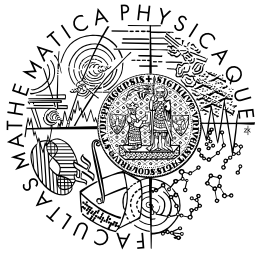

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Martin Šípka

Machine learning through geometric mechanics and thermodynamics

Mathematical Institute of Charles University

Supervisor of the doctoral thesis: doc. RNDr. Michal Pavelka, Ph.D.

Study programme: Mathematical and computer modeling

Study branch: P4F11

Prague 2023

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

First of all, I would like to thank my family for their lasting support and encouragement. Without their help, none of this would be possible, and I would likely, instead of pursuing an academic career, take a shortcut somewhere along the way.

My gratitude also belongs to my dear Barbora. She never let me accept any failure and kept me on the path. Her kind words of encouragement were often needed in times when things, as they often do in science, did not work as I naively thought they would.

I am deeply thankful to my supervisor, Michal, and advisor, Lukáš. They were not only my managers, as it sometimes unfortunately happens, but rather my mentors and allies, without whose help you would not be reading these words.

I would also like to thank Rafa, who was my advisor when visiting foreign lands. His capacity to provide fresh and exciting ideas was staggering, his knowledge vast and often complementary to mine as is usually best.

Ondra from my lab showed me how top-level research is done with just excellent ideas and mathematical proficiency, Avni kept my expectations about machine learning realistic and always knew where the value was, Simon never refused to help me and encouraged me to do my best, Akshay never took things too seriously as no one ever should, Kevin was the only one to agree to join for a Bruins game, Lucia never refused to join me for a coffee when things were not going my way, and Johannes helped me to build some of that proficiency and reliability he certainly has plenty of.

Lastly, I thank my dog Beny, who, even though he cares more about running away than machine learning, reminded me that even that can lead to happiness, forming a realistic backup plan.

Thank you.

Title: Machine learning through geometric mechanics and thermodynamics

Author: Martin Šípka

Institute: Mathematical Institute of Charles University

Supervisor: doc. RNDr. Michal Pavelka, Ph.D., Mathematical Institute of Charles University

Abstract: This thesis studies novel approaches to learning of physical models, incorporating constraints and optimizing path dependent loss functions. Recent advances in deep learning and artificial intelligence are connected with established knowledge about dynamical and chemical systems, offering new synergies and improving upon existing methodologies. We present significant contributions to simulation techniques that utilize automatic differentiation to propagate through the dynamics, showing not only their promising use case but also formulating new theoretical results about the gradient behavior in long evolutions controlled by neural networks. All the tools are carefully tested and evaluated on examples from physics and chemistry, thus proposing and promoting their further applications.

Keywords: Machine learning, Geometrical mechanics, Chemical reaction, Enhanced sampling

Contents

Introduction

The evolution of physics and derived natural sciences is only possible with improvements in how physical laws are written, interpreted, and used in the context of their predictive power. A successful theory is distinguished by incorporating interpretable constants, which derive their roots from a profound comprehension of potential limitations. Often, an idea is formulated in full generality that applies to a broad set of systems, but for most, except a few trivial ones, it is not expected to be used in practice. In science and engineering, a successful method designed for practical applications must rigorously consider two fundamental facets: the complexity and computational cost for realistic systems, and carefully evaluate the simplifications and reductions needed to satisfy the operational constraints of its users.

When evaluating the trade-off between complexity and computational cost, the pursuit of an optimal configuration for a specific task demands advanced insights and intuition. This quest for optimality depends upon the capacity to sustain the desired level of precision while minimizing associated costs or, conversely, adhering to acceptable costs while maximizing precision. The approach of reducing the complexity of a particular model is one of the most essential concepts during physical model creation and implementation.

One of the ways to capture physics and knowledge of natural laws into the reduced models is through mathematics. The branch investigating geometry naturally provides an elegant and often intuitive way to write complex theory in a more approachable and compact manner. Geometrical interpretation also connects multiple disciplines and allows us to visualize and interpret results, uncovering relationships between physics and its mathematical language.

When reducing the complexity and dimensionality of models, incorporating geometry becomes even more critical as a blindly constructed model reduction often lacks a theoretical framework and basis to anchor it within the laws of physics. Geometry gives interpretability and forms a foundation for deeper understanding.

There are examples from history where scientists approached the relationship of geometry and physics from both directions. A case of geometry yielding a physical theory can be illustrated by the formation of the general theory of relativity [1]. Albert Einstein utilized the geometrical concepts of curved Riemann geometry when he formulated General Relativity in the early 20th century. Riemannian geometry became crucial for Einstein to describe the curvature of spacetime as being due to the presence of various forms of matter. On the contrary, Quantum Mechanics became useful before its complete mathematical framework, functional analysis, was developed. The immense importance and experimental evidence of validity became the driving force behind improvements in operator theory and mathematics of infinite dimensional spaces.

A typical way to create a reduced model is to formulate a series of equations with a small set of fittable constants, e.g., material parameters. The correct values in this set are then found by various methods, for example, by least square fitting or gradient-based methods [2]. The resulting model is considered optimized and can be used to simulate and model various scenarios using its predictive power.

With the emergence of machine learning (ML), a new way to construct re-

duced representations of physical relationships emerged. Instead of modeling an equation that has been designed specifically to accommodate a small number of fittable parameters, we can now use the original model in full generality and represent the building blocks as flexible functions with many parameters, optimizing them by advanced techniques such as backpropagation. We are able to pivot from optimizing a small set of physical constants to fitting a function with thousands of parameters thanks to the improvements in ML achieved in recent years.

However, this new frontier of model construction comes with the risk of 'blind fitting,' where ML algorithms, with their flexible, functional forms and vast parameter spaces, can fit data without regard to any form of underlying physics. While these models may deliver impressive results on training datasets, their ability to generalize beyond that and their interpretability can be severely limited. This predicament often leads to models that are impressive at first glance yet often lack their primary purpose of providing trustworthy and immediately usable information about the system of interest.

This thesis advocates for a more conscious and careful use of ML models in capturing physical laws. The approach presented here emphasizes the importance of interpretability and the integration of domain knowledge into the learning process. By connecting theoretical principles with data-driven methods, we can guide the learning algorithms to respect the constraints and symmetries inherent in physical systems.

We aim for a paradigm where ML is not merely a statistical tool but a means to extend our understanding of physics. By carefully crafting the architecture of the models to mirror the structured knowledge of physics and implementing regularization techniques inspired by physical insights, we can embed the essence of physical laws into the models. This incorporation allows the models to learn not just patterns but the governing principles coming directly from geometry and mathematical laws.

The thesis first introduces basic concepts of ML to the audience with a background in mathematics, physics, or chemistry, presenting basic concepts like backpropagation or a loss function. Later, we dive into the details of Hamiltonian learning and chemical reactions, showing how ML impacts those fields.

The thesis provides an overview of three of the papers that were published during the course of the PhD study program. The first one [3] is discussed in Chapter 2, and its main results are summarized. The other two papers [4] and [5] are the basis for the Chapter 3, and they offer two approaches to the modeling of chemical reactions on the molecular level.

1. Machine learning in science

Machine learning, a subfield of artificial intelligence, has helped advance a number of scientific disciplines in recent years. The advances in ML methods and improvements in its critical components (e.g., better neural network architectures) enabled computational systems to adaptively improve their performance by assimilating information from data, thereby progressively refining their predictive or decision-making capabilities [6]. This also meant the necessity for the development of firm theoretical foundations, which sought to comprehend the computational and representational capabilities of neural architectures [7].

