explicitly programmed to do so. Lately, thanks to the constantly improving computational power and the dizzying increase in data availability, this field has become ubiquitous in science as well as everyday life, and a plethora of methods have been accordingly developed. Following this rapid development, many different scientific fields have already benefited from the cross-contamination of such methods[10,11]. These include the broader and generic field of atomistic simulations[12], for example, for the development of accurate yet computer-efficient interaction potentials[13,14] and also the more specific field of the enhanced sampling[15–17].

In this Thesis, we present our contribution to this collective effort with a particular focus on the development of such methods but also providing examples of their potential impact on applications to challenging real systems.

A large part of this work will be devoted to presenting methods for the data-driven design of efficient CVs for enhanced sampling[18–20]. In this regard, we will present the `mlcolvar` library[20] (short for Machine Learning COLlective Variables) we built for the easy development and dissemination of machine-learning CVs. This will also serve as an assist to overview some of the recent developments in this field. Among them, we will discuss in much more detail the Deep Targeted Discriminant Analysis (Deep-TDA) method[18], in which the CV is expressed as the output of a Neural Network (NN) that is optimized according to a classification criterion. In turn, the input of such an NN is meant to be a set of physical descriptors (i.e., distances, angles, coordination numbers..). In standard Deep-TDA, this descriptor set is supposed to be collected from the metastable states only. However, in the Transition Path Informed (TPI-Deep-TDA) variant[19] of the method, information from the transition path ensemble is also included in the training set to improve the specific quality of the CV at the cost of a slightly more laborious computational effort.

Alongside the theoretical framework of such methods and their application to prototypical systems, we will also present an application to the much more challenging case of the λ-transition in liquid sulfur[21,22] and the underlying peculiar chemistry. This liquid-liquid phase transition has indeed attracted much interest in the last century as it is associated with a living polymerization of eight-membered crown-shaped sulfur rings into long linear polymers, which results in an anomalous behavior of physical properties, such as the viscosity and the heat capacity. Even if, in this case, the focus will be more on the application itself, the importance of method development will still be evident. To account for the complex chemistry of the liquid phases of sulfur, we will indeed design a set of topological descriptors combining elements of graph theory[23] and machine learning. In addition, to study such a complex system, we will also make use of state-of-the-art machine-learning interatomic potentials[24], showing the importance of CV-based enhanced sampling simulations for their optimization in an active learning framework[25–28]. Armed with these tools, we will finally shed light on the puzzling mechanisms involved in the λ-transition, which remained uncertain despite previous *ab initio*-based theoretical studies.

Last but not least, we will move back to pure method development, but, in this case, our goal will be the design of a bias potential for enhanced sampling from a new perspective, which is still in its infancy but whose initial results are so promising that we include it here nonetheless. In this approach, rather than promoting transitions between the metastable states by *filling* the basins of the energy landscape[7], we aim at stabilizing the high-energy region of the phase space associated with the transition state region (TS) to increase its sampling. To do so, we will approximate the behavior of the committor function[29] of our system with a classifier-like CV of the Deep-TDA type and we will express our bias as a function of the gradient of such a function, thus allowing the localization on the TS region. Upon proper optimization, the application of such a bias allows us to sample that delicate region of the phase space as effortlessly as the metastable states in the unbiased scenario, providing precious data that can be crucial to the study of complex systems' reactive mechanisms and for effectively training machine-learning potentials or CVs of the TPI-Deep-TDA type.

This Thesis is divided into two parts and is structured as follows. The first part aims at providing the essential theoretical background of the computational techniques that are relevant to the development of the rest of the work. In Chapter 1, we briefly introduce the basics of MD simulations and some essential elements of Statistical Mechanics that are necessary for our purposes. Chapter 2 introduces the topic of rare events and the theoretical background of ES methods and CVs, which will be the central topic of the whole Thesis. In Chapter 3, we concisely present some essential elements of ML that are used to develop the methods presented in the following chapters, with a particular focus on using NNs as universal interpolators.

The second part is devoted to presenting and discussing the results obtained during this Thesis work, with the data-driven CVs being the *fil rouge* of this part, both in terms of methods and of applications. In Chapter 4, we provide a general overview of the data-driven approach to the design of CVs. This allows us to introduce the `mlcolvar` library in Chapter 5, where we present the structure and the motivations of the code alongside some practical examples and a brief review of methods developed in this field during the last years. Among these approaches, we then focus on classifier-like CVs in Chapter 6, where we present the Deep-TDA method and its TPI variant, accompanied by several examples on prototypical systems and a short note on their use in combination with ES methods for kinetics calculations. In Chapter 7, we apply the Deep-TDA model to a much more challenging system as we discuss our study of the complex phenomena involved across the λ-transition of liquid sulfur. Finally, in Chapter 8, we present our new biasing scheme for the extensive sampling of the transition state region and some prototypical applications of such an approach.

Chapter 1

Introduction to Molecular Dynamics

In the last decades, atomistic simulations have become increasingly important in many fields of Science, ranging from Physics and Chemistry to Materials Science and Biology. The value of this approach lies in the possibility of acting as a practical bridge between theoretical and experimental approaches.

Among these methods, Molecular Dynamics (MD) plays a central role thanks to its ability to simulate the time of evolution of groups of atoms, starting from a description of the microscopic interactions and the fundamental laws of classical mechanics. In this introduction, we will limit to a concise introduction to this powerful technique, dwelling only on those aspects that will be more relevant to the development of this Thesis. In particular, we will introduce those concepts that are more relevant to the field of enhanced sampling and that benefit more from the contributions of machine learning approaches. For any further details and a more comprehensive formulation of the following concepts, we refer the Reader to the vast available literature in this field[1,2,4], to which this introduction is largely inspired, and to the references provided in the specific sections.

1.1 The experimental approach of Molecular Dynamics

Molecular Dynamics represents arguably the best example of the so-called *in silico* experimental techniques, as it closely resembles the approach of real experiments in many aspects. In real experiments, if we are interested in studying the behavior of a property of interest for a given sample under certain conditions, we *measure* this quantity using a measuring instrument, e.g., a thermometer. In practice, the measurement is performed and averaged over a certain interval of time, and the statistical error in the result depends on the length of this interval, i.e., in many cases, provided some controlled experimental conditions, the longer the measurement, the better the statistics.

In MD, we basically do the same thing. We prepare our *sample* composed of N particles, we determine a form for the interaction between them, we let them evolve according to classical dynamics until the properties we are interested in no longer change with time, and then we perform our measures. In this case, the last step is not done using measuring instruments but through equations that relate the properties of interest to the positions and momenta of the particles. For example, instead of a thermometer, we could use the equipartition theorem from statistical mechanics to estimate the temperature of our system from the average kinetic energy per degree of freedom α

$$\langle\frac{1}{2}mv_\alpha^2\rangle = \frac{1}{2}k_B T \tag{1.1}$$

In practice, we can then measure the total kinetic energy of the system at a time t and divide it by the number of degrees of freedom N_f ($N_f = 3N - 3$ for fixed total momentum) to obtain the *istantaneous* temperature $T(t)$

$$T(t) = \sum_{i=1}^{N} \frac{m_i v_i(t)^2}{k_B N_f} \tag{1.2}$$

From this expression, it is clear that, as the momenta of the atoms will fluctuate over time, so will the temperature. Thus, to obtain a statistically accurate estimate, we should average over many of such fluctuations in time.

1.2 The Molecular Dynamics algorithm

The easiest way to briefly introduce MD is to look at the essential components of an MD code. One of the strengths of MD is, indeed, the relative simplicity of its logic. In the simplest case, the MD procedure reduces to a short list of operations which involves only a few steps:

1. **Initialization**, performed only once at the beginning
    - We read and set the parameters that determine the simulation's conditions, e.g., number of atoms, timestep, temperature, pressure...
    - We initialize the system, i.e., we set the initial positions and velocities of the particles
2. **Evolution in time**, repeated until the desired simulation length is reached
    - We compute the forces on all the particles
    - We compute the displacements of all the particles by integrating Newton's equations of motion

3. **Computation of physical quantities**, performed after the central loop is completed

In the following sections, we will expand on these components to provide the overall essential information. For an exhaustive treatment, we refer the Reader to the relevant literature[1,2].

1.2.1 Initialization

The initialization of an MD simulation clearly depends on the system we are interested in studying. For example, for the study of a crystalline solid, the initial positions of the atoms have to be chosen such that they respect the crystalline structure of the material and avoid overlaps between the different atoms. Similarly, the simulation parameters are dictated by the thermodynamical conditions that we want to reproduce in the simulation (e.g., temperature and pressure).

The nature and the complexity of the system also influence the choice of the description of the atomic interaction we may want to adopt. The possibilities and details in this sense are countless and will thus be discussed in more detail in Sec. 1.5. Here, we only anticipate that this choice heavily determines the overall cost of the simulation and its scope. For example, the study of chemical reactions in which bonds are broken and formed requires a highly accurate and expensive description based on quantum mechanics, and this poses severe limitations on the size and timescale one can simulate. On the other hand, the folding of proteins can be studied using fast parametric force fields that can tackle such huge systems even on macroscopic timescales.

In this stage, the *size* of the system also needs to be set, that is, the number of atoms in our simulation cell. This could be a delicate choice that, when it comes to complex systems, requires some critical thinking to find a balance between computational cost and accuracy. Ideally, we would surely like to simulate the evolution of macroscopic real-sized systems over long time periods. However, as expected, the computational cost of the simulations increases with their timespan and with the number of simulated atoms. For this reason, in simulations, we should ideally find the best setup to ensure that the simulated system is accurately representative of the real one while the computational costs are minimized. For example, in the case of homogenous bulk systems, it is common practice to apply *periodic boundary conditions* on our simulation cell to mimic the presence of an infinite bulk surrounding it. In this framework, our N-particle cell is treated as the primitive cell of an infinite periodic lattice. Thus, any given particle i will not only interact with the other particles in the simulation cell but also with their *periodic images*. This allows mimicking the bulk environment at an affordable computational cost, provided that we ensure to avoid the self-interaction issue, which may easily lead to artifacts.

One last consideration needs to be done concerning the initialization of the velocities of the particles. Typically, they are either inherited from a previous simulation or randomly extracted from a pre-assigned distribution (e.g., a normal or Maxwell-Boltzmann distribution). In the second case, we need to ensure that the total momentum of the particles is zero (i.e., the momentum of the center of mass of the system is zero) and that the kinetic energy of the system is consistent with the initial temperature we want to simulate. This last constraint is, however, not so strict as the temperature will change during the simulation anyway, and the system will eventually *equilibrate* itself during the earlier stages of the simulation.

1.2.2 Evolution in time

This is the core of an MD program and, at each cycle, accounts for the evolution of the system over an interval of time Δt. This set of operations is then cyclically repeated until the simulated time reaches the desired length.

Force computation This step is by far the slowest of any MD code, as we compute the forces acting on all the atoms in our system starting from the description of the interatomic interaction between them we decided to adopt for the simulation (see Sec. 1.5).

In general, given an interatomic interaction potential $U(\mathbf{x})$, the force $\mathbf{F}_i$ acting on the atom i with coordinates $\mathbf{x}_i$ is proportional to the derivatives of the potential with respect to its position

$$\mathbf{F}_i = -\frac{\partial U(\mathbf{x})}{\partial \mathbf{x}_i} \tag{1.3}$$

The computational cost associated with the computation of this quantity clearly depends on the form of $U(\mathbf{x})$ and the more complex it is, the more expensive the computation. For example, we can consider the simplest case of a two-body interatomic interaction $U_{ij}(\bar{x}_{ij})$ that for two atoms i and j only depends only on their scalar distance $\bar{x}_{ij}$ (e.g., Lennard-Jones or Morse potentials). In order to compute the forces in a system of N particles, we would need to compute all the $N(N-1)/2$ pairwise distances between them. In practical terms, this means that the cost of computing the forces, even in this simple system, already scales as N^2. As one can easily imagine, this scaling law dramatically worsens as we move to complex potentials and even more if we have to rely on quantum mechanics.

Integrating the equations of motion In MD, the motion of the atoms is treated in a *classical* way, in the sense that the trajectory of the ions in time obeys Newton's motion laws of classical mechanics.

This means that, once we have computed the force $\mathbf{F}(t)$ and given the position $\mathbf{x}(t)$ of an atom at time t, its evolution over an interval of time Δt is simply ruled by the equation of motion

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{\mathbf{F}(t)}{2m}\Delta t^2 \tag{1.4}$$

where $\mathbf{v}(t)$ is the velocity at time t.

In practice, the numerical integration of Newton's equation is performed using slightly different but equivalent algorithms. The most popular choice in this sense is the so-called Verlet algorithm, but other integration algorithms are available, for example, the popular Leap-Frog algorithm, and we refer the Reader to the related literature[1] for more details.

In the Verlet formulation, the new positions at a timestep $t + \Delta t$ are computed as

$$\mathbf{x}(t + \Delta t) = 2\mathbf{x}(t) - \mathbf{x}(t - \Delta t) + \frac{\mathbf{F}(t)}{m}\Delta t^2 + \mathcal{O}(\Delta t^4) \tag{1.5}$$

in which the error on the new positions is of order Δt^4, and they are computed without using explicitly the velocities. However, they can still be recovered from the positions as

$$\mathbf{v}(t) = \frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t - \Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^2) \tag{1.6}$$

From the example of the Verlet algorithm, we can see that integrating the equations of motion using a finite timestep Δt induces an error proportional to the size of the timestep itself, with larger timesteps leading to larger errors. An intuitive explanation for such a trend is related to the fact that the assumption that the forces $\mathbf{F}$ are constant over the time interval Δt becomes less reasonable as this increases and completely wrong if the chosen timestep is larger than the characteristic time of the fastest motion of the system. In general, in practice, the choice of the integration interval is thus the result of a trade-off between computational accuracy and speed and depends on the system at hand.

1.2.3 Computation of physical quantities

The final goal of a MD simulation is to compute some relevant properties of interest about our many-body system, possibly allowing a direct comparison with experimental results. As one can imagine, the spectrum of such properties is wide and heterogeneous, and not all of them are computed with the same approach.

For instance, some thermodynamical quantities can be directly expressed as the average values of functions of the atomic positions and momenta. We already discussed the example of the computation of the system's temperature from its average kinetic energy (see Eq. 1.2), and the pressure can be computed with a similar approach using the virial equation for pairwise additive interactions.

However, not all the thermodynamical properties can be computed this way. For example, the entropy S, the Helmholtz free energy H, and the Gibbs free energy G cannot be expressed as a simple average of functions of the atomic positions and momenta, but they shall rather be computed using the tools of statistical thermodynamics (see Sec. 1.3).

Another worth-mentioning quantity that can be easily obtained from simulations is the radial distribution function $g(r)$. This is related to the local structure of the simulated system and, most interestingly, can be compared with the experimental results as it can be expressed as the Fourier transform of the structure factor $S(k)$, which can be experimentally measured using diffraction techniques. In Chapter 7, for example, we will proceed this way while studying liquid sulfur.

All the quantities mentioned above can be defined as *static* equilibrium quantities, and, in principle, they could also be computed using different classes of simulations, e.g., Monte Carlo simulations. However, one of the main strengths of MD is the possibility of also accessing *dynamical* equilibrium quantities, e.g., diffusion, of the system, as it can trustfully reproduce the actual time evolution of the system and not only its statistical behavior. Noteworthily, this feature of MD also allows observing the real dynamics of the system, for example, for the study of reaction mechanisms, as we will show in the case of liquid sulfur.

1.3 An essential compendium of statistical mechanics

In previous sections, we have seen that what we can directly measure from our simulations on the atomistic level, e.g., positions and momenta of the particles, is quite different from what is actually measured in the experiments. Indeed, experimental data are typically related to average properties, averaged both on the measurement time and on the large number of particles in the sample.

In order to compare simulations and experiments, it is thus necessary to define which kind of averages we should aim to compute on our results. In practical terms, this is done by the (powerful) means of statistical mechanics. In the following sections, we shall provide a brief and essential introduction to the basic elements of this branch of Physics that are most functional for the remainder of this Thesis. In this spirit, this is not meant to be a formal and rigorous derivation of such principles, which can be easily found elsewhere[1,4,30], but rather a quick contextualization in the statistical mechanics framework of some concepts that will be relevant in the next chapters.

1.3.1 A surprisingly gentle introduction with quantum mechanics

It may sound surprising, but when it comes to introducing the basic laws of statistical mechanics, starting from the language of quantum mechanics turns out to be one of the easiest paths. Moreover, what we need for our introduction does not go much further than a few basic concepts in this field.

Multiplicity of energy eigenstates First of all, we need to recall that a quantum mechanical system can be found in different states. For simplicity, we will only consider *energy eigenstates*, which are those states that are eigenvectors of the Hamiltonian H of the system. Using Dirac notation[31], this means that for a state $|i\rangle$ of such a kind, we have that $H|i\rangle = E_i|i\rangle$, in which E_i is the energy of the state $|i\rangle$. The degeneracy of such energy levels is usually small for simple systems with just a few degrees of freedom. This means that, for a given energy $E = E'$, the number of states $|j\rangle$ such that $H|j\rangle = E'|j\rangle$ is small. On the other hand, as the complexity and the size of the system increase, so does such degeneracy. This means that if we take a system with a fixed number of particles N and volume V, for a given energy E we will have a number $\Omega(E)$ of eigenstates with that energy in which the system is *equally likely* to be found.

Given these assumptions, we can now imagine having two *weakly interacting* systems, meaning that they can exchange energy with each other, and the total energy is given by the sum of their individual contributions $E = E_1 + E_2$. In this case, for a given value of E_1, the total degeneracy of the system is given by the product of the single multiplicities

$$\Omega(E_1, E_2) = \Omega(E_1) \times \Omega(E_2) \tag{1.7}$$

For convenience, we can directly move to the logarithm of the multiplicity in order to have an additive measure of the subsystems' degeneracy

$$\ln \Omega(E_1, E - E_1) = \ln \Omega(E_1) + \ln \Omega(E - E_1) \tag{1.8}$$

The concept of thermal equilibrium If we now let our subsystems exchange energy, what will be the *most likely* distribution of energies? We know that every energy state of the total system is equally likely, as E is fixed, but the number of eigenstates associated with a certain distribution of the subsystems' energy depends on the corresponding values of E_1 and E_2. In other terms, we want to find the E_1 that maximizes the function $\ln \Omega(E_1, E - E_1)$, which can be found by setting its derivatives with respect to E_1 to zero

$$\left(\frac{\partial \ln \Omega(E_1, E - E_1)}{\partial E_1} \right)_{N,V,E} = 0 \tag{1.9}$$

that corresponds to

$$\left(\frac{\partial \ln \Omega_1(E_1)}{\partial E_1} \right)_{N_1,V_1} = \left(\frac{\partial \ln \Omega_2(E_2)}{\partial E_2} \right)_{N_1,V_2} \tag{1.10}$$

which we can also write for convenience in a more compact form

$$\beta(E_1, V_1, N_1) = \beta(E_1, V_1, N_1) \qquad \text{with} \quad \beta(E, V, N) = \left(\frac{\partial \ln \Omega(E, V, N)}{\partial E}\right)_{N,V} \tag{1.11}$$

If we imagine placing all the energy in one subsystem, such that $E = E_1$, there will be an energy flow from system 1 to system 2 until the above condition is satisfied and the $\ln \Omega$ of the total system is maximized. This happens when there is no net exchange of energy between the two systems, which are said to be in *thermal equilibrium*.

A microscopic definition of entropy and temperature At this point, we can reconnect ourselves to classical thermodynamics[32,33], from which we know that a system is in thermal equilibrium when its entropy S is maximized. This suggests that we can express the entropy of our total system $S(N, V, E)$ as related to $\ln \Omega$

$$S(N, V, E) = k_B \ln \Omega(N, V, E) \tag{1.12}$$

where k_B is the so-called Boltzmann factor.

From classical thermodynamics, we also know that thermal equilibrium between two systems is reached when they are at the same temperature $T_1 = T_2$. This macroscopic equilibrium condition can be readily compared with our microscopic criterion $\beta_1 = \beta_2$, which suggests that we can relate β (see Eq. 1.11)to the absolute temperature T. Indeed, if we consider the classical thermodynamics definition of temperature

$$\frac{1}{T} = \left(\frac{\partial S}{\partial E}\right)_{V,N} \tag{1.13}$$

and we apply it with the microscopic definition of entropy from Eq.1.12, we obtain that

$$\beta = \frac{1}{k_B T} \tag{1.14}$$

which provides us with a microscopic definition of the absolute temperature of our system.

The Boltzmann distribution Now, we can consider the case of a system A, which is in contact with a large heat bath B. If we assume that they are weakly interacting, as we did in the previous example, the total energy of the system will be $E = E_A + E_B$. If we now suppose to consider a specific quantum state i of system A that has energy E_i, it follows that the thermal bath degeneracy will be $\Omega(E_B) = \Omega(E - E_A)$.

It is clear that the probability P_i to find system A in state i will be related to the multiplicity of the heat bath

$$P_i = \frac{\Omega_B(E - E_i)}{\sum_j \Omega_B(E - E_j)} \tag{1.15}$$

If we expand $\Omega_B(E - E_i)$ around $E_i = 0$, which is a reasonable condition if B is much larger than A, and we exploit Eqs. 1.12 and 1.13, we obtain

$$P_i = \frac{\exp(-E_i/k_bT)}{\sum_j \exp(-E_j/k_BT)} \tag{1.16}$$

which is the familiar expression of the well-known *Boltzmann distribution* of the energies of a system at temperature T.

Average energy and the partition function Having an expression for the energy distribution of our system at temperature T, we can now compute its average energy $\langle E \rangle_T$ as a weighted average over all the states i

$$\langle E \rangle_T = \sum_i E_i P_i = \frac{\sum_i E_i \exp(-E_i/k_BT)}{\sum_j \exp(-E_j/k_BT)} \tag{1.17}$$

which can be simplified by introducing the *partition function*

$$Z = \sum_i \exp(-E_i/k_BT) \tag{1.18}$$

to obtain

$$\langle E \rangle_T = -\frac{\partial \ln \sum_i \exp(-E_i/k_BT)}{\partial 1/k_BT} = -\frac{\partial \ln Z}{\partial 1/k_BT} \tag{1.19}$$

Quite conveniently, the partition function can also be used to write Eq. 1.16 in a more concise format as

$$P_i = \frac{\exp(-E_i/k_bT)}{Z} \tag{1.20}$$

At this point, we recall the relation from classical thermodynamics that establishes the relation between the Helmholtz free energy F and the average energy E of a system

$$E = \frac{\partial F/T}{\partial 1/T} \tag{1.21}$$

From here, we can see that F is directly related, up to a constant, to the partition function of our system

$$F = -k_BT \ln Z = -k_BT \ln\left(\sum_i \exp(-E_i/k_BT)\right) \tag{1.22}$$

leading us to one of the fundamental expressions in equilibrium statistical mechanics, which we will largely use in the following chapters.

1.3.2 From the quantum scenario to classical statistical mechanics

In the previous section, we approached statistical mechanics from the point of view of quantum mechanics, leading to an expression for the probability of finding the system at a given temperature T in an eigenstate with energy E_i (see Eq. 1.16).

If we know this probability distribution and we take into account a generic observable A, we can thus obtain the *thermal average* of A in our system

$$\langle A \rangle = \frac{\sum_i \exp(-E_i/k_B T)\, \langle i|\, A\, |i\rangle}{\sum_j \exp(-E_j/k_B T)} = \frac{\sum_i \langle i|\exp(-H/k_B T) A\, |i\rangle}{\sum_j \langle j|\exp(-H/k_B T)\, |j\rangle} \tag{1.23}$$

where $\langle i|\, A\, |i\rangle$ is the expectation value of observable A in the eigenstate $|i\rangle$.

Unfortunately, this expression is of little practical utility if we consider that it would require solving the Scrhödinger equation for our system, which is clearly an unrealistic task for all but the simplest systems. Luckily, Eq. 1.23 can be simplified to a more practical form in the classical limit, in which fine quantum effects are considered negligible. A complete derivation of the expression for the classical thermal average can be found in Ref.[1]. Here, we only report the final result

$$\langle A \rangle = \frac{\int d\mathbf{p}^N d\mathbf{r}^N \exp\{-\beta\left[\sum_i p_i^2/(2m_i) + U(\mathbf{r}^N)\right]\} A(\mathbf{r}^N, \mathbf{p}^N)}{\int d\mathbf{p}^N d\mathbf{r}^N \exp\left\{-\beta\left[\sum_j p_j^2/(2m_j) + U(\mathbf{r}^N)\right]\right\}} \tag{1.24}$$

This expression looks similar to Eq. 1.23, but, importantly, it is formulated in terms of the positions $\mathbf{r}^N$ and momenta $\mathbf{p}^N$ of the particles in our systems, which we can access in our simulations.

1.3.3 Ergodicity

In the previous sections, we introduce the computation of average quantities for a given system in terms of averages over all its possible quantum states, which is usually referred to as *ensemble average* in the statistical mechanics jargon. However, this approach does not match much of what we do in real experiments or in MD simulations. Indeed, in the experiments, we measure the quantity of interest over a certain interval of time to obtain an average value. In other terms, for a given quantity A, what we measure is a *time average*

$$\overline{A} = \lim_{t\to\infty} \frac{1}{t} \int_0^t dt' A(t') \tag{1.25}$$

in which we assume that if we consider a time interval long enough, any effect of the initial conditions is negligible.

Similarly, in simulations, we aim to simulate the system's time evolution to average our calculations over a sufficiently long time. However, we cannot safely assume independence on the initial conditions in this case. In addition to time averages, we should indeed repeat our simulations starting from different initial configurations $\{\mathbf{r}^N(0), \mathbf{p}^N(0)\}$ and average over such a number of simulations.

$$\overline{A}(\mathbf{r}^N, \mathbf{p}^N) = \frac{\sum\limits_{\text{Initial conditions}} \left(\lim\limits_{t\to\infty} \frac{1}{t} \int_0^t dt' A(\mathbf{r}^N, \mathbf{p}^N, \mathbf{r}^N(0), \mathbf{p}^N(0), t') \right)}{\text{Number of initial conditions}} \tag{1.26}$$

If we consider the limiting case in which we sum over all the possible initial conditions that are compatible with the N,V, and E values of our system, we can replace the summation over the initial conditions with an integral. For a generic function $f(\mathbf{r}^N(0), \mathbf{p}^N(0))$ of the initial conditions we obtain

$$\frac{\sum\limits_{\text{Initial conditions}} f(\mathbf{r}^N(0), \mathbf{p}^N(0))}{\text{Number of initial conditions}} \quad \to \quad \frac{\int_E d\mathbf{r}^N d\mathbf{p}^N f(\mathbf{r}^N(0), \mathbf{p}^N(0))}{\int_E d\mathbf{r}^N d\mathbf{p}^N} \tag{1.27}$$

where the integration is restricted to the shell of constant energy E. This average on the *phase space* corresponds to the classical limit of the *ensemble* average we introduced in the previous section. At this point, we can switch the order of time and initial conditions averages

$$\overline{A(r)} = \lim_{t\to\infty} \frac{1}{t} \int dt' \langle A(r, \mathbf{r}^N(0), \mathbf{p}^N(0), t') \rangle_{NVE} \tag{1.28}$$

In this expression, we can soon realize that the ensemble average actually does not depend on the time t' due to the deterministic nature of our simulations. We can thus skip the time averaging in Eq. 1.28 to find

$$\overline{A(r)} = \langle A(r) \rangle_{NVE} \tag{1.29}$$

In other words, we found that if we are interested in studying the average behavior of an observable A, time averaging and ensemble averaging are equivalent in the long time limit. This concept is usually referred to as the *ergodic hypothesis* and assumes that if we wait for a sufficiently long period of time, our system will eventually visit all its microstates. In MD, we generally take the ergodic condition as true to compute averages from our simulations upon the condition that such simulations are sufficiently long to guarantee a meaningful sampling of the portion of the phase space we are interested in.

As a final note, we recall that the path we followed to arrive at Eq. 1.29 should be seen as a *plausible* motivation rather than a proper proof, which would have clearly been out of our interests in this work. Moreover, our derivation was obtained for the so-called NVE ensemble, in which, for our system, the number of particles N, the volume V, and the energy E were fixed. However, often, it can be more realistic and convenient to perform our simulations in different conditions, e.g., NVT and NPT ensembles. To do this, we need to introduce some new elements in our simulations toolbox, which we shall introduce in the following section.

1.4 Molecular Dynamics in different ensembles

In the previous section, we introduced some elements of statistical mechanics that are relevant to MD simulations in the so-called *microcanonical ensemble* (NVE fixed). Indeed, systems that evolve according to Newton's equations, as we do in MD, live in this ensemble.

However, if we think about experimental conditions, we soon realize that performing our simulations in other ensembles may be more reasonable and practical. For example, experiments are often performed at constant temperature, which leads us to the *canonical* (or NVT) ensemble or constant pressure, which brings us to the isobaric-isothermal ensemble (or NPT). In order to extend MD simulations to such ensembles and better mimic the experimental conditions, two approaches have been followed, which we will mention in our brief discussion in the following paragraphs. One is to introduce some stochastic dynamics to the systems, and the other is to modify the dynamical equations of the system with the so-called *extended Lagrangian* approach.

Constant temperature simulations A number of algorithms have been proposed to perform *meaningful* simulations at constant temperature, which go under the name of *thermostats*[34]. We specify *meaningful* because only fixing the temperature does not guarantee by itself that our simulation belongs to the canonical ensemble we want to sample. For example, rescaling the velocities of the particles at each step would lead to a perfectly constant kinetic temperature T_K (obtained using the equipartition theorem), which is actually in contrast with the fluctuations of T_K predicted from the Maxwell-Boltzmann distribution. In addition, a good thermostat should ensure the conservation of some quantity to check that we are actually sampling the correct ensemble and guarantee the ergodic conditions.

The first thermostat to be proposed was from Andersen[35], and it was based on a stochastic procedure, namely on a stochastic scattering of the particles with a thermal bath that would modify some of their velocities. Berendsen[36], on the other hand, proposed to add an additional equation to the equations of motion aimed at driving the system's kinetic energy toward a target value. Unfortunately, this method does not guarantee sampling of the canonical ensemble and should be avoided for production runs.

Both methods are simple and relatively stable but present some drawbacks, making other methods preferable in practice. One example is the Nose-Hoover thermostat[34], which relies on the extended Lagrangian approach. The idea in this framework, in a nutshell, is to increase the dimensionality of our system of N particles from $6N$ to $6N + 2$ and to solve the equations of motion to conserve the energy in the augmented space while the real $6N$-dimensional system is represented by the canonical ensemble. Another valid and popular alternative is the *stochastic velocity rescaling* thermostat[37]. This can be seen as an extension of Berendsen method in which an external force is added to enforce the correct kinetic energy distribution.

Constant pressure simulations To maintain a constant temperature in our simulations, we rescale the velocities of the particles, whereas when we aim for constant pressure simulations, we let the simulation box volume vary to keep the internal pressure, on average, constant.

Some of the ideas we presented for thermostats can also be inherited in the case of *barostats*, and often, the two algorithms are indeed very similar. This is the case, for example, of Berendsen barostat[36], in which pressure can be controlled by means of another additional equation aimed at driving the pressure toward the target value. The two algorithms, as one can imagine, also share the same limitations for what concerns an accurate sampling of the correct ensemble. In contrast, the formulation of Andersen barostat[35] is different from the thermostat as it belongs to the extended Lagrangian type. In this case, the volume is included as an independent dynamical variable in the phase space to mimic the effect of an external piston scaling the volume of the system. Andersen's formulation is suitable only for isotropic deformation of the simulation box, but it was later generalized by Parrinello and Raman[38] for the anisotropic case by allowing the cell to change not only its size but also its shape.

1.5 The description of interatomic interactions

From what we have discussed in the previous section, it is clear that the quality of an MD simulation and the reliability of its results crucially depend on an appropriate choice of the functional form $U(\mathbf{x})$ used to describe the interatomic interactions and from which the atomic forces are computed (see Eq. 1.3). As already anticipated, different *levels of theory* can be adopted in this sense, and, in general, a more accurate description corresponds to more expensive calculations and slower simulations. Luckily, not all systems require a highly accurate and expensive description based on quantum mechanics, but often approximated potentials are enough to accurately reproduce the dynamics of interest. In the following, we will provide a short and high-level summary of the possible description of the interatomic interactions and their typical use cases. A comprehensive discussion on this topic is indeed out of the scope of this work and can be found in the literature[3,39–41].

Ab initio methods Starting from the most accurate methods, we have the so-called *ab initio* or *first principles* approaches, which guarantee an extremely high accuracy as they rely on the solution of quantum mechanical problems[41,42]. Different approaches can be adopted in this sense, but the arguably most popular are the ones from the Density Functional Theory (DFT) family, in which the energy of the system is described as a functional of the electronic density. Unfortunately, the precision of these methods comes at an extremely high computational price, which severely limits the sizes and timescales that can be simulated using the so-called ab initio MD (AIMD) to some thousand atoms and some hundred picoseconds at most. For all these reasons, ab initio approaches are typically employed whenever an extremely accurate description is needed, for example, in the case of systems characterized by complex and reactive chemistry and/or whenever little, if none, information on the system is available.

Semi-Empirical methods A cheaper and much faster alternative to ab initio calculations is provided by the so-called *semi-empirical* methods. They share the same conceptual framework of ab initio methods, but they speed up the computation by neglecting some minor integrals and compensating this approximation with the addition of some empirically tuned parameters[43]. Like every other approximation, this approach can be successful only if the semi-empirical model retains the relevant physics to properly describe the system of interest. When this is the case, for instance, in the case of simple chemical reactions, the speed of simulations can be increased by orders of magnitude with respect to ab initio approaches.

Force Fields Despite these great gains in terms of computational cost, even semi-empirical methods are way too expensive or inefficient to simulate the dynamics of larger systems, such as bulk materials or biomolecules in a solvent. In such cases, one can revert to the wide category of the so-called *force field* potentials, in which the interatomic interactions are analytically described by parametric functions whose computation is fast and efficient. In this formulation, the interaction energy is expressed as the sum of several additive contributions that depend on a (possibly large) number of empirical parameters. The simplest example in this category is the well-known Lennard-Jones potential U_{LJ}

$$U_{LJ}(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{1.30}$$

in which the interaction energy between two atoms i and j only depends on their distance r_{ij} and on the parameters ϵ and σ. These parameters regulate the interaction's intensity and characteristic distance, respectively, and depend on the atomic species at hand. The total energy on atom i is then simply obtained as the sum of the interactions with all the atoms j

$$U^{i}_{LJ} = \sum_{j\neq i} V_{LJ}(r_{ij}) \tag{1.31}$$

In the same spirit, more complex force fields have been proposed, in which more complex terms, such as three-body or angular interactions, are added to improve the expressivity of the model. Examples in this regard range from the relatively simple Stilinger-Weber potential[44], which accurately reproduces the diamond structure of solid silicon with the addition of three-body interactions, to the more complicated force fields used in protein dynamics, such as AMBER[45] or CHARMM[46], or for the description of solvent interaction, for example, the TIP3P and TIP4P models[47] for water.

Thanks to their parametric formulations, force fields allow for huge speedups at the cost of sacrificing the accuracy of finer details. However, in many cases, such an accuracy may not be relevant. For example, in the case of protein folding, the process often involves only a conformational transformation without a major change in the chemical structure apart from hydrogen bonds.

In general, the choice of developing and using force fields is thus motivated by the need to have a trade-off between precision and simulation speed. Indeed, processes involving large numbers of atoms or characterized by long timescales would be completely unreachable with higher levels of theory[40].

Machine Learning Potentials From what we have said so far, it may seem that, in simulations, we will always find ourselves in this dilemma between accuracy and efficiency. However, in the last decade, following the pioneering work of Behler and Parrinello[13], a number of *machine learning interatomic potentials* have been developed to provide accuracies comparable to ab initio methods with a computational cost closer to the one of classical force fields. Such approaches are proving themselves to be game-changers in MD[12], and they have been applied to the study of many complex scenarios, from materials[25] to reactive chemical systems[26,48] or catalytic processes[27,28].

The general idea behind these approaches is to predict the energy and forces of the system of interest using some kind of machine-learning model (see Chapter 3) trained on (large) datasets of reference configurations labeled with the energies and forces obtained with DFT calculations. Many of such methods are based on the definition of some local environment descriptors, which are then fed as inputs to neural networks (NNs). This is the case of the classical example of Behler-Parrinello potentials[13], which employ atom-centered symmetry functions as fixed local descriptors and associate a different NN to each chemical species in the system, and of the DeepMD[24,49] potential we will use in Chapter 7 for the study of liquid Sulfur. In the DeepMD case, many different input descriptors are available[49], including the local frame descriptor, the two-body and three-body embeddings, and the attention-based descriptors[50] we employed for sulfur. Other interesting approaches for neural network potential are based on *graph neural networks*, in which a graph is used to represent the atomic structure. In the most intuitive way, each node of the graph is associated with an atom, and the edges (i.e., the graph connections) are defined on the basis of the interatomic distances. In such a framework, the information is propagated in the network using the *message-passing* scheme, in which the node values and edge features through the different layers are determined by the application of some convolutional filters, which are update functions that depend on the values of the neighboring elements of the graph. Examples in this sense are, for instance, SchNet[51], PhysNet[52], and NequIP[53], which has the additional peculiarity of being based on the recently developed equivariant networks[54].

However, these are just a few of the examples in a field that, in the last years, has been continuously and dramatically evolving. For this reason and considering that an extensive review of this class of approaches is out of the scope of this Thesis, we refer the Reader to the specific literature[12,14,55–57].

Regardless of the chosen architecture, the quality of this approach crucially depends on the quality of the dataset used for the training. First, and somehow trivially, the level of theory employed for the reference calculations has to guarantee a description of the process that is sufficiently accurate. Second, and most importantly, the reference configurations in the dataset shall span the portion of the phase space relevant to the process, also including configurations from the transition state region.

For this reason, in recent years, it has been proposed to apply to the training of machine learning potentials *active learning* strategies aimed at progressively extending the dataset to improve the overall quality of the potential. Some of such approaches, like FLARE[58], are based on on-the-fly estimates of the model uncertainty, triggering new DFT calculations for those configurations in which it exceeds a given threshold and adding such configurations to the training set. In another approach that has become popular in the last years, the active learning procedure is boosted with the help of enhanced sampling techniques (see Sec. 2.2), which are aimed at improving the phase space sampling[26–28]. A practical example of such an approach will be provided in our study of liquid sulfur reported in Chapter 7.

Chapter 2

Molecular Dynamics and rare events

In the previous chapter, we introduced the basics of Molecular Dynamics, discussing how it can be used to simulate the time evolution of many particle systems at the atomistic scale. Unfortunately, such a small scale does not only apply to the size of the systems but also to the timescales we can access with our simulations. Moreover, with respect to such timescales, most natural processes are considered *rare events*, as they occur orders of magnitude slower than that.

Apparently, this may look catastrophic for any application of MD simulations to real systems. However, in the last decades, many *enhanced sampling* techniques have been developed to overcome this intrinsic limitation of standard MD and extend its scope to real-life processes, which we introduce in this chapter and will largely use in the rest of this Thesis.

2.1 The timescale problem

As we have shown in the previous sections, the MD algorithm is intrinsically sequential as the configuration at time $t + \Delta t$ depends on the previous one at time t (see Sec. 1.2). This means that the overall computational cost of a simulation scales linearly with the number of steps. In addition, the timesteps typically used to have a meaningful integration of the equation of motion are of the order of femtoseconds, meaning that a nanosecond-long calculation would require roughly some 10^6 iterations[3]. Even with modern computers and fast force fields, these conditions typically restrict the accessible timescale to the order of milliseconds at most, with such limitations becoming more severe if we increase the complexity of the system and/or if we use higher levels of theory.

Unfortunately, in contrast, several relevant microscopic natural processes occur on the millisecond or longer timescales, which are thus completely out of reach for standard MD simulations. This mismatch between real and simu-

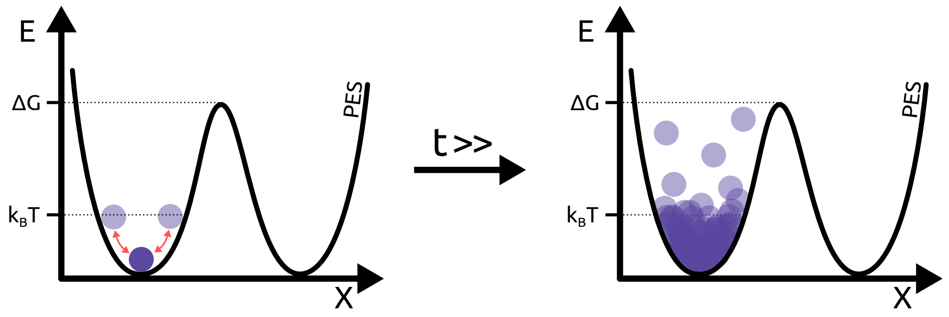


Figure 2.1: Schematic representation of the potential energy surface (PES) in a rare event scenario and the resulting limited sampling on a generic coordinate x. The presence of a large free energy barrier ΔG compared to the available thermal energy $k_B T$ limits the sampling to the bottom of the initial metastable state even after a long time.

lated timescales is related to the fact that many relevant dynamical processes are characterized by several metastable states separated by large energetic barriers, as depicted in Fig. 2.1. Indeed, when the energy needed to overcome these barriers is much larger than the thermal energy available in the system, we found kinetic bottlenecks, and the transitions between metastable basins become *rare events*, which occur on timescales way outside the practical reach of standard simulations[3–5].