The advent of ML can be split across several epochs, beginning with simple models, like perceptron introduced by [8]. The progress continued with notable contributions in the conceptualization and development of Support Vector Machines (SVMs), which enriched the landscape of ML by introducing novel tools for regression and classification, underpinned by the robust theoretical framework of statistical learning theory [9]. Decision Trees [10], and subsequently Random Forests [11], offered an alternative of simple, robust models, well equipped for various data formats and a wide variety of tasks [12].

An era of 'deep learning renaissance,' defined by the breakthroughs achieved through multi-layered neural architectures, has consistently extended the frontiers of feasibilities in ML [13]. This epoch has been characterized by the amplification of model capacities and the augmentation of computational resources, which collectively have led to unprecedented successes in tasks such as image classification, natural language processing, and reinforcement learning [14].

1.1 Science and ML

The advances in computer science have been quickly translated to other sciences that commonly use computing power to process, generate, and simulate data. In scientific exploration, particularly within physics and chemistry, machine learning (ML) has emerged as a helpful tool, dramatically altering traditional methodologies and enhancing the capabilities of researchers. The infusion of ML in these fields synthesizes a new paradigm where predictive modeling, data interpretation, simulation, and experimentation converge to accelerate discovery and innovation.

In physics, machine learning contributes significantly to areas demanding vast computational resources or those involving incomprehensibly large data sets. One such critical area is high-energy physics. Experiments in large hadron colliders produce enormous amounts of data, which traditional data analysis methods struggle to process. ML algorithms help in filtering and analyzing it, enabling researchers to identify novel particles and understand fundamental forces of the universe more efficiently [15].

Moreover, in the realm of condensed matter physics, ML assists in the exploration of phase transitions and the identification of material properties, tasks that typically require intensive computational simulations [16]. Astrophysicists equally leverage ML, employing sophisticated algorithms for the classification and understanding of celestial objects, aiding in the deciphering of cosmic events from data accrued through telescopes and space observatories [17].

In chemistry, machine learning facilitates a new era of accelerated experimentation and discovery. The field of chemical synthesis is notably impacted, with ML models predicting possible reaction outcomes or recommending novel pathways, thereby expediting the synthesis of desired compounds [18]. These advancements are particularly pertinent in drug discovery, where ML algorithms search vast chemical spaces to identify potential therapeutic compounds, significantly shortening development timelines and reducing costs.

Furthermore, in materials science, a discipline interlinked with chemistry, ML is instrumental in the discovery and design of new materials. By integrating historical data on material properties, ML models can predict the properties of unknown compounds, guide the synthesis of materials with desired characteristics, and considerably lower the need for expensive and time-consuming physical trials [19].

Quantum chemistry also benefits immensely, with ML models assisting in solving Schrödinger's equation for increasingly complex systems more efficiently than traditional numerical methods, offering profound insights into molecular behavior and interactions [20].

1.2 Neural Networks

In this thesis, we will be working with neural networks as building blocks. We shall, therefore, briefly explain their functionality. Neural networks serve as interpolators projecting inputs to outputs via a non-linear function parametrized by a vector of so-called weights. As they consist of several interconnected layers of neurons, they can be seen as compositions of high-dimensional mappings. Per the Universal Approximation Theorem, every continuous function from $\mathbb{R}^n$ to $\mathbb{R}^m$ can be approximated by a neural network with one hidden layer and a non-linear activation function [21, 22, 23]. These multi-layered architectures have shown a great capability for tasks like image recognition, natural language processing, and other applications of deep learning [13].

To understand how a simple neural network works, let us consider a perceptron first. The input of a perceptron is a vector of numbers $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ and a vector of weights $\mathbf{w} = [w_1, w_2, \ldots, w_n]$. On the output, a scalar product is computed, including a bias term b as

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b = \mathbf{w}^\top \mathbf{x} + b \tag{1.1}$$

This output z is then passed through an activation function f. Since a perceptron was initially a binary classifier, a step function was used

$$f(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{if } z < \theta. \end{cases} \tag{1.2}$$

To iteratively train the perceptron, we use a simple training algorithm. Assume y_j is the output and t^j is the ground truth value for a datapoint x_j. For all misclassified data points, we then do the weight update:

$$w_{i+1} = w_i + t^j x_i^j, \tag{1.3}$$

while for correctly classified examples, we do nothing. This forms a simple binary classifier that can only guarantee the classification of linearly separable data. While its initial use was, therefore, limited, by modifying and extending perceptron, we can build the modern theory of neural networks.

Instead of classifying the data, we can, with a different choice of an activation function, predict a continuous value. Moreover, by stacking multiple perceptrons, we can increase the ability of a network to capture more complex relationships, creating a so-called multi-layer perceptron (MLP), sometimes named a dense or fully connected neural network. An MLP is, to this day, an important component of many deep learning architectures and designs. Contrary to a single perceptron, an MLP contains the so-called hidden layers, the intermediary outputs of perceptrons before the final layer. The output of a perceptron in a hidden layer becomes a vector, and weights become matrices $\mathbf{W}$. The equation then may look like:

$$\begin{aligned} \mathbf{y}_1 &= f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\ \mathbf{y}_2 &= f(\mathbf{W}_2\mathbf{y}_1 + \mathbf{b}_2) \\ \mathbf{y}_3 &= f(\mathbf{W}_3\mathbf{y}_2 + \mathbf{b}_3) \\ y &= \mathbf{W}_4\mathbf{y}_3 + \mathbf{b}_4 \end{aligned} \tag{1.4}$$

See Figure 1.1. The mathematical operation each neuron typically carries out can be understood as an affine transformation of the inputs followed by the application of a non-linear activation function [14]. Parameters of the affine transformation (weights and biases) are then subject to the optimization procedure, where they are sought so that a prescribed loss function is minimized. The activation function plays a crucial role as it provides non-linearity to the neural network, enabling it to approximate more complicated functions. To understand how such extension can be trained, we need to dive into concepts of loss function and backpropagation.

1.2.1 Loss function and Maximum Likelihood Principle

A loss function will serve as a measure of how far we are from the optimal outcome. It is designed such that when the network predicts the correct value, the loss is zero. An example of a popular loss is a mean squared error between the predicted and actual values

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1.5}$$

where:

- n is the number of observations.
- y_i is the actual value of the i-th observation.
- $\hat{y}_i$ is the predicted value of the i-th observation.

A reader familiar with numerical mathematics may instead know this loss by the term average l_2 error.

There are, however, many more loss functions we can use. We need to look at the Maximum Likelihood Principle to understand where they come from and lay the foundations for deriving various types of losses suitable for different objectives and tasks.

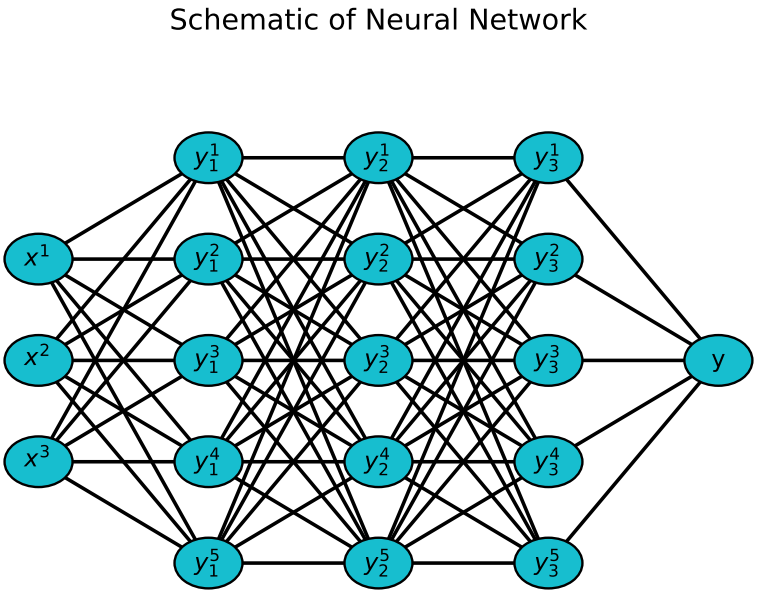

Figure 1.1: The structure of a typical neural network. In this example, we show a neural network used for energy prediction from 3D coordinates of a rigid body from [3]

The common guiding principle in statistics and machine learning, the Maximum Likelihood Principle, is a fundamental parameter estimation concept. It provides a principled way to determine the parameter values of a statistical model that best fits the observed data. In essence, the Maximum Likelihood Principle seeks to find the parameter values that maximize the likelihood of observing the given data under a statistical model. Suppose the likelihood function is denoted by $K(\theta, t_i, x_i)$ and depends on the model parameters, inputs to the model, and true values. In that case, we may find the optimal parameters θ by maximizing the likelihood of seeing true values for given inputs. We will continue to denote likelihood as K to avoid confusing it with a loss function L.

$$\hat{\theta} = \arg\max K(\theta) \tag{1.6}$$

Assume a Gaussian (normal) distribution for the target variable y given the input features X, and the model's predictions are denoted as $\hat{y}$. We aim to maximize the likelihood of observing the data (X, y). Since the data distribution is normal, we can express the likelihood by

$$K(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_i - \hat{y}_i(\theta))^2} \tag{1.7}$$

where n is the number of data points, σ is the standard deviation of the Gaussian distribution, and y_i and $\hat{y}_i$ are the observed and predicted values for the i-th data point, respectively.

We start by taking the logarithm since maximizing a function, and its logarithm is equivalent:

$$\ln K(\theta) = -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \hat{y}_i(\theta))^2. \tag{1.8}$$

Equivalently, we can minimize the negative log-likelihood:

$$-\ln L(\theta) = \frac{n}{2}\ln(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \hat{y}_i(\theta))^2. \tag{1.9}$$

This term is proportional to the Mean Squared Error (MSE) loss (1.5) since the first term is just a constant dependent on the properties of the data distribution. Hence, by maximizing the likelihood, we derive the MSE loss, which is commonly used in linear regression.

Entropy and cross-entropy

Before we continue with different possible loss functions, let us recall two important information theory fundamentals.

1. **Entropy**: The concept of entropy is defined in multiple areas of science. In physics, entropy represents the disorder of a physical system. It can be properly defined, has its physical unit (J/K), and is a fundamental quantity of thermodynamics. Similarly, in the context of information theory, entropy is a measure of the unpredictability or randomness of a system containing certain information. It can be measured in bits and quantifies the average amount of information (or uncertainty) in a random variable's possible outcomes. The Shannon entropy [24] $H(X)$ for a discrete random variable with probability distribution $P(x)$ is defined as:

$$H(X) = -\sum_{x} P(x)\log P(x) \tag{1.10}$$

Here, log is the logarithm base 2, and the sum is over all possible outcomes of the random variable X.

2. **Cross-Entropy**: Cross-entropy, also defined in [24] is used to measure the difference between two probability distributions, especially in machine learning for classification problems. For two discrete probability distributions, P (the true distribution) and Q (the predicted distribution), the cross-entropy of Q relative to P is:

$$H(P, Q) = -\sum_{x} P(x)\log Q(x) \tag{1.11}$$

This measures the average number of bits needed to identify an event from a set of possibilities using the predicted distribution Q instead of the true distribution P.

Cross-Entropy Loss

In classification problems, where the goal is to assign data points to discrete classes or categories, the Cross-Entropy Loss (also known as log loss) is a common

choice. It is derived using a probabilistic model predicting each data point's class probabilities. The likelihood function for a binary classification problem is:

$$K(\theta) = \prod_{i=1}^{n} (p_i^{y_i} \cdot (1 - p_i)^{1-y_i}), \tag{1.12}$$

where n is the number of data points, p_i is the predicted probability that the i-th data point belongs to class 1, and y_i is the actual binary label (0 or 1) for the i-th data point.

To find $\hat{\theta}$, we maximize the likelihood, or equivalently, as previously, its logarithm

$$\ln K(\theta) = \sum_{i=1}^{n} (y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)). \tag{1.13}$$

To derive the loss it is more convenient to not maximize but minimize the negative of previous expression:

$$-\ln K(\theta) = -\sum_{i=1}^{n} (y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)). \tag{1.14}$$

This term is proportional to the Cross-Entropy Loss:

$$\text{Cross-Entropy Loss} = -\frac{1}{n} \sum_{i=1}^{n} (y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)). \tag{1.15}$$

Thus, by maximizing the likelihood of correct classification, we derived the Cross-Entropy Loss, which is commonly used in logistic regression and other classification models. Upon a closer look, the cross-entropy loss is just a special case of a cross-entropy function, as a formula for cross-entropy, a measure of difference between two distributions, can be estimated from examples by

$$H(p, q) = -\sum_{i} p(i) \log(q(i)). \tag{1.16}$$

1.2.2 Backpropagation

Since the losses we just derived are differentiable, we may want to update model parameters θ until they reach optimal value by gradient descent. This is advantageous when a model itself can be differentiated with respect to its parameters and when the exact formula for the optimal θ cannot be efficiently used. Since we will be dealing with neural networks, let's explain how to get the gradient of the loss with respect to θ.

The mean squared error (MSE) loss for a single data point i is defined as:

$$\text{MSE}_i = \frac{1}{2}(y^i - \hat{y}^i)^2 \tag{1.17}$$

where y^i is the true target value, and $\hat{y}^i$ is the predicted value for data point i. The overall MSE loss for the entire dataset is the average of the individual MSE values:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \text{MSE}_i, \tag{1.18}$$

where N is the number of data points.

Backpropagation is used to efficiently compute the gradients of the loss with respect to the network's parameters ($\mathbf{W}_i$ and $\mathbf{b}_i$) such that we can then update the parameters in the direction opposite to the gradient of the loss and this way minimize the loss. The gradients are computed recursively from the output layer back to the input layer. The output layer is simple, with only a trivial chain rule

$$\frac{\partial \text{MSE}}{\partial y} = y - \hat{y} \tag{1.19}$$

$$\frac{\partial \text{MSE}}{\partial \mathbf{W}_4} = \frac{\partial \text{MSE}}{\partial y} \cdot \mathbf{y}_3^T \tag{1.20}$$

$$\frac{\partial \text{MSE}}{\partial \mathbf{b}_4} = \frac{\partial \text{MSE}}{\partial y} \tag{1.21}$$

Hidden Layer 3 (Layer 3)

$$\frac{\partial \text{MSE}}{\partial \mathbf{y}_3} = \frac{\partial \text{MSE}}{\partial y} \cdot \mathbf{W}_4 \tag{1.22}$$

$$\frac{\partial \text{MSE}}{\partial \mathbf{W}_3} = \frac{\partial \text{MSE}}{\partial \mathbf{y}_3} \cdot \mathbf{y}_2^T \tag{1.23}$$

$$\frac{\partial \text{MSE}}{\partial \mathbf{b}_3} = \frac{\partial \text{MSE}}{\partial \mathbf{y}_3} \tag{1.24}$$

Hidden Layer 2 (Layer 2)

Analogous to the third hidden layer. Evaluate $\frac{\partial \text{MSE}}{\partial \mathbf{W}_2}$ and then trivially use chain rule.

Input Layer (Layer 1)

Evaluate $\frac{\partial \text{MSE}}{\partial \mathbf{W}_2}$ using previous values and then obtain derivatives with respect to the weights in layer 1.