Moreover, these limitations become even more severe if we consider that, in order to extract any significant results from our simulations, they should be supported by some statistics. Indeed, if we want to study a given process, the observation of a single transition has a limited value. For example, in the case of a chemical reaction, many reactive events should be observed before drawing any statistically meaningful conclusion.

The timescale problem is for sure one of the main problems of standard MD, and in the last decades, a great effort has been devoted to overcoming such limitations to extend the scope of simulations by accelerating the simulated dynamics. This led to the development of the so-called *enhanced sampling* methods, which we introduce in the following section.

2.2 Enhanced Sampling in Molecular Dynamics

As we have seen in the previous section, one of the main limitations of standard MD is related to the mismatch between the timescales on which many relevant processes occur and the ones we can simulate. In order to overcome this limitation, a number of *enhanced sampling* methods have been developed in the last decades, aimed at accelerating the simulated dynamics and thus extending the scope of simulations to real systems. In the following, we will provide a brief introduction to this class of methods and refer the Reader to the available literature[5,59–62] for further details.

In our discussion, we will focus on the methods in which the potential energy surface of the system is modified to remove kinetic bottlenecks by adding an external *bias potentials*. As we discussed above, these bottlenecks are related to the presence of large free energy barriers between the metastable states, and the aim of the bias potentials is to *lower* those barriers to facilitate the transitions. In practice, such bias potentials are typically expressed as functions of a few selected *order parameters* or *collective variables*, which are, in turn, functions of the atomic coordinates whose fluctuations are relevant in the process of interest.

2.2.1 Sampling in collective variable space

It is common both in Chemistry and Physics to formulate a description of complex problems in terms of a few selected variables. This is the case, for example, of Landau's order parameters in his theory of phase transitions[63]. The need for finding significant variables that can describe the collective behavior of a system is true both in the case of purely theoretical approaches, as in the case of statistical mechanics[64], and probably even more in the case of computational approaches. In the latter case, a concise description of the system is indeed of paramount importance to limit the computational costs while providing a meaningful and easily understandable representation of the system.

In practical terms, these *collective variables* (CVs) should be able to describe the key features of the chemical/physical behavior of the system of interest. Moreover, such CVs should be able to distinguish between all relevant metastable states and to encode the *slow modes* of the system, i.e., its slowest degrees of freedom. From a mathematical point of view, the CVs can be defined as functions, eventually nonlinear, of the atomic coordinates $\mathbf{r}$, i.e., $\mathbf{s}(\mathbf{r}) = (s_1(\mathbf{r}), s_2(\mathbf{r}) \dots s_d(\mathbf{r}))$. In the rest of this section, we will focus more on the theoretical framework of CV-based enhanced sampling, leaving the details about the CVs themselves and their choice to Sec. 2.4.

Having introduced our CVs, we can now consider their equilibrium distribution

$$P(\mathbf{s}) = \int d\mathbf{r}\delta[\mathbf{s} - \mathbf{s}(\mathbf{r})]P(\mathbf{r}) = \left\langle \delta[\mathbf{s} - \mathbf{s}(\mathbf{r})] \right\rangle \tag{2.1}$$

where $P(\mathbf{r})$ is the Boltzmann distribution $P(\mathbf{r}) = \frac{\exp(-\beta U(\mathbf{r}))}{Z}$ determined by a potential $U(\mathbf{r}$ in the coordinates space, $\beta = (k_B T)^{-1}$ is the inverse temperature, $Z = \int d\mathbf{r} \exp(-\beta U(\mathbf{r}))$ is the partition function and $\delta[\mathbf{s} - \mathbf{s}(\mathbf{r})]$ denotes a delta function centered in $\mathbf{s}$.

Following statistical mechanics (see Sec. 1.3), we can express the *free energy* *surface* (FES) in the CV space as the logarithm of the probability distribution

$$F(\mathbf{s}) = -\frac{1}{\beta} \ln P(\mathbf{s}) \tag{2.2}$$

Such a *low-dimensional* FES, as it is a function of a few CVs, is usually much smoother than the original and extremely rugged *high-dimensional* potential energy surface $U(\mathbf{r})$. Moreover, we recall that the free energy is defined up to an additive constant, meaning that the zero level can be shifted arbitrarily.

Upon an appropriate choice of the CVs, the FES contains all the relevant information about the related system. Indeed, it shows the location and relative stability of the different metastable states and the height of the free energy barriers that may separate them. For example, if we consider two metastable states A and B in our system, we can estimate the relative free energy difference between them from the probabilities of the two states

$$\Delta F_{A,B} = -\frac{1}{\beta} \ln \frac{\int_A d\mathbf{s} \exp[-\beta F(\mathbf{s})]}{\int_B d\mathbf{s} \exp[-\beta F(\mathbf{s})]} \tag{2.3}$$

where the subscripts indicate that the integration is restricted to the basins A and B, respectively. We have seen in Sec. 1.3, that if we assume ergodic sampling from our simulations, the FES can be *equivalently* computed as

$$F(\mathbf{s}) = -\frac{1}{\beta} \lim_{t\to\infty} \ln N(\mathbf{s}, t) \qquad \text{with} \quad N(\mathbf{s}, t) = \frac{\int_0^t dt' \delta[\mathbf{s} - \mathbf{s}(\mathbf{R})]}{\int_0^t dt'} \tag{2.4}$$

Here, $N(\mathbf{s}, t)$ is a normalized histogram of the data collected in a *standard*, or *unbiased*, simulation. Despite looking promising at first, Eq. 2.4 is rather useless in practice if we are limited to standard MD. Indeed, as we have seen, in many physical systems, unbiased sampling is extremely limited on the timescales that are computationally affordable and thus fails to be ergodic at all.

2.2.2 Enhanced sampling with bias potentials

Starting from the pioneering work of Torrie and Valleau[6], many *enhanced sampling* methods[60–62] have been developed to overcome the sampling limitations of standard simulations relying on the addition of an external *bias* potential V to the system. Such bias potential is expressed as a function of the CVs, i.e., $V = V(\mathbf{s})$, and is aimed at accelerating the dynamics of the process of interest and enhancing the sampling in the CVs space. It is important to note that, even if the bias explicitly depends on $\mathbf{s}$, it still implicitly depends on the atomic coordinates through the CVs themselves as $\mathbf{s} = \mathbf{s}(\mathbf{r})$.

Static bias potential The simplest scenario we can consider is the one in which we apply to our system a bias potential $V(\mathbf{s})$ that is constant with time. The presence of such a bias potential modifies the energetic landscape of the system, as we schematically depict in Fig. 2.2, thus resulting in a modified probability distribution P_V

$$P_V(\mathbf{s}) = \int d\mathbf{r}\delta[\mathbf{s}-\mathbf{s}(\mathbf{r})]P_V(\mathbf{r}) = \frac{\exp\{-\beta[F(\mathbf{s})+V(\mathbf{s})]\}}{\int d\mathbf{s}\exp\{-\beta[F(\mathbf{s})+V(\mathbf{s})]\}} \tag{2.5}$$

in which $P_V(\mathbf{r})$ is the Boltzmann distribution driven by the biased energy landscape

$$P_V(\mathbf{r}) = \frac{\exp[-\beta[U(\mathbf{r})+V(\mathbf{s}(\mathbf{r}))]]}{Z_V} \tag{2.6}$$

with its partition function $Z_V = \int d\mathbf{r}\exp[-\beta[U(\mathbf{r})+V(\mathbf{s}(\mathbf{r}))]]$.

Comparing Eq. 2.5 and Eq.2.1, we can readily see that the unbiased FES $F(\mathbf{s})$ can be obtained from the biased data, up to a constant, as

$$F(\mathbf{s}) = -\frac{1}{\beta}\log N_V(\mathbf{s}) - V(\mathbf{s}) = F_V(\mathbf{s}) - V(\mathbf{s}) \tag{2.7}$$

in which N_V is the histogram accumulated in the biased simulation and F_V is the (non-physical) free energy surface of the biased system. This tells us that, given a proper sampling of the phase space in the biased simulations, we can recover

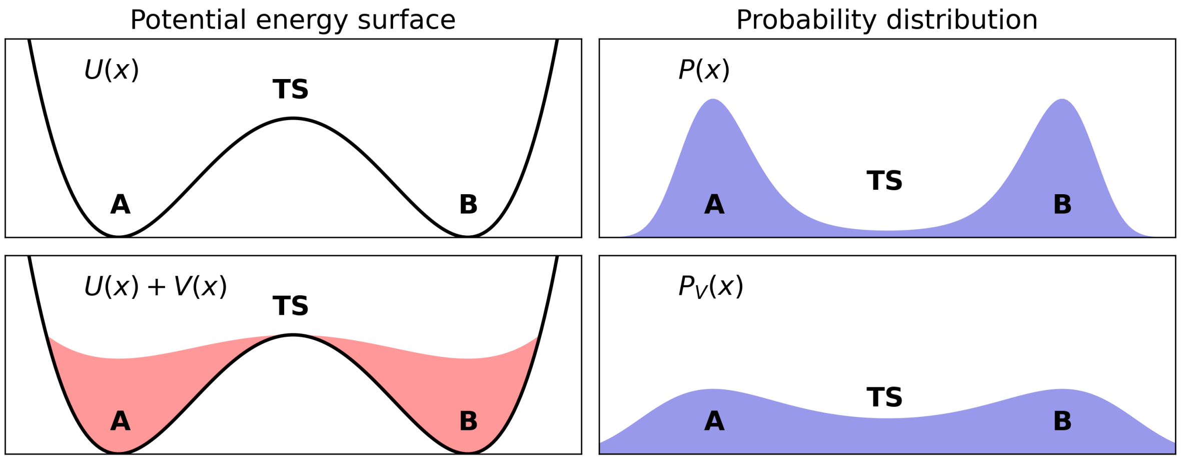


Figure 2.2: Schematic representation of the enhanced sampling approach effect on the energy landscape (left column) and the probability distribution (right column) of a double-well potential model. The top row depicts the unbiased scenario, characterized by large free energy barriers, resulting in a probability distribution peaked only in correspondence to the metastable states. The bottom row depicts the smoothed biased scenario with easy-to-overcome barriers, resulting in a much more spread probability distribution, which is non-zero also in the transition state (TS) region.

the unbiased FES by *reweighting* the histogram from the biased simulation. Similarly, we can observe that the unbiased probability distribution $P(\mathbf{r})$ is related to the biased one $P_V(\mathbf{r})$

$$P(\mathbf{r}) \propto P_V(\mathbf{r}) \exp[\beta V(\mathbf{s}(\mathbf{r})] \tag{2.8}$$

which, in principle, allows us to recover the unbiased average value of any generic observable $O(\mathbf{r})$ by reweighting each configuration by the corresponding bias

$$\langle O(\mathbf{r})\rangle = \frac{\left\langle O(\mathbf{r}) \exp[\beta V(\mathbf{r})]\right\rangle_V}{\left\langle \exp[\beta V(\mathbf{r})]\right\rangle_V} \tag{2.9}$$

where the expectation values on the right-hand side are computed in the biased ensemble.

It should be noted that the reweighting relations provided so far are strictly valid only in the case of a static bias, i.e., constant with time. One way to obtain such a condition in practice is by employing a bias potential that is static by design, as in the case of Umbrella sampling[6] for instance. However, many notable methods, such as Metadynamics[7] and its evolution OPES[65] (On-the-flight Probability Enhanced Sampling) rely on a time-dependent bias to enhance the fluctuations of the system and thus require an extension of the formalism provided so far.

Time-dependent bias potential: the example of Metadynamics The definition of an effective static bias potential is typically difficult in most cases. In practice, it is indeed more common to revert to time-dependent biasing schemes in which the bias potential is determined on the fly as the simulation (and the exploration of the phase space) goes on. In general, if we apply a time-dependent bias potential $V(\mathbf{s}, t)$ to our system, we can still recover the unbiased statistic upon the condition that at every instant t_0 we consider in our statistics, the system is at equilibrium with the instantaneous potential $U(\mathbf{r}) + V(\mathbf{s}, t_0)$. This typically implies that in our simulations, the bias is modified slowly and gently, thus leaving enough time for the system to relax before the next modification, and/or that in the long term, the bias converges to a constant value.

To showcase some features of time-dependent approaches, we will briefly summarize the popular *well-tempered*[8,66] variant of the Metadynamics (WT-MetaD) method as an example. For the sake of brevity, in the following, we will drop the bias dependence on time in our notation, thus simply writing $V(\mathbf{s})$. In WT-MetaD, the bias potential $V(\mathbf{s})$ is built by adding repulsive Gaussians at fixed intervals, which are centered on the point sampled $\mathbf{s}_k$ at the deposition moment k. In a naive way, this approach can be seen as we are slowly *filling* the metastable basins one tiny Gaussian at a time as we proceed with our simulation, as schematically depicted in Fig. 2.3.

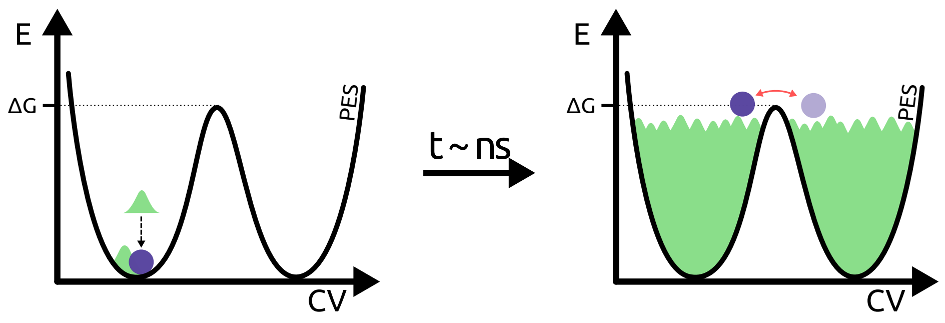


Figure 2.3: Schematic representation of the working principle behind Metadynamics in a rare event scenario described by a potential energy surface (PES) presenting a large free energy barrier ΔG. During the exploration (left panel), repulsive Gaussian bias potentials (in green) are deposited on the fly in the visited positions. At convergence after a relatively short time (typically ~ns), the energetic landscape is filled by the bias potential, thus removing the free energy barriers and promoting transition between the different states.

In mathematical terms, the bias at the n^{th} iteration can thus be written as

$$V_n(\mathbf{s}) = \sum_{k}^{n} \exp\left[-\frac{\beta V_{k-1}(\mathbf{s}_k)}{\gamma - 1}\right] \mathbf{G}(\mathbf{s}, \mathbf{s}_k) \tag{2.10}$$

where $\mathbf{G}(\mathbf{s}, \mathbf{s}_k)$ denotes a Gaussian centered in $\mathbf{s}_k$. The parameter $\gamma > 1$ is called *bias factor*, and determines the spread of the biased probability distribution P_V with respect to the unbiased one P upon convergence

$$P_V(\mathbf{s}) \propto \frac{[P(\mathbf{s})]^{1/\gamma}}{\int d\mathbf{s} [P(\mathbf{s})]^{1/\gamma}} \tag{2.11}$$

The WT-MetaD algorithm is indeed designed to asymptotically converge in the long term to a smooth bias V_γ defined as

$$V_\gamma(\mathbf{s}) = -(1 - \frac{1}{\gamma})F(\mathbf{s}) + c(t) \tag{2.12}$$

where

$$c(t) = \frac{1}{\beta} \log \frac{\int d\mathbf{s} \exp\left[-\beta F(\mathbf{s})\right]}{\int d\mathbf{s} \exp\left[-\beta (F(\mathbf{s}) + V(\mathbf{s}, t))\right]} \tag{2.13}$$

which does not depend on $\mathbf{s}$[8,67].

Such a bias is not designed to completely fill the basins to obtain a flat and purely diffusive energy landscape but rather to reduce the height of the free energy barriers between the metastable states. Moreover, it can be seen that the limit case $\gamma \rightarrow 1$ corresponds to the unbiased scenario, whereas when $\gamma \rightarrow \infty$, we have that $V_\infty(\mathbf{s}) = -F(\mathbf{s})$ resulting in a P_V that is constant everywhere.

In the case of WT-MetaD, the unbiased probability can still be recovered from the biased one. However, one has to account for the time dependency of the bias in the reweighting procedure by including the time-dependent term $c(t)$[8,67] in the weights

$$P(\mathbf{r}) = P_V(\mathbf{r}, t) \exp\left[\beta(V(\mathbf{s}(\mathbf{r}), t)) - c(t))\right] \tag{2.14}$$

However, MetaD presents some flaws even in its well-tempered and more stable variant. For example, its effectiveness strongly depends on the quality of the used CVs, and it may take a long time to converge in some cases. Moreover, this approach still presents long *transient* stages at the beginning of the simulations in which the bias is changing wildly. As a consequence, the configurations sampled in the initial part have to be discarded when computing averages, somehow wasting computational time. Fortunately, some of these limitations have been alleviated in OPES[65], a recent development of MetaD, which will be discussed in more detail in the next section as it will be extensively used in the following sections of this Thesis.

2.3 OPES: On-the-fly Probability Enhanced Sampling

In the previous paragraphs, we have introduced some general features of enhanced sampling methods, starting from a motivation for such methods and coming to the concepts of bias potentials and reweighting of unbiased statistics. In the last part, we then used the example of time-honored Metadynamics to provide a more practical example of these concepts. In the following, we will introduce OPES[65,68,69] (On-the-fly Probability Enhanced Sampling), a powerful evolution of Metadynamics that improves on many aspects of its ancestor.

The foundational idea of OPES is rather simple, which is to build the bias potential according to an estimate of the probability distribution rather than based on the energy as we do in MetaD. Indeed, OPES method aims at sampling a given target distribution $p^{tg}(\mathbf{s})$ in the configurational space $\mathbf{s}$, which is different from the equilibrium Boltzmann distribution $P(\mathbf{s}) \propto \exp[-\beta U(\mathbf{s})]$. This is done by incrementally building a bias potential $V(\mathbf{s})$ according to

$$V(\mathbf{s}) = \frac{1}{\beta} \ln \frac{P(\mathbf{s})}{p^{tg}(\mathbf{s})} \tag{2.15}$$

This biasing scheme generally allows for faster convergence of the bias. It indeed quickly becomes *quasi-static* and, as a consequence, the properties of interest, such as the free energy, can be computed using standard reweighting (see Eq. 2.9) using almost all the sampled points from our simulations as we do not have long transient parts to discard from our trajectories.

Depending on the type of target distribution we choose, different OPES biases can be used, each of them with specific strengths and limitations. In the following, we will start from the first OPES formulation, OPES-MetaD, which we will also use to showcase some of the general features of this powerful method.

OPES-MetaD In OPES-MetaD, the target distribution resembles the WT-MetaD[66] one we introduced in the previous section

$$p^{tg}(\mathbf{s}) = p^{WT}(\mathbf{s}) \propto [P(\mathbf{s})]^{1/\gamma} \tag{2.16}$$

in which γ is the so-called biasfactor, which determines the spread of this distribution. Similarly to MetaD, the bias is optimized on the fly, but in this case, it is obtained by reweighting the unbiased probability distribution $P_n(\mathbf{s})$ at iteration n using a Gaussian kernel density estimation[70] (KDE)

$$P_n(\mathbf{s}) = \frac{\sum_k^n w_k G(\mathbf{s}, \mathbf{s}_k)}{\sum_k^n w_k} \tag{2.17}$$

As a technical detail, the number of kernels used in the simulation is kept low by means of a kernel-merging compression algorithm.

In this framework, the bias at step n can be expressed as

$$V_n(\mathbf{s}) = (1 - \frac{1}{\gamma})\frac{1}{\beta} \ln\left(\frac{P_n(\mathbf{s})}{Z_n} + \epsilon\right) \tag{2.18}$$

in which $\epsilon \ll 1$ is a regularization term to prevent the logarithm from going to zero, and the normalization factor Z_n depends on the region of the CV space Ω_n that has been explored up to step n

$$Z_n = \frac{1}{|\Omega_n|} \int_{\Omega_n} P_n(\mathbf{s}) d\mathbf{s} \tag{2.19}$$

Of course, as the exploration will change during the simulation, it will have an impact on the bias. Thanks to the Z_n term, in OPES-MetaD, we usually have a very quick initial exploratory phase in which the bias is coarsely determined to be only slightly refined to convergence in the remainder of the simulation. This feature brings many advantages, such as faster convergence and the possibility of using a simple reweighting scheme without cropping almost any initial transient. Another interesting consequence is that OPES-MetaD allows for a quick evaluation of poor or suboptimal CVs, which, for example, do not really capture the relevant modes of the system. In such a case, it will likely get stuck in one metastable state for a long time without showing any transition because the initial coarse bias will be ineffective due to the poor CVs, and it will not change much during the simulation.

Conveniently, in practice, OPES-MetaD depends on very few parameters: the optimization pace, the initial bandwidth of the Gaussians kernels (which is automatically optimized on the fly thereafter), and the expected height of the free energy barrier that is supposed to be overcome in the process. The pace is usually determined similarly to MetaD, for instance, optimizing the bias every 500 steps. The initial kernel width can be estimated from the smaller standard deviation of the CVs in the minima. Whereas, as far as the barrier is concerned, a reasonable choice is enough in most cases as it determines the choice of γ and ϵ, with the biasfactor being much a less critical parameter here with respect to what it is in MetaD.

Many examples of the application of OPES-Metad to prototypical systems will be discussed in Chapters 5 and 6.

OPES-Explore As the name suggests, this variant of the model is oriented to a faster exploration of the CV space at the cost of a slower convergence. This approach is indeed more similar to an improved version of MetaD in terms of sampling, and it still allows for good sampling when used with suboptimal, if not poor, CVs at variance with OPES-Metad.

In OPES-Explore, the target distribution is still the well-tempered distribution, but the bias is not expressed as a function of the unbiased equilibrium probability $P(\mathbf{s})$, but it is built on the fly from an estimate $P^{WT}(\mathbf{s}) \propto [P(\mathbf{s})]^{1/\gamma}$ of the distribution that is being sampled during the simulation

$$p_n^{WT}(\mathbf{s}) = \frac{1}{n}\sum_{k}^{n} G(\mathbf{s}, \mathbf{s}_k) \tag{2.20}$$

where $\mathbf{s}_k$ is the CV value sampled at step k.

The construction of the bias potential is also slightly different as, in this case, it is slower and more gradual. The bias at iteration n in OPES-Explore is given by

$$V_n(\mathbf{s}) = (\gamma - 1)\frac{1}{\beta}\ln\left(\frac{p^{WT}(\mathbf{s})}{Z_n} + \epsilon\right) \tag{2.21}$$

The free energy surface (FES can be obtained from OPES-Explore trajectories in two ways. One is to compute it from the probability estimate directly

$$F_n(\mathbf{s}) = -\frac{\gamma}{\beta}\ln p_n^{WT}(\mathbf{s}) \tag{2.22}$$

The other is via importance sampling reweighting

$$F_n(\mathbf{s}) = -\frac{\gamma}{\beta}\ln\left[\sum_{k}^{n}\exp\left[\beta V_{k-1}(\mathbf{s}_k)\right]\right] \tag{2.23}$$

At variance with the OPES-Metad case, in which these two approaches were equivalent, in this case, they can differ significantly, especially in the first part of the simulation in which the bias is far from convergence.

In Chapter 7, we will discuss how we used OPES-Explore for the study of the processes and structure involved in liquid sulfur's λ-transition.

OPES-Flooding The OPES-Flooding[69] variant was developed for efficiently collecting unbiased transition and studying kinetic rates by combining the OPES biasing scheme and the ideas of conformational flooding[71] and hyperdynamics[72]. In these two works, the Authors demonstrated that kinetic and dynamical information could still be recovered from biased simulations, upon the condition that the bias is not added in the *transition state* region.

Given these assumptions, the OPES-Flooding idea, which we schematically depict in Fig. 2.4, is then rather simple as we build a bias that is aimed at *partially filling* only one of the metastable basins such that we can escape from it in a reasonable time and that the transition state region is not affected by the bias itself. This last condition is ensured by the introduction of an additional parameter, the so-called *excluded region* $\mathbf{s}_{\text{exc}}$, and imposing that no bias is deposited for $\mathbf{s} > \mathbf{s}_{\text{exc}}$. On the other hand, for $\mathbf{s} < \mathbf{s}_{\text{exc}}$ the deposited bias still increases the probability of observing the transition.

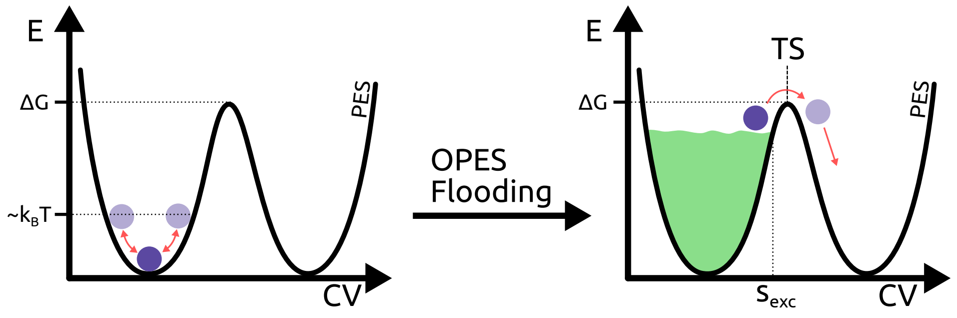


Figure 2.4: Schematic representation of the working principle of OPES-Flooding for a potential energy surface (PES) presenting a large free energy barrier ΔG that hinders the transitions between two basins, thus limiting the unbiased sampling to the bottom of the metastable states (left panel). The OPES-Flooding bias is constructed to *partially fill* the basin without affecting the transition state (TS) region. To do so, the bias is not deposited beyond the *excluded region* value s_{exc} of the considered collective variable (CV). With the help of such a bias, the system can escape from the initial metastable state following an unbiased transition trajectory (right panel).

In practice, this approach allows the collection of a number of unbiased reactive trajectories from which precious information on the kinetics of the process can be extracted, as well as information on the transition-state-related configurations. The first outcome, however, is not relevant for the purposes of our thesis, and we thus refer the Reader to the corresponding literature[69] for further details, whereas the second will be exploited in the methods presented in Sec. 6.3.

2.4 Collective variables

In the previous sections, we have seen that the concept of collective variables (CVs) is of central importance in the context of enhanced sampling. However, we still have not dwelt enough on this topic, which will be central to the development of the rest of this Thesis, so we do it now.

From a mathematical point of view, the CVs can be defined as functions, eventually nonlinear, of the atomic coordinates $\mathbf{r}$, i.e., $\mathbf{s}(\mathbf{r}) = (s_1(\mathbf{r}), s_2(\mathbf{r}) \dots s_d(\mathbf{r}))$. Moreover, they should be continuous, differentiable, and invariant with respect to the symmetries of the system (e.g., translation, rotation, and permutation of identical atoms).

Despite sounding somehow abstract, the concept of collective variables should probably be familiar to Physicists and Chemists in practice, even if under different names. It is indeed common both in Chemistry and Physics to describe complex problems on the basis of a few selected variables. In Physics, for instance, this is the case of the *order parameters* used for describing phase transitions in Landau's theory[63]. Whereas in Chemistry, the progress of a chemical reaction is often expressed as a function of a generic *reaction coordinate*.

In computational approaches, however, the need for a *dimensionality reduction* is not only motivated by the need for a concise description of the process from a conceptual point of view but also from the very practical problem of limiting the computational costs. Indeed, in our discussion on enhanced sampling methods, we mentioned that bias potentials are defined as functions of CVs, but we did not mention that the overall cost of such simulations scales exponentially with the number of CVs on which the bias is applied. Consequently, in practice, it is uncommon to bias more than a couple of CVs at a time, which motivates the interest in making our CVs *as collective as possible*.

The meaning of *collective* in this context refers to the idea that our CVs should be able to combine together all the relevant degrees of freedom that describe the key features of the chemical/physical process we are interested in. For example, in the case of a chemical reaction in which two small organic molecules have to react to a single product, one can try to use the distance between the center of masses of the two molecules as a CV, which will combine information about the atomic positions of many atoms and of the two molecules into a single variable. A good CV is then supposed to encode the so-called *slow modes* of the system, which are those degrees of freedom whose fluctuations are most relevant to the process of interest and should also be able to distinguish between all the relevant metastable states and the transition state as well. For example, in the toy model reported in Fig. 2.5, we can see that a proper CV should be able to distinguish the metastable states A and B, and also the transition state region (TS). In this sense, it is clear that the x coordinate is an awful CV, as all such regions are superimposed when projected on the x-axis. On the other hand, the

y coordinate is a better CV as the overlap between the different regions is much smaller. However, even the y coordinate is still a *sub-optimal* CV when it is used in an enhanced sampling context, as it does not encode any information about the bent path that one has to follow to go from A to B (and back) passing through the transition state. In other terms, even this CV does not encode all the *slowest modes* of our system.

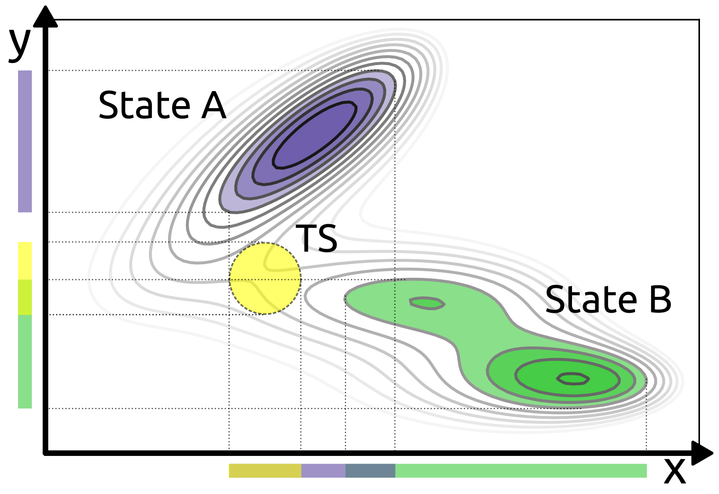


Figure 2.5: Schematical representation of the quality of two possible collective variables (CV), i.e., x and y coordinates, for the Müller-Brown potential energy surface. The metastable states A and B are highlighted in purple and green, respectively, whereas the transition state (TS) region is colored in yellow. The three regions are projected along the two CVs and represented with the same color scheme.

Traditionally, the determination of CVs has been driven by physical and chemical intuition, and the choice reverted to physical descriptors such as distances, angles, or coordination numbers or to simple mathematical functions of the coordinates as in the case of the RMSD-based CVs used in protein simulations[73,74]. However, this approach can fall short as the complexity of the studied systems increases. This motivated the development in the last decades of many machine-learning-based approaches aimed at automatically determining CVs in a data-driven way, as we will largely discuss in Chapters 4, 5 and 6.

Chapter 3

Machine Learning basics

The past decade has seen an exponential rise in the application of the so-called *machine learning* (ML) techniques in many fields of science as well as everyday life. This has been possible thanks to the constantly improving computational power at constantly decreasing prices, the dizzying increase in data availability, and the plethora of more and more efficient algorithms and architecture proposed by the machine-learning community. The first factor directly relates to the time-honored Moore's law[75], which predicted back in the 60s a hardware performance increase rate of roughly a factor of two every two years. Since then, that empirical prediction has proven correct, and it is still likely to be thanks to the advent of modern GPUs, which greatly contributed to the development and widespread of ML. The increasing data availability comes as a natural consequence of this development. In everyday life, almost everyone owns a smartphone, uses search engines on the internet, or interacts with social media, thus generating (big) data that can be used, for instance, to tailor advertisement campaigns based on the users' interests. On the other hand, in the case of computational science, the constantly increasing computational power now allows the collection of huge datasets that only twenty years ago would have been impossible even to dream of. Last but not least, we have what is somehow both the consequence and the driving motion for such vertical advancements, which is the development of ML techniques. Indeed, large datasets are useless if we do not have a practical way to extract the relevant information from them. In the same way, super-fast GPU wouldn't be needed if our algorithms were still limited to simple linear regression.

In the following sections, we will provide a limited, to say the least, introduction to the world of ML algorithms. We do this as this Thesis vastly relies on the contribution that ML can bring to computational science and simulations. However, we approach this world as users, and thus, we will provide an introduction that aims at being more practical than comprehensive. Moreover, many of the applications we will present in the rest of this Thesis are based on rather basic machine-learning algorithms whose implementation in practice is greatly simplified by the many available code libraries in this field. For all these reasons, our treatment will be limited to the topics that are most relevant to the purposes of this work, and we refer the Reader to the available literature[9,76,77] for a complete overview also pointing out some references specifically tailored to a Physicists audience[10,11].

3.1 What is Machine Learning

We can define Machine Learning (ML) as a field of study oriented to the development of techniques that are aimed at automatically learning from data and making predictions without having been explicitly programmed to do so. In any ML approach, three key ingredients are needed: a *model*, a *dataset* to learn from, and an *objective function* (or *loss function*) that encodes the task that needs to be accomplished. As the purpose of this Chapter is to provide a practical introduction to some relevant elements of ML, we will introduce such ingredients in a general way and will try to contextualize them with the help of a simple but didactic example, such as the fitting of pairs of points (x, y) using polynomial regression.

The model The *model* is typically defined as a function $f_\theta(\mathbf{x})$ of some *input features* $\mathbf{x}$ that depends on a set of *parameters* θ. Such parameters are *trainable*, in the sense that they can be tuned such that the model is optimized for the given task. The model (and also the objective function, as we will see in a moment) often presents also *non-trainable* parameters, which we refer to as *hyperparameters*. In the ML jargon, the mathematical form of f_θ is typically referred to as the *architecture* of the model, and different choices can be adopted for different tasks. In the following, we will focus only on the case of feed-forward Neural Networks (NNs), which will be discussed at length in Sec. 3.3, but many alternatives can be found in literature, for example, graph neural networks[78] (GNN) and convolutional neural networks[9] (CNN). In our polynomial regression example, the model is represented by the parametric form of our polynomial

$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \cdots + \theta_n x^n \tag{3.1}$$

which depends on the set of trainable parameters $\theta = \{\theta_0, \theta_1, \theta_2 \dots \theta_n\}$, whereas the grade n of the polymer used for fitting can be seen as a hyperparameter.

The objective function The objective function $L(\mathbf{y})$ is a mathematical function as well, which typically depends on the output of the model $\mathbf{y} = f_\theta(\mathbf{x})$. This quantity should formalize in mathematical terms the task that we want to accomplish, and, in general, it is defined in a way such that the model can be optimized by minimizing it. As one can imagine, the nature of this function is determined by the problem and the task at hand. In the case of our example, the task is rather simple: we want the output of our model y', for any of the given input x in our dataset of size N, to be as close as possible to the corresponding y value. Possible objective functions for this purpose are then the mean squared error (MSE) or the root mean square error (RMSE), which are often employed for curve fit optimization

$$L_{\mathrm{MSE}} = \frac{1}{N}\sum_{i}^{N}(y-y')^2 \qquad L_{\mathrm{RMSE}} = \sqrt{\frac{1}{N}\sum_{i}^{N}(y-y')^2} \tag{3.2}$$

Optimization of the model At this point, we have a model, a dataset, and an optimization criterion. Thus, we *only* need to find the right combination of parameters that minimizes our loss function, or, in other terms, we need to *train* our model for our task. But how can we do that? In general, this is done in an iterative way and by using some variant of the *gradient descent* technique[9]. This optimization algorithm aims to find the local minimum of our loss function in the space of parameters θ by moving at each iteration, or *epoch*, of a step in the direction of its gradient. In practice, this is done by updating the parameters of our model at each epoch according to

$$\theta_{i+1} = \theta_i - \lambda \nabla_{\theta_i} L[f_{\theta_i}(x)] \tag{3.3}$$

Here, λ is the so-called *learning rate*, which determines the size of the step we make at each iteration. This hyperparameter should be chosen wisely to have a proper optimization as if it is too small, the procedure would take much longer than needed. On the other hand, if it is too large, it would struggle to converge.

In Eq. 3.3, we actually reported the *vanilla* version of the gradient descent approach, which is referred to as *steepest* descent. However, even if intuitive, this is often not the best choice and, in practice, more advanced variants are employed. Among those, the probably most popular choice is the ADAM optimizer[79], but other options, such as the stochastic gradient descent[80], are available.

Validation, Overfitting, and Underfitting So far, we have always said that the goal of our model's training is to minimize the loss function. However, this is strictly true only if we are interested only in *fitting* the available and not in making *predictions* with our model. However, this latter scenario is often the case in ML.

For instance, if we imagine that in our example, x is a temperature and y is some physical quantity that should depend on the temperature, what we are trying to learn with our model would be the thermal dependence of such quantity. In this case, we may be interested not only in a model that does not only fit well the points we already have in our training set but that can also predict the value of $\hat{y}$ for different temperatures that were not included in our initial

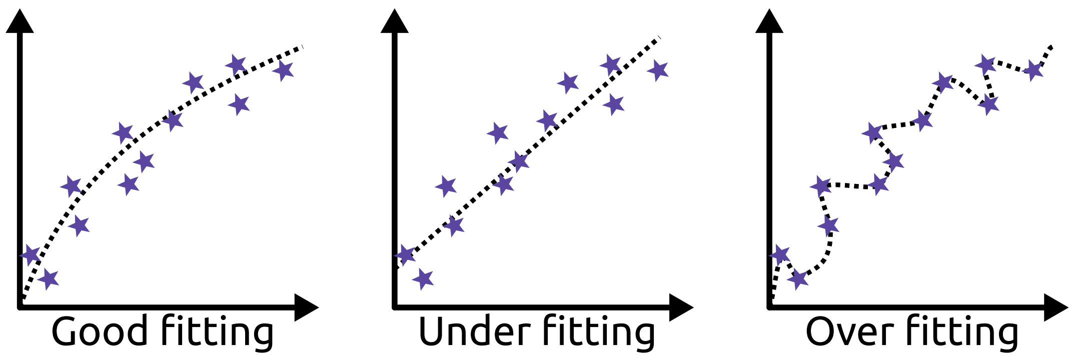


Figure 3.1: Schematic representation of different learning outcomes in machine learning in the simple example of data interpolation. The left panel reports the case of a proper fitting of the data, with the model able to follow faithfully the trend in the training data. The central panel depicts the case of underfitting, in which the model learned a representation of the training data that is too simple and rough to properly describe them. The right panel shows a case of overfitting, in which the model representation is too specific and tailored to the training data, thus losing any generalization capability.

dataset. In such a case, we may have an unpleasant experience with our model of what is commonly known as *overfitting*, which is that our model performs extremely well during the training, i.e., minimizes the loss on the training data, but performs poorly on new data, i.e., gives wrong predictions on new data (see right panel of Fig.3.1). Fortunately, many techniques have been developed to monitor and prevent overfitting. For example, it is a common practice to randomly *split* the available dataset in two parts: the *training set*, which usually includes some 80%-90% of the data and is used for the parameters optimization, and the *validation set*, which does not take part in the optimization but is used for monitoring the performance on the model on unseen data. This approach often allows spotting overfitting when present. Indeed, in that case, the score of our model would be much better on the training set than on the validation one. To limit and prevent this problem, once we have the diagnostic tool of *cross-validation*, we can *early-stop* the training before the model starts to overfit the training data by monitoring the trend in training and validation losses. Another example in this sense is the so-called *dropout* approach, in which, at each iteration, we include the stochastic possibility of turning off some of the parameters in the model, thus including an artificial noise to the training.

At the other end of the spectrum of pathological behaviors for our ML models, we have *underfitting*, in which the model predictions are wrong simply because the model did not learn what it was supposed to (see central panel of Fig. 3.1). In this case, the possible causes are usually the model being too simple or the training time being too short. Following these considerations, it is thus safer to say that, to consider our optimization successful, we have to find the set of parameters that best minimizes our loss function before the eventual onset of overfitting (see left panel of Fig. 3.1).

3.2 Different types of Machine Learning

In general, ML approaches can be divided into three broad categories depending on the different learning approaches[10,11]: supervised, unsupervised, and reinforcement learning.

Supervised learning In *supervised* learning, we want to obtain a model that can accomplish a given task after being trained on an example dataset. This approach is called supervised because we are somehow guiding our model toward the solution. This is the case of classification and regression algorithms. For instance, image recognition algorithms are trained on *labeled* dataset. For example, in such a case, we may want to train a model to predict the species of the animal in a given picture starting from a dataset of images of different animals (cats, cows, capybaras..) *labeled* with the name of the species of the animal in the picture. More classical examples of supervised learning are linear and polynomial regression and decision trees. In computational Physics and Chemistry, supervised learning is widely used, for example, for the training of ML interatomic potentials[11], as we have mentioned in Sec. 1.5.

Unsupervised learning In *unsupervised* learning, we are interested in finding (hopefully nontrivial) structural patterns in our data without prior knowledge. This means that in our data, no labels are available. This is, for example, the case of clustering, in which we aim to group data points in a way such that the points in each group share some common properties. A basic example of unsupervised learning is the time-honored principal component analysis[81] (PCA) with all its variants. In Science, such approaches are mostly used for analysis[11], with some relevant exceptions, as we will discuss, for example, in Sec. 4.2.1.

Reinforcement learning The last category is the so-called *reinforcement learning*, which aims at mimicking human learning more closely. In this approach, the model is trained to react to the environment based on some reward function. Even if we do not use this approach in this work, a few examples are the algorithms employed in the motion of robots and the models trained to play *human* games such as chess or go.

3.3 Neural Networks

Until this point, we have provided only a few hints of the actual mathematical form of our ML models. In this section, we should make up for this by introducing *neural networks* (NNs), also called *multilayer perceptrons*, which are ar-

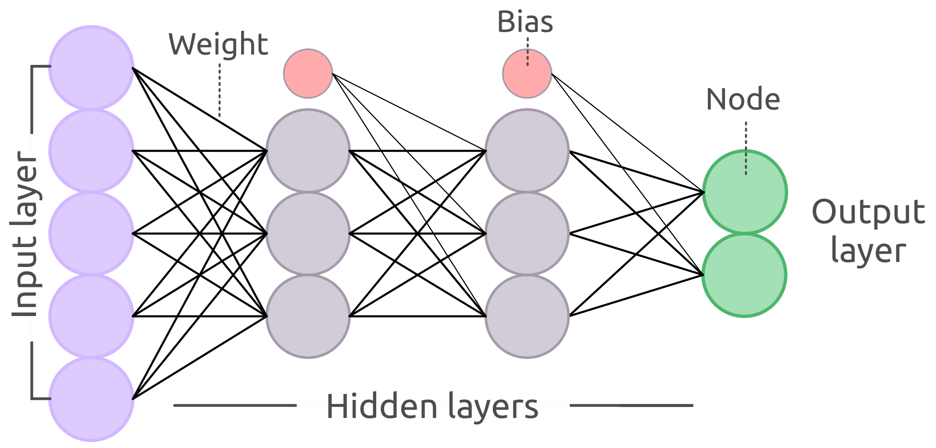


Figure 3.2: Schematics of the components of a feed-forward neural network that combines the information of an input layer (purple) through some hidden layers (grey) into an output layer (green). The circle depicts the network nodes, whereas the lines represent the weighted connections between them. The red circles in the upper part represent the bias applied in the hidden layers.

guably the quintessential ML models. These are indeed the models that stand behind the ML revolution of the past decades[9,11], and they will be widely used in the remainder of this Thesis. However, for our purposes, it is enough to limit our discussion to the simple, but yet most diffused, case of feed-forward neural networks (FFNNs), whereas we refer the Reader to the literature for other flavors of this powerful approach[9,76]. In Fig. 3.2, we schematically depict the structure of a simple NNs and we shall refer to it as we discuss its components.

The network structure The goal of feed-forward neural networks is to approximate some function f^*. In this sense, they can be seen as *universal interpolators*, a more powerful and nonlinear extension of regression models. These models are called *feed-forward* because the information flows from the inputs $\mathbf{x}$ (purple in Fig. 3.2) through the model f_θ, which depends on some parameters θ, to the output $\mathbf{y} = f_\theta(\mathbf{x})$ (green in Fig. 3.2).

They are called *networks* because they are typically represented by a composition of many different functions, which are referred to as the *layers* of the network. The number of such layers determines the *depth* of the network, and their basic elements are the so-called *neurons* or *nodes* (circles in Fig. 3.2). Each neuron i takes an input vector of d input features $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and produces a scalar output $a_i(\mathbf{x})$. The neurons are then stacked into the layers, with the output of each layer serving as the input of the next one. The first layer of the network is called the *input layer*, the last one is the *output layer*, whereas the layers in between are usually called *hidden layers*. The whole layered structure of the network, on the other hand, is usually referred to as the *architecture* of our model.

Neuron operations The output of a neuron $a_i(\mathbf{x})$ in essentially all cases can be decomposed into two conceptually separated operations. One is a linear combination of the inputs, and the other is a non-linear function $\sigma(z)$. This second part is usually, but not necessarily, the same for all the neurons in the network and is usually referred to as the *activation function*.

For a given input $\mathbf{x}$, the linear transformation has the form of the dot product with a set of node-specific *weights* $\mathbf{w}^{(i)}$ (lines in Fig. 3.2) which is followed by a shift via a node-specific *bias* $\mathbf{b}^{(i)}$ (red circles in Fig. 3.2)

$$z^{(i)} = \mathbf{w}^{(i)} \cdot \mathbf{x} + \mathbf{b}^{(i)} \tag{3.4}$$

In this framework, the set of weights $\mathbf{w}$ and biases $\mathbf{b}$ represents the set of tunable parameters of our model $\theta = \{\mathbf{w}, \mathbf{b}\}$.

If we now apply the nonlinear activation function, we can obtain the overall input-output function as

$$a_i(\mathbf{x}) = \sigma_i(z^{(i)}) \tag{3.5}$$

Among the many available activation functions[10], popular choices are sigmoids, the hyperbolic tangent ($\tanh$), or the rectified linear units (ReLU). In practice, the computational and training properties of the neurons depend, of course, on the chosen activation. For example, in some cases, we may prefer to have a *softer* nonlinearity, as for the case of sigmoids and the $\tanh$ function, whereas, in some others, we may prefer the ReLU stronger activation, which also does not saturate for larger input values. We should also note that the derivatives of these nonlinearities are quite different from each other. For example, the derivatives of the hyperbolic tangent are generally smoother than the ones of ReLU functions, and this feature could be relevant if we are interested not only in our model's output but also in its derivatives, as we will see in the biasing scheme presented in Chapter 8.

Optimization and Backpropagation We have already seen that the optimization of ML models is done by minimizing an objective function via gradient descent or one of its variants. In the case of NNs, the procedure is the same, but it is made computationally more difficult due to the presence of multiple hidden layers. The solution to this complication is the *backpropagation algorithm*[82], or simply *backprop*, which allows for efficient computation of gradients by exploiting the layered structure of the network. At its core, this method is nothing more than the ordinary chain rule for partial differentiation. The idea is then to use the chain rule to express in a computationally efficient way the complex derivatives over the whole network as the product of the simple derivatives of its components, i.e., linear combinations and activations.

During each iteration of the optimization procedure, we then have different stages[9,10]

1. **Forward propagation**: we apply our feed-forward neural network f_θ to our input $\mathbf{x}$ to compute the output $\mathbf{y} = f_\Theta(\mathbf{x})$

2. **Compute loss**: we evaluate our objective function $L(\mathbf{y})$ on the output of the model $\mathbf{y} = f_\theta(\mathbf{x})$
3. **Back propagation**: we compute the gradients of the loss with respect to the parameters $\nabla_\theta L(\mathbf{y})$ using the chain rule
4. **Parameters optimization**: we update the parameters θ of the model according to the gradients computed with backpropagation and the chosen learning rate.

Autoencoders: the unsupervised twin of neural networks Even if neural networks are generally employed in the case of supervised learning, they share the general framework with the *autoncoder* (AE) architecture, which is usually used in the unsupervised setting and is worth mentioning. Such an architecture, apart from the characteristic details of specific variants[9,76], is composed of two concatenated networks: the *encoder* and the *decoder*. In the simplest scenario, the encoder E_{θ_E} is nothing more than a feed-forward neural network, which compresses some high-dimensional input features $\mathbf{x}$ into a low-dimensional output space $\mathbf{y} = E_{\theta_E}(\mathbf{x})$, which is commonly referred to as the *latent space*. The decoder D_{θ_D}, on the other hand, is a *reversed* neural network, in the sense that it maps the compressed latent space $\mathbf{y}$ back to a space $\mathbf{x}' = D_{\theta_D}(\mathbf{y})$, which has the same dimension of the input one. Overall, the input-output relation in the autoencoder is given by the composition of the two parts $f_\theta(\mathbf{x}) = D_{\theta_D}(E_{\theta_E}(\mathbf{x})) = \mathbf{x}'$.

Autoencoders can be used, for example, to find maximum variance modes in a set of data in a similar way to what is done in PCA. To do that, they are trained, according to a *reconstruction loss*, to find a latent space representation from which the decoder can still *reconstruct* the original input, i.e., $\mathbf{x}' = f_\theta(\mathbf{x}) = \mathbf{x}$. Then, the decoder is discarded, and the encoder is used to operate the dimensionality reduction to the latent space. Another fascinating outcome is related to their use as generative models. For example, if in our dataset we have pairs of configurations corresponding to two different times t and $t + \Delta t$, in principle, we could train an autoencoder to predict the evolution after the Δt interval, i.e., $\mathbf{x}' = f_\theta(\mathbf{x}(t)) = \mathbf{x}(t + \Delta t)$[83].

Machine learning libraries, an honorable note We should point out that nowadays, many technical details we discussed in the previous paragraphs are most of the time passed under the hood as they are already implemented in high-level open-source libraries such as PyTorch[84] or Keras[85]. Such libraries indeed provide all the building blocks for creating and efficiently training state-of-the-art NN models as well as tools for dataset management. One of the founding pillars of these codes is the *automatic differentiation*, which allows for fast computation of the derivatives needed, for example, in backpropagation. Moreover, it should be noted that the utility of this last feature is not limited to the ML scenario but can also be exploited as a tool to access the derivatives of complex functions easily. As a final note, we must highlight that such libraries have greatly contributed to making ML accessible also to users from different backgrounds, thus making its diffusion to many fields of Science possible.

Chapter 4

The data-driven approach to collective variables

As we have seen in the introductive chapters, many enhanced sampling methods depend on identifying some collective variables (CVs) that can encode the relevant physics of the system and process we are interested in. Traditionally, CVs have been determined on the basis of physical/chemical intuition and by trial and error. Typical examples could be the use of simple physical descriptors[73] such as distances, angles, and coordination numbers in chemical reactions or simple functions such as the RMSD in the case of proteins. This approach offers the advantage of a transparent physical interpretation, but it can fail to capture the complex behavior of many molecular systems as, this way, it is easy to overlook important slow features that might hinder convergence.

In the last decade, machine learning (ML) techniques have been applied to the challenge of designing effective CVs, leading to a number of methods suitable for different scenarios according to the available data.

In the following sections, we will introduce some general aspects of such an approach in a *data-oriented* perspective, also mentioning some relevant methods for CVs design.

4.1 The learning scenario

As we have seen in Chapter 3, learning CVs in a data-driven way implies that we need a model function, which depends on some parameters, a dataset, which we can use to optimize this function, and an objective function, which formalizes the criterion on which we want to build our CV.

Dataset and input descriptors The dataset in this scenario typically consists of samples collected from MD simulations. These can be either unbiased, i.e., obtained with standard MD simulations, or biased, i.e., obtained by applying some enhanced sampling scheme (see Sec. 2.2). However, the raw cartesian coordinates of the atoms in our simulations cannot be used directly for our purposes as they are not invariant to the symmetries of the system (e.g., translation, rotation, permutation of identical atoms). The most popular choice is then to pre-process them to obtain a set of input features that can satisfy such symmetry requirements and provide an invariant description of the system. As an alternative, alignment[86] of data-augumentation[87] procedures have also been proposed. However, such approaches can become expensive and problematic when it comes to large systems and when chemical reactions are involved. On the other hand, the use of input features can also be helpful in incorporating some physical knowledge into the model in the same spirit by which CVs were historically designed. For example, physical descriptors such as distances and angles can be used as input features if we want to describe a chemical reaction[88,89] or, in the case of liquid-solid phase transitions, bond order parameters[90] or structure factor peaks can be used[91].

Typically, in an approach based on machine learning, several of these input features are combined together in a model function to build our CV. This alleviates the need to carefully choose which physical descriptors to use and also allows including those variables that may account for finer details of the process of interest. For example, if we have to pick a CV based on intuition for a chemical reaction in which two small molecules have to bond together, the distance between the bonding atoms would probably be the most obvious (and relevant) descriptor for the process. However, it may also be that other distances are affected by the reaction and thus, our CV would benefit from including also such degrees of freedom. Indeed, in general, for our approaches to be effective, we need the set of features we include in our dataset to carry enough information for the description of the process, at least when considered *collectively*.

Functional form of the model The functional form of the model function typically depends on a trade-off between model expressiveness and interpretability. For example, linear models (e.g., linear discriminant analysis LDA[92], principal component analysis PCA[81]) are readily interpretable but they often require a pre-identification of a set of relevant features to be effective. On the other hand, nonlinear models are far more expressive, but additional procedures may be needed to provide a meaningful interpretation[93–95].

Among nonlinear models, neural networks (NNs, see Sec. 3.3) have become extremely popular in recent years. We have indeed mentioned in the previous chapters that NNs can be used as universal interpolators to represent complex functions of many inputs and many outputs. In practice, this provides the right flexibility for an effective learning procedure also in the case of complex systems. Moreover, in the scenario of enhanced sampling, they lend themselves quite well because they are continuous and differentiable functions of the inputs by construction, and the derivatives are also easily accessible using the backpropagation algorithm and exploiting the automatic differentiation features of machine learning libraries. Furthermore, NNs are designed to effortlessly handle even a large number of input features, thus making the choice of the input descriptors less and less critical.

Learning objectives We have mentioned already that, in general, the design of CVs is guided by three main learning objectives

1. Operate a *dimensionality reduction* of the system to maximize the information enclosed in the CV space[86,87,96,97]
2. *Distinguish* between the different metastable states[15,88,98]
3. Reflect the *long-term evolution* of the system, that is, to describe its *slowest modes*, which are related to the transition between metastable states[83,99–104]

In the context of enhanced sampling, the third objective is typically the most important, as it is related to the deeper nature of our process. However, in practice, it is not always possible to directly use this as an operational criterion, for example, because our dataset is limited. Indeed, we often find ourselves in a chicken-and-egg situation[17,105] in which we need good data to extract good CVs, but, at the same time, we need good CVs to collect good data. In other terms, extracting efficient CVs typically requires a proper exploration of all the relevant states and transitions between them, but this exploration typically already requires effective CVs.

For this reason, the first two criteria have been widely used as surrogate objectives as they don't need extensive dynamical information about the system. Even if this approach can eventually lead to *sub-optimal* CVs, it still allows, in many cases, to obtain the best from limited data. Moreover, iterative approaches can also be used to refine our CVs as new (and better) data become available, either by performing multiple iterations following the same scheme or by progressively enforcing more CVs objectives from the list above[17,105,106].

4.2 Learning approaches

Each of the learning objectives we introduced in the previous paragraph requires different types of data, which progressively become richer in information. In the following, we discuss from this perspective the main three different learning approaches that can be adopted for a data-driven determination of CVs, which are summarized in Fig. 4.1.

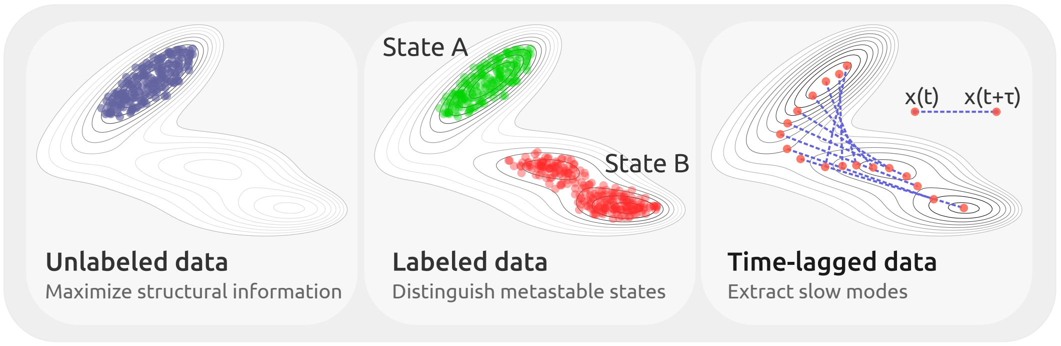


Figure 4.1: Type of data and the respective optimization criteria that can be used to design data-driven collective variables

4.2.1 Working with unlabeled data: unsupervised learning

The first scenario is the one in which we only have a collection of generic samples from MD simulations. In this case, our possibilities are limited to using unsupervised machine learning techniques, which are aimed at automatically finding structural patterns within the data[107,108]. In terms of the learning objectives we have listed in the previous paragraph (see Sec. 4.1), this approach only satisfies the first and most generic one. However, an unsupervised approach has the advantage of being applicable to any kind of MD simulation in principle. This allows the use of data from unbiased near-equilibrium simulations as well as those out-of-equilibrium or even those in which it is impossible to recover the unbiased dynamics.

In this spirit, unsupervised learning has been used in atomistic simulations to identify CVs based solely on sets of configurations obtained from MD trajectories. A classical example of such an approach is the linear principal component analysis[81,109] (PCA), which aims at finding the maximum variance directions within the data. The scope of the same principle has been extended by the use of autoencoding[96,105] (AE) NNs (see Sec. 3.3), which are trained to learn a compressed representation with the constraint that the original data can still be reconstructed starting from there.

In order to compensate for the lack of information that can be obtained from the data in an unsupervised setting, these approaches are often applied in iterative routines, for example, by alternating cycles of enhanced sampling and CV discovery[96,105,110]. This way, at each iteration, one can collect new data that can be used to train (possibly after reweighting) a *more-informed* CV that will eventually lead to a better exploration of the phase space. For example, in the case of chemical reactions, one may start from some reactant molecules and try to iteratively search for possible products.

4.2.2 Classifying metastable states: supervised learning

The second scenario presents an additional piece of information, which is being aware of the metastable state of interest for our system. In the example of a chemical reaction, these can be the reactants and product basins, or, in the case of a protein conformational equilibrium, they can be the folded and unfolded states. Other examples can be the bound and unbound state of a ligand with respect to a binding site or the different phases in a material.

From a machine-learning point of view, in this case, we are dealing with a *labeled* dataset, which means that each configuration can be associated with a specific metastable state of the system. Such a dataset can be easily collected in a rare-event scenario by running standard MD simulations in each basin. In the rare-event setting, the system will indeed remain trapped for a long time in that metastable state due to the presence of large free energy barriers that prevent spontaneous transition, making the labeling of the data straightforward.

When such data are available, we can optimize our CVs not only to operate a dimensionality reduction but also to discriminate between the metastable states, which corresponds to the second learning objective. Among the linear methods based on classification, support vector machines[15] and linear discriminant analysis[98] (LDA) have been applied for CV design. Also in the case of supervised approaches, nonlinear generalizations based on NNs have been proposed (Deep-LDA[88] and Deep-TDA[18,19], see Chapter 6) and applied to a wide range of systems[111–114].

The strength of supervised approaches is that they provide an easy way to insert into our model some previous knowledge, either in the form of a labeling of the states or also regression of some physical observables. As the resulting CVs are *aware* of the possible metastable states of the system, they can be a first hypothesis for the sampling of reactive trajectories between such states, thus giving access to even better data that can be used to further refinements of our CV model.

4.2.3 Extracting the slow modes: time-informed learning

The third and arguably best scenario is the one in which we know the metastable states of our system and we already have access to reactive trajectories between them. In the case of biophysics, this scenario can be realized by means of long unbiased simulations performed on supercomputers with dedicated hardware[115] or by sampling under different thermodynamic conditions (e.g., higher temeprature). However, in most of the systems, such as chemical reactions and phase transitions, the free energy barriers are often so large that cannot be overcome with these approaches. For this reason, the most common source of reactive trajectories is represented by enhanced sampling simulations. Such simulations can be performed either via CVs-independent methods[17], or by using CVs based on physical intuition as well as based on the data-driven approaches we have discussed so far. In this regard, it should also be noted that in this case, it is necessary to recover the unbiased dynamics from the biased simulations. This is a task that is not always simple and some approximations have been proposed and discussed, for example, in Ref.[106]

If we have access to reactive trajectories, time-informed learning can be used to find the slow modes that rule the long-term evolution of the system, thus fulfilling the third learning objective from our list (see Sec. 4.1). In the context of rare events we are interested in, the slow modes are indeed typically associated with the transitions between the long-lived metastable states. One linear example of this category is the time-lagged independent component analysis[116–118] (TICA), which aims at finding the linear combination of the input data that are maximally autocorrelated. Several nonlinear extensions of such a criterion have been proposed with the aim of providing more flexibility to the model function to achieve a better representation of the relevant slow modes[17,102,119]. Also time-lagged autoencoders have been applied for extracting CVs from reactive trajectories by learning a compressed representation from which future configurations can be predicted[83,120].

4.3 Combinining different approaches: multi-task learning

All the approaches described in the previous sections are related to the optimization of a specific loss function depending on a specific type of dataset. However, in principle, different approaches could also be combined and different types of data can be used at the same time. Such a practice is common in the machine learning field and is usually referred to as *multi-task* learning. Typically, this approach is used to improve the model's generalization capability and to better exploit *all* the available data. By multi-task learning, we refer to a broad class of algorithms that are applied for the optimization of machine learning models on multiple objective functions at the same time. This concept has been explicitly used in the context of supervised learning of CVs[121], and other CVs from the literature can also be framed in a multi-task format, for example, neural-network-based CVs that are optimized on a linear combination of loss functions. One example in this sense is the EncoderMap method[97], in which the normal reconstruction loss used for the training of autoencoders is paired with the Sketch-map[122] loss. The aim of the Sketch-map loss is to give more structure to the learned CV by enforcing that the distances in the low-dimensional latent space are proportional to the real distances in the high-dimensional space of the input features. In a similar way, in the Variational Dynamics Encoder (VDE) method[120], a function that aims at maximizing the autocorrelation of the CV is added to the optimization of a time-lagged variational autoencoder (VAE).

Even if following a multi-task learning approach on the same dataset can already bring some advantages to our CVs, more information can be encoded in the model if we consider using different types of data. For instance, for a given system, we may have both labeled and unlabeled data, which, of course, carry different types of information. For example, imagine that we start our study on a system from some labeled data from the metastable states, and we use such data to find a first CV with a supervised learning approach. Then, by performing enhanced sampling simulation on this CV, we can explore the phase space a little bit more, thus generating new data with our simulations that should not be wasted but rather used to obtain a better CV, for example, by incorporating a new loss based on unsupervised learning methods to our previous supervised CV model.

Chapter 5

The `mlcolvar` library

In the previous sections, we have provided a description of the construction of data-driven CVs, focusing on the dependence of the optimization tasks on the type of available data and mentioning the possibility of combining them to improve our models. From this discussion, it is already clear that the range of possible approaches and methodologies in this field is extremely wide and heterogeneous. Even if this flourishing of methods in the last years has certainly been beneficial for the enhanced sampling community, it still lacks a common framework. Indeed, the available implementations typically support only one or, at best, a few of the methods reported in the literature, and the interfaces with enhanced sampling codes are limited to specific cases. This, of course, has detrimental effects on the further development of these methodologies as well as on their utilization in practice.

These considerations, accompanied by the experience of Prof. Parrinello's group in methods for CV design, led us to the development of `mlcolvar`, a Python library aimed at simplifying the construction of data-driven CVs and their deployment in the context of enhanced sampling. The goal of`mlcolvar`, short for Machine Learning COLlective VARiables, is twofold. On one hand, to have a unified framework to help test and utilize (some of) the CVs proposed in the literature. On the other, to have a modular interface that simplifies the development of new approaches and the contamination between them.

The library is based on the machine learning library PyTorch[84] and the high-level Lightning package[123], which simplifies many aspects of the overall training procedure. This provides a simpler user interface and allows focusing only on the CV design and optimization. The library is an open-source project that is accessible on Github[1] or on the Python Package Index (PyPI)[2]. In principle, the library can also be used as a standalone tool, for example, for analysis of MD

[1]https://github.com/luigibonati/mlcolvar
[2]`pip install mlcolvar`

Figure 5.1: Logo of the `mlcolvar` library, which is meant to be interfaced with PLUMED enhanced sampling plugin, whose logo is a pigeon. To symbolize the bond between the two codes, we chose a capybara, a graceful rodent known to have pseudo-symbiotic friendships with small birds.

trajectories, but its main purpose is to be employed to create CVs that can be used in combination with enhanced sampling methods through the PLUMED C++ software[124,125].

In the following sections, we will present the main aspects of the library, such as the general optimization workflow and the structure of the code. In doing so, we will also review some of the recent methods in the field that are already (or that can easily be) implemented in the library. At the end of the chapter, we will then provide a series of prototypical examples in different learning scenarios. However, a complete description of the library is provided in the documentation and the tutorials[3] available online.

5.1 The CVs optimization workflow

The best way to introduce `mlcolvar` is by showing how it can be used for our purposes. For this reason, we will show the typical workflow used for CVs optimization with the help of the library, starting from the raw data to an optimized model ready to be used in enhanced sampling simulations. In this section, we will just set the context, providing a practical overview while leaving the technical details of the individual components of the workflow to the next sections.

The `mlcolvar` workflow consists of only a few steps, which are schematically depicted in Fig. 5.2. To provide a more practical reference to our discussion, we also provide in Listing 5.1 a working example of the corresponding few lines of code that are necessary for the implementation of such a workflow at the user level.

As a starting point in our workflow, we have the collection of data from MD simulations with the help of PLUMED[124,125]. These data are then imported into the Python ecosystem by means of the `utils` functions available in the library (step 1). They are then framed in a `DictModule` object (step 2), which allows efficient handling of the data in the training process. For example, it optionally

[3]https://mlcolvar.readthedocs.io/en/

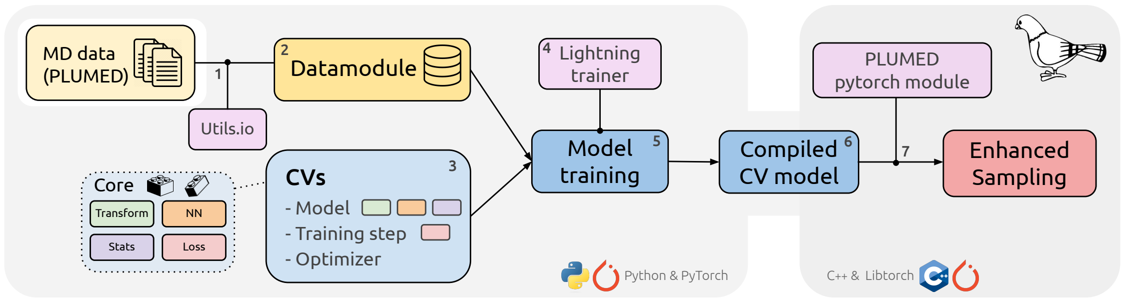


Figure 5.2: Schematic summary of the workflow for the construction of data-driven CVs in `mlcolvar`. Data from MD simulations are loaded and framed in a datamodule. A CV is selected from ready-to-use ones (`mlcolvar.cvs`) or built from the implemented building blocks (`mlcolvar.core`). After training, the model is compiled with the TorchScript language to be deployed to PLUMED for using it as CV to enhance sampling. A corresponding example of code is given in list. 5.1.

```python
# Setup
import torch,lightning
from mlcolvar.data import DictModule
from mlcolvar.cvs import AutoEncoderCV
from mlcolvar.utils.io import create_dataset_from_files

# 1. Import training data (e.g. PLUMED COLVAR files)
dataset = create_dataset_from_files('./COLVAR')
# 2. Create a Lightning datamodule which splits dataset in train/valid
datamodule = DictModule(dataset, lenghts=[0.8,0.2])
# 3. Choose a model and hyper-parameters
cv_model = AutoEncoderCV(encoder_layers=[45,30,15,2])
# 4. Define a trainer object
trainer = lightning.Trainer(max_epochs=1000)
# 5. Optimize parameters
trainer.fit(cv_model, datamodule)
# 6. Compile the model with TorchScript
cv_model.to_torchscript('model.ptc')
# 7. Use it in PLUMED via the pytorch module
```

Listing 5.1: Example of the typical workflow of CVs optimization with `mlcolvar`, as schematically depicted in Fig. 5.2. The input features are calculated by PLUMED and the end result is a serialized model that can be deployed in PLUMED via the LibTorch C++ interface.

divides the data into training and validation datasets (e.g., for early stopping or hyperparameter searching[9]) and sets up the mini-batches for the training (step 3). During the training, we optimize the `cv_model` we decided to use. In most cases, this can be initialized as one of the ready-to-use CV classes already implemented in the library. As an alternative, a custom model class can be implemented by the user starting from the `core` building blocks and the template

CV classes.

The `cv_model` object contains the trainable parameters of the model, the definition of the loss function we want to optimize on, and the optimizer used for the training. The whole optimization procedure is made extremely simple thanks to the Lightning package[123]. All the required operations are indeed automatically handled by the Lightning `Trainer` object (step 4). Conveniently, Lightning takes care not only of the optimization tasks but also of ancillary features such as the application of early stopping, storing metrics, generating log files and checkpoints, and automatically managing the process acceleration with GPUs, thus greatly simplifying the final user experience.

Once our `cv_model` has been optimized on the data, we can deploy it via the TorchScript language to make it transferable and accessible for production (step 6). Indeed, the frozen model can be imported in PLUMED using the `pytorch` model interface and used as CV for our enhanced sampling simulations (step 7).

We must stress that the workflow we discussed can be implemented by a `mlcolvar` user in 6 lines of code, with none of them requiring any in-depth coding knowledge. In order to make the process simpler, we also initialized all the defaults so as to offer reasonable starting points that can work well in most cases. Nevertheless, all the classes in the library are easily customizable and offer a flexible framework to control, for example, input standardization, network architecture, loss functions, and training hyperparameters. The technical details of such customizations are out of the purposes of this Thesis, but they are discussed at length in the documentation and the tutorials available online.

5.2 High-level structure of the code

We now proceed to present the overall structure of the `mlcolvar` library. This is designed to be modular in order to facilitate the implementation of new methods by only adding the necessary building blocks while inheriting the common elements of CV models. As we have said, the library relies on the machine learning libraries PyTorch[84] and Lightning[123]. However, we reimplemented some features to make them more flexible and suitable for our purposes and to simplify the overall interpretability of the code.

The library contains four main modules, which are `data`, `core`, `cvs`, and `utils`.

mlcolvar.data In `mlcolvar.data`, we provide PyTorch- and Lightning-compatible classes that simplify and improve the efficiency of data access. Moreover, to simplify the readability of the code, we based the implementation of these elements on a dictionary-like structure. The key elements are:

- `DictDataset`: A dictionary-like PyTorch `Dataset` that maps keys (e.g., data, labels, targets, weights) to tensors.

- `DictLoader`: A PyTorch `DataLoader` that wraps a `DictDataset`. This class is optimized to significantly reduce the data access time during training and to combine multiple datasets for multi-task training.
- `DictModule`: A Lightning `DataModule` that takes care of automatically splitting a `DictDataset` into training and validation (and optionally test) sets and returning the corresponding dataloaders.

mlcolvar.core In `mlcolvar.core`, we implemented the building blocks that are used for the construction of the CV classes. We organized them into the following submodules:

- `nn`: learnable modules (e.g., neural networks).
- `loss`: loss functions for the CVs optimization.
- `stats`: statistical analysis methods (e.g., PCA, LDA, TICA).
- `transform`: non-learnable transformations of data (e.g., normalization, descriptors calculation).

All of them are implemented as Python classes that inherit from `torch.nn.Module`. In particular, the `mlcolvar.nn` module contains a class that constructs a general feed-forward neural network which can be customized in several aspects, such as activation functions, dropout, and batch-normalization.

mlcolvar.cvs The `mlcolvar.cvs` module includes ready-to-use CV classes, grouped by the type of data used for their optimization in the following sub-packages:

- `unsupervised`: methods that require input data characterizing single MD snapshots.
- `supervised`: require labels of the data (e.g., the metastable states they belong to) or a target to be matched in a regression task.
- `timelagged`: require pairs of time-lagged configurations, typically from reactive trajectories, to extract the slow modes.

Since they are the key element of this library, the structure of CVs is described in more detail in Sec. 5.3.

mlcolvar.utils Finally, in `mlcolvar.utils`, one can find a set of miscellaneous tools for a smoother workflow. For example, we implemented here helper functions to create datasets from text files, as well as to compute free energy profiles along the CVs.

5.3 The structure of CV models

The CVs in `mlcolvar` are defined as classes that inherit from a `BaseCV` class and `LightningModule`. The former defines a template for all the CVs along with common helper functions, including the handling of data pre- and post-processing. The latter is a Lightning class, which adds several functionalities to simplify training and exporting the CV. In particular, `LightningModule` not only encapsulates the model with its architecture and parameters but also implements the training step (and hence defines the loss function) as well as the optimization method.

The structure of the CVs in `mlcolvar` is designed to be modular. The core of each model is defined as a series of building blocks (typically implemented in `mlcolvar.core`) that are by default executed sequentially, although this can easily be changed by overloading the forward functions in `BaseCV`. An example where this is necessary are AutoEncoders-based CVs. In this case, the building blocks are normalization, encoder and decoder, but the CV is the output of the encoder, not the final output of all the blocks.

New CVs require implementing a `training_step` method that contains the steps that are executed at each iteration of the optimization. Moreover, the loss function and the optimizer settings are saved as class members to allow for easy customization and it is possible to add preprocessing and postprocessing layers. This allows to speed up the training by applying the transformations only once to the dataset and later including them in the final model for production. In addition, it allows post-processing to be performed on the model after the training stage (e.g., standardizing the outputs).

Finally, multi-task learning is supported through the `MultiTaskCV` class that takes as input a given model CV as the main task together with a list of (auxiliary) loss functions which will be evaluated on a list of datasets. The samples from different datasets go through the same network but enter only one of the loss functions (see panel d of Table 5.1). The loss function used in this case is a linear combination of each specific loss. Each loss function can optionally be preceded by task-specific layers that are also optimized during the training but are not evaluated to compute the CV. Examples of task-specific models are the decoder used for the reconstruction task in autoencoders (see Fig. 5.7 for an example) or a classifier/regressor used for supervised tasks[121].

5.4 Deploying the CV in PLUMED

Once optimized, the CVs are exported using just-in-time compilation to the TorchScript language, returning a Python-independent and transferable model. We wrote an interface within the PLUMED[124,125] software that allows these exported models to be loaded through the LibTorch library (PyTorch C++ APIs). This is implemented in the `pytorch` module of PLUMED as a function that takes as input a set of descriptors and returns the CVs alongside their derivatives with respect to the input. An example of a minimal input file is shown in Listing 5.2.

This means that the CVs can be immediately used in combination with all enhanced sampling methods implemented in PLUMED (among which we find, for example, Umbrella Sampling[6], Metadynamics[7] and its many variants[74], Varia-

```
            # 1. Compute input features (e.g. pairwise distances)
            d1: DISTANCE ATOMS=1,2
            ...
            dN: DISTANCE ATOMS=17,19
            # 2. Load model exported by mlcolvar
            cv: PYTORCH_MODEL FILE=model.ptc ARG=d1,...,dN
            # 3. Apply bias potential (e.g. with OPES)
            opes: OPES_METAD ARG=cv.node-0 PACE=500 BARRIER=40

```

Listing 5.2: Example of PLUMED input file in which the calculation of input features is requested, the CV model exported from `mlcolvar` is loaded, and an enhanced sampling calculation is performed on it.

tionally Enhanced Sampling[16,126], On-the-fly Probability Enhanced Sampling[65], just to name a few) and within the supported molecular dynamics codes (including but not limited to LAMMPS[127], GROMACS[128], AMBER[45], CP2K[129], QUANTUM ESPRESSO[130], and ASE[131]), which allows simulating a wide range of complex processes in (Bio)Physics, Chemistry, Materials Science, and more.

Note that the interface is very general and thus can be used not only to compute collective variables but also to test CVs defined through complex functions by taking advantage of PyTorch's automatic differentiation capabilities (in the spirit of the PYCV PLUMED module[132] based on Jax). For example, it has been used to construct CVs from the eigenvalues of the adjacency matrix[133].

5.5 Methods for CVs optimization

In this section, we briefly present the methods implemented in the `mlcolvar` library for the identification of collective variables. Furthermore, we will mention some related methods and show how they can be constructed based on the building blocks of the library. In Table 5.1 we summarized them together with an overview of the common architectures. Note that this does not want to be an extensive review or comparison of the different methods, but rather a concise reference describing the different approaches that can be employed in a given scenario. The intention is to trace a path that runs in two directions. The first is the type of data we have available (unlabeled, labeled or time-informed), which leads us to configure the learning process in different ways. The second is the increase in the expressiveness of models, moving from linear methods to models based on neural networks. Many of the latter methods can be seen as finding the best feature space that maximizes the efficacy of linear statistical methods. Learning the CVs, therefore, becomes equal to solving a variational principle, of which neural networks are excellent candidates as basis functions due to their properties of representing arbitrary functions. For this reason, in

Data	**Objective**	**Method**	**Architecture**
Unlabeled	*Maximize structural information*	PCA	linear
		AutoEncoder (AE)	**A**
		Variational AE (VAE)	**A**
		EncoderMap	**A**
Labeled	*Distinguish metastable states*	LDA	linear
		Deep-LDA	**B**
		Deep-TDA	**C**
Time-lagged	*Slow modes*	TICA	linear
		Deep-TICA/SRV	**B**
	Slow modes + structural information	Time-lagged AE (TAE)	**A**
		Variational Dynamics Encoder (VDE)	**A**

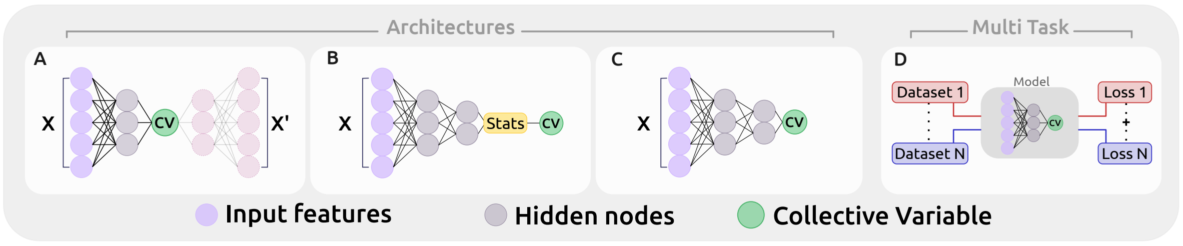


Table 5.1: (top) List of the methods implemented in `mlcolvar` for building CVs grouped accordingly to the data used for their optimization. In italics the method which can be easily implemented by following the related notes. (bottom) Sketch of the different neural network CV architectures. a) AutoEncoder, composed by an encoder that maps the input data to a latent space (the CVs) and a decoder that reconstructs it. b) Feed-forward NN used to transform the inputs before applying a statistical method (e.g., LDA, TICA). The CVs are obtained as linear combinations of the NN outputs (e.g. DeepLDA, SRV/Deep-TICA). c) Feed-forward NN whose outputs are used directly as collective variables (e.g. DeepTDA).

each section, we start discussing a linear statistical method and then move to the neural-network CVs, which are implemented in `mlcolvar`.

5.5.1 Unsupervised methods

Principal Component Analysis (PCA)[81] is a linear dimensionality reduction technique that projects the data on the principal components, i.e., the directions of maximum variance. These directions correspond to the eigenvectors of the data covariance matrix, while its eigenvalues measure the amount of explained variance. PCA is typically used to process inputs and provide whitened descriptors for other models or even directly as CVs, in which case only the very first few components are used.

AutoEncoders (AEs) are a class of NNs consisting of two main components: an encoder and a decoder (see panel a in Table 5.1 and Sec. 3.3). The encoder maps the input descriptors into a low-dimensional latent space, while the decoder performs the inverse task, i.e., reconstructing the original input from its low-dimensional representation. AEs are trained by minimizing the reconstruction loss, which is usually measured as the mean square error (MSE) between the input and its reconstructed output. The latent space thus learns a minimal set of features that maximally preserves the information on the input structures, and, in this sense, AEs can be viewed as a non-linear generalization of PCA. During the simulation, the output of the encoder is used as CV, while the decoder is used only during training[105,110].

Variational AutoEncoders (VAEs)[134] are a probabilistic variant of AEs, which mainly differ from standard autoencoders in that the data in the latent space is pushed to follow a predefined prior distribution, which is normally a Gaussian distribution with zero mean and unit variance, i.e., $\mathcal{N}(0, 1)$. This acts as a regularizer and encourages the network to learn a continuous and smoother latent space representation. This is accomplished by modifying both the network architecture and the loss function. First, the encoder learns to output the mean and variance of a Gaussian distribution, and the sample that goes through the decoder is drawn from this Gaussian. Second, the encoder/decoder parameters are optimized to minimize a linear combination of the reconstruction loss and the Kullback-Leibler (KL) divergence between the Gaussian learned by the encoder and the prior distribution $\mathcal{N}(0, 1)$. As CV, the implementation in the `mlcolvar` library then uses only the output of the encoder corresponding to the mean (i.e., ignoring the variance output).

Related models Another unsupervised learning algorithm based on NNs is the EncoderMap[97]. This method combines an autoencoder with the cost function of Sketch-map[135]. Sketch-map is a multidimensional scaling-like algorithm that aims to preserve the structural similarity, i.e. to reproduce in low-dimensional space the distances between points in the high-dimensional space. In `mlcolvar`, this can be easily implemented by subclassing the AutoEncoder CV and adding the sketch-map objective to the loss function. In a similar spirit, also the Multiscale Reweighted Stochastic Embedding[136], which combines a NN with the t-stochastic neighbor embedding (t-SNE) cost function can be implemented.

5.5.2 Supervised methods

Here, we only provide a short introduction to this class of methods that is functional to the discussion of the `mlcolvar` library. However, in Chapter 6, we expand this discussion to present more in detail some of the following methods.