Since we now have the gradient with respect to the parameters, we can trivially use gradient descent. This process iteratively refines the parameters and minimizes the MSE loss.

1.2.3 Batched backpropagation

It is not practical to evaluate an average gradient of the loss for the entire dataset and do an update afterward. Instead, an approach called Stochastic Gradient Descent (SGD) works by splitting a large dataset into smaller chunks - mini-batches and then performing incremental updates along the loss gradient of the corresponding batch. This way, many smaller updates are done instead of one large update, contributing to more stochasticity and higher stability. Aggregating data into batches also introduces possible parallelization opportunities. Modern hardware, such as Graphical Processing Units and Tensor Processing Units, can greatly benefit from simple parallel operations such as matrix multiplications.

Many variations of this algorithm exist, such as the Adam algorithm (short for Adaptive Moment Estimation) [25] that works like SGD but utilizes momentum term to make the optimization smoother. We shall use Adam optimizer in this work. Although much time could be spent on the various techniques of deep learning, we refer an interested reader to book [14] and proceed to the actual application in Hamiltonian systems.

2. Hamiltonian systems

Hamiltonian equations stand as a significant area of interest in learning and representation theory. This stems from their remarkable ability to model many practical and complex systems efficiently. Mechanical systems, such as particles or rotating bodies, are described very intuitively using Hamiltonians and are often used in practical numerical implementations. Hamiltonian equations for control and learning of mechanical systems are, therefore, extensively studied in domains like robotics, where the effective representation of systems is of paramount importance. Quantum mechanics also leverages the Hamiltonian approach to describe and formulate theories, connecting seemingly unrelated and very different fields.

Moreover, the Hamiltonian equations find an important application in the field of molecular dynamics, a branch of simulations indispensable in studying the movement and behavior of atoms and molecules. Understanding them allows researchers to understand complex interactions on the smallest of scales. Even for such abstract objects, the motion follows from the scalar Hamiltonian, more specifically from its potential energy surface and initial conditions. In Chapter 3, we will also investigate Hamiltonian learning in this context.

In this chapter, we seek to improve on the fundamentals of Hamiltonian learning. We shall investigate how to represent the equation's building blocks and discuss some further constraints on the formulations. We note that we will use an Einstein summation convention unless explicitly written.

A canonical Hamiltonian equations are usually written in the form

$$\begin{aligned} \dot{\mathbf{q}} &= \frac{\partial H(\mathbf{q},\mathbf{p})}{\partial \mathbf{p}}, \\ \dot{\mathbf{p}} &= -\frac{\partial H(\mathbf{q},\mathbf{p})}{\partial \mathbf{q}}. \end{aligned} \tag{2.1}$$

in this work, we shall not restrict ourselves to this canonical form, but we will work with a more general formulation using Poisson brackets. More specifically, we shall follow the GENERIC formalism described, for example, in [26]. A Poisson bracket is a skew-symmetric bilinear algebra on the space $\mathcal{F}(\mathcal{M})$ of smooth functions on $\mathcal{M}$ given by

$$\{\bullet,\bullet\} : \mathcal{F}(\mathcal{M}) \times \mathcal{F}(\mathcal{M}) \to \mathcal{F}(\mathcal{M}). \tag{2.2}$$

One of the requirements is the Leibniz rule

$$\{F, HG\} = \{F, H\}G + H\{F, G\}, \tag{2.3}$$

while another important concept is the Jacobi identity

$$\{F, \{H, G\}\} + \{H, \{G, F\}\} + \{G, \{F, H\}\} = 0. \tag{2.4}$$

A manifold equipped with a Poisson bracket is called a Poisson manifold.

We can interpret Hamiltonian dynamics as an evolution on a Poisson manifold. An evolution of a quantity $\mathbf{x}$ can be written as

$$\dot{\mathbf{x}} = X_H(\mathbf{x}) = \{\mathbf{x}, H\}. \tag{2.5}$$

The above properties have desired physical consequences. Skew-symmetry trivially implies the conservation of energy

$$\dot{H} = \{H, H\} = 0, \tag{2.6}$$

while the independence on an additive constant to energy is secured by the Leibnitz rule. An object called a Poisson bivector can be obtained by calculating the quantity

$$L(dF, dH) := \{F, H\}. \tag{2.7}$$

Jacobi identity can be written, in a coordinate-free approach, as

$$\mathcal{L}_{\mathbf{X}_H} L = 0, \tag{2.8}$$

where $\mathcal{L}_{\mathbf{X}_H}$ is a Lie derivative along the Hamiltonian evolution [27]. The Lie derivative is a concept in differential geometry and tensor calculus, named after the mathematician Sophus Lie. It is a tool used to measure the change of a tensor field along the flow of another vector field. In simple terms, it helps in understanding how a quantity changes as you move along the curves defined by a vector field. The equation (2.8) guarantees not only energy but also a Poisson bivector is conserved along the evolution.

We can immediately illustrate the approach we just built on canonical Hamiltonian equations (2.1). We would consider a Poisson bracket in a form

$$\{F, H\} = L^{kl} \frac{\partial F}{\partial x^k} \frac{\partial H}{\partial x^l}, \tag{2.9}$$

and assuming a local coordinate system x^i where $\mathbf{x}$ represents the pair $(\mathbf{q}, \mathbf{p})$ get

$$\dot{x}^i = L^{ij} \frac{\partial H(\mathbf{x})}{\partial x^j} \tag{2.10}$$

One immediately sees that by choosing L as

$$L = \begin{pmatrix} 0 & \mathbf{1} \\ -\mathbf{1} & 0 \end{pmatrix} \tag{2.11}$$

Hamiltonian canonical equations follow

$$\begin{pmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{p}} \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{1} \\ -\mathbf{1} & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial H(\mathbf{q},\mathbf{p})}{\partial \mathbf{q}} \\ \frac{\partial H(\mathbf{q},\mathbf{p})}{\partial \mathbf{p}}. \end{pmatrix} \tag{2.12}$$

We will further need one more definition. A Casimir C of a Poisson bracket is a function such that the following holds

$$\{C, F\} = 0 \quad \forall F \tag{2.13}$$

One can then define a symplectic system as one with only constant Casimirs, while a non-symplectic system has a non-constant Casimir. While a definition via symplectic form is usually used, we will be working with this equivalent definition as it is sufficient for our purposes, since will also only consider finite dimensional systems in this thesis. The definition naturally introduces two types of systems: symplectic and non-symplectic. There has been considerable effort in efficient Hamiltonian learning in recent years, yet the vast majority of publications focused solely on the symplectic version of the equations. Notable works include, for example, [28, 29]. Learning the same equations but following different initial assumptions, namely action minimization, can be achieved by Lagrangian Neural Networks [30].

2.1 Non-symplectic systems

Much less attention has been given to the learning of non-symplectic systems despite their importance in physics. Poisson Neural Networks PNN [31] and our alternative Direct Poisson Neural Network DPNN [3] are, to our knowledge, the only attempt made to address this issue.