Linear Discriminant Analysis (LDA)[92] is a statistical analysis method that aims to find the best linear combination of input variables that maximally separates the given classes. This is achieved by maximizing the so-called Fisher's ratio which measures the ratio of between-class variance to the within-class one. Similarly to PCA, the discriminant components are found via the solution of a (generalized) eigenvalue problem involving the within and between-class covariance matrices. If for PCA the eigenvalues represent the variance, here they measure the amount of separation between states along the relevant eigenvectors. Note that for LDA the number of non-zero eigenvalues (and hence of CVs that can be used) is $C-1$ with C being the number of metastable states. A variant, called harmonic-LDA (HLDA), has been employed for CVs design[89,98].

Neural-network based LDA (Deep-LDA) A non-linear generalization of LDA can be obtained by transforming the input features via a NN[88,137], and then performing LDA on the NN outputs (see Table 5.1, panel b). In this way, we are transforming the input space in such a way that the discrimination between the states is maximal. During the training, the parameters are optimized to maximize the LDA eigenvalues (Fisher's loss). The CV(s) are then obtained by projecting the NN output features along the LDA eigenvectors. This has the advantage of obtaining orthogonal CVs. In the case of two states, maximizing the Fisher's loss is equal to maximizing the single eigenvalue, while in the multiclass scenario, we can either maximize their sum or just the smallest one[137]. Since the LDA objective is not bounded, a regularization must be added to avoid the projected representation from collapsing into delta-like functions, which would not be suitable for enhanced sampling applications[88]. It is worth noting that, for a single CV, this result can be obtained equivalently by using the output of a neural network optimized with Fisher's criterion, which becomes the ratio of between-class and within-class variance of the output. This can be easily done with `mlcolvar` by combining a NN with a single output with the Fisher's loss.

Targeted Discriminant Analysis (Deep-TDA) In Deep-TDA[18], the discrimination criterion is achieved with a distribution regression procedure. Here, the outputs of the NN are directly used as CVs (see Table 5.1, panel c), and the parameters are optimized to discriminate between the different metastable states. This is achieved by choosing a target distribution along the CVs equal to a mixture of Gaussians with diagonal covariances and preassigned positions and widths, one for each metastable state. This targeted approach performs particularly well in the multi-state scenario, as it allows to exploit information about the dynamics of the system (i.e. a precise ordering of the states) to reduce further the dimensionality of the CVs space with respect to LDA-based methods.

This method and its variants[19] will be explained much more in detail in the next chapters, also with the help of examples and showing practical applications to challenging systems.

5.5.3 Time-informed methods

Time-lagged Independent Component Analysis (TICA) TICA[117,118] is a dimensionality reduction method that identifies orthogonal linear combinations of input features that are maximally autocorrelated, and thus represent the directions along which the system relaxes most slowly. For a given lag-time, these independent components are determined as the eigenfunctions of the autocorrelation matrix associated with the largest eigenvalues, which are connected to their relaxation timescales. These have been shown to approximate the eigenfunctions of the transfer operator[138], which is responsible for the evolution of the probability density toward the Boltzmann distribution. TICA has been applied both to enhanced sampling[139] and to extract the CVs from biased simulations[140,141].

Neural network basis functions for TICA (Deep-TICA) Similarly to LDA and Deep-LDA, we can consider a nonlinear generalization of TICA by applying a NN to the inputs before projecting along the TICA components (see Table 5.1, panel b). This corresponds to using NNs as basis functions for the variational principle of the transfer operator[118,138]. Similar architectures have been proposed, which all aim to maximize the TICA eigenvalues (typically the sum of the squares is maximized)[17,102,142]. The implementation in `mlcolvar` follows the Deep-TICA[17] method. In addition, we implemented different ways of reweighting the data[60,106,140] as well as a reduced-rank regression estimator[143] to learn more accurate eigenfunctions.

Time-lagged AutoEncoders Another class of methods that work with pairs of time-lagged data is based on autoencoding NNs. Time-lagged autoencoders (TAEs)[83] have the same architecture as standard ones, but the encoder/decoder parameters are optimized to find a compressed representation capable of predicting the configuration after a given lag-time rather than reconstructing the inputs. Thus, the decoder takes the CV at time t and uses it to reconstruct the time-lagged inputs at $t + \tau$. In `mlcolvar`, this can be simply achieved using an `AutoEncoderCV` but with a dataset in which the output targets are time-lagged configurations.

Similarly, one can also consider a time-lagged variant of the variational autoencoder, as done in the Variational Dynamics Encoder (VDE) architecture[120]. To build a VDE, one needs to simply optimize a time-lagged VAE with an additional term in the loss function which maximizes the autocorrelation of the latent space. It is worth noting that both TAEs and VDEs tend to learn a mixture of slow and maximum variance modes[119], at variance with the non-linear generalizations of TICA which only learn slowly decorrelating modes.

5.6 Application examples

As discussed, the methods implemented in the library have been extensively applied to study a wide range of atomistic systems. In this section, we provide a didactic overview of the library's capabilities by focusing on a simple toy model. Specifically, we showcase an example for each of the CV categories presented above with the intent of highlighting the versatility of the implementation and how different workflows may be chosen depending on the available data. For atomistic examples, we refer the reader to the next chapters or to the documentation of the `mlcolvar` library, which includes notebooks demonstrating the use of the library with systems taken from the literature on data-driven CVs.

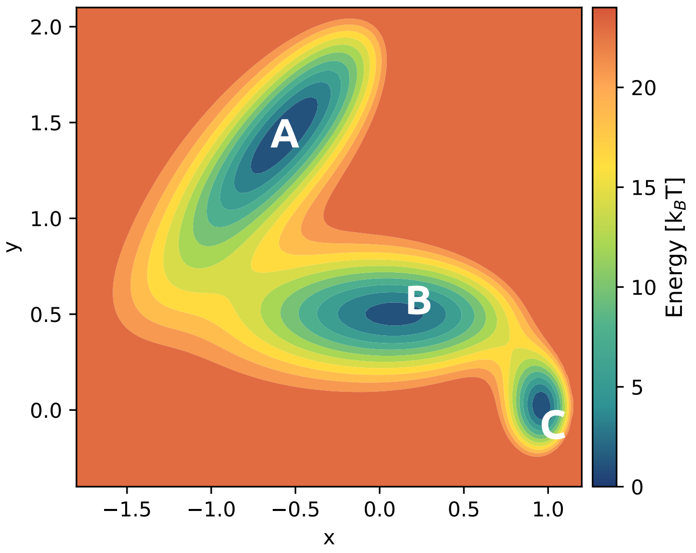


Figure 5.3: Three-state potential energy landscape for the movement of a particle in two dimensions, used as a toy model in sec. 5.6. The analytic expression of the potential energy, obtained by modifying the Muller-Brown potential to have three states, is available on the GitHub repository in the Jupyter notebooks implementing the examples.

In the following, we consider a particle moving in two dimensions under the action of the three-state potential depicted in Fig. 5.3 built out of a sum of Gaussians. The input features of the NNs are taken to be the x and y position of the particle. The activation function is chosen to be the shifted softplus[51], which is well suited for differentiating the CVs. The parameters are optimized via gradient descent using the ADAM optimizer with a learning rate of 10^{-3}. The dataset is split into training and validation, and early stopping is used to avoid overfitting. All the simulations are performed using the simple Langevin dynamics code[144] contained in the `ves`[126] module of PLUMED, and the biased simulations are performed using the OPES[65] method with a pace of 500 steps, the automatic bandwidth selection, and a barrier parameter equal to 16 $k_B T$.

5.6.1 Unsupervised setting: state discovery

We first start with the least informed scenario in which we only have data limited to a single metastable state and aim at exploring the potential energy surface. Starting from the first unbiased data in state A, we adopt an iterative procedure akin to the MESA[105] method in which we train an `AutoEncoderCV`, perform a short biased run, and add the collected configurations to the training dataset. This workflow is repeated until necessary, e.g. all the states have been discovered. We performed 16 iterations of 10^5 steps each and reported the results in Fig. 5.4. In panel a, we colored the sampled regions according to the iteration in which they were first visited, while in panel b, we report the time evolution of the variable y, which is able to distinguish the different states. At first, the AE drives the sampling along the direction of maximum variance of state A (blue dots), but after a few iterations, it is able to discover state B as well. Finally, after 10 iterations, state C is also visited, and from there the system visits all three states, although only one or two transitions are observed per iteration.

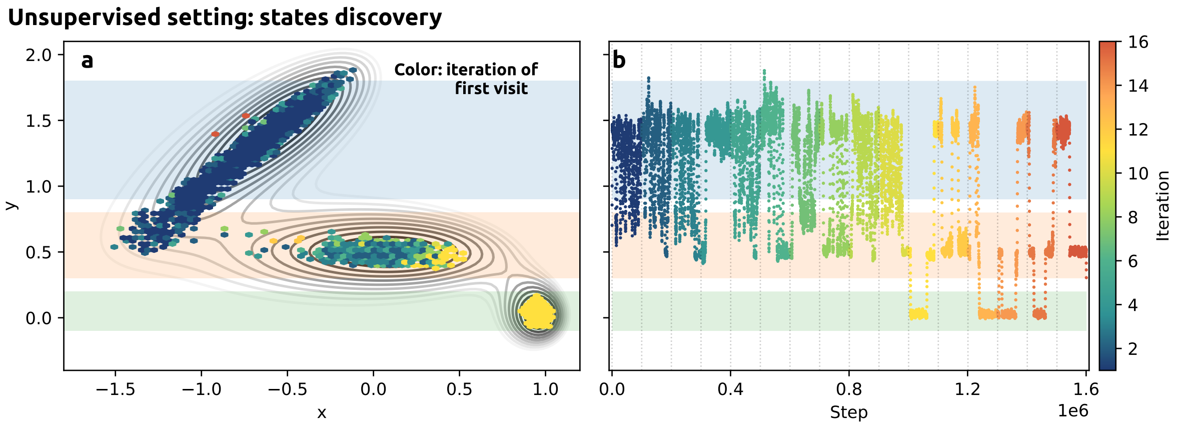


Figure 5.4: MD simulations of the toy model in Fig. 5.3 biasing the autoencoder CV. The colored shaded regions in the background indicate the three metastable states. **a**) Exploration of the energy landscape. Each region of the space is colored according to the iteration in which it was visited for the first time, as reported in the colorbar. The contour lines denote the isolines of the 2D potential energy. **b**) Time evolution of the y coordinate along different iterations of the autoencoder CV. The points are colored according to the iteration number, as reported in the colorbar.

5.6.2 Supervised setting: CVs as classifiers

Once the three states of the system have been discovered, we can step up to more refined CV models based on supervised learning and aim for a comprehensive sampling of the free energy landscape, e.g. to converge a free energy profile. For each of the three states, we run short unbiased MD runs and collect labeled samples of the three states (see Fig. 5.5c). Then, we train a `DeepTDA` CV with a single component and a target distribution of three equidistant Gaussians (ordered as A, B, C for increasing CV values). This is motivated by the consideration that during the exploratory phase described above only transitions of the kind $A \leftrightarrow B$ and $B \leftrightarrow C$ are observed. Enhancing the sampling along the DeepTDA CV results in multiple transitions between the different states as reported in fig. 5.5d. We observe that the sampling follows approximately the minimum free energy path connecting the states. Furthermore, the multiple transitions induced by this trial CV allow converging the free energy profile (reported in the inset of panel d).

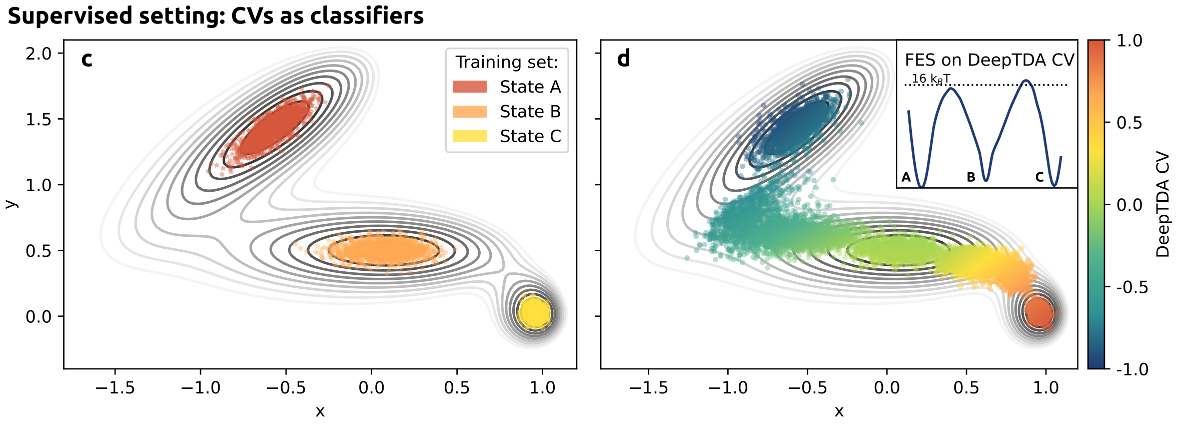


Figure 5.5: **c**) Training set of the three-state Deep-TDA CV, each state is depicted in a different color according to the legend. **d**) Points sampled with MD simulations of the toy model in Fig. 5.3 biasing the Deep-TDA CV. The colormap provides the CV value for the points and the inset reports the free energy surface (FES) computed from the sampled data projected along the Deep-TDA CV.

5.6.3 Time-lagged setting: improving CVs

One scenario that often occurs in practice is when we have a suboptimal simulation capable of promoting just a few transitions before getting stuck due to some slow orthogonal mode not being accelerated. In this context, time-informed methods such as `DeepTICA` can be used to extract (approximations of) the slowly decorrelating modes that hamper simulations' convergence and thus design better CVs. As an example, we took a simulation performed using the position `y` as the CV, which is the prototype of an order parameter capable of distinguishing the metastable states but not the transition states between them. As can be seen from Fig. 5.6e, this results in poor sampling. Following the Deep-TICA scheme, we optimized a new CV based on this data, which, when biased, leads to a much more diffusive sampling capable of converging the simulation in a fraction of the time (Fig. 5.6f).

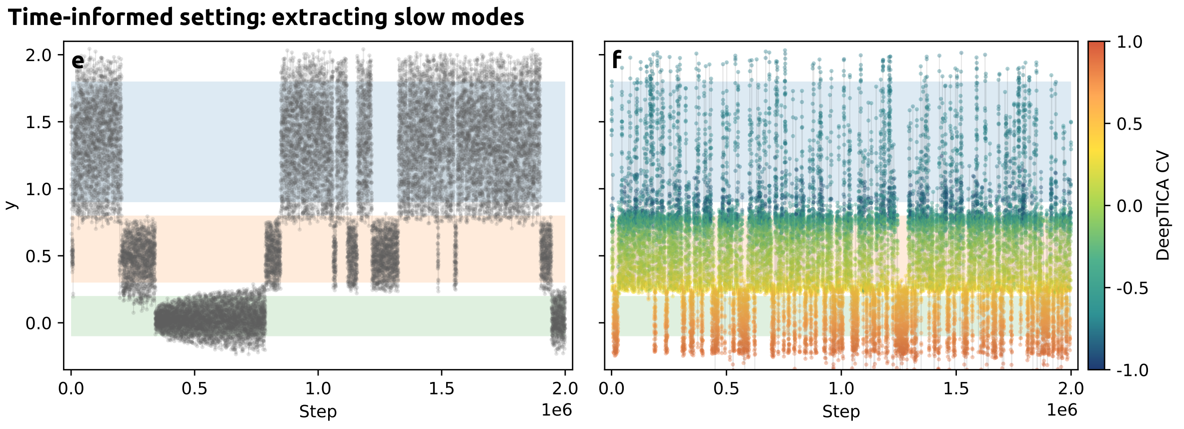


Figure 5.6: Time series of MD simulations of the toy model in Fig. 5.3, the colored shaded regions in the background indicate the three metastable states e) Suboptimal OPES simulation biasing the y coordinate, used as training set for the Deep-TICA CV. **f**) Time series of the simulation biasing the Deep-TICA CV as well as the previous (static) bias. The colormap reports the Deep-TICA CV value.

5.6.4 Multitask learning: a semi-supervised application

Finally, we show how the library can be used to combine different data in a multitask framework to improve data-driven CVs. To this end, we take as an example the two datasets from sec. 5.6.2. The first is the labeled dataset which has been used to construct the Deep-TDA CV (Fig. 5.5c), while the second is composed by the unlabeled configurations generated by biasing the Deep-TDA CV (Fig. 5.5d). We define a single `MultiTaskCV` that, as schematically depicted in Fig.5.7, is composed of an autoencoder optimized upon two different losses: the first is the reconstruction loss on the unlabeled dataset (blue path in Fig.5.7), and the second is the TDA loss acting on the latent space optimized on the labeled dataset (red path Fig.5.7). In this way, one obtains a semi-supervised approach in which each component benefits from the other. Indeed, the autoencoder CV is regularized to distinguish the metastable states, and at the same time, the classifier CV is informed about regions outside the local minima.

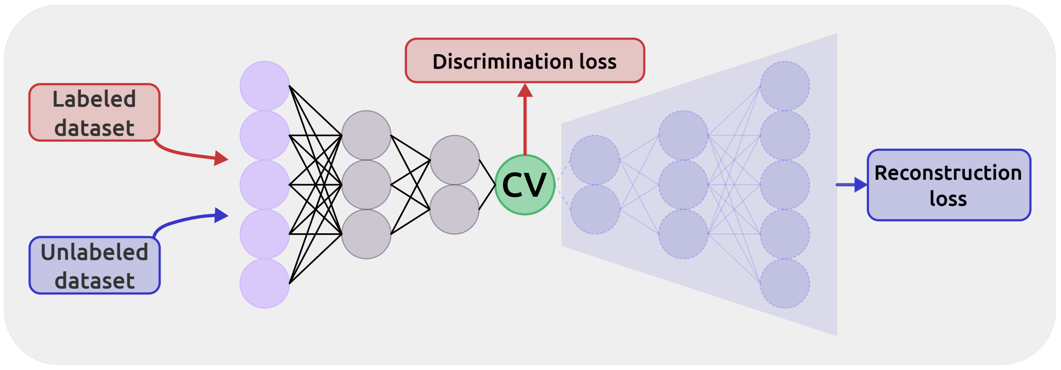


Figure 5.7: Example of a multitask CV employed for semi-supervised learning. The model is based on an autoencoder architecture and is optimized on two separate datasets with different criteria. Data from an unlabeled dataset (blue path) contributes to an unsupervised loss function (i.e., reconstruction MSE loss), which depends on the output of the decoder. Data from a labeled dataset (red path) contributes to a supervised loss function that depends directly on the CVs space, which is the output of the encoder (e.g., TDA loss).

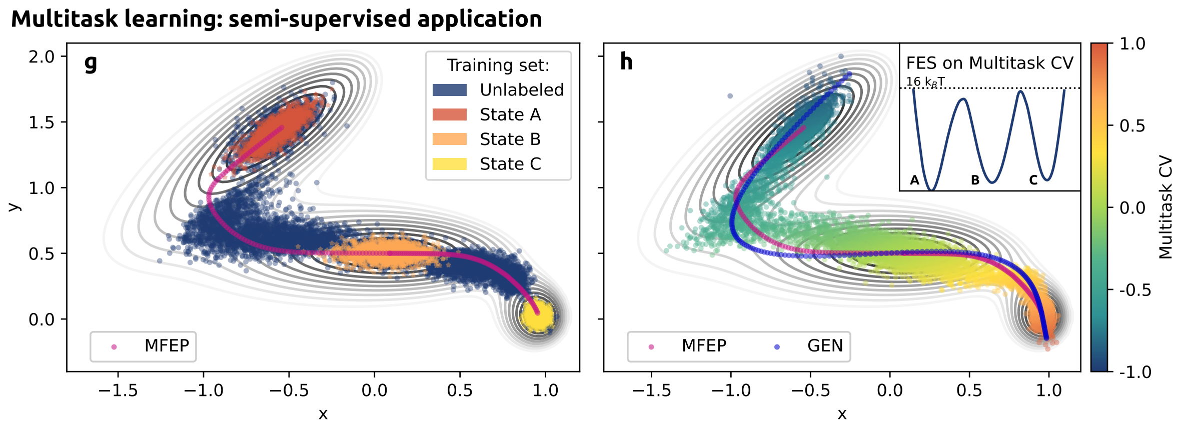


Figure 5.8: **g**) Training set for the semi-supervised multitask CV (see Fig. 5.7). **h**) Points sampled with MD simulations of the toy model in Fig. 5.3 biasing the MultiTask CV. The colormap provides the CV value for the points and the inset reports the free energy surface (FES) computed from the sampled data projected along the multitask CV. The blue dotted path (GEN) is generated by applying the decoder to a collection of evenly spaced samples in the latent space. In both **g** and **h**, the purple path represents the minimum free energy path (MFEP).

As a result, when this multitask CV is employed to enhance sampling, it follows the minimum free energy path (purple dotted line in Fig. 5.8h) more closely than the simulation using only Deep-TDA, which was exploring higher-energy pathways (Fig. 5.8g). Moreover, the multitask approach allows for inspecting the CV space by using the model in a generative way. To this end, we examine how a path connecting the states in the latent space is mapped by the decoder into the reconstructed input space. This generates a set of configurations shown in Fig. 5.8h that traces the free energy path between the three states remarkably well.

Chapter 6

Collective variables as classifiers

In the previous chapters, we have seen that one of the possible approaches that can be followed for the data-driven design of collective variables is the one of supervised learning. In this chapter, we will present in more detail the class of *classifier-like* collective variables, which are optimized based on the classification of different sets of labeled data. More in detail, we will discuss the Deep Targeted Discriminant Analysis (Deep-TDA) method[18], and we will introduce its *transition path informed* variant[19] (TPI-Deep-TDA), providing in both cases examples on some prototypical systems.

6.1 Learning collective variables from the metastable states

In Chapter 2, we have seen that most of the interesting processes in nature are rare events that are characterized by infrequent transitions between some long-lived metastable states. We have also seen that physical descriptors, such as distances, angles, and coordination numbers, are used to characterize the different states of the system. In the case of rare events, configurations generated in unbiased MD runs in the metastable basins are typically well-separated in the multidimensional space of such descriptors and when such descriptors are combined into low dimensional CV, a necessary, if not sufficient, condition is that data belonging to different basins remain separated.

This motivated the use of a classifier-like approach for CV design[15,98], considering that these models are specifically designed to distinguish between classes of data. Classifiers are indeed trained on datasets in which the data are labeled according to some classes, for example, images of different types of fruit, and aim at predicting the correct class for the given samples. Typically, in machine learning, such methods are used for *prediction*, whereas, in the case of CV design, we are more interested in their ability to encode the structure of our dataset into the latent space of the model output, which is our CV.

With these ideas in mind, it has been suggested that *discriminant analysis* could be helpful in the context of CVs design[15,88,98] starting from data harvested in the metastable basins. As we have anticipated in the previous chapter, the first applications of such an approach relied on a linear method from classical statistics, which is the time-honored Linear Discriminant Analysis[92] (LDA). Given a set of input features $\mathbf{x}$ of dimension d, this aims at finding the best linear combination $\mathbf{W}^*$ of them that can maximize the *separation* between the given N_c classes, which in our case are the metastable states. The separation is expressed in terms of the so-called *Fisher's ratio*, which depends on the *within-class* $\mathbf{S}_w$ and *between-class* $\mathbf{S}_b$ scatter matrices of our data. The first depends on the fluctuations within the single basins and is given by the average of the classes covariances $\mathbf{S}_i$

$$\mathbf{S}_w = \frac{1}{N_c}\sum_i^{N_c} \mathbf{S}_i \tag{6.1}$$

The second, on the other hand, is the covariance of the class means μ_i and measures the fluctuation between classes

$$\mathbf{S}_b = \frac{1}{N_c}\sum_i^{N_c} (\mu_i - \mu)(\mu_i - \mu)^T \tag{6.2}$$

where μ is the average of the N_c class means. The Fisher's ratio that has to be maximized to find the best projection $\mathbf{W}^*$ is then expressed as

$$\underset{\mathbf{W}}{\text{argmax}} \frac{\mathbf{W}\mathbf{S}_b\mathbf{W}^T}{\mathbf{W}\mathbf{S}_w\mathbf{W}^T} \rightarrow \mathbf{W}^* \tag{6.3}$$

Even if this approach[98], and its *harmonic variant*[89] (HLDA), has been successfully applied for the design of CVs for simple systems, it still presents some limitations. Indeed, a linear scheme is often not flexible enough to discriminate well between the states when we deal with complex systems and high-dimensional data. Moreover, in LDA, the discriminant components are found via the solution of a (generalized) eigenvalue problem involving the covariance matrices, which somehow limits the number of input features.

To alleviate these limitations, the LDA approach has thus been generalized with the hybrid Deep-LDA approach[88,137], in which a strong nonlinear feature is added to the standard LDA scheme with the help of NNs. In Deep-LDA, the high-dimensional input descriptors of the model are indeed preprocessed by a NN into a low-dimensional latent space where the application of the LDA scheme is simplified as the NN objective function directly depends on the Fisher's ratio. This way, one obtains a nonlinear and more powerful generalization of the

LDA method that allows designing CVs for applications as diverse as chemical reactions[88], crystallization of solids[111], and ligand binding[113].

This empirical evidence is somehow reassuring for our purposes. Indeed, even if the idea of building a CV based on a discrimination criterion *only* may sound reasonable, there is not any *a priori* guarantee that it should be effective because it does not encode any explicit information about the transition state of our process. Furthermore, the hybrid Deep-LDA approach is rather attractive because, in theory, there is no limit to the number of descriptors that can be used.

Nonetheless, this method still presents room for improvement both from the conceptual and practical point of view. For example, we mentioned that Deep-LDA still requires the solution of the LDA eigenvalue problem in the latent space returned by the NN. However, in principle, this step could be skipped by expressing the CV as the output layer of the NN and by maximizing a *Fisher-like ratio* directly in the CV space, thus simplifying the overall procedure.

Another limitation of LDA and Deep-LDA is related to the case of systems with multiple metastable states ($N_c > 2$). In this case, Deep-LDA always forces us to use a CV that has $N_c - 1$ components, as it is canonically required to account for all the possible transitions between those states. However, in many cases, this is not necessary as the transitions between some of the states are not physically allowed. This is the case, for example, of a chemical reaction in which the same reactants can evolve into different products that cannot interconvert or of sequential reactions that present a series of stable intermediates. In such cases, the dimensionality of the CV space can be reduced beyond the standard $N_c - 1$ *rule*. However, this requires some previous knowledge of the system and a different and more flexible approach to the discrimination task we have used so far.

In the next sections, we will present this approach, also showing how it can help us in addressing the last limitation of Deep-LDA, which is its lack of (explicit) information about the transition state region.

6.2 Deep Targeted Discriminant Analysis

If we think about the shape of the CV space that we aim at learning with a discrimination approach, we can soon realize that it is rather simple to imagine, as it basically needs to guarantee that the metastable states are well-defined when projected in the CVs space. For example, if we consider the case of a two-state system A and B, we would like to obtain a CV space in which the projections of the data from two basins are well distinguished from each other. In other terms, this means that we want their probability distributions in the CV space not to overlap with each other. From this perspective, we can reframe the Deep-LDA approach in a simpler way. Instead of starting from some data and optimizing a model over a discrimination criterion to find a proper CV space, we can choose our *target* CV space and train a model to map the descriptor space into such a space, which is the idea behind the Deep Targeted Discriminant Analysis[18] method (Deep-TDA).

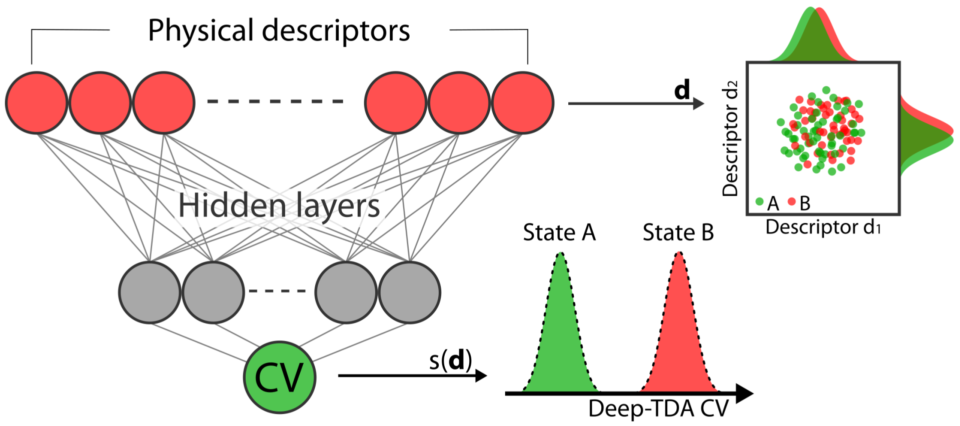


Figure 6.1: Schematic representation of the construction of the Deep-TDA CV. A set of physical descriptors $\mathbf{d}$ is fed as input of a feed-forward NN whose output layer directly gives the CVs. A series of nonlinear transformations through the NN hidden layers progressively compress the dimension using as objective function the distance in the projected space from a target distribution in which the states are well-discriminated. To further illustrate the process in the case of two states A and B, we have a sketch of the intricate data distribution with respect to some of the physical descriptors from the input set $\mathbf{d}$ and, below, the well-resolved distributions in the Deep-TDA CV $s(\mathbf{d})$.

6.2.1 Methods

Two-state scenario As for Deep-LDA, given two states A and B, characterized by a set of descriptors $\mathbf{d}$, we want to construct a CV s by finding a one-dimensional projection along which the two states are well-discriminated. If we skip the linear step of Deep-LDA by directly expressing the CV as the output of the NN, we can reformulate the optimization criterion of the CV s to make it more direct and flexible. To do so, we train the NN so that the distribution of the metastable states along the CV follows a preassigned bimodal distribution (see Fig. 6.1). The target distribution to be used in this framework can be, in principle, of any type as long as it guarantees proper discrimination between the states. A natural choice is to use as a target two Gaussian distributions of preassigned positions and widths, such that the A configurations are distributed according to one of the two Gaussians and B configurations according to the other. The ability of a target distribution to discriminate between the two states can be measured with a Fisher-like ratio

$$F = \frac{(\mu_A - \mu_B)^2}{\sigma_A^2 + \sigma_B^2} \tag{6.4}$$

where μ_A and μ_B are the average values for the two states in the CV space and σ_A^2 and σ_B^2 their variances. However, as F rapidly grows to very large values when the separation between the states is increased, we prefer to use $\Delta = \sqrt{F}$ as a parameter to characterize our target function. Because, in the end, we want to use s as a CV and not only as a discriminator, the choice of such a parameter requires some experimentation. Indeed, to be effective, s needs to assume properly interpolating values when dealing with transition states. In practice, this implies that Δ can be neither too small nor too large. If Δ is too small, the CV is not able to discriminate correctly between states. On the other hand, if Δ is too large, s cannot describe well the transition state. As a rule of thumb, we found that values in the range $25 < \Delta < 50$ are appropriate choices.

As far as the objective function is concerned, we could have enforced the target distributions using Kulback-Leibler divergences. However, in the case of Gaussians, this would have been an overkill and thus, we simply impose that the two distributions have preassigned positions and widths. Thus, each state k contributes to the loss function two terms, one that enforces its center L_k^μ and the other its width L_k^σ

$$L_k^\mu = (\mu_k - \mu_k^{tg})^2 \qquad L_k^\sigma = \left(\sigma_k - \sigma_k^{tg}\right)^2 \tag{6.5}$$

The whole loss function is then given by the linear combination of all the contributions.

$$L = \sum_k \left[\alpha L_k^\mu + \beta L_k^\sigma\right] \qquad k = A, B \tag{6.6}$$

in which the hyperparameters α and β are chosen such that the two terms are scaled to roughly the same order of magnitude at the earlier stages of the optimization. Conveniently, the loss function in Eq. 6.6 tends to zero as we approach the target distribution, thus convergence is simple to monitor.

Multi-state scenario The extension to the multi-state case is straightforward in the Deep-TDA framework. We recall that in a system with N_s states in the general case, one needs to define $(N_s - 1)$ CVs. We build the CVs by imposing a target that is a linear superposition of N_s multivariate Gaussians with diagonal covariances. Each Gaussian is then defined by $N_\rho = (N_s - 1)$ CV positions and covariances, thus leading to the following loss function

$$L = \sum_k^{N_s} \sum_\rho^{N_\rho} \left[\alpha L_{k,\rho}^\mu + \beta L_{k,\rho}^\sigma\right] \tag{6.7}$$

where ρ are the components of the CVs space. The location of the different Gaussians is arbitrary, but as before, attention has to be paid to the relative distances and widths.

Thanks to the targeted approach, there are circumstances in which, using Deep-TDA, one can reduce the number of CVs. This reduction is possible if the reaction involves a series of steps that one can sequentially align, for example, if it involves a series of stable intermediates or presents different products that cannot interconvert. In these situations, the topology of the problem is actually linear and the number of CVs can be reduced to one with clear computational advantages. Of course, this reduction is only possible because we are using information on the dynamics of the system.

6.2.2 Examples

For didactical purposes, we report first the application of Deep-TDA to the study of alanine dipeptide in vacuum. This simple but instructive example will confirm that, in the two-state case, Deep-TDA works just as well as Deep-LDA, as to be expected. After dealing with two-state test cases, we apply Deep-TDA to the multi-state case of the hydrobromination of propene and of a double intramolecular proton transfer reaction, where the advantage of using Deep-TDA becomes clearly apparent. In the following paragraphs, we focus on showcasing the Deep-TDA application to prototypical systems, thus leaving more technical details out of discussion. However, such details are synthetically reported in Sec. 6.2.3.

Alanine Dipeptide As it is well known, the torsional equilibrium of alanine dipeptide at room temperature is a two-state system. In this small molecule, the two Ramachandran angles ϕ and ψ are known to be good CVs and we could have used these angles as descriptors. However, following Ref.[88], we tested the robustness of the method by using as descriptors the 45 distances between the heavy atoms.

The isolines of Deep-TDA CV are reported in panel **a** of Fig. 6.2 and are equivalent to those of Deep-LDA and follow the underlying energetic landscape in the Ramachandran plot even far from the training basins. Using Deep-TDA CV and OPES[65] (see Sec. 2.3) effectively encourages many transitions between the two metastable basins. The performances are comparable to the nearly ideal set of Ramachandran angles as confirmed by the consistency of the estimates of the FES along the ϕ torsion angle and of the free energy difference between the basins, as shown in panels **b** and **c** of Fig. 6.2.

Hydrobromination of propene Having assessed that Deep-TDA is a good substitute for Deep-LDA, we now turn to discuss cases in which Deep-TDA can give a competitive advantage. One of them is the case of the hydrobromination of propene. Starting from the same reactants (R), this chemical reaction can lead to two products, identified as Markovnikov (M) and Anti-Markovnikov (A).

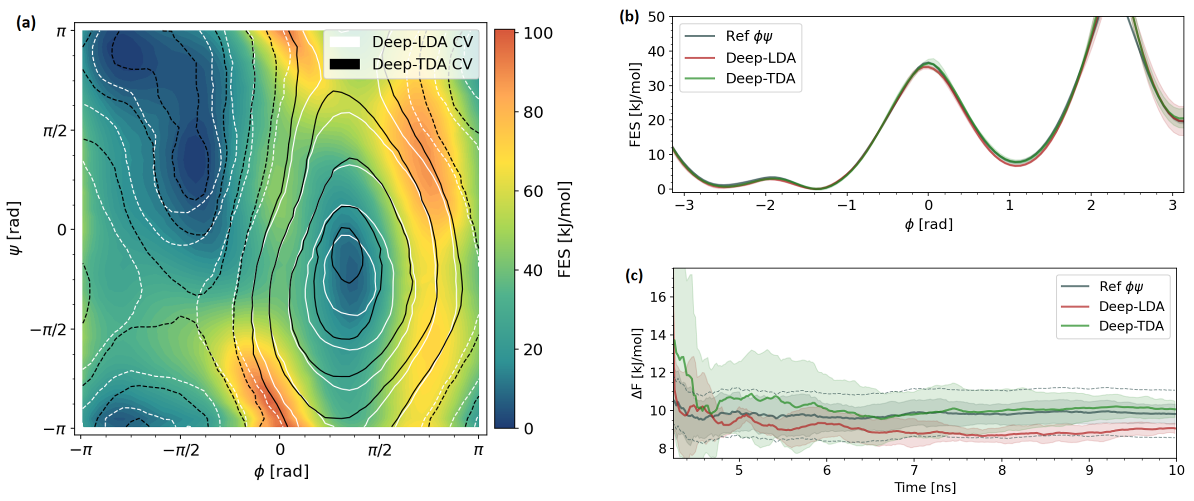


Figure 6.2: Results of OPES enhanced simulations of the folding of alanine dipeptide. **(a)** Comparison of the isolines of Deep-LDA (white) and Deep-TDA (black) CVs on top of the energetic landscape in Ramachandran angles $\phi\psi$ plane. Solid lines are for positive CV values, dashed lines for negative ones. **(b-c)** Comparison of energy estimates from OPES simulations using Deep-LDA, Deep-TDA or a reference $\phi\psi$ as CVs. The results are averaged on five independent simulations for each CV to get the mean solid line and the standard deviation error bars. **(b)** FES profile estimates along the ϕ Ramachandran angle. **(c)** ΔF between the metastable basins estimates, obtained as functions of the simulation time. The dashed lines give the $\pm 0.5 K_b T$ range on the reference curve.

This is clearly a multi-state scenario and we can tackle this problem in two ways. One in which, following the prescription of LDA, we determine two CVs (i.e., $N_s - 1 = 2$). The second takes advantage of the fact that products A and M interconvert with a very low probability. Thus, one is in a scenario where the problem can be mapped into a linear sequence of reactions $A \leftrightarrow R \leftrightarrow M$ and reduce the number of CVs from two to one, as anticipated earlier.

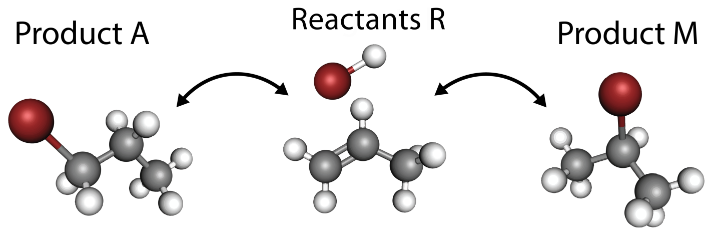


Figure 6.3: Reaction scheme for the hydrobromination of propene. The same reactants R can lead to two possible products, anti-Markovnikov A and Markovnikov M, depending on the addition position of the halide.

We compare here the two possibilities, starting from the standard two CVs approach. In this case, we have chosen a very simple target that is a sum of multivariate Gaussian functions with diagonal covariances placed at the vertices of an equilateral triangle. The positions and widths of the Gaussians were chosen so as to satisfy the criteria described earlier (see Sec. 6.2.1).

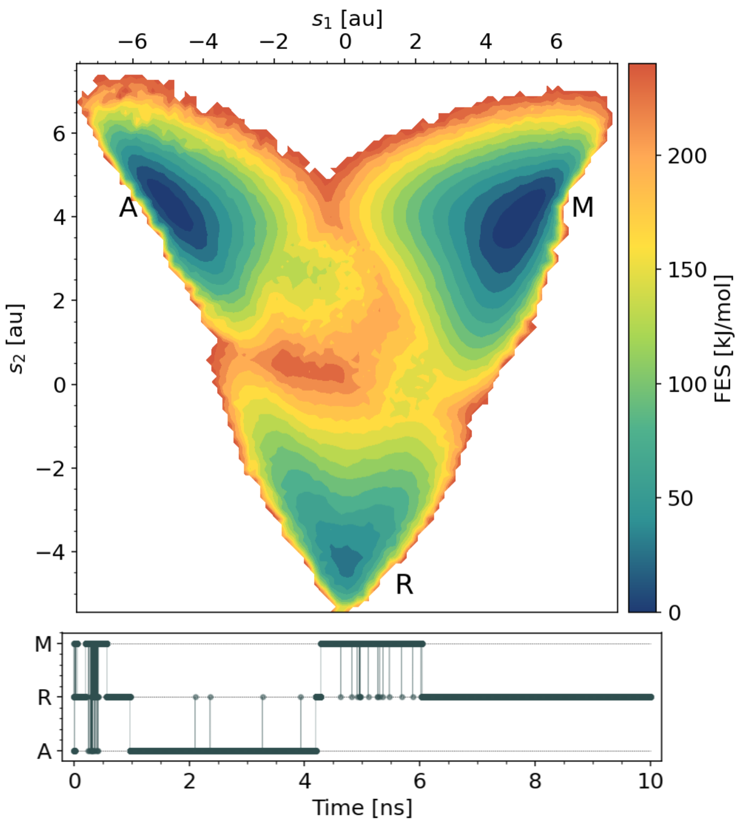


Figure 6.4: Upper panel: FES estimate for the hydrobromination of propene in the plane of the Deep-TDA CVs s_1 and s_2. The basins are labeled as anti-Markovnikov A, reactants R and Markovnikov M. In the SI, an error analysis can be found. **Lower panel**: State occupation during a single 10ns OPES run with the two-dimensional Deep-TDA CV. As indicator functions, we use, for each metastable state, its descriptors density, modeled as a Gaussian mixture[145]. The density was trained on the same unbiased data and uses the same descriptors employed in the generation of the Deep-TDA CV.

As in Ref.[146], we used as descriptors the contact functions

$$c_{ij}(r) = \frac{1 - \left(\frac{r}{\sigma_{ij}}\right)^n}{1 - \left(\frac{r}{\sigma_{ij}}\right)^m} \tag{6.8}$$

where r are the pairwise atomic distances and σ_{ij} is the typical bond length between the involved species i and j. In our case, the σ_{ij} parameters of Eq.6.8 were set as σ_{BrC}=1.9Å, σ_{CC}=1.7Å, σ_{BrH}=1.4Å, σ_{CH}=1.2Å, whereas the exponents have been set to n=6 and m=8 to have a smooth behavior over a wide range of distances. The contacts between hydrogen atoms have not been used in the training. In order to focus the sampling on the relevant reaction, we also break the permutational symmetry and allow only the H in the initial HBr molecule to react[89,145] (see Sec. 6.2.3). Even if this implies a limitation of the permutational entropy of our system, the effects of such an approximation are negligible when compared to the large free energy barriers associated with the reaction at the studied temperature.

The two-dimensional free energy surface is shown in the upper panel of Fig. 6.4 and reflects the expected free energy order of the different states. However, the non-linearity of the CVs distorts the FES and makes it difficult to identify the transition paths, eventually requiring further analysis in the fashion of Ref.[147]. In fact, an analysis of the OPES trajectories showed that the system was able to undergo many transitions involving the reactants basin R but never made a direct transition between A and M, which thus appears extremely unlikely. Thus, the only relevant transitions were of the type $A \leftrightarrow R \leftrightarrow M$ (see the lower panel in Fig. 6.4). This suggested using a one-dimensional CV thus we

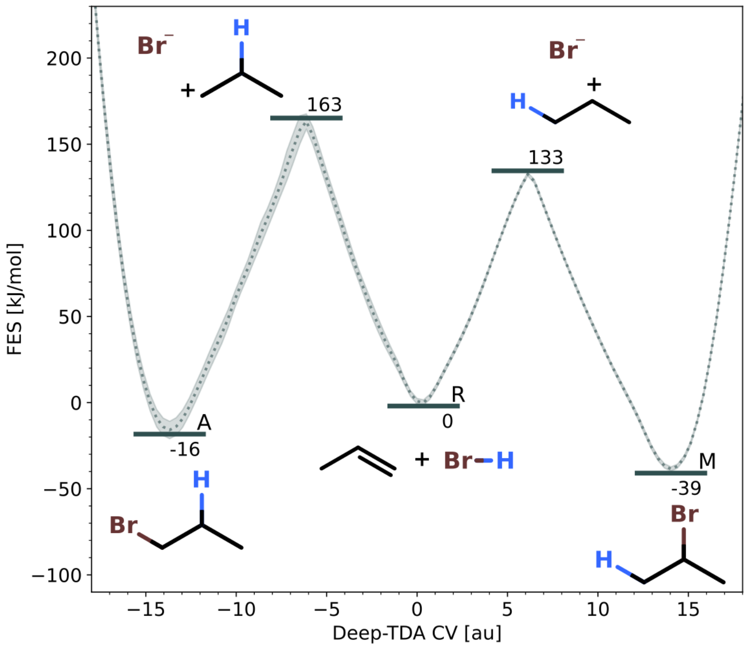


Figure 6.5: FES estimate profile for the multi-state hydrobromination of propene reaction projected along the Deep-TDA CV, with the indication of the different metastable and transition states. The dotted line gives the average FES profile with the related error. The free energy of each state is expressed in kJ/mol.

designed a one-dimensional target that at the center has the reactants R and at the sides, the products A and M and used our Deep-TDA method. When used in OPES, the CV was able to drive transitions from reactants to products without attempting any $A \leftrightarrow M$ direct transition. The resulting one-dimensional FES in Fig. 6.5 is more easily readable with respect to the two-dimensional one in Fig. 6.4, with the different states and transition states clearly marked as done in the standard representation of chemical processes. This representation also illuminates the fact that the selectivity towards the Markovnikov product is due to the kinetics of the process rather than to its thermodynamics.

Double proton transfer in diamino-benzoquinone The one-dimensional approach can also be applied to chemical reactions in which a number of reactive steps take place in a well-defined order. One such case is the double proton transfer of 2,5-diamino-1,4-benzoquinone. For this reaction, a two-step mechanism has been proposed[148,149] with a stable intermediate (I) between the reactant (R) and product (P), which are respectively the enol and keto stable forms of the compound (see Fig. 6.6). To determine the one-dimensional Deep-TDA CV, we have thus designed a target with the correct ordering of the states so as to account for transitions of the type $\mathrm{R} \leftrightarrow \mathrm{I} \leftrightarrow \mathrm{P}$. As descriptors, we have used the heavy atom coordination numbers to preserve all the symmetries of the system.

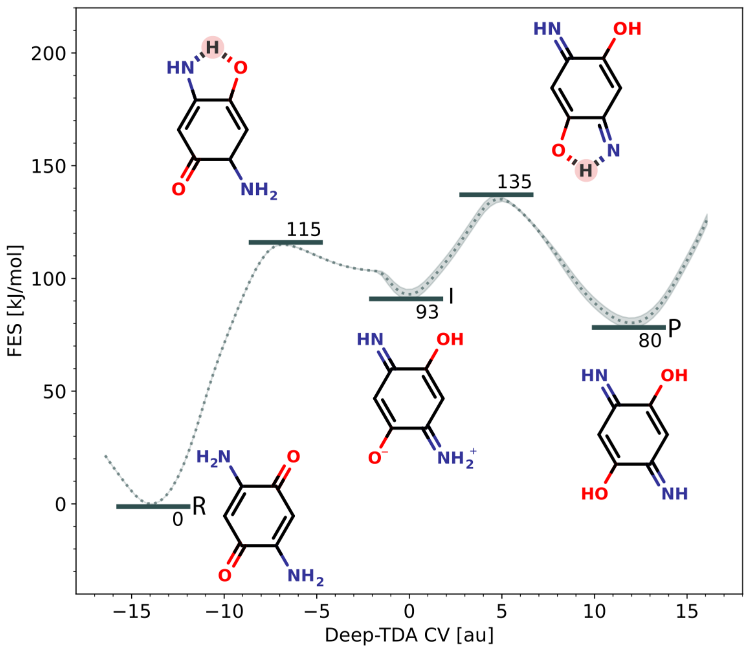


Figure 6.6: FES for the double intramolecular proton transfer reaction in 2,5-diamino-1,4-benzoquinone projected along the Deep-TDA CV. The metastable states are, from left to right: keto form (**R**), intermediate (**I**) and enol form (**P**). The dotted line gives the average FES profile with the related error. The free energy of each state is expressed in kJ/mol.

When used in biased OPES simulations, also here the Deep-TDA CV was able to promote efficiently the different reaction steps. Also in this case the resulting one-dimensional FES (Fig. 6.6) clearly shows the three metastable states and gives a neat representation of the reaction profile with free energies compatible with those obtained from static calculations[149]. In biased runs, the system was also able to explore less likely rotational isomers, different from the dominant ones in the training set, with a slight effect on the shape of the I and P basins. The possible relative orientations of the heteroatoms hydrogens are sketched in Fig.6.7. The most sampled conformers adopt the A-type configuration, as it is the one that favors the most the exchange of hydrogen between the two functional groups and is also stabilized by a hydrogen bond.

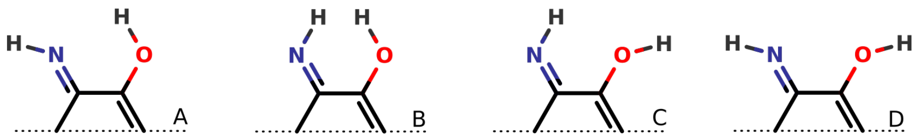


Figure 6.7: Sketch of the possible relative dispositions of the hydrogens in E and I metastable states. A-type conformers are the most likely.

6.2.3 Additional computational details

Alanine Dipeptide The alanine dipeptide simulations have been carried out using the GROMACS-2019.6[128] MD engine patched with PLUMED2.7[124,125]. We have used the Amber99-SB[45] force field with a 2fs timestep. To sample the NVT ensemble, we have used the velocity rescaling thermostat[37] set at 300K. In the OPES simulations driven by the Deep-TDA CV, we have used `SIGMA=0.2` to match the standard deviation of the basins we had set in the target distribution, `BARRIER=30` kJ/mol and `PACE=500`. We used the 45 distances between heavy atoms as input of a NN with structure {45, 24, 12, 1} nodes/layer. The target function parameters were $\mu_A^{tg} = -7$, $\mu_B^{tg} = 7$ and $\sigma^{tg} = 0.2$. The resulting Δ separation was thus $\Delta = 50$. The hyperparameters for the loss function (Eq.6.6) were $\alpha = 1$ and $\beta = 250$. We performed five statistically independent 10ns simulations to compute the average values reported in Sec. 6.2.2.

Hydrobromination of propene The hydrobromination of propene simulations have been carried out using the CP2K-7.1[129] software package patched with PLUMED2.7[124,125] at PM6 semi-empirical level. The integration step was 0.5fs and we used the velocity rescaling thermostat[37] set at 300K with a time constant of 100fs.

In Fig. 6.8, the reactant molecules with the atomic labels used to identify the atoms in PLUMED are reported. Following Ref.[145], harmonic restraints have been applied on some of the distances to guarantee that only H^{11}, Br, C^1 and C^2 atoms participate in the reaction and prevent undesired reactions to occur:

- To keep the molecules close to each other and in suitable conditions to react, some restraints have been applied on the distances most affected by the reaction (d_{1-10}, d_{1-11}, d_{2-10}, d_{2-11}, d_{10-11}) in the form $k(d_{ij} - d_0)^2$, with $k = 250$KJ/mol and d_0=3Å.
- To prevent the formation of elemental hydrogen, all the hydrogen-hydrogen distances d_{ij} (i, j in `ATOMS=3,4,6,7,8,9,11`) have been constrained to be greater than 1.4Å with a restraint in the form $k(d_{ij} - d_0)^2$, with $k = 250$kJ/mol and d_0=1.4Å.
- *Non-reactive* hydrogens (`ATOMS=3,4,6,7,8,9`) have been *locked* to the respective carbon atoms (`ATOMS=1,2,5`) with restraints on all the distances d_{HC} in the form $k(d_{HC} - d_0 + o_i)^2$, with $k = 450$kJ/mol, d_0=1.4Å and o_i=0.2.

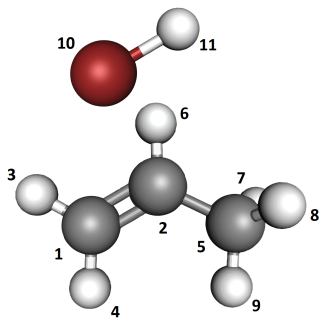


Figure 6.8: Sketch of the molecules involved in the hydrobromination of propene with the atomic labels.

For the training of the two-dimensional Deep-TDA CV, we have used the 34 contacts described in Sec. 6.2.2 as input of a NN with structure {34, 24, 12, 2} nodes/layer. The target function parameters in the $\{s_1, s_2\}$ CVs space were $\mu_A^{tg} = [-5, 4.3]$, $\mu_R^{tg} = [0, -4.3]$, $\mu_M^{tg} = [5, 4.3]$, to have the three states on the vertices of an equilateral triangle with side 10, and $\sigma^{tg} = [0.2, 0.2]$, giving $\Delta = 35$ for each pair of states. The hyperparameters for the loss function (Eq.6.7) were $\alpha = 1$ and $\beta = 250$. In the OPES simulations, we have biased the 2D Deep-TDA CVs using `SIGMA=0.2,0.2` to match the standard deviations along the two CVs of the basins in the target distribution, `BARRIER=240` kJ/mol and `PACE=500`. The results reported in Sec. 6.2.2 were averaged over four statistically independent 10ns simulations.

For the training of the one-dimensional Deep-TDA CV, we have used the same 34 contacts described above as input of a NN with structure {34, 24, 12, 1} nodes/layer. The target function parameters were $\mu_A^{tg} = -14$, $\mu_R^{tg} = 0$, $\mu_M^{tg} = 14$ and $\sigma^{tg} = 0.2$, giving $\Delta = 50$ for each of the products with respect to the reactants basin. The hyperparameters for the loss function (Eq.6.6) were $\alpha = 1$ and $\beta = 250$. In the OPES simulations, we have biased the Deep-TDA CV using `SIGMA=0.2` to match the standard deviation of the basins in the target distribution, `BARRIER=240` kJ/mol and `PACE=500`. The results reported in Sec. 6.2.2 were averaged over six statistically independent 10ns simulations.

Double proton transfer in diamino-benzoquinone The intramolecular proton transfer simulations have been carried out using the CP2K-7.1[129] software package patched with PLUMED2.7[124,125] at PM6 semi-empirical level. The integration step was 0.5fs and we used the velocity rescaling thermostat[37] set at 300K with a time constant of 100fs.

The coordination numbers used for the training have been computed with PLUMED according to Eq. 6.8. They have been computed on each heavy atom with respect to the element couples: CC, CN, CO, CH, NH, OH. The σ_{ij} parameters in Eq. 6.8 were set as σ_{CC}=1.7Å, σ_{CN}=1.7Å, σ_{CO}=1.6Å, σ_{CH}=1.2Å, σ_{NH}=1.1Å, σ_{OH}=1.2Å, whereas the exponents have been n=6 and m=8 to have a smooth behavior over a wide range of distances.

For the one-dimensional Deep-TDA CV training, we used these 28 coordination numbers as input for a NN with structure {28, 24, 12, 1} nodes/layer. The target function parameters were $\mu_R^{tg} = -14$, $\mu_I^{tg} = 0$, $\mu_P^{tg} = 14$ and $\sigma^{tg} = 0.2$, giving $\Delta = 50$ for the R and P states with respect to the intermediate I. The hyperparameters for the loss function (Eq.6.6) were $\alpha = 1$ and $\beta = 250$.

In the OPES simulations, we have biased the Deep-TDA CV using `SIGMA=0.2` to match the standard deviation of the basins in the target distribution, `BARRIER=120` kJ/mol and `PACE=500`. The average values in Sec. 6.2.2 are obtained by averaging over ten statistically independent 10ns simulations.

6.3 Including information about the transition paths

In previous sections, we have mentioned that one of the main limitations of discriminant-based CVs is related to their dataset being limited to the metastable states only. This approach is surely convenient in practice, as the training data from the metastable basins are easy to collect by means of standard MD simulations, and we have shown that, nonetheless, it is also effective in driving transitions across the phase space region related to the transition state that is not explicitly represented in the dataset.

However, in that region, our model is forced to extrapolate due to the absence of training data and is thus much less accurate. It is indeed well-known that neural networks can be universal interpolators but they can eventually perform worse when it comes to extrapolation[150–152]. In the case of CVs for enhanced sampling, this can easily lead to sub-optimal CVs with slower convergence and less focused sampling. For example, a Deep-CV trained only on data from the minima may drive successful transitions, but they may not follow the true *minimum free energy path*, as it does not encode any explicit information about it.

These considerations led us to propose improving the quality of the Deep-

TDA CV by including in the training set information related to the transition state obtained from reactive trajectories. This led us to the *transition path informed* (TPI-Deep-TDA) variant[19] of the method, which we will introduce and compare with the standard Deep-TDA approach in the next sections.

6.3.1 Methods

In Deep-TDA and other discriminant-based CVs, one assumes that a model that is trained only to discriminate between the metastable states will also provide a meaningful description of the transition state region. This is, in general, not a bad assumption since the extrapolation takes into account the fact that the CV has to smoothly join the metastable state regions. However, sometimes, the performance of such a CV can be far from optimal. To improve the quality of the Deep-TDA CV, we propose to incorporate information from the *transition path ensemble* (TPE) obtained from reactive trajectories. The CV design protocol is depicted in Fig. 6.9 and described below for an example two-state system.

- **Step 1:** We collect data on a set of descriptors by running unbiased simu-

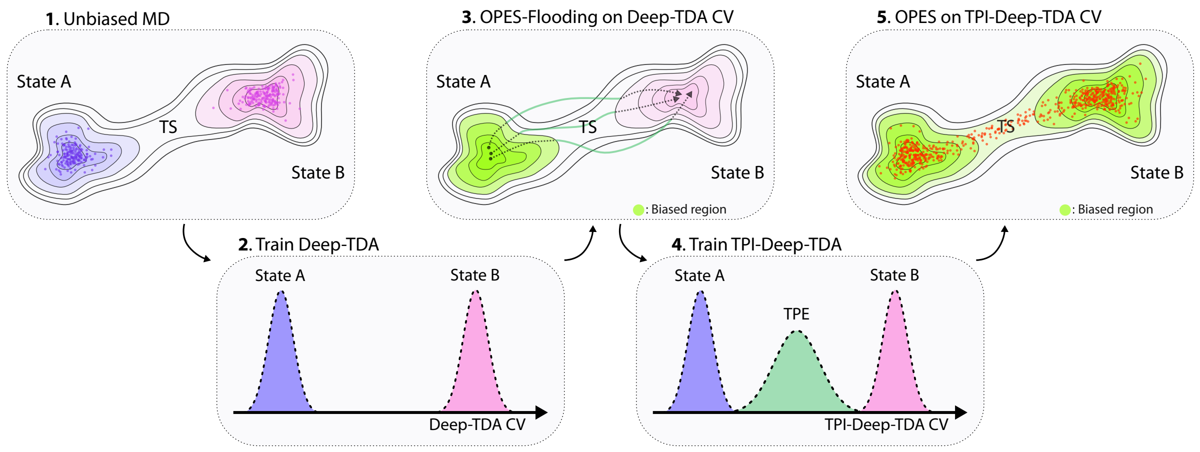


Figure 6.9: A schematic representation of the TPI-Deep-TDA CV construction for a two-state system. **1.** A set of physical descriptors **d** is collected by unbiased MD runs in the metastable basins of the system **2.** A Deep-TDA CV is trained such that the data from states A and B are distributed according to two Gaussians in the CV space **3.** Unbiased reactive trajectories are sampled using a set of OPES-Flooding[69] simulations along the Deep-TDA CV. The bias (green shade) is deposited only in one of the basins and excluded from the TS region. Only the sections of each reactive path that fall outside the metastable basins are taken as part of the TS-region dataset (marked in green) **4.** A second TPI-Deep-TDA CV is trained to fit the TPE data distribution to a third wider Gaussian (painted in green) between the metastable states A and B **5.** The TPI-Deep-TDA CV is biased in OPES[65] to drive transitions between A and B applying bias (green shade) along the TS path.

lations in the metastable basins of the system (panel 1).

- **Step 2:** The data collected in the metastable states are used to train a standard Deep-TDA[18] CV as discussed in Sec. 6.2 (panel 2).
- **Step 3:** Using the Deep-TDA CV thus generated, we perform a set of OPES-Flooding[69] (see Sec. 2.3) simulations and harvest several reactive trajectories. We select from the reactive trajectories only those configurations that lie outside the metastable basins (panel 3).
- **Step 4:** The new configurations thus obtained are added to the initial dataset and we train the new TPI-Deep-TDA CV. To do so, we modify the target distribution used for the Deep-TDA CV in Step 2 by adding a third wider Gaussian, placed between the ones related to the metastable states, and optimizing the NN to fit the TPE data distribution in the CV space to such Gaussian (panel 4).
- **Step 5:** The TPI-Deep-TDA CV is finally used to perform OPES simulations to calculate the free energy landscape (panel 5).

The rationale behind the choice of different target widths for the Gaussian functions is to reflect the structure of the physical data. Let us, for simplicity, consider only a two-state case, A and B. In this case, the data can be divided into three groups: those that belong to basin A, those that belong to basin B, and those that belong to the transition paths ensemble (TPE). The data in A and B are localized at different positions in the high-dimensional descriptors space. Therefore, a mapping into separately localized Gaussians is a natural one. In contrast, the TPE data are spread across the region in between the metastable states. To best mimic this structure while still sticking to a Gaussian representation, we introduce a third Gaussian centered between the A and the B Gaussians. The corresponding width σ_{TPE} is larger than those of A and B, bridging the intermediate region, and has a negligible overlap with either A and B Gaussians. We also emphasize that such overlaps, if too large, would reduce the ability of the CV to distinguish the various states and consequently degrade its performance.

At this stage, we point out that the transition path ensemble used for training our proposed CV can, in principle, be sampled using various alternative schemes, including transition path sampling[153], aimless shooting[154], transition interface sampling[155], metadynamics of paths[156], etc. It should also be possible to collect configurations specific to the TS region by applying the transition-state-oriented bias we propose in Chapter 8. Nonetheless, here we choose to use the OPES-Flooding[69] algorithm (see Sec. 2.3) as it does not require any pre-existing knowledge on the location of the TS, a piece of information that is not often readily available. In addition, by proceeding this way, we also get the advantage of conveniently recovering the kinetics of the process, as we will briefly discuss in Sec. 6.4.

6.3.2 Examples

For didactical purposes, we first compare the performances of the standard Deep-TDA CV with the TPI variant on a simple but instructive toy model system, namely the Müller-Brown potential. To further prove the potential advantages of this refined approach, we then apply it to the study of the conformational equilibrium of chignolin protein in an explicit solvent environment and to a binding-unbinding process of a small molecule and a calyxarene host.

As done in Sec. 6.2, we first center our discussion on showcasing the important features of the method and report the most technical details of the following results in a separate section (see Sec. 6.3.3).

Müller-Brown Potential The two-dimensional Müller-Brown potential is often used to test the efficiency of enhanced sampling methods. As we shall see below, it is a system in which a standard Deep-TDA CV performs rather well. Still, the TPI extension of the Deep-TDA approach to include transition path data is able to further improve the CV performance and speed up convergence.

In order to understand this different behavior, we can first compare the training sets used for the two CVs reported in Fig. 6.10, which consists of the x and y coordinates. In the case of standard Deep-TDA, the data set is indeed limited to the metastable basins (pink and blue in Fig. 6.10), while in the case of the TPI variant, it also spans the transition region (green in Fig. 6.10). To obtain

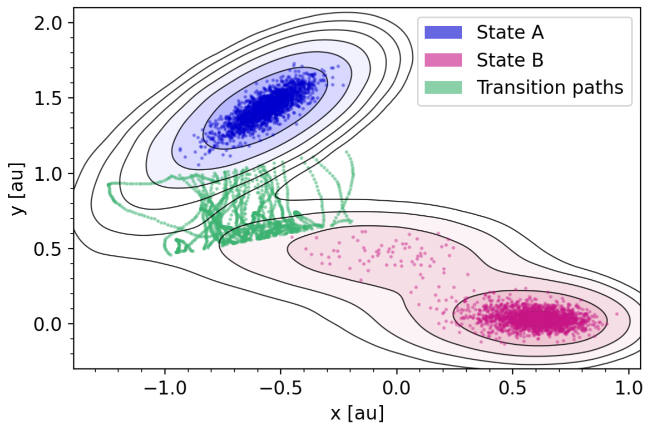


Figure 6.10: Scatter plot of the configurations for the training of deep learning collective variables for the Müller-Brown potential. The training data for the standard Deep-TDA CV is limited to the metastable states (blue and pink), whereas for the TPI-Deep-TDA CV, configurations from the transition path ensemble (green) are also included. The isolines of the true free energy surface are depicted in black.

the additional data on the TPE, 30 OPES-Flooding simulations were performed using the Deep-TDA CV s with a filling factor $\Delta E = 10\ k_BT$. From the sampled configurations, only those not belonging to any of the two basins (i.e., whose s

values were in the interval $-3.5 < s < 2.0$) were considered to be part of the TPE (reported in green in Fig. 6.10).

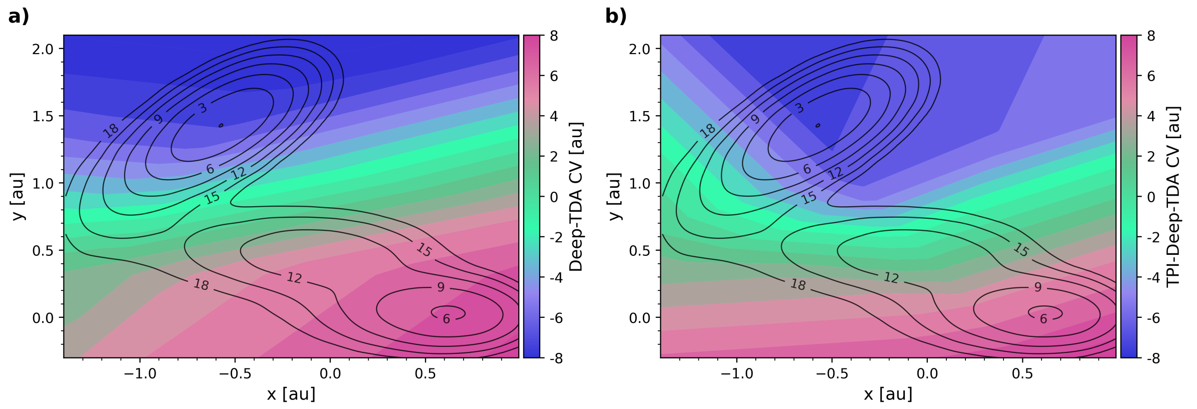


Figure 6.11: Contour plots of **(a)** standard Deep-TDA and **(b)** TPI-Deep-TDA CVs for the Müller-Brown potential. These CVs describe poorly the regions that are far from the training data. This confirms the inability of NN to extrapolate to data-poor regions. However, since such regions are not physically interesting, this is of little practical consequence.

This improvement in the training set results in an improvement of the CV quality as well, as we can appreciate by comparing Deep-TDA and TPI-Deep TDA CVs isolines (see Fig.6.11). In the case of Deep-TDA, the CV can distinguish well between the metastable states but it does not follow precisely the gradient of the underlying energy landscape. (Fig.6.11, panel a). On the other hand, the TPI-Deep-TDA CV isolines follow the free energy gradient more closely (Fig.6.11, panel b). As a consequence in the first case, the system is pushed by the bias to explore a larger than necessary portion of the transition state region and less relevant regions (Fig.6.12, panel a). In the TPI variant, on the other hand, the nature of the transition state region is more closely encoded in the CV, thus facilitating the transitions between the metastable states and the sampling points closer to the minimum free energy path (Fig.6.12, panel b). This allows reaching convergence slightly faster using TPI-Deep-TDA (Fig.6.12, panel c). The free energy difference between the two basins has been computed by integrating the free energy landscape in both sides of the dividing surface $s = 0$

$$\Delta F = -k_B T \ln \frac{\int_0^{s_{max}} \exp\left[-F(s)/k_B T\right] ds}{\int_{s_{min}}^{0} \exp\left[-F(s)/k_B T\right] ds} \tag{6.9}$$

where $F(s)$ is the free energy surface along the CV s and s_{max} and s_{min} are the minimum and maximum values of the explored CV space. This approach is used for computing the ΔF for the next systems as well, considering that in both Deep-TDA CV and TPI-Deep-TDA CV, the two metastable basins are situated symmetrically around $s = 0$ by construction.

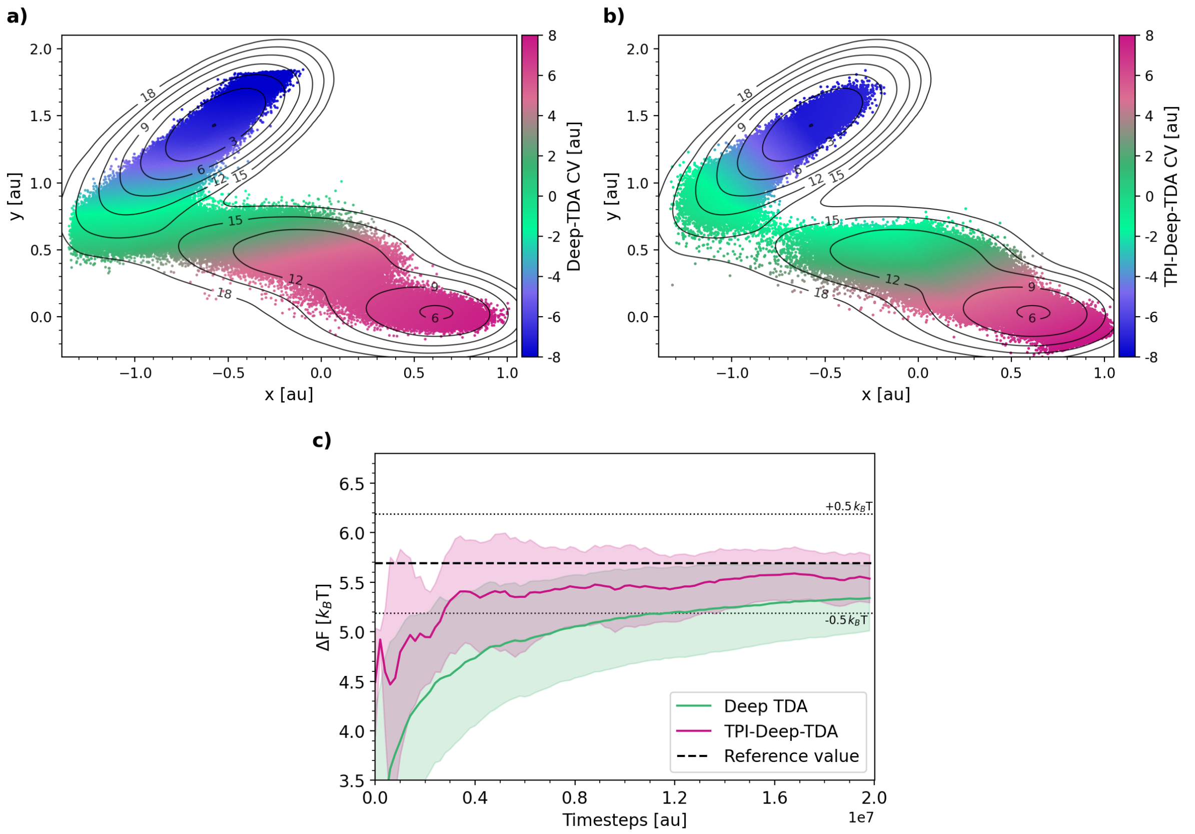


Figure 6.12: Scatter plot of the points visited performing OPES simulations using **(a)** Deep-TDA and (**b**) TPI-Deep-TDA CVs on the Müller-Brown potential, whose isolines are given in black. The colormap indicates the value of the corresponding CV. The density of the scatter plot near the TS region is lower in the case of TPI-Deep-TDA CV indicating that the transitions take place through the minimum energy path.**(c)** Convergence of the free energy difference ΔF between the basins with simulation time. The solid line and the shaded region report respectively the average and standard deviation computed from three independent trajectories. The reference value of 5.69 k_BT was obtained by numerical integration[68] and the dotted lines mark the $\pm 0.5 k_BT$ range around that value.

Chignolin folding The folding and unfolding of chignolin in an explicit solvent environment is a popular benchmark for testing computational methods on proteins. Conveniently, this system was also studied by means of long unbiased simulations with dedicated supercomputers[115] that provide precious reference values and unbiased trajectories.

To study this system, we trained two TPI-Deep-TDA CVs using as input descriptors the contacts between the 10 α-carbons (C_α) of the peptidic chain. This results in a better convergence with a tighter confidence interval if compared to the standard Deep-TDA (Fig. 6.13). In less than 200 ns, the free energy difference between the folded and the unfolded state converged within one k_BT of the results obtained from $\sim$ 100 μs long unbiased simulation[115]. In contrast,

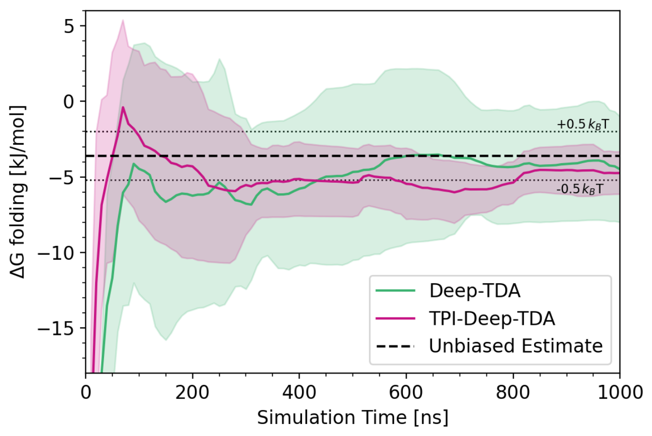


Figure 6.13: Comparison of the convergence with simulation time of the free energy of Chignolin folding as obtained from OPES simulation with Deep-TDA and TPI-Deep-TDA CVs. The solid line and the shaded region report, respectively, the standard deviation computed from three independent trajectories. The dashed line reports the reference value obtained with long unbiased simulations[115]. The $\pm 0.5 k_B T$ range around this value is marked by the thin dotted lines.

using standard Deep-TDA CV, it took around 500 ns to reach convergence. Moreover, the uncertainty in the free energy difference between the folded and the unfolded states was larger than that of TPI-Deep-TDA. One of the reasons for the increased efficiency is due to the fact that when using TPI-Deep-TDA, the exploration of the FES is limited to the minimum free energy path as observed in the 2D model potential.

It should be noted that in practical systems, it is often difficult to run multiple independent simulations to assess the uncertainty of the predicted free energy landscape. Due to the tighter confidence interval, the use of TPI-Deep-TDA CV increases the chance of obtaining a free energy surface that closely resembles the true free energy landscape of the system for the given force field.

Ligand receptor binding Lastly, we tested our TPI-Deep-TDA approach for the binding of G2 guest to the tetramethylated octa-acid (OAMe) host used in the Statistical Assessment of Modeling of Proteins and Ligands[157] (SAMPL5) challenge.

To describe the binding transition, we adopted the same 13-dimensional descriptor set proposed in previous work from our Group by Rizzi et al.[113,158] in which more details can be found. To summarize them, this set of descriptors consisted of the linear distance between the binding pocket of the host and the center of mass of the guest (z), the water coordination of 8 fixed virtual atoms placed along the path of ligand (un)binding, and the water coordination of 4 heavy atoms of the G2 guest molecule.

Unlike the previous two examples, the ligand-receptor distance (z) is used as the CV for the OPES-Flooding simulations, instead of the Deep-TDA CV. One reason for this choice is to showcase the versatility of our approach. Another reason is that this distance CV is the most obvious and straightforward description of the ligand binding process and is commonly used in the literature.

Following the TPI-Deep-TDA approach, the binding free energy converges rapidly (i.e., less than 50 ns) to a value of -6.08 ± 0.78 kcal/mol similar to the result obtained by Rizzi et al.[113] of -6.19 ± 0.08 kcal/mol (Fig. 6.14, panel b) who studied this system in great detail using identical force field parameters.

The computed binding free energy is not in perfect agreement with the experimental results[157] of -5.04 kcal/mol, likely due to the approximate nature of the empirical force field. However, we could reproduce the result of Rizzi et al.[113] with reasonable accuracy, confirming the usefulness of our CV, and we also observed frequent transitions between the bound and the unbound states during the course of the OPES simulations.

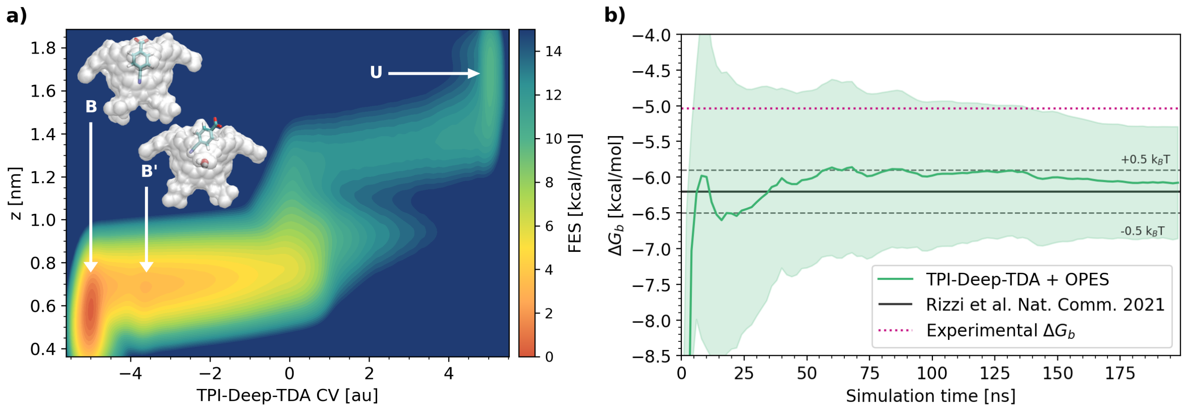


Figure 6.14: **(a)** 2D free energy surface projected along the TPI-Deep-TDA CV and the vertical component z of the distance between the ligand and the binding site. Three metastable states are visible: unbound (U), bound (B), and semi-bound (B'). Representative structures of the B and B' states are provided in the inset. **(b)** Convergence with time of binding free energy of G2 guest in OAMe octa-acid host using TPI-Deep-TDA CV. The solid line and the shaded region report, respectively, the standard deviation computed from three independent trajectories. As a reference, we report in black the computational value obtained in Ref.[113] with a similar setup and in pink the experimental value from Ref.[157]. The dashed lines give the $\pm 0.5 k_B T$ range on the computational reference value.

In panel a of Fig. 6.14, we show the average free energy surface from three independent simulations, projected along the ligand-receptor distance (z) and the TPI-Deep-TDA CV. As in Ref.[113], we found that there is a bound state (B) and an intermediate semi-bound configuration (B'). This latter state appears as a shallow minimum close to the B state. In the B' state, the binding pocket is occupied by one water molecule, which prevents the guest from attaining the minimum energy bound configuration (Fig. 6.14, inset of panel a). The nature of this state is discussed in detail in previous work[113] and the free energy difference between this shallower minimum and the true bound state was reported to be approximately 2 kcal/mol, which is in agreement with our results.

6.3.3 Additional computational details

Müller-Brown Potential The simulations were for the Müller-Brown potential performed using Langevin dynamics based on the Bussi-Parrinello algorithm[144] as implemented in the `ves_md_linearexpansion`[126] module of PLUMED[124,125]. The damping constant in the Langevin equation was set to 10/time-unit. The time unit was defined arbitrarily and corresponds to 200 timesteps and natural units ($k_B T = 1$) were used in all the calculations.

The initial dataset for the training of a standard Deep-TDA CV was collected with unbiased simulations in the two metastable states for 10^6 time steps. From each trajectory, a total of 10000 points were randomly selected. As descriptors and input of the NN, we used the x and y coordinates. We used a NN with structure {2, 6, 4, 1} nodes/layer. The target mean and width of distributions corresponding to the two minima (A and B) were $\{\mu_A = -7.0, \mu_B = 7.0\{$ and $\{\sigma_A = 0.5, \sigma_B = 0.5\{$. The hyperparameters α and β were chosen as 1 and 250. The Adam[79] optimizer was used with a learning rate of 10^{-3} and a L_2 regularization term with a hyperparameter of 10^{-5}.

To train the TPI-Deep-TDA CV, we used a total of 1714 configurations distributed roughly in equal numbers between the metastable states and the TPE configurations obtained from OPES-Flooding trajectories. The target mean and width of the Gaussians for the training were selected as $\{\mu_A = -7.0, \mu_{TPE} = 0.0, \mu_B = 7.0\}$ and $\{\sigma_A = 0.5, \sigma_{TPE} = 1.0, \sigma_B = 0.5\}$. The NN training procedure was identical to what was done in the Deep-TDA case.

A set of three OPES simulations per each Cv was conducted with identical parameters for comparison, setting $\Delta E = 20$ kJ/mol and running for 2×10^7 time steps. During the simulations, half harmonic walls have been applied at $s = \pm 8.5$ for the Deep-TDA CV, and $s = \pm 8.0$ for the TPI-Deep-TDA CV to restrict the exploration of excessively high energy conformations.

Chignolin folding For the study of folding and unfolding of chignolin, we performed our simulations using the CHARMM22* force field[159], and the solvent has been modeled by the CHARMM TIP3P[160] force field, sharing the same setup used for long unbiased simulations on this system[115] to have a direct comparison with those results. For the same reason, we kept the simulation condition consistent with that work. All simulations were performed in NVT ensemble at 340K with GROMACS v2021.5[128] patched with PLUMED v2.9[124,125].

To collect the initial dataset for the training of the CVs, we computed the 45 α-carbons pairwise contacts over 50ns-long unbiased molecular dynamics simulations in the folded and unfolded states. In the case of the unfolded simulation, a harmonic wall was applied along the RMSD order parameter at RMSD = 3 Å to prevent a spontaneous folding of the protein. The initial Deep-TDA CV was trained on these two states setting the target distribution means and widths as $\{\mu_A = -7.0, \mu_B = 7.0\}$ and $\{\sigma_A = 0.2, \sigma_B = 0.2\}$. For each state, 48000 structures were sampled and used to train a NN with structure {45, 24, 12, 1} nodes/layer. For the optimization, the Adam[79] optimizer was used with a learning rate of 0.001 and L_2 regularization of 10^{-5}.