PNN work leverages the Darboux-Weinstein theorem [32] and seeks transformation to Darboux-Weinstein coordinates where the Poisson bivector is constant. DPNN paper, on the other hand, chooses to address the issue differently. The Poisson bivector and Hamiltonian are learned simultaneously. To ensure the system we are learning obeys all physical laws, the Jacobi identity is either implicitly conserved by the neural network design or imposed with an additional loss function. The problem of respecting and incorporating physical constraints into the learning process is a highly discussed matter in the field of representation and learning. In the case of our work DPNNs, we are dealing with the identity $\mathcal{L}_{\mathbf{X}_H} L = 0$ that appears to be simple and potentially easy to fulfill in computer simulations. However, the contrary is true in the general case. To understand, we need to transition to a coordinate system and express the Lie derivative. In local coordinates x^i, the identity becomes

$$J^{ijl} = L^{kl}\frac{\partial L^{ij}}{\partial x^k} + L^{ki}\frac{\partial L^{jl}}{\partial x^k} + L^{kj}\frac{\partial L^{li}}{\partial x^k} = 0. \tag{2.14}$$

This is a partial differential equation for Poisson bivector L. To solve it generally in high dimension (e.g., for a complicated multiparticle system), we need to implement an efficient PDE solver in high dimension. Since our L is represented by a neural network, it is straightforward to implement the equation solver by introducing the loss function $\left(J^{ijl}\right)^2$. To minimize the loss, or in ML language, to train the network to find the minimum with respect to the loss is then a way to solve the (2.14).

Another way to approach such constraints is to seek analytical solutions whenever possible. Luckily, the Jacobi identity has been investigated thoroughly in the past, and there are some interesting analytical solutions in 3D and partially in 4D as well [33, 34, 35]. Using the well-known matrix-vector isomorphism in 3D

$$\mathbf{L} = \begin{pmatrix} 0 & -J_z & J_y \\ J_z & 0 & -J_x \\ -J_y & J_x & 0 \end{pmatrix} \leftrightarrow \mathbf{J} = (J_x, J_y, J_z). \tag{2.15}$$

Jacobi identity becomes a 3D vector equation

$$\mathbf{J} \cdot (\nabla \times \mathbf{J}) = 0, \tag{2.16}$$

with a general solution in the form

$$\mathbf{J} = \frac{1}{\phi}\nabla C \tag{2.17}$$

where C is, interestingly, a Casimir mentioned above and ϕ is a scalar function. Thus, instead of learning the entire Posisson bivector, we can just learn two scalar functions in 3D, and the Jacobi identity constraint will be automatically satisfied.

What happens when we ignore the Jacobi identity altogether? Will the learning suffer? Will the learned system possess some inherent defects for certain applications? Can we sometimes omit the constraint entirely? Such a question is a topic of major discussion in the ML community.

2.2 Constrains and ML

Incorporating various levels of external knowledge in machine learning systems has been used to improve the process of learning and pattern recognition. A thesis is that methods tailored to a specific task tend to perform better as parts of the problem are encoded in the general structure of the algorithm. Another perspective is to look at constraints as part of a more significant concept called inductive biases.

Inductive biases collectively create a set of assumptions that the model uses to model the relationship between the data and its predictions. [36]. Examples may include a linear regression assuming an approximately linear relationship or a convolutional neural network leveraging the spatial structure of an image in its kernel design and general architecture [37].

There is an ongoing and active discussion about the amount and purpose of inductive inductive biases in ML tasks. While it might seem that the more information we correctly assume, the easier will be the task of parameter fitting, the contrary is often the case. Let us illustrate the dilemma in a simple example from physics.

We consider the task of learning the forces of an unknown system via a neural network approximator. We seek to find a function f such that it matches an unknown Hamiltonian gradient on a set of points $\mathbf{q}_i$. In other words, we are trying to get the best approximation

$$f(\mathbf{q}, \theta) \approx \nabla_{\mathbf{q}} H(\mathbf{q}). \tag{2.18}$$

We are, however, facing a dilemma. We can construct a network with the number of inputs equal to the number of outputs, which is equal to the dimension of the system. The other way would be to proceed in a more sophisticated way, design a scalar function $H'(\mathbf{q}, \theta)$, and after the inference, calculate its gradient with respect to $\mathbf{q}$ obtaining forces. The gradient calculation can be done efficiently using automatic differentiation. A physicist would likely prefer the second option as it guarantees the conservation of Hamiltonian when evolving the trajectory field in time (see equation (2.6)). However, is it, besides capturing conservative property, also learning more accurate forces, or is an extra gradient operation creating a complicated system that is more difficult to train? Luckily, we are able to compare in this case, as such a model problem has been investigated thoroughly in the domain of molecular dynamics with machine-learned Hamiltonians. Hamiltonians are routinely trained when their ground truth is based on quantum calculations because the inference from a machine learning model is much faster than the quantum calculation itself. Given enough training points, we can thus predict the result of a quantum calculation within an acceptable error. The paper [38] concludes, that directly learning forces is more data efficient than learning energy and proposes the constrain inclusion by other means, namely by error

correction. On the other hand, most of the currently benchmarked and used models, reported, e.g., here [39], use the learning of energies mainly because the issue of non-conservativeness is considered critical and outweighs the benefits of data efficiency. It is, therefore, not an easy task to judge the importance of a physical constraint in learning workflows, and the purpose of the calculation must be understood before making the decision.

2.3 Jacobi identity

Our work [3] discusses the constraints of Jacobi identity. How does it influence learning errors, and when is it worth including it? The three versions of Jacobi identity enforcement were selected.

- **(WJ)** Ignore Jacobi identity. Training $\mathbf{L}(\mathbf{x})$ and $H(\mathbf{x})$ *without* the Jacobi identity. The loss function consists only of the L_2 error measuring the discrepancy between the training steps and steps obtained by implicit midpoint rule (IMR) iterations with bivector $\mathbf{L}$ and Hamiltonian $H(\mathbf{x})$ encoded by the networks:

$$\mathcal{L}_{\mathrm{WJ}} = c_{\mathrm{mov}} \sum_n |((\mathbf{x}_{n+1})_{\mathrm{exact}} - \mathbf{x}_n) - ((\mathbf{x}_{n+1})_{\mathrm{NN}} - \mathbf{x}_n)|^2 \tag{2.19a}$$

  where n goes over all training steps. Each point $(\mathbf{x}_{n+1})_{\mathrm{NN}}$ is calculated from $\mathbf{x}_n$ by the IMR method (using the $\mathbf{L}$ and H that are being trained and encoded by the neural networks). Prefactor c_{mov} is used to scale the loss function to numerically advantageous values (typically we use $c_{\mathrm{mov}} = 10$).

- **(SJ)** Training $\mathbf{L}(\mathbf{x})$ and $H(\mathbf{x})$ with *soft* Jacobi identity, where the L_2-norm of the Jacobiator (2.14) is a part of the loss function. The loss function is thus

$$\mathcal{L}_{\mathrm{SJ}} = \mathcal{L}_{\mathrm{WJ}} + c_{\mathrm{Jac}} \sum_n \sum_{ijk} |J^{ijk}(\mathbf{x}_n)|^2, \tag{2.19b}$$

  where i, j, and k go over the dimension of the system (here 3, 4, or 6). Prefactor c_{Jac} is used to scale the loss function into better numerical values (typically, we use $c_{\mathrm{Jac}} = 10$).

- **(IJ)** Training $C(\mathbf{x})$ and $H(\mathbf{x})$ with *implicitly* valid Jacobi identity, based on the general solution of Jacobi identity in 3D (2.20). The loss function is the same as in the case of the WJ method, but this time Jacobi identity is valid automatically,

$$\mathcal{L}_{\mathrm{IJ}} = \mathcal{L}_{\mathrm{WJ}}. \tag{2.19c}$$

Schematically, the approaches are explained in 2.1 and 2.2. The **(IJ)** approach leverages the fact that the general solution of the Jacobi identity (2.16) in 3D is known. The general solution of Jacobi identity (2.16) is

$$\mathbf{J} = \frac{1}{\phi} \nabla C \tag{2.20}$$

for arbitrary functions ϕ and C, where C is a Casimir function and ϕ is usually called the Jacobi last multiplier. This way, if we parametrize functions H, ϕ,

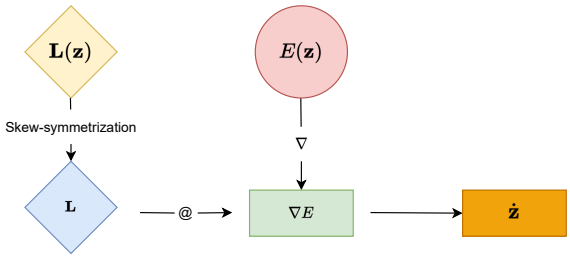

Figure 2.1: Scheme SJ (Soft Jacobi) of the methods that learn both the energy and Poisson bivector.