Using the Deep-TDA CV s, OPES-Flooding simulations were performed starting from the folded state to sample the TPE. The barrier parameter ΔE was set to 15 kJ/mol and the excluded region boundary s_{exc} was -3.0. As shown in Ref.[69], the use of a relatively low barrier parameter ΔE is indeed necessary to avoid biasing the transition state so that we can recover unbiased transition pathways and kinetics. From these simulations, a total of 20 successful transitions were observed from which structures corresponding to $-5.0 < s < 5.0$ were associated with the TPE and used for training the TPI-Deep-TDA CV.

The TPI-Deep-TDA CV was then trained following an identical protocol as the standard Deep-TDA CV except for the addition of the TPE-related Gaussian in between the folded and the unfolded states. A wider distribution (i.e., $\mu_{TPE} = 0.0$ and $\sigma_{TPE} = 1.5$) was used for the TPE Gaussian as these structures do not belong to a single metastable state but include multiple configurations sampled during the entire pathway.

We then compared the performance of the Deep-TDA and TPI-Deep-TDA CVs by performing three independent 1 μs long OPES simulations for each case. Unlike the OPES-flooding, a barrier parameter (ΔE) of 30 kJ/mol was used in these simulations to ensure adequate exploration of both free energy minima within the simulation timescale.

Ligand receptor binding For the study of the G2-OAMe ligand binding, we performed our simulations using the Generalized AMBER Force Field (GAFF)[161] used in previous work[113] to have a direct comparison with those results. For the same reason, we kept the simulation condition consistent with that work. All simulations were performed in NVT ensemble at 300K with GROMACS v2021.5[128] patched with PLUMED v2.9[124,125].

For training the Deep-TDA CV, we performed 50 ns of unbiased MD simulations in both bound and unbound states. To sample the transition path ensemble OPES-Flooding simulations were conducted starting from the bound configuration using the z distance as CV. A total of 13 successful transitions from the bound to the unbound state were sampled from which 170,000 data points were extracted according to the criteria 0.8 nm $< z < 1.3$ nm. An equal number of points were sampled from the unbiased simulations in each metastable state leading to a total of 510000 points as training data. For the TPI-Deep-TDA model, we used the same descriptors based on water coordination proposed by Rizzi et al.[113] (see Sec. 6.3.2) as input of a NN with structure {12, 8, 6, 4, 1} nodes/layer. The target mean and width of the distributions were chosen as follows: $\{\mu_A = -5.0, \mu_{TPE} = 0.0, \mu_B = 5.0\}$ and $\{\sigma_A = 0.2, \sigma_{TPE} = 0.5, \sigma_B = 0.2\}$. For the optimization, the Adam[79] optimizer was used with a learning rate of 0.0025 and L_2 regularization of 10^{-5}.

Subsequently, the two-dimensional CV space consisting of the z and the TPI-Deep-TDA order parameter was used for enhanced sampling and three independent 200 ns long OPES simulations were performed with $\Delta E = 40$ kJ/mol.

6.4 A note on computing kinetics with Deep-TDA and OPES-Flooding

Even if kinetics is not the main focus of this Thesis and we will thus not discuss it in depth, it is still worth mentioning an interesting application of Deep-TDA in this sense. However, for further information on kinetic calculations from biased simulations, we refer the reader to the good and recent review of this topic in Ref.[162].

In Sec. 2.3, we introduced the *flooding* variant of the enhanced sampling method OPES, which is used to efficiently collect unbiased reactive trajectories. These can be used for the computation of the kinetics of a given process of interest, or, as we have seen in the previous section (see Sec 6.3), to enrich our training set for CV design with configurations from the transition path ensemble.

We have seen that this method is based on the *conformational flooding* approach and requires the definition of an *excluded region* s_{exc} parameter to avoid the deposition of the bias on the transition region to be effective[69,162]. The choice of this parameter may not be particularly important if we are only interested in sampling reactive paths to collect configurations for a dataset, but it is crucial for kinetics calculations. It has indeed been shown in Ref.[69] that a careful choice of the s_{exc} value and of the maximum deposited bias is of the utmost importance for efficient and accurate results in this sense.

However, with most CVs, also the ones obtained with the help of machine learning, the determination of such a parameter is far from trivial or, at least, far from optimal. In most cases, it is indeed rather difficult to identify *a priori* the position of the transition state region along a CV, and thus one often reverts to trial and error, a practice that, in the absence of a reference value, could be quite tedious. In contrast, in Deep-TDA, the shape of the CV space is determined beforehand with the choice of the target, and it is thus much easier to set a reasonable value for s_{exc} on the first try. This reasoning is even more true in the case of the TPI variant, in which the localization of the transition state region in the CV space is explicitly enforced during the training. However, we have seen that this approach needs a dataset that is more laborious to obtain, possibly making the eventual gains negligible when compared to the additional effort.

To demonstrate the possible gains from the use of OPES-Flooding along Deep-TDA CVs, we can consider the case of the folding of chignolin and the (un)binding process we presented in Sec. 6.3.2. In that case, we used OPES-Flooding simulations to obtain configurations from the TPE. From the same unbiased reactive trajectories, we computed the kinetics for the two systems as described in Ref.[69] and the results are reported in table 6.1 alongside some reference values.

System	Number of runs	Mean first passage time (τ)	p-value	95% Confidence interval	Acceleration factor	Reference value
Chignolin (unfolding)	20	3.08 μs	0.665	1.94 - 4.70 μs	316	2.2 $\pm$ 0.4 μs (Ref.[115])
Host-Guest unbinding (OAMe-G2)	13	3.78 ms	0.338	1.82 - 5.52 ms	1.5×10^6	2.02 ms (Ref.[158])

Table 6.1: The kinetics obtained using OPES-Flooding during the training of the TPI-Deep-TDA CV. The p-values are computed from 2 stample Kolmogorov-Smirnov test[163]. The 95% confidence intervals were computed as $\{\frac{2\sum_i^n t_i}{\chi^2_{2n}(0.975)}, \frac{2\sum_i^n t_i}{\chi^2_{2n}(0.025)}\}$ as suggested by Kaminsky[164]. The t_i refers to the rescaled time for the i-th transition, and $\chi^2_{2n}(\alpha)$ is critical value of two-tailed chi-squared test with $2n$ degrees of freedom and $p = \alpha$.

First of all, in both cases, the results are in agreement with the reference within the 95% confidence interval with the reference value and the *p-values* show a good agreement with the ideal Poisson distribution for the mean first passage times (MFPT). In the case of Chignolin folding, the reference is obtained from long unbiased simulations performed on dedicated supercomputers[115]. For this system, it should also be noted that the computational efficiency, as measured in terms of the acceleration factor is almost two times that achieved using Deep-TICA and Deep-LDA CVs in Ref.[69], suggesting a superiority of the Deep-TDA CV for computing rates using OPES-Flooding simulations.

As far as the host-guest system is concerned, the reference value for the binding time[158] comes from the application of the Gaussian Mixture Based Enhanced Sampling[145] (GAMBES) method. In this case, what is impressive is the possibility of obtaining millisecond timescale ligand residence time from nanosecond long simulations, i.e., acceleration factor $\approx 10^6$.

These results, even if somehow preliminary, are promising for future developments for the application of the Deep-TDA CVs in kinetics computation. Moreover, as we already anticipated in the previous section, such results conveniently come as a *side product* of the TPI-Deep-TDA workflow we introduced just above.

Chapter 7

An application to liquid Sulfur

In the previous chapters, we have discussed different methods for the data-driven design of collective variables for enhanced sampling. In particular, in Chapter 6, we have introduced the Deep-TDA[18] method. However, the applications we have presented so far have been more oriented toward didactic purposes and to prove the validity of these approaches rather than tackling relevant problems.

In the following sections, we will shift our attention to a direct application of the Deep-TDA method to the study of the challenging mechanisms and structures involved in the *lambda transition* of liquid Sulfur.

7.1 Sulfur's complex phase diagram and the λ-transition

Elemental sulfur exhibits a very complex phase diagram that has attracted much attention for decades. This richness derives from the sulfur's propensity to be twofold coordinated, which results in the possibility of either forming ring-like structures (S_π) or polymeric chains (S_∞) of competing energy.

Phase diagram overview The ring-like arrangements dominate the stable solid structures[21,22,165] (orthorhombic α-S, monoclinic β-S and γ-S) with polymeric arrangements reported only for pressures higher than 2.0 GPa[166–168].

Among the cyclic polymorphs, small rings are preferred, and the 8-membered crown-shaped rings (S_8, see Fig. 7.1) are by far the most stable configurations. However, evidence of a minority fraction of rings with sizes ranging from 6 to 32 atoms (S_π) has also been reported[169,170].

A mixture of these two structural models has also been proposed to describe the

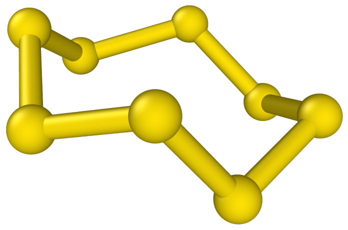


Figure 7.1: Structure of the eight-membered crown-shaped sulfur rings (S_8), the most stable among the sulfur's cyclic polymorphs.

liquid phase, and several liquid-liquid phase transitions have been reported in a wide temperature and pressure range[21,22,169,171,172]. For instance, a transition between a low-density liquid, richer in rings, and a high-density liquid, richer in polymers, has been reported with a transition line that terminates into a critical point at around 1000 K and 2.0 GPa[169].

The λ-transition in liquid sulfur, state of the art In the following sections, we will focus on a small part of the phase diagram, namely in the proximity of the so-called *lambda transition*[22,171,173]. This transition occurs at $T_\lambda = 432$ K and ambient pressure and results in an anomalous behavior of physical properties like heat capacity[22], viscosity[173], and density[174]. The behavior observed around T_λ has been widely studied and associated with the onset of a living polymerization of the S_8 rings[175,176]. It is believed that before the λ-transition, ring-like structures dominate, but as one increases the temperature, the polymeric fraction of the liquid slowly grows and suddenly increases around T_λ.

Over the years, extensive experimental studies have been conducted to characterize the species involved in the transition[21,177–180]. However, they still provide limited insight into the underlying dynamical process and a detailed picture of this phenomenon is still missing. Previous *ab initio*-based theoretical studies[181–185] have partially filled this gap, but the scope of these simulations has been limited by the computational costs of first principles methods. Indeed, all these studies have been limited to relatively short timescales (some 100 ps) and/or simulation cells with only a few hundred atoms. Clearly, such limitations are quite severe when studying polymeric systems, which by definition involve a large number of atoms and often exhibit slow dynamics[176,186].

7.2 Methods

In recent years, the combination of machine learning interatomic potentials (see Sec. 1.5) and enhanced sampling methods (see Sec. 2.2) in an active learning framework has proven effective in overcoming computational difficulties similar to the ones we just mentioned for liquid sulfur in a number of complex processes. Some examples are the liquid-liquid transition in phosphorus[26], the nucleation and phase diagram of gallium[25], the decomposition of urea[48] or the dynamical nature of heterogenous catalytic processes[27,28]. When applied to the study of liquid sulfur, such an approach allowed us to overcome the limitation of standard MD and to simulate systems of thousands of atoms for timescales of the order of nanoseconds.

We have indeed mentioned in Sec. 1.5 that machine-learning potentials allow performing *ab-initio-quality* molecular dynamics (MD) simulations at a fraction of the cost of first principles methods. However, to be accurate enough for the study of reactive processes, they need to be optimized on a large set of reference configurations. It is then of the utmost importance that the training set is not limited to low-energy configurations from the metastable states but also includes configurations related to their transition states. Unfortunately, complex systems such as liquid sulfur belong to the category of *rare-events* (see Chapter 2), in which reactive events are hindered by the presence of large free energy barriers, dramatically complicating the collection of training configurations with standard MD simulations.

Fortunately, enhanced sampling methods are aimed at overcoming this limitation so as to improve the sampling of reactive events, as we discussed in Sec. 2.2. In this way, configurations from reactive trajectories become accessible and it is thus possible to enrich the training set with the necessary configurations. As we already mentioned, many enhanced sampling methods rely on the identification of a small number of collective variables (CVs) (see Ref. 2.4). In the case of liquid sulfur, the complex structural changes that take place close to the λ-transition are difficult to express in simple geometrical terms, and we use instead a more abstract CV that results from a combination of graph theory and machine learning.

Provided this short overview, in the following sections, we explain in detail the different elements of our approach.

7.2.1 Machine learning potential for interatomic interactions

To simulate with molecular dynamics a process as complex as the polymerization of liquid sulfur and its inverse, an extremely accurate description of the interatomic interaction is of utmost importance. Typically, this requires the use of electronic structure methods such as density functional theory (DFT), which can faithfully describe the changes in the forming and breaking of chemical bonds. Unfortunately, such calculations are still computationally too expensive to perform extensive sampling even on small systems, especially when rare events are involved.

To address this issue, many *ab-initio-quality* force fields based on machine learning have been proposed following the strategy pioneered by Behler and Parrinello[13] (see Sec. 1.5). In this approach, the interatomic interactions are modeled through a feed-forward NN, which is trained to faithfully predict the energies and forces obtained with DFT calculations on a large set of reference configurations. For example, in Fig. 7.2, we report the extremely good agreement between the energy and forces predicted by our machine-learning potential and the ones obtained from DFT calculations on the training and test sets.

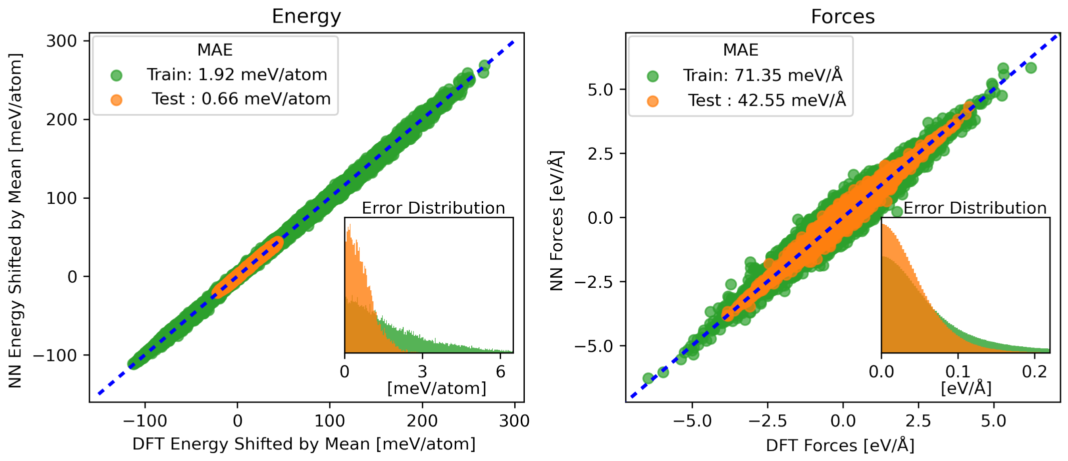


Figure 7.2: Comparison of the energies (left panel) and atomic forces (right panel) calculated on the training set (green) and test set (orange) using DFT and the final NN potential for liquid sulfur. Energies in the left panel are shifted by the mean value of the DFT atomic energies. Bottom-right insets illustrate the probability distributions of the absolute difference in energies and forces between the DFT and final NN model. Top-left insets report the mean absolute errors (MAE).

In our specific case, for the description of the atomic interactions, we employed the DeepMD method[24] in the attention-based Deep Potential scheme[50], which is able to give a much more accurate reproduction and prediction of DFT energies and atomic forces in the case of liquid sulfur, as it is shown in Fig. 7.3. Concerning both Fig. 7.2 and Fig. 7.3, it should be noted that the different error distributions on the test and training sets are due to a different fraction of *problematic* configurations on the two datasets. Indeed, most of the *out-of-equilibrium* configurations collected through the active learning cycles (see next paragraph) were added to the training set to improve the overall quality of the model.

Building a proper dataset with active learning The most challenging task for a successful application of our workflow is the building of a training set able to cover the relevant configurational space. For instance, liquid sulfur exhibits a wide range of ring-like and chain-like metastable structures with different sizes[21], and it is thus necessary to include these configurations in the training set as well as those related to the transition state of the interconversion process.

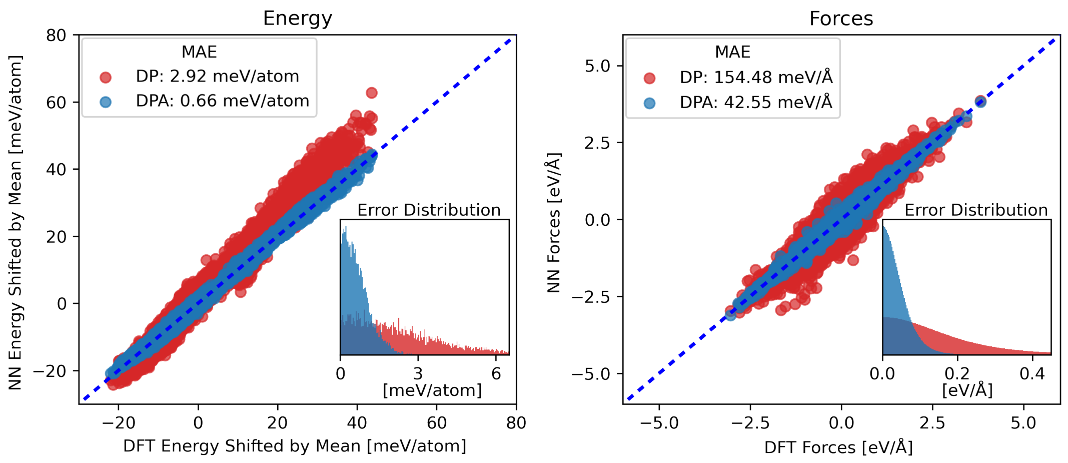


Figure 7.3: Comparison of energies (left panel) and atomic forces (right panel) calculated on the test set using DFT and NN potentials trained with the Attention-based Deep Potential scheme[50] (DPA, blue) and the standard Deep Potential-Smooth Edition scheme[24] (DP, red). Energies in the left panel are shifted by the mean value of the DFT atomic energies. Insets illustrate the probability distributions of the absolute difference in energies and forces between the DFT and NN models. For the standard DP potential, we trained on the same training set as that of our final NN potential (DPA), as well as the same hyperparameters except for increasing the training steps from 3×10^6 to 5×10^6. The test is the same as that used to validate our final NN potential.

To improve and simplify this step, active learning strategies boosted by the use of enhanced sampling methods have been applied to study several complex systems[25–28,48,187]. In this approach, we alternate iteratively cycles of refinement of the potential and cycles of enhanced sampling. The latter allows us to obtain, at an affordable computational cost, crucially relevant reactive configurations, which are then labeled with DFT calculations and added to the training set to train a more accurate potential for the next sampling cycle.

In our specific case, to enhance the sampling of our simulations, we applied the OPES[65,68] method (see Sec. 2.3) combined with a combination of the Deep-TDA[18] CV (see Sec. 6.2) and some elements of graph theory (see Sec. 7.2.2).

The active learning protocol The whole procedure of exploring the relevant atomic configurations to be included in the training set consisted of a few steps.

First, we ran a series of unbiased AIMD simulations in the NPT ensemble on systems of 128 atoms for times ranging from 2 to 10 ps. These simulations were performed in the 500 ~ 1200 K temperature range and 0.2 ~ 3.0 GPa pressure range to collect configurations in both the polymeric and ring phases of liquid sulfur. Indeed, approaching high temperature and pressure, S_8 rings are destabilized in favor of the S_∞ polymers, thus inducing a spontaneous and fast transition.

From these simulations, we collected about 7800 atomic configurations to build the initial training set and start our active learning procedure, which alternate cycles of training and sampling according to the following steps:

- **Step 1:** We train a committee of four NN potentials using different initial weights and the previous iteration's updated training set. For the first iteration, we shall use the initial training set of AIMD configurations.
- **Step 2:** We perform a series of simulations using one of the four NN potentials trained in Step 1 to explore new relevant atomic configurations. Not limited to AIMD simulations anymore, we expanded our systems from 128 atoms to 512 atoms and ran enhanced sampling simulations using OPES combined with our topological CVs (see Sec. 7.2.2). These biased simulations are essential for exploring the active atomic configurations along the polymerization of S_8 rings.
  During the simulations, we monitor the reliability of the potential on the sampled configurations based on the committee maximal standard deviation σ[188] of the atomic forces predicted by the four NN potentials:
  $$\sigma = \max_i \sqrt{\frac{1}{4}\sum_{\alpha=1}^{4} \|\mathbf{F}_i^\alpha - \overline{\mathbf{F}_i}\|^2} \tag{7.1}$$
  where $\mathbf{F}_i^\alpha$ is the atomic force on the atom i predicted by the NN potential α, and $\overline{\mathbf{F}_i}$ is the average force on the atom i over the NN potentials committee. To minimize the number of new relevant atomic configurations to be added to the training set while ensuring maximum diversity, we follow the same strategy described in Ref.[27], with the low (σ_l) and up (σ_u) bound values set to 0.15 and 0.4, respectively. We refer the readers to Ref.[27] for further details.
- **Step 3:** We calculate the DFT energies and forces for the configurations selected in Step 2 and include them in the training set for the next iteration.

Following our previous works[25–28,48], this active learning process is repeated until less than 10% of the sampled atomic configurations fall into the candidate list of Step 2. At the end of the procedure, our dataset included roughly 1.5×10^5 atomic configurations, with almost 90% of them consisting of 512 atoms.

7.2.2 Topological collective variables

In order to successfully drive transitions between the ring and polymer phases of liquid sulfur with enhanced sampling, the design of a proper collective variable (CV) is crucial, as we have discussed in Chapter 2.

In Chapter 4, we have seen that machine learning (ML) techniques have been applied in the last decade to the challenge of designing effective CVs, leading to a number of methods suitable for different scenarios according to the available data[20]. We mentioned that such methods generally combine large sets of physi-

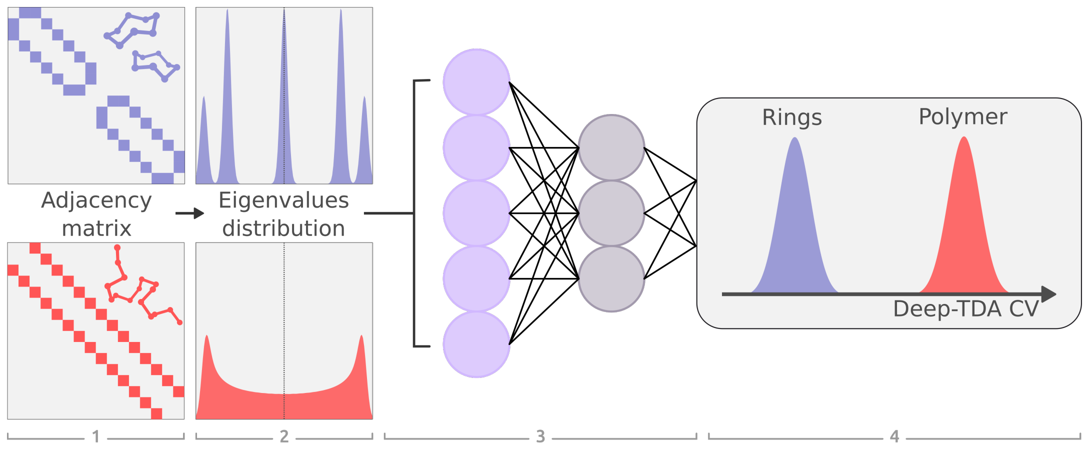


Figure 7.4: a Schematic representation of the construction of the topological collective variable (CV) for polymerization in liquid sulfur. For each configuration, we build the corresponding adjacency matrix and compute its eigenvalues distribution with a continuous histogram. The values of such histograms are fed as inputs of a neural network (NN) that combines them and returns the CV as output, according to the Deep-TDA CV scheme[18]. For the training of the NN, we build a dataset of configurations from pure rings (blue) and pure polymer (red) phases. The NN is optimized such that the projection of the training data in the CV-space matches a pre-assigned target distribution in which the states are well-discriminated.

cal descriptors with the help of NNs, which are then optimized on specific object functions depending on the circumstances.

Here, we use the Deep-TDA[18] method (see Sec. 6.2), in which the CV model is optimized according to a classification criterion, and we build our descriptors set from the adjacency matrix of the system. The whole protocol of our CV design process is schematically depicted in Fig. 7.4 and we shall refer to it as we discuss its components.

In Deep-TDA, the CV is the output of a feed-forward NN (panel 3 in Fig. 7.4) whose inputs are a set of physical descriptors collected with short unbiased runs in the metastable basins that are supposed to be visited in the process of interest. The NN is optimized such that the training data, when projected in the CV space, are distributed according to a preassigned target distribution (panel 4 in Fig. 7.4). This target is defined as a series of Gaussians with fixed positions and widths, one for each state, such that data from different basins are localized in different regions of the CV space.

Topological descriptors from graph theory Even if we have seen how Deep-CVs greatly simplified the CV design procedure, the effectiveness of these methods is still affected by the choice of input descriptors as they should be informative on the slow modes of the process (see Chapter 4). In this sense, the complex processes that take place at the λ-transition presented us with the new challenge of finding proper descriptors that could describe the ring opening and formation while retaining permutational invariance. Since the system undergoes enormous changes in connectivity, it is natural to look for variables that are able to describe such changes. We thus resort to graph theory[23] and, in particular, to the adjacency matrix (panel 1 in Fig. 7.4) and its eigenvalues (panel 2 in Fig. 7.4). The fact that the eigenvalues reflect the connectivity can be intuitively understood if we consider, for instance, that the values $\pm\sqrt{2}$ and 0 reflect a S_8 ring arrangement, and the multiplicity of such eigenvalues is related to the number of rings present in the system.

The use of the adjacency matrix eigenvalues to build CVs is not new[133,189], but here, rather than using a selected number of them as CVs, we shall use all eigenvalues as descriptors. Then, to standardize this information and remove any dependence on the indexing choice for the single atoms, we computed the histogram of such eigenvalues and used the value on the underlying bins as inputs of our Deep-TDA model (panel 2 in Fig. 7.4).

Adjacency matrix The adjacency matrix $\mathbf{A}$ is a concise way adopted in graph theory[23] to represent the connectivity of a system starting from the interatomic distances between the particles. The $\mathbf{A}_{ij}$ element of the matrix is either $\mathbf{A}_{ij} = 1$, if the scalar distance d_{ij} between atom i and j is lower than a d_{cutoff}, or $\mathbf{A}_{ij} = 0$ otherwise. However, in practice, the application of a sharp cutoff is not suitable in an enhanced sampling context, as it would lead to discontinuous derivatives of the matrix $\mathbf{A}$ with respect to the atomic coordinates. For this reason, to apply the cutoff, we used a sharp but continuous switching function $S(d_{ij})$ in the form of a Fermi-like function

$$S(d_{ij}) = \frac{1}{1 + \exp\left(\frac{d_{ij} - d_{cutoff}}{q}\right)} \tag{7.2}$$

where the q value was chosen to obtain the sharpest behavior with numerically stable derivatives, i.e., $q = 0.25$, and d_{cutoff} was set to 2.6 based on the typical sulfur-sulfur bond distances.

Training of Deep-TDA model To train our Deep-TDA CV, we used a two-state model, using unbiased data collected in the pure ring and pure polymeric phases (see Fig. 7.4 whole). Our choice was motivated by the experimental evidence that suggests that the relevant properties in the λ-transition region depend on the relative fraction of these two phases. We also note that even if the pure polymeric phase is considered allegedly non-physical, it still can be used to simplify the training of the model by making the relevant polymer-related features more evident. Nonetheless, considering that the polymer concentrations reported in the experiments do not exceed 60/70%, we applied a harmonic restraining potential along the CV in correspondence to polymeric contents above such concentrations to avoid the useless exploration of unphysical regions during our biased simulations.

Even if the histogram of the adjacency matrix eigenvalues is the input of

Deep-TDA NN, the inputs of the whole CV model, in our case, are the positions of all the atoms in the system. From there, using the additional preprocessing tools available in the `transform` branch[1] of the `mlcolvar` library (see Chapter 5), we compute within the model:

1. All the interatomic pairwise distances between the atoms
2. The adjacency matrix starting from the distances and applying a smooth cutoff (see previous paragraph)
3. The full eigenvalues spectrum of the adjacency matrix using PyTorch[84] tools
4. The histogram of the eigenvalues with 100 bins in the range (-2.2,2.2) using a Gaussian expansion to ensure continuous derivatives
5. The Deep-TDA CV as the output of an NN that takes the value of the bins of the histogram of the adjacency matrix eigenvalues as inputs

For the training of the Deep-TDA model, the targets were chosen to be $\mu_A = -25$ and $\mu_B = 25$ for the centers of the distributions and $\sigma_A = 0.2$ and $\sigma_B = 0.2$. The training set was composed of 18000 configurations for each of the two states for a total of 36000 configurations. Including input and output layers, the architecture was {100, 64, 32, 1} nodes/layer, and the activation function was the rectified linear unit (ReLU). The learning rate for the optimization was set to 0.001.

7.2.3 Additional computational details

Code and software All *ab-initio* molecular dynamics simulations (AIMD) and single-point energies and forces needed for training neural network (NN) potential were performed using the CP2K 9.2 code[129]. We also double-checked that the forces calculated using CP2K code were consistent with those obtained with Quantum Espresso code[130]. The Bader's charges analysis was obtained by analyzing the DFT electronic densities with the Bader[190] code.

The training of the NN potentials for the interatomic interactions has been done using the DeepMD-kit package[191]. The NN potential-based Molecular dynamics (MD) simulations were performed using LAMMPS[127] MD engine with the DeepMD-kit software plugged in to describe the interatomic interactions. For enhanced sampling simulations, we used the PLUMED[124] plugin patched with LAMMPS.

The training of the machine learning collective variable (Deep-TDA CV) has been done using the `mlcolvar`[20] library based on PyTorch[84] (see Chapter 5). In particular, we used the `transform`[1] branch of the library, which includes the tools for adjacency-matrix-related calculations. The CVs have been deployed to PLUMED using the interface provided in the `pytorch` module of PLUMED (see Sec. 5.4).

[1]https://github.com/luigibonati/mlcolvar/tree/transform

The atomic displacement analysis was performed using the Python interface of the visualization and post-processing code Ovito[192]. The GUI of the same code has been used for the rendering of the molecular snapshots reported in Sec. 7.3.

AIMD simulations In AIMD simulations, energies and forces were computed using the Perdew–Burke–Ernzerhof (PBE) exchange-correlation density functional[193]. The Kohn and Sham orbitals were expanded in a m-DZVP Gaussian basis and the plane wave expansion of the electronic density was truncated at an energy cutoff of 300 Ry. The core electrons were treated using the Goedecker-Teter-Hutter (GTH) pseudopotentials[194,195] optimized for PBE. To reduce the computational cost, only the Γ-point was used to sample the supercell Brillouin zone.

All AIMD simulations were performed in the NPT ensemble with a time step of 2.0 fs. Temperature and pressure were controlled using Nosé-Hoover thermostat[196] and a Nosé-Hoover-like barostat[197] with coupling constants of 0.05 ps and 0.5 ps, respectively. To mitigate the computational costs, only a smaller cubic simulation cell consisting of 128 atoms was used. Nonetheless, the results discussed in the following were calculated on larger cubic simulation cells, i.e., 512 and 3456 atoms.

Single-point energies and forces calculations The energies and forces used for NN training were computed using the same exchange-correlation density functional (PBE) and pseudopotential (GTH) as that of the AIMD simulations. We used the energy cutoff of 350 Ry and we added the D3 dispersion corrections[198]. Single-point calculations have been performed on cells with 128 and 512 atoms. For atomic configurations of 128 atoms, we used k-points grids of $2\times2\times2$. In contrast, only the Γ-point was used for atomic configurations composed of 512 atoms. We indeed checked that for such a bigger system, the accuracy on energies and forces using only the Γ-point and the one obtained using the $2\times2\times2$ k-points grid were almost indistinguishable.

Training of neural network potentials The NN potentials were trained following the DeepMD method[24] in the attention-based Deep Potential scheme[50]. The cutoff radius was set to smoothly decay from 0.5 Å to 7.5 Å. The maximum possible number of neighbors in the cutoff radius was set to 90 and the number of layers in the attention scheme was set to 3. We used three hidden layers with $\{30, 60, 120\}$ nodes/layer for the embedding network and four hidden layers with $\{240, 240, 240, 240\}$ nodes/layer for the fitting network. The size of the embedding matrix was set to 16. The learning rate was set to decay from 1.0×10^{-3} to 5.0×10^{-8} and the batch size was set to 8. The prefactors of the energy and the force terms in the loss function change from 0.01 to 5 and from 1000 to 1, respectively. The final NN model was trained for 3.0×10^{6} steps.

NN potential-based MD simulations The results reported in the following have been obtained via NN-potential-based MD simulations. Specifically, we performed unbiased and biased simulations in the NVT ensemble, using the global velocity rescaling thermostat[37] with a relaxation time of 0.05 ps and periodic boundary conditions (PBC) on cubic simulation cells. The timestep of the simulations was set to 1.0 fs.

The unbiased approach has been used to study the radial distribution function, the structure factor, and the mobility of the atoms. For this latter analysis, we simulated a system consisting of 512 atoms (box 24.8Å) starting from configurations generated from biased simulations, whereas, for the others, we simulated a much larger model made of 3456 atoms (box 46.9Å). In both cases, the box sizes were chosen to be consistent with the experimental densities.

For the enhanced sampling simulations, we applied the On-the-fly probability[65,68] (OPES) biasing scheme (see Sec. 2.3) to the machine-learning-based Deep-TDA CVs (see Sec. 6.2 and Sec. 7.2.2). This more expensive enhanced sampling approach has been used to simulate the dynamics of a system of 512 atoms to study the polymerization mechanisms of sulfur.

7.3 Results

In the first part of the results, we compare the results from unbiased simulations with the available diffraction data and we briefly report on atomic mobility in the different phases, showing in both cases consistent results with the experiments. In the second part, we study, with the help of enhanced sampling simulations, the polymerization and depolymerization processes that are supposed to take place close to the λ-transition. Based on nanoseconds-long reactive simulations and the results of a charge distribution analysis, we also propose reaction mechanisms to finally shed light on the puzzling formation and breaking of the polymers.

7.3.1 Static quantities: radial distribution function and structure factor

The radial distribution function $g(r)$ and the structure factor $S(k)$ depend on the structural ordering of the atoms and their features on the relative concentration of the phases in the sample. For this reason, these quantities have been experimentally monitored at different temperatures around the λ-transition[199]. However, the corresponding characterization of the S_8 fraction at such temperatures is still affected by significant discrepancies in results obtained with different experimental techniques[171,176,200,201].

This is the typical scenario in which theoretical modeling can provide a helpful contribution to the experiments. Simulations can indeed access the *pure* phases (i.e., only rings S_8 and only polymers S_∞), which are never found in the experiments. Despite sounding somehow unphysical, this information can provide clear and instructive footprints of the corresponding structures. One can then obtain the $g(r)$ and $S(k)$ for all the intermediate different compositions through a linear combination of the contribution from the pure S_8 and S_∞ phases. For example, the radial distribution function $g_\alpha(r)$ at a given concentration of rings α is

$$g_\alpha(r) = \alpha g_{S_8}(r) + (1-\alpha) g_{S_\infty}(r) \tag{7.3}$$

and similarly the structure factor $S_\alpha(k)$

$$S_\alpha(k) = \alpha S_{S_8}(k) + (1-\alpha) S_{S_\infty}(k) \tag{7.4}$$

To obtain the reference values for this approach, we analyzed 200 configurations collected over one ns of equilibrated dynamics of 3456 atoms (box size 46.9 Å) in the pure S_8 and S_∞ phases.

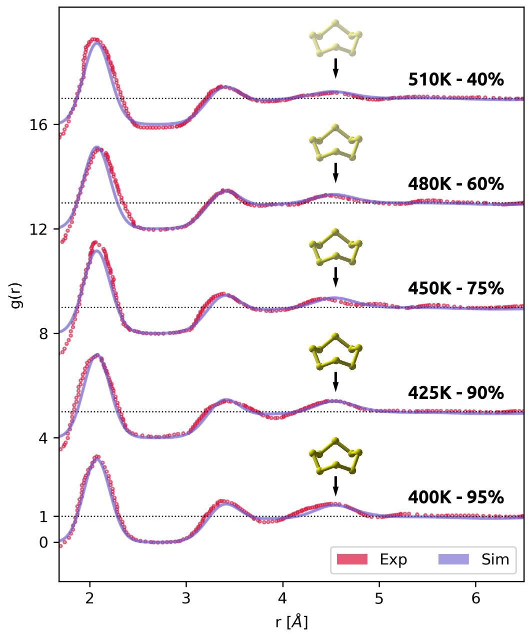


Figure 7.5: Radial distribution functions $g(r)$ for liquid sulfur at temperatures around the T_λ. Simulated results at different ring concentrations (solid blue lines) are compared with experimental data[199] at different temperatures (red void dots). Each experimental temperature and the matching percentage of rings are reported close to the corresponding curve. The $\pm 5\%$ interval on the ring concentration for each simulated curve is given as a shaded blue area. The black arrow marks the third peak signal associated with the S_8 species. The black dotted lines mark the $g(r) = 1$ value for each couple of curves as they are offset by four units in the vertical direction for visualization purposes.

Radial distribution function The radial distribution function $g(r)$ provides the probability of finding two atoms at a given interatomic distance r and can be readily computed from simulated trajectories[1]. In Fig. 7.5, we compare the experimental data[199] from X-Ray diffraction (XRD) at ambient pressure and different temperatures with our estimates for different S_8 concentrations (95% to 40% S_8 rings). The considered concentrations are chosen from the range provided by experimental results[171,202] and then refined to improve the matching with the experimental data. For a meaningful and direct comparison with the experimental data, we also applied a Gaussian convolution to the simulated results.

Our results remarkably reproduce the evolution with the temperature of the third peak around 4.5Å (see the arrow in Fig. 7.5). This elusive signal is associated with third neighboring distances in the S_8, thus tends to disappear as the temperature and the polymer fraction increase. Moreover, the first and second coordination shells (2.05Å and 3.3Å) peaks, which come from both S_8 and S_∞ phases and remain almost constant with temperature, are also accurately reproduced, with our results being almost indistinguishable from the experimental ones.

Structure factor Even if the radial distribution function is readily available from simulations, the actual quantity that can be measured experimentally is the structure factor $S(k)$. This is the counterpart in the reciprocal space of the $g(r)$, and the two quantities can be expressed as the Fourier transform of each other. Accordingly, we compute $S(k)$ from our estimate of the $g(r)$ as

$$S(k) = 1 + \rho_0 \sum_{0}^{L/2} 4\pi r^2 [g(r) - 1] \frac{\sin(kr)}{kr} \delta r \tag{7.5}$$

where k is the wavevector, L the box size and ρ_0 the average system's atomic density[199].

The comparison of the resulting $S(k)$ with the experimental data[199] is reported in Fig. 7.6. Our results show excellent agreement with the experiment also in this case, especially for the second and third peaks. Moreover, our approach allows us to reproduce remarkably well the first peak in the region at small k values, despite some (small) differences between our results and the experimental data, which could be caused by the cutoff applied in our deep-potential and, to a lesser extent, by the aforementioned scale issues. We remark on this result as previous theoretical studies, based on first-principles approaches, could not reproduce this signal, most likely due to their limited time and size scales. This region indeed corresponds to large distances in the real space. Therefore, it can be heavily affected by a small simulation box or limited statistics.

7.3.2 Atomic mobility: displacement analysis

One of the main features of the λ-transition is its sudden increase in viscosity above T_λ. At the atomic level, this should correspond to a decreased mobility of

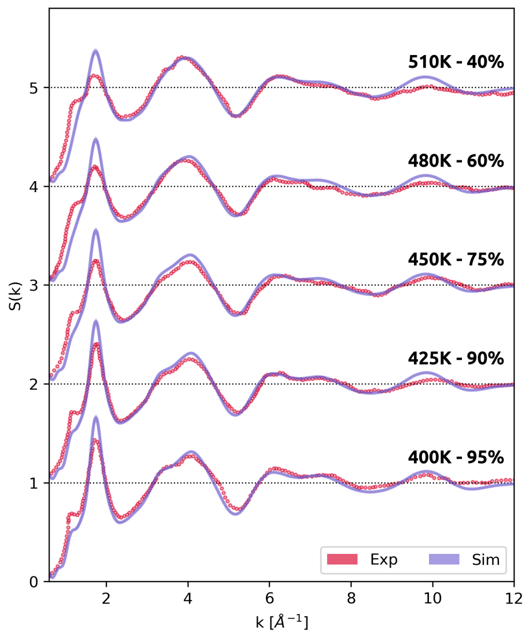


Figure 7.6: Structure factor $S(k)$ for liquid sulfur at temperatures around the T_λ. Simulated results at different ring concentrations (solid blue lines) are compared with experimental data[199] at different temperatures (red void dots). Each experimental temperature and the matching percentage of rings are reported close to the corresponding curve. The $\pm 5\%$ interval on the ring concentration for each simulated curve is given as a shaded blue area. The black dotted lines mark the $S(k) = 1$ value for each couple of curves as they are offset by one unit in the vertical direction for visualization purposes.

the atoms.