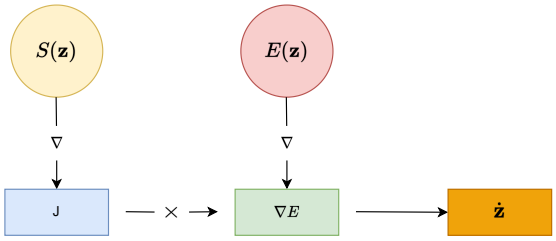

Figure 2.2: Scheme IJ (Implicit Jacobi) of the learning method implicitly enforcing Jacobi identity.

and C by neural networks, we can learn the system that is automatically Jacobi compatible.

The main examples areas where it was illustrated were a 2D particle in a harmonic potential, a rotating rigid body described by the angular momentum $\mathbf{m}$, a 3D particle in a harmonic potential, heavy top dynamics, and an interesting case of Shivamoggi equations.

1. **2D particle in harmonic potential:** This example is a case of a symplectic 4-dimensional physical system guided by a Hamiltonian function $H(\mathbf{q}, \mathbf{p}) = \mathbf{q}^2 + \mathbf{p}^2$. We can simulate and learn the system easily. The Jacobi identity was prescribed and investigated by implementing **WJ** and **SJ** approaches.

2. **Rigid body rotation:** An example of a 3D non-symplectic system. Has Casimirs. All three possibilities, **WJ**, **SJ**, and **IJ**, were used.

3. **Heavy top rotations:** Six dimensional non-sympectic system. **WJ** and **SJ** were used.

4. **3D particle in a harmonic potential** Same as 2D particle. Symplectic dynamics with **WJ** and **SJ** compared.

5. **Shiwamoggi equation** A system described in works of [40, 41]. 4D Hamiltonian equations with an interesting form of Jacobi identity. **WJ** and **SJ** used.

As non-symplectic systems might be unfamiliar to some readers, we shall introduce rigid body dynamics here in full detail. In the context of this problem, define a rigid body as a rotating object that is isolated from the outside world.

For example, a rotating satellite in a vacuum. The state variable is a three-dimensional vector $\mathbf{M}$. The Poisson bracket for this system is

$$\{F, H\}^{(RB)}(\mathbf{M}) = -\mathbf{M} \cdot \frac{\partial F}{\partial \mathbf{M}} \times \frac{\partial H}{\partial \mathbf{M}}, \tag{2.21}$$

Notice that any function of the magnitude of $\mathbf{M}$ is conserved by the bracket and is, therefore, a Casimir. Consider $\mathcal{F}(\mathbf{M}^2)$. By realizing that

$$\frac{\partial \mathcal{F}(\mathbf{M}^2)}{\partial \mathbf{M}_i} = 2\frac{\partial \mathcal{F}(\mathbf{M}^2)}{\partial \mathbf{M}^2}\frac{\mathbf{M}^2}{\partial \mathbf{M}_i} = 2\mathbf{M}_i\frac{\partial \mathcal{F}(\mathbf{M}^2)}{\partial \mathbf{M}^2} \tag{2.22}$$

We can arrive to

$$\dot{\mathcal{F}}(\mathbf{M}^2) = -\mathbf{M}_i\epsilon_{ijk}\frac{\partial \mathcal{F}}{\partial \mathbf{M}_j}\frac{\partial H}{\partial \mathbf{M}_k} = -2\mathbf{M}_i\mathbf{M}_j\epsilon_{ijk}\frac{\partial \mathcal{F}(\mathbf{M}^2)}{\partial \mathbf{M}^2}\frac{\partial H}{\partial \mathbf{M}_k} \tag{2.23}$$

which is an expression simultaneously symmetric and antisymmetric in i, j, therefore necessarily zero. This is initially a counterintuitive example as it analogously means that changing the energy by adding any function dependent on the magnitude of angular momentum does not change the dynamics. In other words, we can only determine the Hamiltonian or the energy of the system up to a function of $\mathbf{M}^2$. This is what a Casimir can look like in practice. The tensor of inertia, often denoted as II, is a symmetric 3×3 matrix for a rigid body in three-dimensional space. It's defined with respect to a chosen origin, usually the center of mass of the body or a fixed point in space. The elements of this tensor are given by integrals over the mass distribution of the body. Before we formulate the Hamiltonian, we need to introduce a tensor of inertia. The tensor of inertia is a matrix

$$\mathbf{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \tag{2.24}$$

With this form of inertia, we could write Hamiltonian as

$$H(\mathbf{M}) = \frac{1}{2}\mathbf{M}_i\mathbf{I}_{ij}\mathbf{M}_j \tag{2.25}$$

We can simplify the Hamiltonian by diagonalizing the tensor of inertia, meaning finding and working in a coordinate system in which the tensor becomes a diagonal matrix. In a diagonalized form, the tensor of inertia is written as:

$$\mathbf{I} = \begin{pmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{pmatrix}$$

Here, I_1, I_2, and I_3 are the principal moments of inertia, and they correspond to the eigenvalues of the original tensor of the inertia matrix. The axes corresponding to these principal moments are orthogonal and are the eigenvectors of the tensor. This simplifies the Hamiltonian to

$$H = \frac{1}{2}\left(\frac{M_x^2}{I_x} + \frac{M_y^2}{I_y} + \frac{M_z^2}{I_z}\right), \tag{2.26}$$

The full equations can now be written as

$$\dot{\mathbf{M}} = \mathbf{M} \times \frac{\partial H}{\partial \mathbf{M}} \tag{2.27}$$

We also showcase results on the rigid body system. We choose to do so because the system implements all three approaches to fulfill Jacobi's identity. We will compare two aspects of learning. How close is the predicted trajectory to the simulated ground truth, and how well is L matched? To quantify the match of Poisson bivector L, we will introduce a compatibility relation. From the original equation (2.10), it is clear that the multiplication of L by a constant K and simultaneous division of H by the same constant maintains the overall dynamics. To account for this, the compatibility condition for two Poisson bivectors

$$\mathbf{J}_1 \cdot (\nabla \times \mathbf{J}_2) = \mathbf{J}_2 \cdot (\nabla \times \mathbf{J}_1). \tag{2.28}$$

is used. This way, we can distinguish whether the $\mathbf{J}_1$ and $\mathbf{J}_2$ are the same, except for the constant factor, or whether they represent different Poisson bivectors corresponding to the different Hamiltonian functions. It is desired and useful to learn the original Hamiltonian function and $\mathbf{J}$ that was used to generate data rather than a combination of different functions that just happens to fit the data well. The known true $\mathbf{J}_2$ is obtained by the introduced matrix vector isomorphism and compared with neural network predicted $\mathbf{J}_1$ by minimizing loss

$$\mathcal{L}_{comp} = \left(\mathbf{J}_1 \cdot (\nabla \times \mathbf{J}_2) - \mathbf{J}_2 \cdot (\nabla \times \mathbf{J}_1)\right)^2. \tag{2.29}$$

The detail about the network training can be seen in 2.3 The trajectory error is similar for all three learning approaches (see figure 2.4. From the shape of distributions, it is hard to judge the best method in this case. Our paper [3] also calculated the average errors that were in favor of soft Jacobi implementation, with implicit Jacobi being the second best. However, the differences are marginal and could differ with, for example, different random seeds. A more important difference was when we investigated the error from the true Poisson bivector $\mathbf{J}_{true}$. The result in the figure 2.4 suggests that the **IJ** version learns evolution represented by L much closer to the real one. Following (SJ), we can arrive at the perhaps unexpected conclusion that the Jacobi identity does not significantly improve the error itself, yet the building blocks are constructed more in line with the theoretical framework. Finally, Figure 2.5 shows errors in learning the trajectories $\mathbf{M}(t)$. All three methods learn the trajectories well, but in this case, the SJ method works slightly better.