Especially in the polymeric phase, the atomic motions become sluggish and a direct calculation of the viscosity is impossible. However, in order to have an insight into the dynamics, we compute and compare the atomic displacements after 10, 25, 50, and 100 ps for ring concentrations that resemble conditions below T_λ (100% rings concentration), slightly above (75%) and well above (55%).

The results from this analysis are reported in Fig. 7.7 and clearly show that, on average, atoms in the molecular S_8 phase have the highest mobility. On the other hand, as the polymeric content increases, a peak in the distribution starts to appear below the 2Å threshold of the first coordination shell. This comes from the polymer atoms, which mostly oscillate around their positions rather than showing any net drift, thus inducing the rise in the viscosity.

7.3.3 Reaction mechanisms and charge analysis

Having assessed the reliability of our potential prediction when compared to the experiments, we move to the study of the chemical mechanisms involved

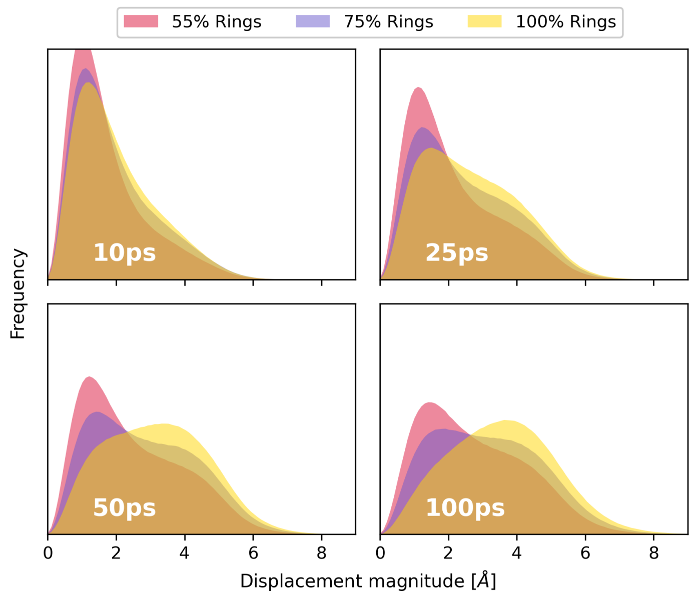


Figure 7.7: Histogram of the atomic displacements in liquid sulfur at different ring concentrations, given in the legend, and different lag times, indicated by white labels.

in the λ-transition in liquid sulfur with the crucial help of enhanced sampling simulations and our topological CV.

In the following, we propose mechanisms for the polymerization process and the opposite reaction for forming the rings. We note that we report only those mechanisms that we found to be dominant in our simulations, i.e., they were observed in the majority of several independent simulations, and for all of them, we double-checked the agreement of our potential with DFT calculations to avoid artifacts.

For each mechanism, we provide a prototypical example and an analysis of the instantaneous charges involved in the process. For each configuration, this latter analysis stands on the analysis of the Bader charge distribution[190] as obtained from the DFT charge density.

Polymerization mechanism The polymerization of liquid sulfur, which we schematically depict in Fig. 7.8, resembles many elements of the standard *ring-opening* polymerization[186]. The first step requires the formation of an active center from which the polymerization can propagate, and this forms when one of the crown-shaped S_8 monomeric units undergoes sufficiently large thermal fluctuations to open. The ring deformation indeed induces a polarization of the local charges, which is shown by the colormap in Fig. 7.8. Negative charges concentrate on the under-coordinated terminal atoms, thus making them active. At this point, they can either react together to close the ring again, or they can look for new neighbors on a different ring nearby. As the active terminal interacts with the guest ring, its charges are forced to reorganize. This induces a defor-

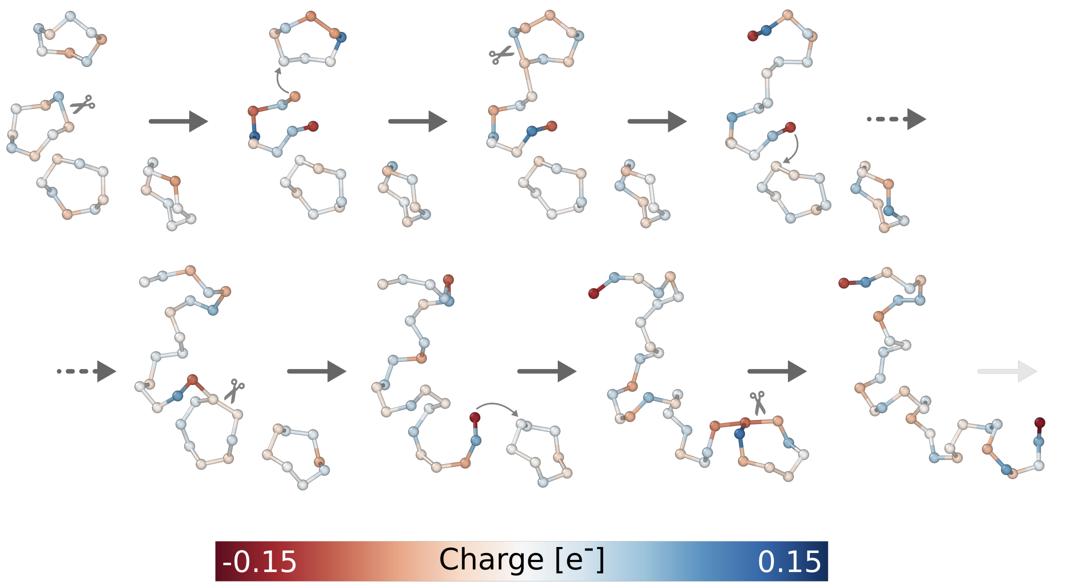


Figure 7.8: Polymerization mechanism in liquid sulfur starting from four S_8 rings. The atoms are colored according to the instantaneous charge obtained by computing the Bader's charge from the DFT electronic density. For visualization purposes, only the relevant atoms are represented from the 512 in the simulation cell.

mation in the guest ring that may eventually open it, leading to the formation of the first oligomer. Right after the opening of the second ring, the charges along the new short chain quickly reorganize, and negative charges concentrate in the chain tails. This makes the terminals active again and ready to further propagate the polymerization to other rings as described above.

Even if the ideal crown-shaped S_8 rings dominate the liquid phase, our calculations confirmed the experimental evidence[21] that other cyclic monomers (S_n, $n \neq 8$) and sub-stable isomers of S_8 rings, which report in part in Fig. 7.9, also contribute to this polymerization process.

Overall, from our results, it appears clear that the under-coordinated nature of the chain tails mainly drives the polymerization. This makes the terminal atoms highly reactive, as indicated by the negative charge localization, and eager to find new partners.

A second key factor lies in the possibility of stabilizing the charge unbalance (on average) over the whole polymer. In agreement with previous theoretical studies[182], we also found shorter chains to be rather unstable and to often revert to rings if left relaxing with short unbiased dynamics, at variance with the longer chains, which remained stable. We suppose this is due to the different capabilities of delocalizing, on average, the unbalanced charges over the chain, which thus plays a key role in the overall polymerization process.

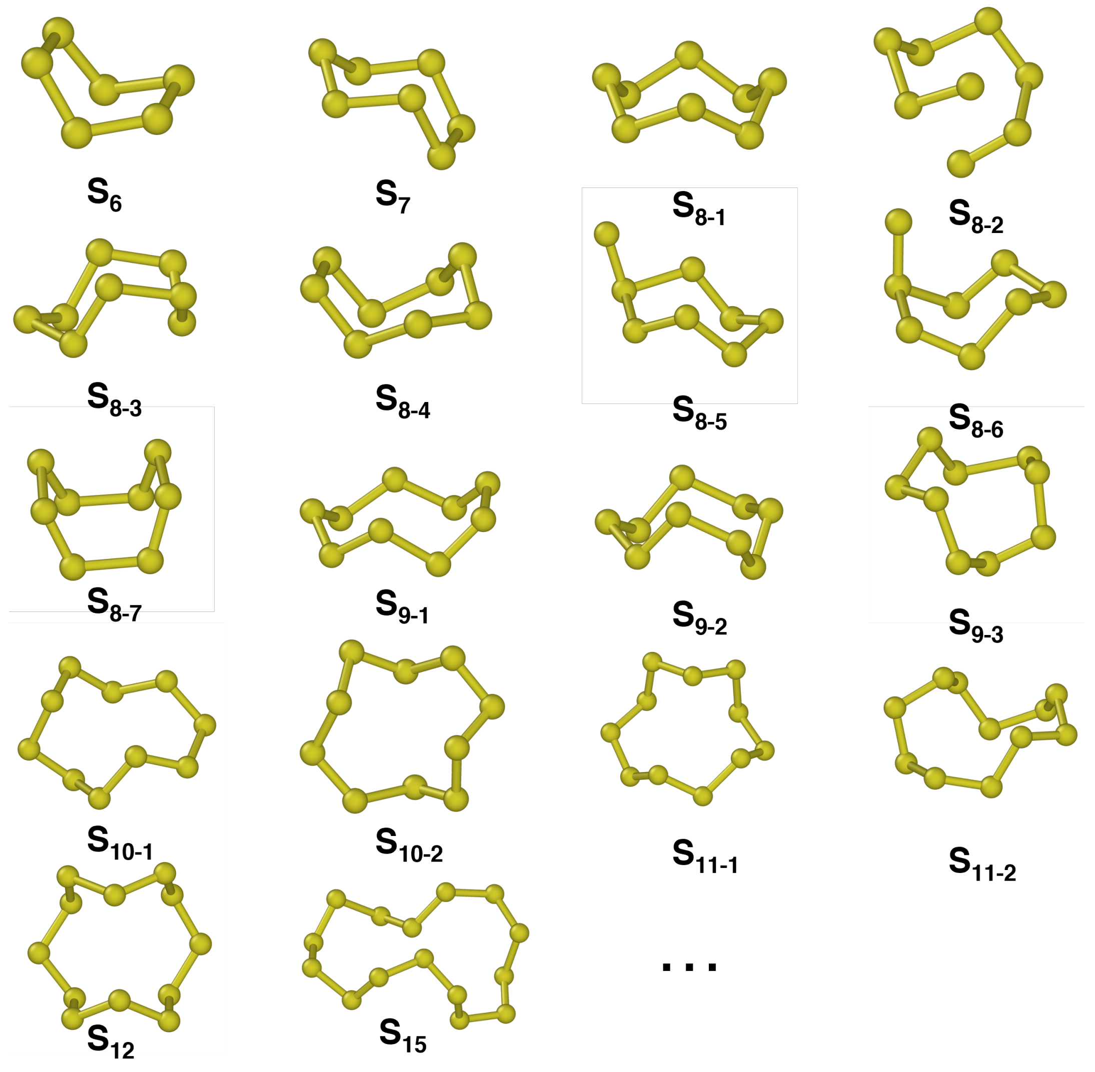


Figure 7.9: Snapshots of typical cyclic configurations ($6 < n < 15$) observed in our simulations. Such configurations were already reported in the literature and validated with DFT calculations. See Ref.[21] and the references hereby.

Ring formation mechanisms The formation of the rings starting from the polymer can occur either at the end of the chain (see Fig. 7.10 **a**), as one would intuitively suppose, but somehow surprisingly, also in the middle, as we schematically depict in Fig. 7.10 **b**.

In the first case (see Fig. 7.10**a**), the tail of the chain is characterized by a charge unbalance and higher mobility with respect to the rest of the chain. The first element, as mentioned in the previous paragraph, is specifically due to the under-coordinated nature of the sulfur terminal atoms, which leads to their negative polarization. This is the *chemical* driving force for the reaction, as it makes such atoms eager for new partners and ready to react. On the other hand, the

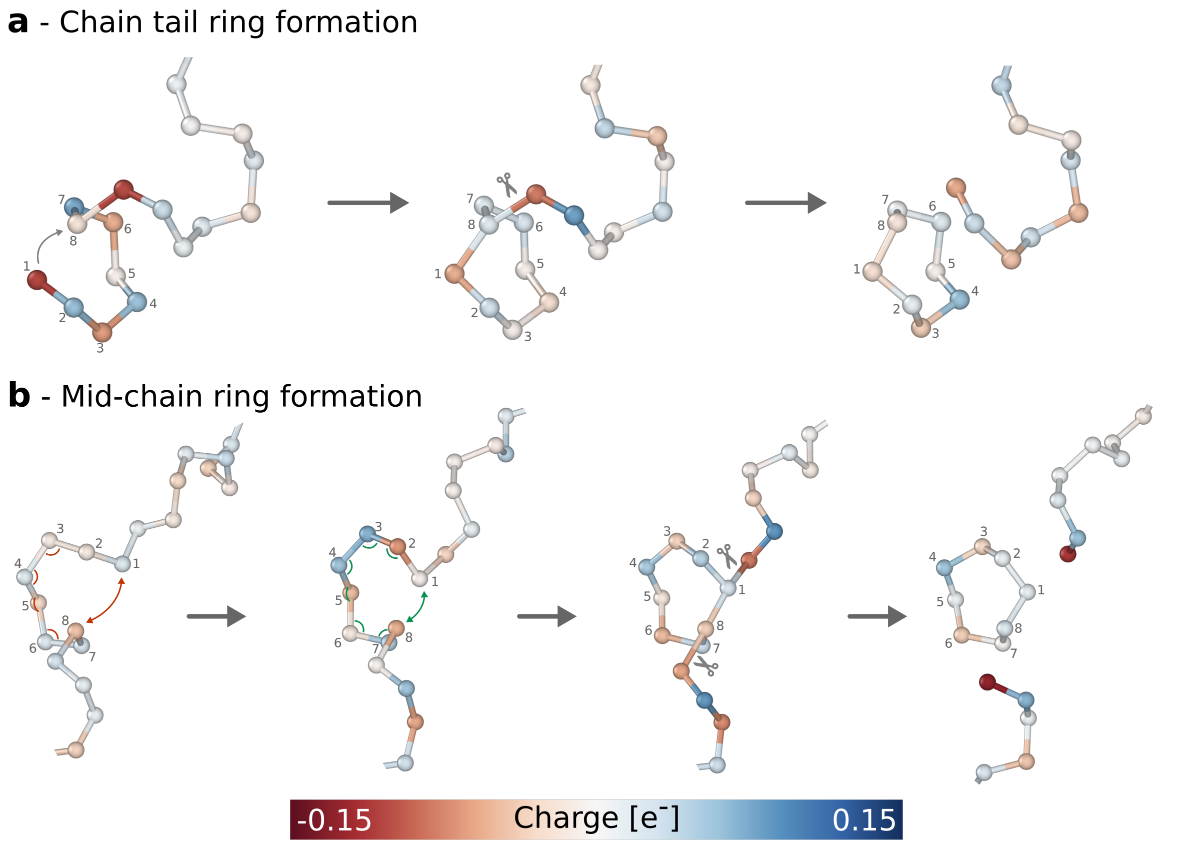

a - Chain tail ring formation
b - Mid-chain ring formation
-0.15
Charge [e⁻]
0.15

Figure 7.10: Mechanisms for the formation of rings in liquid sulfur starting from S_∞ polymers. **a** Formation of a ring from the tail of the polymeric chain. **b** Formation of a ring in the middle of the chain. The atoms are colored according to the instantaneous charge obtained by computing the Bader's charge from the DFT electronic density. For visualization purposes, only the relevant atoms are represented from the 512 in the simulation cell.

higher mobility reflects the general behavior of polymeric chain ends and poses the right *conformational* conditions. Indeed, to react, the tail has to fold onto the chain to form the loop that will eventually lead to the ring. Of course, this loop is most stable if the terminal atom folds such that it interacts with its 7th neighbor, thus ensuring the S_8 arrangement, but this same mechanism can also lead with less probability to some of the different-sized rings we reported in Fig. 7.9.

In the second mechanism, the scenario is significantly different as the atoms involved in the formations of the loop belong to the bulk of the polymeric chain (see Fig. 7.10 **b**). Such atoms are indeed fully coordinated, meaning that they are much less reactive than in the first case and that any polarization they could show is strictly instantaneous. We found that to compensate for this weaker reactivity, it is crucial that the arrangement of the atoms resembles as much as possible that of one of the stable S_π rings. Of course, the choice shall preferably be for the ideal crown-shaped S_8. Thus, we report this case in Fig. 7.10 **b**.

In the first frame, the eight-membered loop that starts to appear in the middle of the chain still has a wrong combination of angles and distances. On the other hand, in the configuration reported in the second frame, the sequence of angles becomes favorable, and the distance between the 1st and 8th S atoms reduces enough they can interact. As a consequence of this interaction, the adjacent atoms show a weak negative polarization (see third frame), which becomes stronger when the ring finally separates from the original chain.

Chapter 8

Transition-state-oriented bias potential from classifiers

In the previous chapters, our focus was oriented toward developing and applying machine-learning collective variables in the context of enhanced sampling to favor the transition between metastable states in a rare-event scenario. In particular, we have largely discussed methods and applications of classifier-like collective variables.

In the following, we will still follow this thread but from a much different perspective. Indeed, we will introduce a method for constructing a bias potential for the sampling of transition state regions and show how to build such a bias starting from the idea of classifier CVs.

8.1 A new approach to enhanced sampling

In previous chapters, we have largely discussed the sampling problems related to rare events in Molecular Dynamics (see Chapter 2) and we have presented enhanced sampling methods for alleviating these limitations (see Chapter 2). As we discussed, most of the methods in this category, such as metadynamics[7,66,74] or OPES[65,68], are meant to accelerate the dynamics of a process by *filling* the metastable basins in the energy landscape to reduce the energy barrier associated with the transitions between them. In this manner, the increased number of state-to-state transitions allows computing a converged estimation of the free energy of the process[5,74], thus allowing a proper characterization of the process itself.

However, when it comes to studying a natural process, free energy is not the only important information one may want to obtain from simulations. One problem of great interest is the identification of the transition state (TS) through which the system has to pass to translocate from one basin to the other. For example, identifying the TS is considered the holy grail when it comes to chemical reactions[203], as it provides precious information about reaction mechanisms and rates[204], or when dealing with proteins, it can provide information on their dynamics[205–207]. Moreover, we have seen how configurations around the TS are crucial in training machine learning reactive interatomic potentials[208] (see Sec. 1.5) and can be used to improve the performances of data-driven collective variables[19] (see Sec. 6.3).

Unfortunately, sampling of the higher-energy regions associated with the transition state (TS) often remains difficult, even if enhanced sampling methods are applied. This comes as no surprise, considering that transition states are found at saddle points[204] (or local maxima) even in the biased energy landscape and are still less likely to be visited.

In the following, we propose a method for extensive sampling of the transition state region by changing the paradigm of the way the bias potential is constructed. Instead of *filling the basins*, we propose to *dig a hole* in the transition state to lower its energy to the point that, in the biased landscape, it becomes a minimum that can be thus thoroughly sampled.

A committor-inspired way out of a chicken and egg problem The key element to achieve our goal of *stabilizing* the TS region is the design of a bias potential that is localized on the TS region itself. At first sight, this appears to be a chicken and egg problem since to localize the bias on the TS region, we need a good sampling of that region, and in turn, to get a good sampling, we need such a localized bias. However, a way out of this dilemma can be found by taking inspiration from Kolmogorov studies of stochastic processes[209], in particular from the fascinating concept of *committor function* in such processes.

For a given configuration, the committor function $q(\mathbf{x})$ gives the probability that starting from a given configuration $\mathbf{x}$, the system relaxes to a certain metastable state[29]. Thus, in a rare event scenario where the points in A have a very small probability of moving to B $q(\mathbf{x}) \approx 0$ for $\mathbf{x} \in A$, similarly $q(\mathbf{x}) \approx 1$ when $\mathbf{x} \in B$.

At this point, if we consider the derivatives of such a function with respect to its inputs $\mathbf{x}$, we can easily find that their square modulus $|\nabla q(\mathbf{x})|^2$ will be different from zero only in the transition region where it will be peaked. This, in principle, supplies us with a tool for automatically localizing the TS region, provided that we can estimate $q(\mathbf{x})$. Moved from these considerations, we propose to build our *transtion-state-oriented* bias as a function of the derivatives of (an approximation of) the committor function $q(\mathbf{x})$ for our system.

Even if the determination of the real committor function is still an open challenge, we argue that it can be approximated, at least coarsely, with a classifier-like collective variable that mimics the committor boundary conditions that the value of $q(\mathbf{x})$ should be constant within a metastable state.

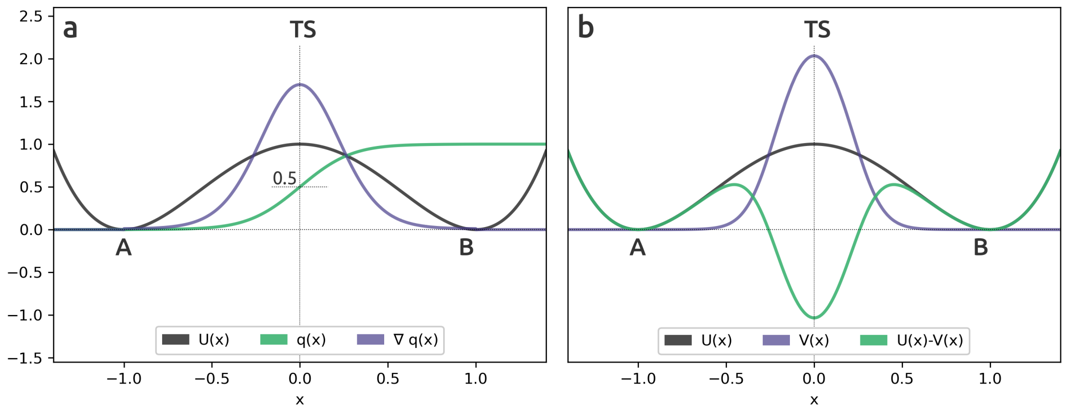


Figure 8.1: Schematic representation of the rationale behind our bias potential in a one-dimensional double-well model potential $U(x)$ (black line). The gradient $\nabla q(x)$ (panel **a**, green line) of the analytic committor function $q(x)$ (panel **a**, purple line) is used to build a bias potential $V(x)$ (panel **b**, purple line) centered on the transition state (TS) of the system. This is applied to the natural potential $U(x) - V(x)$ (panel **b**, green line) to lower the TS region energy to a minimum.

8.2 Methods

8.2.1 Building a bias potential from the committor function

Here, for brevity, we only discuss the feature of the committor function relevant to our purposes. We thus refer the interested reader to the specific literature from transition path theory (TPT) for more details[29,210].

Given a system characterized by two states A and B ($U(x)$ in Fig.8.1), in the framework of TPT, the committor function $q(\mathbf{x})$ is the probability that a trajectory starting at $\mathbf{x}$ reaches first B rather than A, or, in the TPT jargon, it is committed to B. Points on the isocommittor surface $q(\mathbf{x}) = 0.5$ are of particular interest as they have an equal probability of committing to A or B. For this reason, such points are usually associated with the transition state of the process.

The determination of the committor function is, however, far from trivial. From the definition, it follows that $q(\mathbf{x}) = 0$ when $\mathbf{x} \in A$ and $q(\mathbf{x}) = 1$ when $\mathbf{x} \in B$. In between, when $\mathbf{x} \notin (A \cup B)$, $q(\mathbf{x})$ is continuous and monotonous (see panel **a**). Even if in that region the committor function obeys the backward Kolmogorov differential equations

$$\begin{cases} \nabla U \cdot \nabla q - \beta^{-1} \Delta q = 0 & \mathbf{x} \in \Omega \setminus (A \cup B) \\ q(\mathbf{x}) = 0 & \mathbf{x} \in A \\ q(\mathbf{x}) = 1 & \mathbf{x} \in B \end{cases} \tag{8.1}$$

an analytic solution can be obtained only in simple 1D models as the double-well potential reported in Fig.8.1.

Nonetheless, the differential problem of Eq. 8.1 has an equivalent variational formulation in which $q(\mathbf{x})$ is found by minimizing the functional K

$$\min_q K : K = \frac{1}{Z} \int_{\Omega \setminus (A \cup B)} |\nabla q(\mathbf{x})|^2 e^{-\beta U(\mathbf{x})} d\mathbf{x} \tag{8.2}$$

Interestingly, this formulation depends on the gradient ∇q, which by definition is peaked in correspondence with the transition state and goes to zero towards the metastable states (see panel **a**).

Inspired by this framework, we propose to build a transition-state-focused static bias potential $V(\mathbf{x})$ as a function of ∇q (see panel **b**) or at least some approximation of that, as we shall see later

$$V(\mathbf{x}) = \lambda \log \left(|\nabla q(\mathbf{x})|^2 + 1 \right) \tag{8.3}$$

Here, we use the logarithm to have a smoother function and ensure that its argument is greater than one, such that $V(\mathbf{x}) \geq 0 \ \forall \ \mathbf{x}$. The parameter λ is meant to scale the intensity of the bias potential.

In contrast with other biasing schemes, such as metadynamics[7], the purpose of this bias is not to enhance the sampling by filling the energy landscape in the metastable basins but rather by lowering the energy of the transition state region. Upon an appropriate choice of λ, this bias potential allows transforming the TS state region, which is a hard-to-sample saddle point of the natural $U(\mathbf{x})$, into a fictitious local minimum (see panel **b**) that can be sampled effortlessly. In practice, λ should be neither too small nor too large. In the first case, the bias would be too weak to be effective, whereas in the second, it could lead to unstable dynamics. From our experience, given an expected barrier height $\tilde{E}_{TS}$, we found that values in the range of $0.25\,\tilde{E}_{TS} < \lambda < 2\,\tilde{E}_{TS}$ are appropriate choices.

We note here that, as $V(x)$ is sharply localized at the TS, it may be that the biased energy landscape still presents some residual barriers. For example, in the toy model in Fig. 8.1, one still has to overcome a smaller barrier when starting from the original metastable basins. However, in most cases, we found such barriers to be small enough not to hinder the evolution toward the TS, but in case, our approach could also be applied alongside other static or dynamic biasing schemes.

8.2.2 Approximating the committor function with classifier-like collective variables

As discussed above, our bias potential conceptually depends on the committor function $q(\mathbf{x})$, whose determination is far from trivial. The analytic solution of the Kolmogorov equations (Eq. 8.1) is indeed inaccessible for any real system, and even numerical approximated approaches are severely limited by the curse of dimensionality. Similar limitations also arise for the multidimensional integrals in the variational formulation in Eq.8.2. For these reasons, we do not aim at finding the true committor function, but we satisfy ourselves with an approximation based on classifier-like collective variables, which we can design with the help of machine learning techniques, as we have seen in the previous chapters.

To motivate our choice, we recall that in machine learning, classifier functions are trained to discriminate among different classes by returning as output different values for each class[9]. It is clear that, in our case, this criterion is somehow equivalent to the boundary conditions that apply to the committor function, i.e., being either 0 or 1 in the metastable states.

In this sense, adapting the Deep-TDA[18] method (see Sec. 6.2) to our purposes comes naturally. In Deep-TDA, the CV is built as the output of a NN, which takes as input a large set of descriptors collected in unbiased runs in the metastable basins. The NN is optimized to return a CV space in which the distribution of the training data matches a preassigned target distribution in which the different states are well distinguished. In the original method, this target distribution is defined as a series of Gaussians, one for each state, of preassigned positions and widths. Here, to better approximate the classifier-like behavior of the committor function, we choose as target distribution the sum of two delta functions, one associated with one state, located at minus one, and the other associated with the other state, located at one.

We note that even if the training is focused on the metastable regions only, the CV function is still continuous between them, and its derivatives are accessible using the automatic differentiation engines of ML libraries such as PyTorch[84].

8.3 Results

In the following, we test our biasing scheme in three prototypical examples, showing how it can lead to an extensive sampling of regions related to the transition states in these systems. For didactical purposes, we start by discussing the general workflow in the case of the diffusion of a particle in the Muller-Brown potential surface toy model we have already met in the previous chapters. We then move to the study of the proton transfer reaction in the tropolone molecule. Finally, we tackle the folding of the chignolin protein.

8.3.1 Müller-Brown potential

The two-dimensional Muller–Brown potential, whose isolines are depicted in Fig. 8.2, as we have already mentioned earlier in this Thesis, is often used to test the efficiency of enhanced sampling methods in a controlled manner. Performing unbiased simulations of the diffusion of a particle over this potential, one can only sample the bottom of the two metastable states (purple points in Fig. 8.2) due to the barrier that separates them.

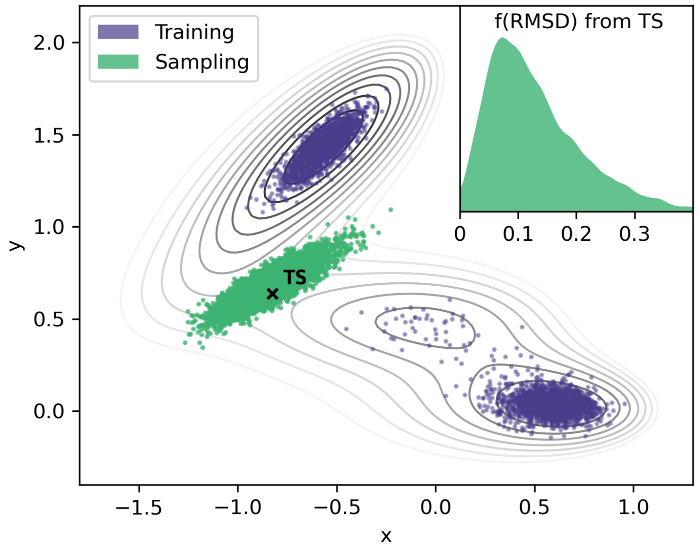


Figure 8.2: Sampling of the transition state (TS) region in the Muller-Brown two-dimensional potential (green points). The bias potential is trained using configurations limited to the two metastable basins (purple points). The reference TS position (black cross) is identified as the point with the highest energy along the minimum free energy path. The top-right inset shows the distribution of the displacements from the TS reference.

Starting from such data, we can train our classifier CV using the x and y coordinates of the particle as descriptors and build the bias potential to direct the sampling toward the TS region. The results of short biased simulations starting from the two basins are reported in Fig. 8.2, clearly showing the extensive sampling of the TS region (green points). Moreover, as shown in the inset, most of the sampled points are concentrated very close to the TS reference, located at the maximum energy point along the minimum free energy path. We want to remark on this point, considering that the only information given to the model was limited to the metastable states.

8.3.2 Proton transfer in Tropolone

Tropolone is a simple cyclic molecule, depicted in the inset of Fig. 8.3, which undergoes an intramolecular proton transfer (PT) reaction. Apart from the hydrogen atom involved in the PT, the molecule is symmetric. The two oxygen atoms, depicted in red, can indeed alternatively be found in the keto (=O) or enol (-OH) form, depending on the position of the exchanged proton. This also determines the arrangements of the conjugated double bonds over the seven-membered carbon ring.

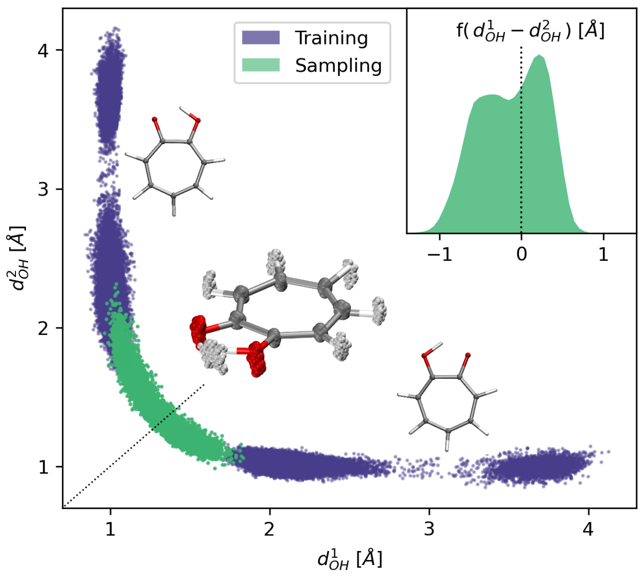


Figure 8.3: Sampling of the transition state (TS) region in the proton transfer in tropolone molecule (green points) projected in the space defined by the distances of the hydrogen atom from each of the two oxygen. The superimposition of 30 random configurations is shown in the central inset. The bias potential is trained using configurations (purple points) from two metastable states of the molecule. Each of them presents two rotational isomers, depending on the orientation of the OH bond, which can easily interconvert at 300K. Only the most stable form is reported in the smaller insets. A qualitative reference for the TS position is located along the $d^1_{OH} = d^2_{OH}$ diagonal (black dotted line). The top-right inset shows the distribution of the displacements from the TS reference.

In Fig. 8.3, we report the points collected in two unbiased runs in the metastable states used to train the classifier CV (purple points). In this case, as input descriptors for our model, we used the interatomic distances between heavy atoms in the system and the hydrogen-oxygen distances related to the hydrogen involved in the PT. The training set points are scattered in the space defined by the distances of the hydrogen atom from each of the two oxygens (d^1_{OH} and d^2_{OH}), in which the chemical and rotational isomers are clearly distinguishable. In this space, the TS can be intuitively localized in the region where such distances are almost equivalent and the hydrogen is bridging between the two oxygen atoms, which is marked in Fig. 8.3 by the thin dotted diagonal line.

The points sampled using our novel biasing scheme (depicted in green) are remarkably distributed in this region. Moreover, they are mostly concentrated in the region d^1_{OH} - $d^2_{OH} = 0$ (see top-right inset) and the sampled configurations clearly resemble the expected TS. In the central inset, we show a superimposition of randomly selected samples from our biased trajectory which clearly shows the PT hydrogen midway between the oxygens and the fluctuations of the ring hydrogens due to the induced resonance of the ring double bonds.

8.3.3 Folding of Chignolin

In Sec. 6.3.2, we have already introduced chignolin as a small protein often used to benchmark enhanced sampling methods on biological systems. In an aqueous solution, it can be found in two stable conformations, folded and unfolded, which are schematically depicted as cartoons in the left side of Fig. 8.4.

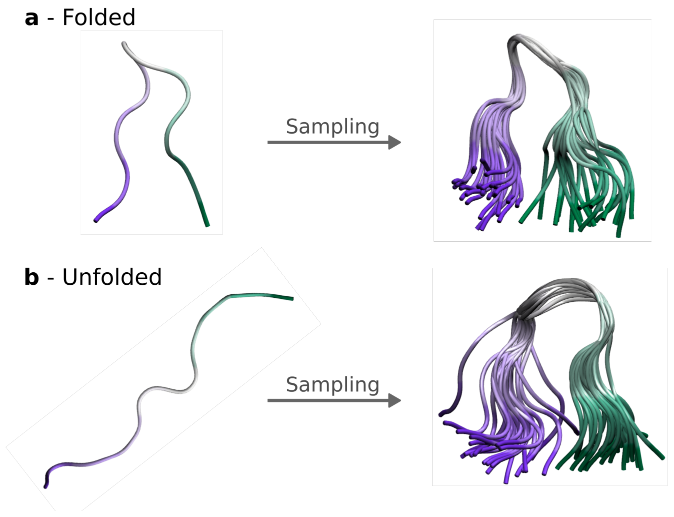


Figure 8.4: Cartoon of the sampling of the transition state region (TS) of the chignolin folding process starting from the **a** folded and **b** unfolded states. The left-hand cartoons show the two metastable configurations. The right-hand ones show a superimposition of 30 random configurations obtained with our biased sampling approach. The color shows the index reference of the atoms in the main chain.

To test our method on this more complex system, we used all the contacts between the α-carbons of the main chain as descriptors for our model, as already done in Refs.[17,19,69] (see also Sec. 6.3.2). Applying the bias thus obtained and starting independent runs in the folded and unfolded state, we could sample, in less than one hour using a standard workstation, folding TS configurations that are remarkably comparable with the TS configurations reported by D.E. Shaw group[115], which were obtained from milliseconds-long unbiased dynamics on dedicated hardware. Indeed, our results from biased dynamics, which are sketched on the right of Fig. 8.4, also show the formation of the *hairpin bend* in the protein chain, even when starting from the unfolded state. Besides this

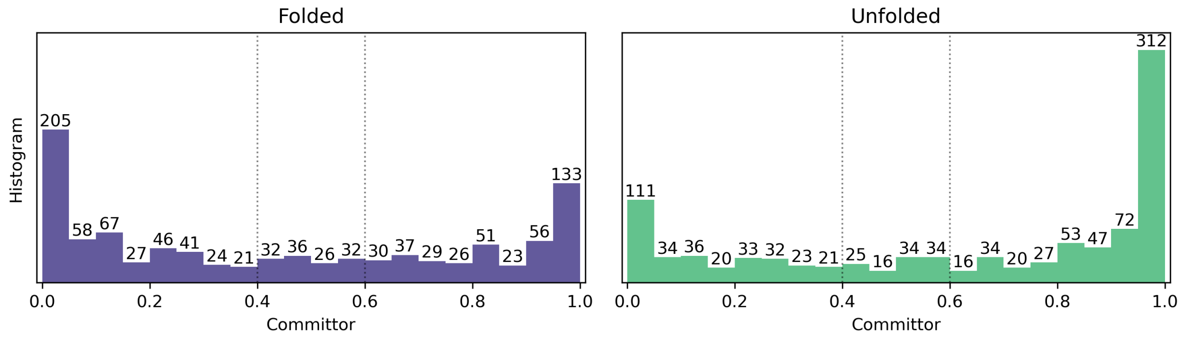


Figure 8.5: Committor analysis on sets of 1000 random configurations sampled with our transition-state-oriented bias starting from the folded state (left) and unfolded state (right). The vertical dotted lines mark the 0.4-0.6 interval associated with the transition state.

qualitative comparison, and in lack of clear TS-defining descriptors, to check if the sampled configurations were actually close to the TS we performed a standard committor analysis. We extract 1000 random configurations from each trajectory and let them relax unbiased in 50 independent replicas to estimate the corresponding committor value. The results of this analysis, which is reported in Fig. 8.5, show that the committor was in the $0.4 - 0.6$ range, thus close to the ideal value, for $\sim 10/15\%$ of the tested configurations. In this regard, we remind that our goal here is not the sampling of the TS itself but rather of the region in its surroundings. For example, if one wanted to optimize a NN force field to study this folding process, the configurations close to the TS would be similarly useful. Moreover, once such a set of configurations is available, it could be used to further improve the quality of our bias potential or as the object of more detailed analysis focused on the TS identification only.

8.4 Additional computational details

General computational details For all our models, we used the `DeepTDA` model implemented in the `mlcvolar` library (see Chapter 5) to train a classifier-like CV from which we could obtain our TS-oriented bias. The `tanh` function has been used as activation function for all the NN in our models to guarantee smoother derivatives.

To deploy the bias to PLUMED, we slightly modified the interface available in the `pytorch` module of the code (see Sec. 5.4) to directly compute the bias within the PLUMED C++ code from the classifier-like CV value using the tools from the LibTorch library[84]. The interface was then modified to return this bias as a PLUMED variable which was then used to bias the simulation using the `BIASVALUE` command.

In all the systems, the biased simulations were performed starting from both metastable basins.

Müller-Brown potential The simulations for the Müller-Brown potential were performed using Langevin dynamics[144] as implemented in the `ves_md_linearexpansion`[126] module of PLUMED[124,125]. The damping constant in the Langevin equation was set to 10/time-unit and natural units ($k_B T = 1$) were used in all the calculations.

The dataset for the training of the classifier-based bias consisted of 2000 configurations for each metastable state. As descriptors and input of the NN, we used the x and y coordinates. We used a NN with structure {2, 32, 16, 1} nodes/layer. The number of training epochs was set to 2000.

We ran the biased simulations starting from both metastable states for 5×10^5 steps and the α factor in Eq. 8.3 was set to 12.

Proton transfer in Tropolone The simulation of the intramolecular proton transfer in tropolone have been carried out using the CP2K-8.1[129] software package patched with PLUMED[124,125] at PM6 semi-empirical level. The integration step was 0.5fs and we used the velocity rescaling thermostat[37] set at 300K with a time constant of 100fs. The superimposition of snapshots reported in Fig. 8.3 was realized using the VMD[211] visualization software.

The dataset for the training of the classifier-based bias consisted of 20000 configurations for each metastable state obtained from unbiased MD trajectories. For our model, we used the `DeepTDA` model implemented in the `mlcolvar` library. As input descriptors for our model, we used the interatomic distances between heavy atoms in the system and the hydrogen-oxygen distances related to the hydrogen involved in the PT. We used a NN with structure {38, 24, 12, 1} nodes/layer. The number of training epochs was set to 2000.

We ran the biased simulations starting from both metastable states for 10^6 steps and the α factor in Eq. 8.3 was set to 45.

Chignolin folding The computational setup and the training set for the study of chignolin were the same as presented in Sec. 6.3.3, we thus refer the Reader to that section for all the necessary details.

For the training, in this case, we used 24000 structures for each state and NN with structure {45, 24, 12, 1} nodes/layer, and the number of training epochs was set to 2000.

We ran the biased simulations starting from both metastable states for 5×10^6 steps and the α factor in Eq. 8.3 was set to 12.

Conclusions and perspectives

The field of atomistic simulations, as many sides of our everyday lives, has been greatly impacted by the rapid development of machine learning techniques in the last few years. As an example, it suffices to mention the development of *ab initio*-quality machine learning interatomic potentials that allow for accurate simulations of complex systems at a fraction of the cost of *first principle* methods, as we have seen in the case of liquid sulfur. This beneficial cross-contamination opens up new paths for future research and makes accessible the study of phenomena that, even just a few years ago, would have been completely unreachable for computational approaches.

In this Thesis, we showed that the impact of such methodologies has also boosted the field of ES simulations in its goal of extending the scope of standard MD simulations. A striking example in this sense can be the transition-state-oriented biasing scheme we proposed, which allows for effortless sampling of this delicate region of the phase space that we typically struggle to access. Indeed, such a result would not have been feasible without the ML tools.

We have also discussed how much effort has been devoted to incorporating ML techniques into determining CVs in a data-driven and semi-automatic way. In this regard, we reported our contribution to this collective effort, also showing an example of the practical impact of such new methods on real-life systems. However, in this direction, there will still be room for improvement in the future. For example, the methods we developed and presented are based on *standard* feed-forward NNs, which are fed with sets of physical descriptors as inputs. However, the use of descriptors and their choice can still be problematic. For this reason, an interesting outlook for future research in this field could be using graph NNs to describe molecular systems.

Another perspective for the near future, which comes naturally as this Thesis is mostly oriented toward method development, is related to extending the applications of such methods to new complex systems. In this regard, we do believe that the `mlcolvar` library and its much-simplified interface will boost progress in this direction, possibly with the development of simple graphical user interfaces (GUIs) to open its use to a broader audience of users that may not be comfortable with coding.

Despite the aforementioned flourishing of ML-aided methods in atomistic simulations, so far, we have most likely only been scratching the surface of the endless possibilities and outcomes of the synergy between these two worlds. For example, besides developing new methods, an open and relevant topic in this sense is related to the interpretability of our simulations and models. Indeed, ML methods bring outstanding contributions in terms of the expressivity of the models, but this often comes at the cost of their easy interpretability. For instance, this is the case of data-driven CVs obtained by combining many physical descriptors, which allow us to perform effectively enhanced sampling simulations but often make it difficult to obtain direct physical insights from such calculations. Possible solutions in this direction could be found by performing *relevance analysis*, in which we look for those variables that are most relevant in our mathematical model, or by improving the *sparsity* of our models with a proper pruning of the inputs aimed at removing the less relevant ones or applying *symbolic regression* techniques to identify functional forms of easier interpretation for our models.

To conclude, another interesting perspective that arises from what we have presented in this Thesis is related to the possibility of developing new enhanced sampling methods by exploiting biasing schemes inspired by the transition-state-oriented we proposed. Hopefully, such new approaches will provide new tools in the enhanced sampling toolbox that can also help to promote transitions between the metastable states in the spirit of other time-honored methods like Metadynamics and Umbrella Sampling.

Publications

Articles published in peer-reviewed journals

[1]E. Trizio and M. Parrinello, “From enhanced sampling to reaction profiles”, J. Phys. Chem. Lett. **12**, 8621–8626 (2021).

[2]D. Ray, E. Trizio, and M. Parrinello, “Deep learning collective variables from transition path ensemble”, J. Chem. Phys. **158**, 204102, `10.1063/5.0148872` (2023).

[3]L. Bonati, E. Trizio, A. Rizzi, and M. Parrinello, “A unified framework for machine learning collective variables for enhanced sampling simulations: mlcolvar”, J. Chem. Phys. **159**, 014801 (2023).

Preprints and articles submitted or in preparation

[1]M. Yang, E. Trizio, and M. Parrinello, “Structure and polymerization of liquid sulfur across the λ-transition”, In peer review (2024).

[2]E. Trizio, P. Kang, and M. Parrinello, “Topic: transition-state-oriented bias potential”, In preparation (2024).

Bibliography

[1]D. Frenkel and B. Smit, *Understanding molecular simulation: from algorithms to applications*, Vol. 1 (Elsevier, 2001).

[2]M. P. Allen and D. J. Tildesley, *Computer simulation of liquids* (Oxford university press, 2017).

[3]R. LeSar, *Introduction to computational materials science: fundamentals to applications* (Cambridge University Press, 2013).

[4]M. Tuckerman, *Statistical mechanics: theory and molecular simulation* (Oxford university press, 2023).

[5]O. Valsson, P. Tiwary, and M. Parrinello, “Enhancing important fluctuations: rare events and metadynamics from a conceptual viewpoint”, Annu. Rev. Phys. Chem. **67**, 159–184 (2016).

[6]G. M. Torrie and J. P. Valleau, “Nonphysical sampling distributions in monte carlo free-energy estimation: umbrella sampling”, J. Comput. Phys. **23**, 187–199 (1977).

[7]A. Laio and M. Parrinello, “Escaping free-energy minima”, Proc. Natl. Acad. Sci. **99**, 12562–12566 (2002).

[8]M. Bonomi, A. Barducci, and M. Parrinello, “Reconstructing the equilibrium boltzmann distribution from well-tempered metadynamics”, J. Comput. Chem. **30**, 1615–1621 (2009).

[9]I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT Press, 2017).

[10]P. Mehta et al., “A high-bias, low-variance introduction to machine learning for physicists”, Phys. Rep. **810**, 1–124 (2019).

[11]G. Carleo et al., “Machine learning and the physical sciences”, Rev. Mod. Phys. **91**, 10.1103/RevModPhys.91.045002 (2019).

[12]F. Noé, A. Tkatchenko, K.-R. Müller, and C. Clementi, “Machine learning for molecular simulation”, Annu. Rev. Phys. Chem. **71**, 361–390 (2020).

[13]J. Behler and M. Parrinello, “Generalized neural-network representation of high-dimensional potential-energy surfaces”, Phys. Rev. Lett **98**, 146401 (2007).

[14]J. Behler and G. Csányi, “Machine learning potentials for extended systems: a perspective”, Eur. Phys. J. B **94**, 1–11 (2021).

[15]M. M. Sultan and V. S. Pande, “Automated design of collective variables using supervised machine learning”, J. Chem. Phys. **149**, 094106 (2018).

[16]L. Bonati, Y.-Y. Zhang, and M. Parrinello, “Neural networks-based variationally enhanced sampling”, Proc. Natl. Acad. Sci. **116**, 17641–17647 (2019).

[17]L. Bonati, G. Piccini, and M. Parrinello, “Deep learning the slow modes for rare events sampling”, Proc. Natl. Acad. Sci. **118**, e2113533118 (2021).

[18]E. Trizio and M. Parrinello, “From enhanced sampling to reaction profiles”, J. Phys. Chem. Lett. **12**, 8621–8626 (2021).

[19]D. Ray, E. Trizio, and M. Parrinello, “Deep learning collective variables from transition path ensemble”, J. Chem. Phys. **158**, 204102, 10.1063/5.0148872 (2023).

[20]L. Bonati, E. Trizio, A. Rizzi, and M. Parrinello, “A unified framework for machine learning collective variables for enhanced sampling simulations: mlcolvar”, J. Chem. Phys. **159**, 014801 (2023).

[21]R. Steudel, *Elemental sulfur and sulfur-rich compounds I*, Vol. 2 (Springer Science & Business Media, 2003).

[22]B. Meyer, “Elemental sulfur”, Chem. Rev. **76**, 367–388 (1976).

[23]J. A. Bondy and U. S. R. Murty, *Graph theory* (Springer Publishing Company, Incorporated, 2008).

[24]L. Zhang, J. Han, H. Wang, R. Car, and W. E, “Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics”, Phys. Rev. Lett **120**, 143001 (2018).

[25]H. Niu, L. Bonati, P. M. Piaggi, and M. Parrinello, “*Ab initio* phase diagram and nucleation of gallium”, Nat. Comm. **11**, 2654 (2020).

[26]M. Yang, T. Karmakar, and M. Parrinello, “Liquid-liquid critical point in phosphorus”, Phys. Rev. Lett. **127**, 080603 (2021).

[27]M. Yang, U. Raucci, and M. Parrinello, “Reactant-induced dynamics of lithium imide surfaces during the ammonia decomposition process”, Nat. Catal., 1–8 (2023).

[28]L. Bonati et al., “Non-linear temperature dependence of nitrogen adsorption and decomposition on Fe(111) surface”, ChemRxiv, 10.26434/chemrxiv-2023-mlmwv (2023).

[29]W. E and E. Vanden-Eijnden, “Transition-path theory and path-finding algorithms for the study of rare events”, Annu. Rev. Phys. Chem. **61**, 391–420 (2010).

[30]D. Chandler, *Introduction to modern statistical mechanics* (Oxford University Press, 1987).

[31]P. A. M. Dirac, “A new notation for quantum mechanics”, Math. Proc. Camb. Philos. Soc. **35**, 416–418 (1939).

[32]K. G. Denbigh, *The principles of chemical equilibrium: with applications in chemistry and chemical engineering*, 4th ed. (Cambridge University Press, 1981).

[33]P. W. Atkins and J. De Paula, *Atkins' physical chemistry* (Oxford university press, 2022).

[34]P. H. Hünenberger, "Thermostat algorithms for molecular dynamics simulations", in *Advanced computer simulation: approaches for soft matter sciences i*, edited by C. Dr. Holm and K. Prof. Dr. Kremer (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005), pp. 105–149.

[35]H. C. Andersen, "Molecular dynamics simulations at constant pressure and/or temperature", J. Chem. Phys. **72**, 2384–2393 (2008).

[36]H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath", J. Chem. Phys. **81**, 3684–3690 (1984).

[37]G. Bussi, D. Donadio, and M. Parrinello, "Canonical sampling through velocity rescaling", J. Chem. Phys. **126**, 014101 (2007).

[38]M. Parrinello and A. Rahman, "Polymorphic transitions in single crystals: A new molecular dynamics method", J. Appl. Phys. **52**, 7182–7190 (1981).

[39]D. C. Rapaport, *The art of molecular dynamics simulation* (Cambridge University Press, 2004).

[40]O. Guvench and A. D. MacKerell, "Comparison of protein force fields for molecular dynamics simulations", in *Molecular modeling of proteins*, edited by A. Kukol (Humana Press, Totowa, NJ, 2008), pp. 63–88.

[41]F. Jensen, *Introduction to computational chemistry* (John wiley & sons, 2017).

[42]E. Kaxiras, *Atomic and electronic structure of solids* (Cambridge University Press, 2003).

[43]W. Thiel, "Semiempirical quantum–chemical methods", WIREs Comput. Mol. Sci. **4**, 145–157 (2014).

[44]F. H. Stillinger and T. A. Weber, "Computer simulation of local order in condensed phases of silicon", Phys. Rev. B **31**, 5262–5271 (1985).

[45]R. Salomon-Ferrer, D. A. Case, and R. C. Walker, "An overview of the amber biomolecular simulation package", WIREs Comput. Mol. Sci. **3**, 198–210 (2013).

[46]B. R. Brooks et al., "Charmm: the biomolecular simulation program", J. Comput. Chem. **30**, 1545–1614 (2009).

[47]W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, "Comparison of simple potential functions for simulating liquid water", J. Chem. Phys. **79**, 926–935 (1983).

[48]M. Yang, L. Bonati, D. Polino, and M. Parrinello, "Using metadynamics to build neural network potentials for reactive events: the case of urea decomposition in water", Catal. Today **387**, 143–149 (2022).

[49]J. Zeng et al., “DeePMD-kit v2: A software package for deep potential models”, J. Chem. Phys. **159**, 054801 (2023).

[50]D. Zhang et al., “DPA-1: pretraining of attention-based deep potential model for molecular simulation”, arXiv preprint arXiv:2208.08236 (2022).

[51]K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, “Schnet–a deep learning architecture for molecules and materials”, J. Chem. Phys. **148**, 241722 (2018).

[52]O. T. Unke and M. Meuwly, “Physnet: a neural network for predicting energies, forces, dipole moments, and partial charges”, J. Chem. Theory Comput. **15**, 3678–3693 (2019).

[53]S. Batzner et al., “E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials”, Nat. communications **13**, 2453 (2022).

[54]M. Geiger and T. Smidt, *E3nn: euclidean neural networks*, 2022.

[55]O. T. Unke et al., “Machine learning force fields”, Chem. Rev. **121**, 10142–10186 (2021).

[56]M. Pinheiro, F. Ge, N. Ferré, P. O. Dral, and M. Barbatti, “Choosing the right molecular machine learning potential”, Chem. Sci. **12**, 14396–14413 (2021).

[57]E. Kocer, T. W. Ko, and J. Behler, “Neural network potentials: a concise overview of methods”, Annu. Rev. Phys. Chem. **73**, PMID: 34982580, 163–186 (2022).

[58]J. Vandermause et al., “On-the-fly active learning of interpretable bayesian force fields for atomistic rare events”, npj Comput. Mater. **6**, 20 (2020).

[59]A. F. Voter, F. Montalenti, and T. C. Germann, “Extending the time scale in atomistic simulation of materials”, Annu. Rev. Mater. Research **32**, 321–346 (2002).

[60]Y. I. Yang, Q. Shao, J. Zhang, L. Yang, and Y. Q. Gao, “Enhanced sampling in molecular dynamics”, J. Chem. Phys. **151**, 070902 (2019).

[61]J. Hénin, T. Lelièvre, M. R. Shirts, O. Valsson, and L. Delemotte, “Enhanced sampling methods for molecular dynamics simulations”, Living J. Comput. Mol. Sci. **4**, 1583 (2022).

[62]A. S. Kamenik, S. M. Linker, and S. Riniker, “Enhanced sampling without borders: on global biasing functions and how to reweight them”, Phys. Chem. Chem. Phys. **24**, 1225–1236 (2022).

[63]L. D. Landau and E. M. Lifshitz, *Statistical physics: volume 5*, Vol. 5 (Elsevier, 2013).

[64]R. Zwanzig, *Non equilibrium statistical mechanics* (Oxford university press, 2003).

[65]M. Invernizzi and M. Parrinello, “Rethinking metadynamics: from bias potentials to probability distributions”, J. Phys. Chem. Lett. **11**, 2731–2736 (2020).

[66]A. Barducci, G. Bussi, and M. Parrinello, “Well-tempered metadynamics: a smoothly converging and tunable free-energy method”, Phys. Rev. Lett. **100**, 020603 (2008).

[67]P. Tiwary and M. Parrinello, “A time-independent free energy estimator for metadynamics”, J. Phys. Chem. B **119**, PMID: 25046020, 736–742 (2015).

[68]M. Invernizzi and M. Parrinello, “Exploration vs convergence speed in adaptive bias enhanced sampling”, J. Chem. Theory Comput., `10.1021/acs.jctc.2c00152` (2022).

[69]D. Ray, N. Ansari, V. Rizzi, M. Invernizzi, and M. Parrinello, “Rare event kinetics from adaptive bias enhanced sampling”, J. Chem. Theory Comput.

[70]B. W. Silverman, *Density estimation for statistics and data analysis* (Routledge, 2018).

[71]H. Grubmüller, “Predicting slow structural transitions in macromolecular systems: conformational flooding”, Phys. Rev. E **52**, 2893–2906 (1995).

[72]A. F. Voter, “A method for accelerating the molecular dynamics simulation of infrequent events”, J. Chem. Phys. **106**, 4665–4677 (1997).

[73]G. Bussi and D. Branduardi, “Free-energy calculations with metadynamics: theory and practice”, in *Reviews in computational chemistry volume 28* (John Wiley & Sons, Ltd, 2015) Chap. 1, pp. 1–49.

[74]G. Bussi and A. Laio, “Using metadynamics to explore complex free-energy landscapes”, Nat. Rev. Phys. **2**, 200–212 (2020).

[75]R. R. Schaller, “Moore’s law: past, present and future”, IEEE spectrum **34**, 52–59 (1997).

[76]Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, nature **521**, 436–444 (2015).

[77]J. Schmidhuber, “Deep learning in neural networks: an overview”, Neur. Netw. **61**, 85–117 (2015).

[78]F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model”, Trans. Neur. Netw. **20**, 61–80 (2009).

[79]D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization”, 3rd ICLR (2014).

[80]*On-line learning in neural networks*, Publications of the Newton Institute (Cambridge University Press, 1999).

[81]I. T. Jolliffe and J. Cadima, “Principal component analysis: a review and recent developments”, Philos. Trans. Royal Soc. A: Math. Phys. Eng. Sci. **374**, 20150202 (2016).

[82]D. E. Rumelhart and D. Zipser, “Feature discovery by competitive learning”, Cogn. Sci. **9**, 75–112 (1985).

[83]C. Wehmeyer and F. Noé, “Time-lagged autoencoders: deep learning of slow collective variables for molecular kinetics”, J. Chem. Phys. **148**, 241703 (2018).

[84]A. Paszke et al., “Pytorch: an imperative style, high-performance deep learning library”, Adv. Neur. Inf. Process. Syst. **32**, 10.48550/arXiv.1912.01703 (2019).

[85]F. Chollet et al., *Keras*, https://keras.io, 2015.

[86]B. Hashemian, D. Millán, and M. Arroyo, “Modeling and enhanced sampling of molecular systems with smooth and nonlinear data-driven collective variables”, J. Chem. Phys. **139**, 214101 (2013).

[87]W. Chen, A. R. Tan, and A. L. Ferguson, “Collective variable discovery and enhanced sampling using autoencoders: innovations in network architecture and error function design”, J. Chem. Phys. **149**, 072312 (2018).

[88]L. Bonati, V. Rizzi, and M. Parrinello, “Data-driven collective variables for enhanced sampling”, J. Phys. Chem. Lett. **11**, 2998–3004 (2020).

[89]G. Piccini, D. Mendels, and M. Parrinello, “Metadynamics with discriminants: a tool for understanding chemistry”, J. Chem. Theory Comput. **14**, 5040–5044 (2018).

[90]P. J. Steinhardt, D. R. Nelson, and M. Ronchetti, “Bond-orientational order in liquids and glasses”, Phys. Rev. B **28**, 784 (1983).

[91]Neha, V. Tiwari, S. Mondal, N. Kumari, and T. Karmakar, “Collective variables for crystallization simulations - from early developments to recent advances”, ACS omega, 10.1021/acsomega.2c06310 (2022).

[92]M. Welling, “Fisher linear discriminant analysis”, Department Comput. Sci. Univ. Tor. (2005).

[93]O. Fleetwood, M. A. Kasimova, A. M. Westerlund, and L. Delemotte, “Molecular insights from conformational ensembles via machine learning”, Biophys. J. **118**, 765–780 (2020).

[94]H. Jung et al., “Machine-guided path sampling to discover mechanisms of molecular self-organization”, Nat. Comput. Sci., 10.1038/s43588-023-00428-z (2023).

[95]P. Novelli, L. Bonati, M. Pontil, and M. Parrinello, “Characterizing metastable states with the help of machine learning”, J. Chem. Theory Comput. **18**, 5195–5202 (2022).

[96]J. M. L. Ribeiro, P. Bravo, Y. Wang, and P. Tiwary, “Reweighted autoencoded variational bayes for enhanced sampling (rave)”, J. Chem. Phys. **149**, 072301 (2018).

[97]T. Lemke and C. Peter, “Encodermap: dimensionality reduction and generation of molecule conformations”, J. Chem. Theory Comput. **15**, 1209–1215 (2019).

[98]D. Mendels, G. Piccini, and M. Parrinello, “Collective variables from local fluctuations”, J. Phys. Chem. Lett. **9**, 2776–2781 (2018).

[99]R. T. McGibbon, B. E. Husic, and V. S. Pande, “Identification of simple reaction coordinates from complex dynamics”, J. Chem. Phys. **146**, 1–18 (2017).

[100]M. Schöberl, N. Zabaras, and P.-S. Koutsourelakis, “Predictive collective variable discovery with deep bayesian models”, J. Chem. Phys. **150**, 024109 (2019).

[101]Y. Wang, J. M. L. Ribeiro, and P. Tiwary, “Past–future information bottleneck for sampling molecular reaction coordinate simultaneously with thermodynamics and kinetics”, Nat. Comm. **10**, 3573 (2019).

[102]A. Mardt, L. Pasquali, H. Wu, and F. Noé, “Vampnets for deep learning of molecular kinetics”, Nat. Comm. **9**, `10.1038/s41467-017-02388-1` (2018).

[103]P. Tiwary and B. J. Berne, “Spectral gap optimization of order parameters for sampling complex molecular systems”, Proc. Natl. Acad. Sci. **113**, 2839–2844 (2016).

[104]F. Noé and C. Clementi, “Collective variables for the study of long-time kinetics from molecular trajectories: theory and methods”, Current opinion structural biology **43**, 141–147 (2017).

[105]W. Chen and A. L. Ferguson, “Molecular enhanced sampling with autoencoders: on-the-fly collective variable discovery and accelerated free energy landscape exploration”, J. Comput. Chem. **39**, 2079–2102 (2018).

[106]H. Chen and C. Chipot, “Chasing collective variables using temporal data-driven strategies”, QRB Discov. **4**, e2 (2023).

[107]G. A. Tribello and P. Gasparotto, “Using dimensionality reduction to analyze protein trajectories”, Front. Mol. Biosci. **6**, 46 (2019).

[108]M. Ceriotti, “Unsupervised machine learning in atomistic simulations, between predictions and understanding”, J. Chem. Phys. **150**, `10.1063/1.5091842` (2019).

[109]A. Amadei, A. B. Linssen, and H. J. Berendsen, “Essential dynamics of proteins”, Proteins: Struct. Funct. Bioinform. **17**, 412–425 (1993).

[110]Z. Belkacemi, P. Gkeka, T. Lelièvre, and G. Stoltz, “Chasing collective variables using autoencoders and biased trajectories”, J. Chem. Theory Comput. **18**, 59–78 (2022).

[111]T. Karmakar, M. Invernizzi, V. Rizzi, and M. Parrinello, “Collective variables for the study of crystallisation”, Mol. Phys., `10.1080/00268976.2021.1893848` (2021).

[112]N. Ansari, V. Rizzi, and M. Parrinello, “Water regulates the residence time of benzamidine in trypsin”, Nat. Comm., `10.1038/s41467-022-33104-3` (2022).

[113]V. Rizzi, L. Bonati, N. Ansari, and M. Parrinello, “The role of water in host-guest interaction”, Nat. Comm. **12**, 2–8 (2021).

[114]S. Das, U. Raucci, R. P. P. Neves, M. J. Ramos, and M. Parrinello, “How and when does an enzyme react? unraveling α-amylase catalytic activity with enhanced sampling techniques”, ACS Catal. **13**, 8092–8098 (2023).

[115]K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw, “How fast-folding proteins fold”, Science **334**, 517–520 (2011).

[116]L. Molgedey and H. G. Schuster, “Separation of a mixture of independent signals using time delayed correlations”, Phys. Rev. Lett. **72**, 3634 (1994).

[117]Y. Naritomi, S. Naritomi, and Fuchigami, “Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: the case of domain motions”, J. Chem. Phys. **134**, 65101 (2011).

[118]G. Pérez-Hernández, F. Paul, T. Giorgino, G. D. Fabritiis, and F. Noé, “Identification of slow molecular order parameters for markov model construction”, J. Chem. Phys. **139**, 15102 (2013).

[119]W. Chen, H. Sidky, and A. L. Ferguson, “Capabilities and limitations of time-lagged autoencoders for slow mode discovery in dynamical systems”, J. Chem. Phys **151**, 64123 (2019).

[120]C. X. Hernández, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, and V. S. Pande, “Variational encoding of complex dynamics”, Phys. Rev. E **97**, 062412 (2018).

[121]L. Sun et al., “Multitask machine learning of collective variables for enhanced sampling of rare events”, J. Chem. Theory Comput. **18**, 2341–2353 (2022).

[122]M. Ceriotti, G. A. Tribello, and M. Parrinello, “Simplifying the representation of complex free-energy landscapes using sketch-map”, Proc. Natl. Acad. Sci. **108**, 13023–13028 (2011).

[123]W. Falcon and T. P. L. team, *Pytorch lightning*, version 2.0.2, 2023.

[124]G. A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, and G. Bussi, “Plumed 2: new feathers for an old bird”, Comput. Phys. Comm. **185**, 604–613 (2014).

[125]“Promoting transparency and reproducibility in enhanced molecular simulations”, Nat. Methods **16**, 670–673 (2019).

[126]O. Valsson and M. Parrinello, “Variational approach to enhanced sampling and free energy calculations”, Phys. Rev. Lett. **113**, 090601 (2014).

[127]A. P. Thompson et al., “LAMMPS-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales”, Comput. Phys. Commun. **271**, 108171 (2022).

[128]M. J. Abraham et al., “Gromacs: high performance molecular simulations through multi-level parallelism from laptops to supercomputers”, SoftwareX **1**, 19–25 (2015).

[129]T. D. Kühne et al., "CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations", J. Chem. Phys. **152**, 194103 (2020).

[130]P. Giannozzi et al., "Advanced capabilities for materials modelling with Quantum ESPRESSO", J. Physics: Condens. Matter **29**, 465901 (2017).

[131]A. H. Larsen et al., "The atomic simulation environment—a python library for working with atoms", J. Phys. Condens. Matter **29**, 273002 (2017).

[132]T. Giorgino, "Pycv: a plumed 2 module enabling the rapid prototyping of collective variables in python", J. Open Source Softw. **4**, 1773 (2019).

[133]U. Raucci, V. Rizzi, and M. Parrinello, "Discover, sample, and refine: exploring chemistry with enhanced sampling techniques", J. Phys. Chem. Lett **2022**, `10.1021/acs.jpclett.1c03993` (2022).

[134]D. P. Kingma and M. Welling, "Auto-encoding variational bayes", arXiv preprint arXiv:1312.6114, `10.48550/arXiv.1312.6114` (2013).

[135]G. A. Tribello, M. Ceriotti, and M. Parrinello, "Using sketch-map coordinates to analyze and bias molecular dynamics simulations", Proc. Natl. Acad. Sci. **109**, 5196–5201 (2012).

[136]J. Rydzewski and O. Valsson, "Multiscale reweighted stochastic embedding: deep learning of collective variables for enhanced sampling", J. Phys. Chem. A **125**, 6286–6302 (2021).

[137]M. Dorfer, R. Kelz, and G. Widmer, "Deep linear discriminant analysis", in (2016).

[138]J. H. Prinz et al., "Markov models of molecular kinetics: generation and validation", J. Chem. Phys. **134**, 174105 (2011).

[139]M. M. Sultan and V. S. Pande, "Tica-metadynamics: accelerating metadynamics by using kinetically selected collective variables", J. Chem. Theory Comput. **13**, 2440–2447 (2017).

[140]J. McCarty and M. Parrinello, "A variational conformational dynamics approach to the selection of collective variables in metadynamics", J. Chem. Phys. **147**, 204109 (2017).

[141]Y. I. Yang and M. Parrinello, "Refining collective coordinates and improving free energy representation in variational enhanced sampling", J. Chem. Theory Comput. **14**, 2889–2894 (2018).

[142]W. Chen, H. Sidky, and A. L. Ferguson, "Nonlinear discovery of slow molecular modes using state-free reversible vampnets", J. Chem. Phys. **150**, 214114 (2019).

[143]V. Kostic et al., "Learning dynamical systems via koopman operator regression in reproducing kernel hilbert spaces", arXiv preprint arXiv:2205.14027, `10.48550/arXiv.2205.14027` (2022).

[144]G. Bussi and M. Parrinello, “Accurate sampling using langevin dynamics”, Phys. Rev. E **75**, 056707 (2007).

[145]J. Debnath and M. Parrinello, “Gaussian mixture-based enhanced sampling for statics and dynamics”, J. Phys. Chem. Lett. **11**, 5076–5080 (2020).

[146]V. Rizzi, D. Mendels, E. Sicilia, and M. Parrinello, “Blind search for complex chemical pathways using harmonic linear discriminant analysis”, J. Chem. Theory Comput. **15**, 4507–4515 (2019).

[147]L. Maragliano, A. Fischer, E. Vanden-Eijnden, and G. Ciccotti, “String method in collective variables: minimum free energy paths and isocommittor surfaces”, J. chemical physics **125**, 024106 (2006).

[148]H. Rumpel and H. H. Limbach, “Nmr study of kinetic hh/hd/dd isotope, solvent and solid-state effects on the double proton transfer in azophenine”, J. Am. Chem. Soc. **111**, 5429–5441 (1989).

[149]M. Fatollahpour and H. Tahermansouri, “Dft study of the intramolecular double proton transfer of 2, 5-diamino-1, 4-benzoquinone and its derivatives, and investigations about their aromaticity”, Comptes Rendus Chimie **20**, 942–951 (2017).

[150]P. Haley and D. Soloway, “Extrapolation limitations of multilayer feedforward neural networks”, Proc. Int. Jt. Conf. on Neur. Netw. **4**, 25–30 vol.4 (1992).

[151]E. Barnard and L. Wessels, “Extrapolation and interpolation in neural network classifiers”, IEEE Control Syst. Mag. **12**, 50–53 (1992).

[152]K. Xu et al., “How neural networks extrapolate: from feedforward to graph neural networks”, arXiv preprint arXiv:2009.11848 (2020).

[153]C. Dellago, P. G. Bolhuis, F. S. Csajka, and D. Chandler, “Transition path sampling and the calculation of rate constants”, J. Chem. Phys. **108**, 1964–1977 (1998).

[154]R. G. Mullen, J.-E. Shea, and B. Peters, “Easy transition path sampling methods: flexible-length aimless shooting and permutation shooting”, J. Chem. Theory Comput. **11**, 2421–2428 (2015).

[155]T. S. Van Erp, D. Moroni, and P. G. Bolhuis, “A novel path sampling method for the calculation of rate constants”, J. Chem. Phys. **118**, 7762–7774 (2003).

[156]D. Mandelli, B. Hirshberg, and M. Parrinello, “Metadynamics of paths”, Phys. Rev. Lett. **125**, 026001 (2020).

[157]J. Yin et al., “Overview of the sampl5 host–guest challenge: are we doing better?”, J. Comput. Mol. Design **31**, 1–19 (2017).

[158]J. Debnath and M. Parrinello, “Computing rates and understanding unbinding mechanisms in host–guest systems”, J. Chem. Theory Comput. **18**, 1314–1319 (2022).

[159]S. Piana, K. Lindorff-Larsen, and D. E. Shaw, “How robust are protein folding simulations with respect to force field parameterization?”, Biophys. J. **100**, L47–L49 (2011).

[160]A. D. J. MacKerell et al., “All-atom empirical potential for molecular modeling and dynamics studies of proteins”, J. Phys. Chem. B **102**, 3586–3616 (1998).

[161]J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, “Development and testing of a general amber force field”, J. Comput. Chem. **25**, 1157–1174 (2004).

[162]D. Ray and M. Parrinello, “Kinetics from metadynamics: principles, applications, and outlook”, J. Chem. Theory Comput. **19**, PMID: 37585703, 5649–5670 (2023).

[163]M. Salvalaglio, P. Tiwary, and M. Parrinello, “Assessing the reliability of the dynamics reconstructed from metadynamics”, J. Chem. Theory Comput. **10**, 1420–1425 (2014).

[164]K. S. Kaminsky, “Confidence intervals for the exponential scale parameter using optimally selected order statistics”, Technometrics **14**, 371–383 (1972).

[165]L. Crapanzano, W. A. Crichton, G. Monaco, R. Bellissent, and M. Mezouar, “Alternating sequence of ring and chain structures in sulphur at high pressure and temperature”, Nat. Mater. **4**, 550–552 (2005).

[166]S. Geller, “Pressure-induced phases of sulfur”, Science **152**, 644–646 (1966).

[167]M. Lind and S. Geller, “Structure of pressure-induced fibrous sulfur”, J. Chem. Phys. **51**, 348–353 (1969).

[168]W. A. Crichton, G. B. M. Vaughan, and M. Mezouar, “In situ structure solution of helical sulphur at 3GPa and 400°C”, Zeitschrift für Kristallographie - Cryst. Mater. **216**, 417–419 (2001).

[169]“Liquid–liquid transition and critical point in sulfur”, Nature **584**, 382–386 (2020).

[170]R. Bellissent, L. Descotes, F. Bouc, and P. Pfeuty, “Liquid sulfur: local-order evidence of a polymerization transition”, Phys. Rev. B **41**, `10.1103/PhysRevB.41.2135` (1990).

[171]R. Steudel, “Liquid sulfur”, Top. Current Chem., 81–116 (2012).

[172]L. Liu et al., “Chain breakage in liquid sulfur at high pressures and high temperatures”, Phys. Rev. B **89**, 174201 (2014).

[173]G. E. Sauer and L. B. Borst, “Lambda transition in liquid sulfur”, Science **158**, 1567–1569 (1967).

[174]K. M. Zheng and S. C. Greer, “The density of liquid sulfur near the polymerization temperature”, J. Chem. Phys. **96**, 2175–2182 (1992).

[175]A. V. Tobolsky and A. Eisenberg, “Equilibrium polymerization of sulfur”, J. Am. Chem. Soc. **81**, 780–782 (1959).

[176]V. F. Kozhevnikov, W. B. Payne, J. K. Olson, C. L. McDonald, and C. E. Inglefield, “Physical properties of sulfur near the polymerization transition”, J. Chem. Phys. **121**, 7379–7386 (2004).

[177]R. Winter et al., “The structural properties of liquid sulphur”, J. Phys. Condens. Matter **2**, 8427 (1990).

[178]M. Stolz, R. Winter, W. S. Howells, R. L. McGreevy, and P. A. Egelstaff, “The structural properties of liquid and quenched sulphur II”, J. Phys. Condens. Matter **6**, 3619 (1994).

[179]C. Biermann, R. Winter, C. Benmore, and P. Egelstaff, “Structural and dynamic properties of liquid sulfur around the λ-transition”, J. Non-Crystalline Solids **232-234**, 309–313 (1998).

[180]A. G. Kalampounias, K. S. Andrikopoulos, and S. N. Yannopoulos, “Probing the sulfur polymerization transition in situ with Raman spectroscopy”, J. Chem. Phys. **118**, 8460–8467 (2003).

[181]H. Flores-Ruiz and M. Micoulaut, “Crucial role of s 8-rings in structural, relaxation, vibrational, and electronic properties of liquid sulfur close to the λ transition”, J. Chem. Phys. **157**, `10.1063/5.0090953` (2022).

[182]R. O. Jones and P. Ballone, “Density functional and Monte Carlo studies of sulfur. I. Structure and bonding in Sn rings and chains (n=2–18)”, J. Chem. Phys. **118**, 9257–9265 (2003).

[183]S. Munejiri, F. Shimojo, and K. Hoshino, “Photo-induced structural change in liquid sulphur”, J. Phys. Condens. Matter **12**, 7999–8008 (2000).

[184]J. S. Tse and D. D. Klug, “Structure and dynamics of liquid sulphur”, Phys. Rev. B **59**, 34–37 (1999).

[185]M. W. Wong, Y. Steudel, and R. Steudel, “Novel species for the sulfur zoo: isomers of S8”, Chem. Phys. Lett. **364**, 387–392 (2002).

[186]P. C. Hiemenz and T. P. Lodge, *Polymer chemistry* (CRC press, 2007).

[187]L. Bonati and M. Parrinello, “Silicon liquid structure and crystal nucleation from *ab initio* deep metadynamics”, Phys. Rev. Lett **121**, 265701 (2018).

[188]Y. Zhang et al., “DP-GEN: a concurrent learning platform for the generation of reliable deep learning based potential energy models”, Comput. Phys. Comm. **253**, 107206 (2020).

[189]F. Pietrucci and W. Andreoni, “Graph theory meets *Ab Initio* molecular dynamics: atomic structures and transformations at the nanoscale”, Phys. Rev. Lett. **107**, 085504 (2011).

[190]G. Henkelman, A. Arnaldsson, and H. Jónsson, “A fast and robust algorithm for bader decomposition of charge density”, Comput. Mater. Sci. **36**, 354–360 (2006).

[191]H. Wang, L. Zhang, J. Han, and W. E, “DeePMD-kit: a deep learning package for many-body potential energy representation and molecular dynamics”, Comput. Phys. Comm. **228**, 178–184 (2018).

[192]A. Stukowski, “Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool”, Model. Simul. Mater. Sci. Eng. **18**, {10.1088/0965-0393/18/1/015012} (2010).

[193]J. P. Perdew, K. Burke, and M. Ernzerhof, “Generalized gradient approximation made simple”, Phys. Rev. Lett. **77**, 3865–3868 (1996).

[194]S. Goedecker, M. Teter, and J. Hutter, “Separable dual-space gaussian pseudopotentials”, Phys. Rev. B **54**, 1703–1710 (1996).

[195]C. Hartwigsen, S. Goedecker, and J. Hutter, “Relativistic separable dual-space gaussian pseudopotentials from H to Rn”, Phys. Rev. B **58**, 3641–3662 (1998).

[196]D. J. Evans and B. L. Holian, “The Nose–Hoover thermostat”, J. Chem. Phys. **83**, 4069–4074 (1985).

[197]S. Melchionna, G. Ciccotti, and B. L. Holian, “Hoover NPT dynamics for systems varying in shape and size”, Mol. Phys. **78**, 533–544 (1993).

[198]S. Grimme, S. Ehrlich, and L. Goerigk, “Effect of the damping function in dispersion corrected density functional theory”, J. Comput. Chem. **32**, 1456–1465 (2011).

[199]K. S. Vahvaselkä and J. M. Mangs, “X-ray diffraction study of liquid sulfur”, Physica Scripta **38**, 737 (1988).

[200]W. J. Klement and J. C. Koh, “Polymer content of sulfur quenched rapidly from the melt”, J. Phys. Chem. **74**, 4280–4284 (1970).

[201]L. Crapanzano, “Polymorphism of sulfur: Structural and Dynamical Aspects”, Theses (Université Joseph-Fourier - Grenoble I, 2006).

[202]S. Teil, “Schnelle chromatographische trennung und bestimmung der schwefelhomocyclen s „(n= 6—28) mittels HPLC”, Fresenius Z Anal. Chem. **326**, 543–546 (1987).

[203]T. G. Solomons and C. B. Fryhle, *Organic chemistry* (John Wiley & Sons, 2008).

[204]E. V. Anslyn and D. A. Dougherty, *Modern physical organic chemistry* (University science books, 2006).

[205]R. L. Baldwin and G. D. Rose, “Is protein folding hierarchic? ii. folding intermediates and transition states”, Trends Biochem. Sci. **24**, 77–83 (1999).

[206]C. Cecconi, E. A. Shank, C. Bustamante, and S. Marqusee, “Direct observation of the three-state folding of a single protein molecule”, Science **309**, 2057–2060 (2005).

[207]S. E. Jackson, “How do small single-domain proteins fold?”, Fold. Design **3**, R81–R91 (1998).

[208]J. Behler, "Perspective: Machine learning potentials for atomistic simulations", J. Chem. Phys. **145**, 170901 (2016).

[209]A. Kolmogoroff, "Über die analytischen methoden in der wahrscheinlichkeitsrechnung", Math. Ann. **104**, 415–458 (1931).

[210]P. Metzner, C. Schütte, and E. Vanden-Eijnden, "Illustration of transition path theory on a collection of simple examples", J. Chem. Phys. **125**, 084110 (2006).

[211]W. Humphrey, A. Dalke, and K. Schulten, "VMD – Visual Molecular Dynamics", J. Mol. Graph. **14**, 33–38 (1996).

Acknowledgments

First, I want to sincerely thank Prof. Michele Parrinello. Not only for being a great and passionate supervisor but also for always doing that with a smile and a joyful spirit. All his advice has been precious for coming to this Thesis, but the possibility of having fun while doing it was priceless. I then have to thank Prof. Francesco Montalenti. Without his advice, I probably would not have undertaken this path, and having him as a tutor has been a pleasure.

How not to mention the colleagues and friends I have met during these PhD years at IIT: Ana, Andrea, Axel, Francesco, Davide, Dani, Dhiman, Jayashrita, Luigi, Narjes, Nicolò, Manyi, Pedro, Pengchao, Peilin, Shivam, Simone, Sudip, Valerio, Umberto. Being able to know such different cultures and backgrounds is a precious experience that goes beyond science. Among them, I owe special thanks (science-wise and not) to Davide, who helped me to take my very first steps in this group, and Umberto, who, despite his continuous threats to my hair, has been a great companion in designing our amazing farewell gifts. Thanks to Luigi for all his random ideas and all the sentences left blank, and to Andrea for being the only one capable of understanding *The Office* jokes. But above all, thanks to both of them for letting me put a purple capybara as the logo of our library. I also need to mention the patience of Francesco and Simone, who have been great roommates, sharing biscuits, problems, and laughters. Dulcis in fundo, I want to thank An(it)a and Manyi for taking turns being my mother and grandmother. Having them around has been *lovely*, to say the least.

If I managed to arrive sane at the end of this work, great merit also belongs to the people who have been surrounding me along the way. I am lucky enough to have new friends to thank, like all my swimming teammates, who have welcomed me with open arms, making me feel at home here in Genova. However, I also cannot thank enough the old ones who have always been there for me: Alice, Elisa, Filippo, Leonardo, Mattia, Sihem, Sofia and Vittorio.

Last but not least, I want to say thank you to those in my closest circle. To my parents, Francesca and Pasquale, for their love and support along the way and all the advice that I often pretend not to care about. To my brother Vincenzo and (my kind of sister) Chiara. At the cost of repeating myself Thesis after Thesis, I am still surprised by the utility of your advice. And finally, a special thanks to Aritz for being Aritz, I could not have asked for more.