2.4 Distinguishing Dissipative Evolution

Since Jacobi identity does not decrease the overall error of a predicted trajectory, why implement it at all? When is a situation we care about, the building blocks of an evolution important? Is there a practical use besides theoretical investigation? An interesting use case of a developed framework arises when a dissipative system is considered. We implement the equations

$$\dot{\mathbf{M}} = \mathbf{M} \times E_{\mathbf{M}} - \frac{\tau}{2} \mathbf{\Xi} \cdot E_{\mathbf{M}} \tag{2.30}$$

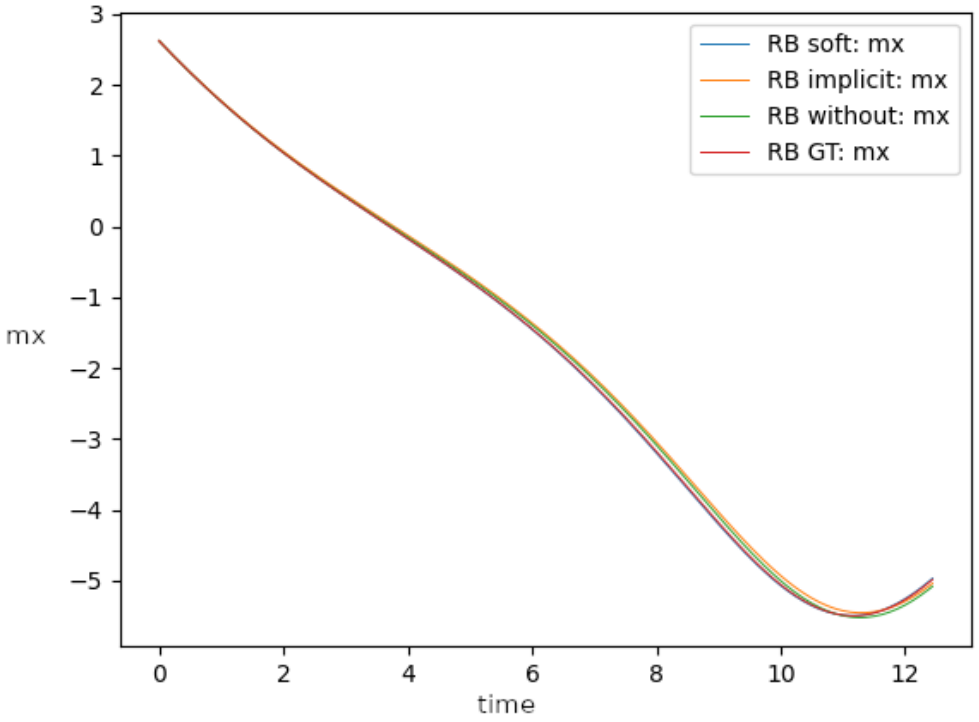

Figure 2.3: Comparison of an exact trajectory (GT) and trajectories obtained by integrating the learned models. All three methods fit the trajectories well. Here, as well in the subsequent figures, mx stands for the x-component of vector $\mathbf{m}$.

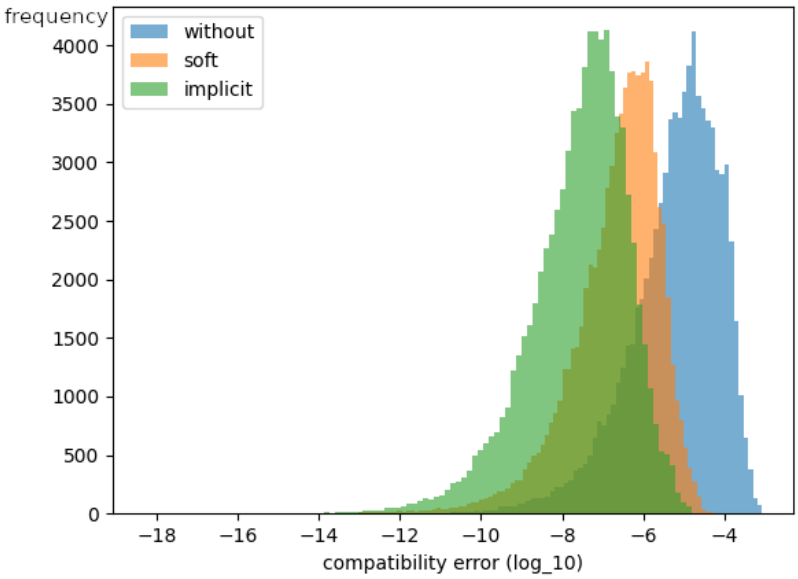

Figure 2.4: Rigid body: Compatibility errors for RB evaluated as $\log_{10}$ of the loss function 2.29. That equality is satisfied if and only if the two Poisson bivectors describe the same Hamiltonian system. The distribution of errors is approximately log-normal. The Compatibility error of the IJ method is the lowest, followed by SJ and WJ.

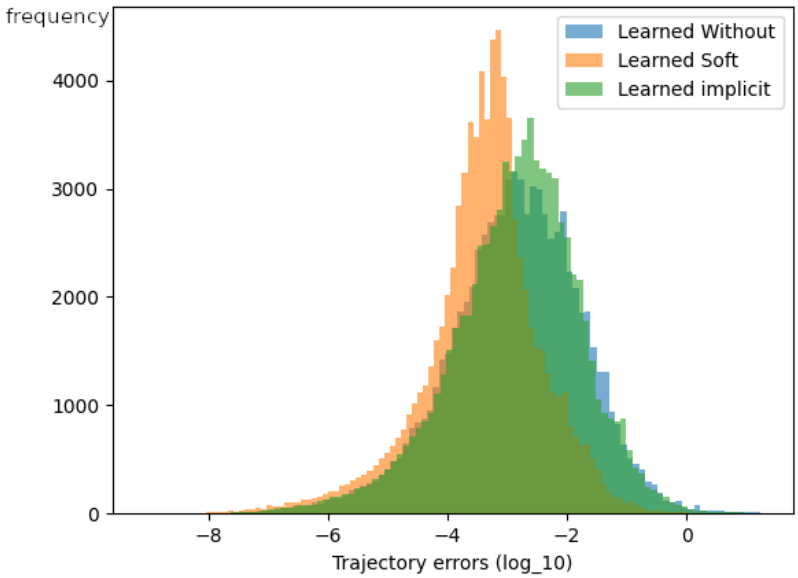

Figure 2.5: Rigid body: Distribution of $\log_{10}$ of squares of errors in $\mathbf{M}$.

where τ is a positive dissipation parameter and where $\boldsymbol{\Xi} = \mathbf{L}^T \cdot d^2E \cdot \mathbf{L}$ is a symmetric positive definite matrix (assuming that energy be positive definite) constructed from the Poisson bivector of the rigid body $L^{ij} = -\epsilon^{ijk}M_k$ and energy $E(\mathbf{M})$.

We now construct two sets of data. The first one is generated by non-dissipative dynamics, and the second by dissipative ones. Then, assuming we do not know anything about the dissipative term, how do we distinguish these two sets? In other words, how do we know a dissipative evolution based only on the data points and nothing else? We attempt to learn both evolutions by assuming they are non-dissipative, i.e., fitting (2.10) to the points and investigating the loss. The trajectory (loss in predicted $\mathbf{M}$ is in the figure 2.6. We can see that while **WJ** learned the evolution well in this case, probably by learning a vector field close to dissipative evolution, not being too constrained. The **SJ** struggles to capture the dynamics well while **IJ** straight-up fails. The constrained conservative learning is unable to find evolution close enough to the dissipative Hamiltonian field. Such a method inspired by machine learning of GENERIC with constraints can thus be used to effectively distinguish dissipative and purely reversible systems without any prior knowledge about the underlying building blocks in (2.10).

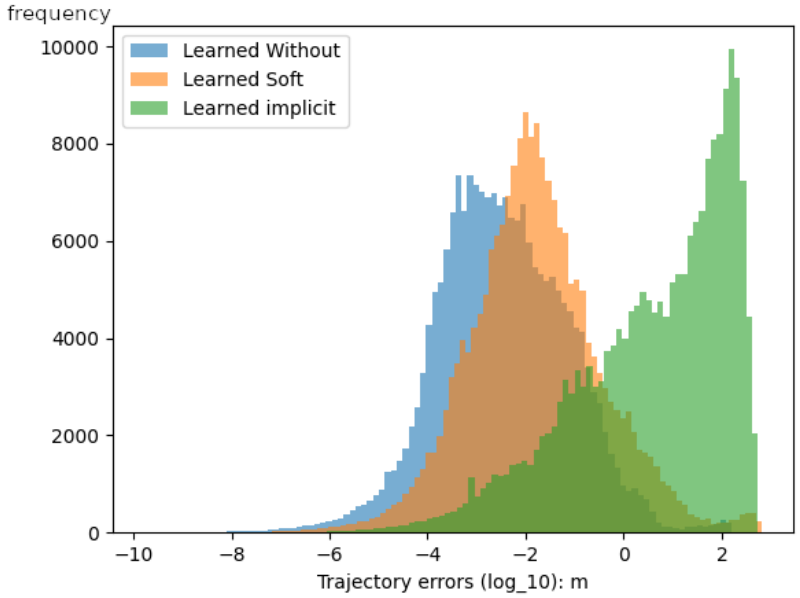

Figure 2.6: Dissipative rigid body dynamics: Distribution of $\log_{10}$ of squares of errors in $\mathbf{M}$. Learned by a non-dissipative model with different methods of enforcement of Jacobi identity.

3. Chemical reactions

Booming progress in chemistry accelerated by machine learning has been a driving force of innovation in recent years. From works that investigate drug candidates screening databases of billions of molecules [42] to machine-learned potentials that are extremely accurate and orders of magnitudes faster than methods used just a couple of years ago [43].

One of the promising areas of interest is the intersection of Hamiltonian simulation and chemical reaction exploration. By knowing the Hamiltonian function of an atomistic system, we possess all the necessary information needed to capture the evolution of the investigated system in silico [44]. Using simulations with the trajectories of atoms and molecules governed by the laws of motion derived from the Hamiltonian, scientists can analyze and predict the outcomes of chemical reactions, exploring the energy landscapes and identifying stable conformations and transition states [45].

The simulation of Hamiltonian evolution for molecular Hamiltonians is called Molecular Dynamics (MD). This temporal evolution provides insights into the thermodynamic properties, reaction kinetics, and mechanisms that underlie complex chemical processes. Moreover, it illuminates reaction barriers — critical indicators of reaction rates and pathways. By estimating these barriers, researchers can predict not only the feasibility of reactions but also their spontaneity and product yield under varying conditions.

The reaction barrier can be defined as a difference in energy a system needs to overcome to traverse between two metastable states. To intuitively define all the remaining necessary terms, a reactive event on a molecular level is a structural change from one energy minimum to another along a transition path that passes a saddle point we will call a transition state (TS). As a small change in transition state energy translates to orders of magnitude different probability of passing it, we consider only the most likely, lowest transition state energy path. For simple low-dimensional systems, such a barrier isn't hard to evaluate, and most likely transition path is not hard to find with tools like the string method [46] or Metadynamics [47]. Due to its small volume, it is possible to explore virtually every corner of the space. The problem becomes rather complicated with increasing dimensionality. Most of the systems we would like to understand in chemistry are high-dimensional since every new atom means more bonds, angles, and dihedral angles to consider. A small zeolite crystal can already have a couple hundred variables that need to be accounted for to consider the landscape and transition paths fully.

Naturally, therefore, it is beneficial to develop a dimensionality reduction technique that would project many degrees of freedom to only a few. This is a notoriously difficult yet important task in many areas across mathematics, physics, economy, and virtually every scientific discipline. The fundamental problem of focusing only on the important and ignoring noise is arguably the basis and starting point of every theory. The tools and methods we can use to address the issue vary from simple intuition-guided selection to sophisticated machine-learning methods that are improved with more and more data available on the subject.

3.1 Dimensionality reduction

Let us have a set of points $\mathbf{x} \in \mathcal{X}$ with dimensionality N. We seek to find a mapping ξ to a set of points $\mathbf{r} \in \mathcal{D}$ with dimensionality $n \ll N$ such that $r_i = \xi(x_i)$. The requirements on ξ can vary based on the broader context of the assignment we are addressing. A famous general and widely used example is Principal Component Analysis (PCA) [48]. This particular method constructs an orthogonal linear transformation that projects the data to a low-dimensional space such that the components capture as much variance in the data as possible.

3.1.1 Variational Autoencoder and reducing dimension

There are multiple variants of PCA that are capable of capturing non-linear relationships, such as kernel variants of PCA or manifold learning methods like Local Linear Embedding (LLE) [49]. Another widely used non-linear dimensionality reduction technique is called an autoencoder. In short, the autoencoder has two neural networks as building blocks. One of them is called an encoder $\mathbf{z} = E(\mathbf{x}, \phi)$ and the other a decoder $\mathbf{x} = D(\mathbf{z}, \theta)$. The encoder encodes from the original space where information lives in ($\mathcal{X}$) to the reduced ($\mathcal{D}$). The decoder then projects back to the original space $\mathcal{X}$. The main idea of this approach is that the compression of the data in the reduced space captures the most important and high-variance modes so that the decoder can then reconstruct as much information as possible. To achieve this objective, we introduce a reconstruction loss

$$\mathcal{L}_R = (\mathbf{x} - D(E(\mathbf{x}, \phi), \theta))^2 \tag{3.1}$$

A very popular extension of the Autoencoder, a Variational Autoencoder (VAE), modifies the approach and introduces stochastic noise. While traditional autoencoders learn a deterministic encoding function, VAEs, as introduced by Kingma and Welling [50], incorporate stochasticity by defining a probability distribution for the latent representation. This fundamental shift allows VAEs not only to perform dimensionality reduction but also to act as generative models.

In a VAE, the encoder network $E(\mathbf{x}, \phi)$ maps inputs to a distribution over the possible latent variables instead of a single point. This process is often represented by predicting not the values $\mathbf{z}$ directly but parameters of a Gaussian distribution, i.e., a mean $\boldsymbol{\mu}$ and a variance $\boldsymbol{\sigma}^2$, and then using a random number generator to obtain final $\mathbf{z}$. The latent variable $\mathbf{z}$ is then sampled from this distribution using a reparameterization technique that enables backpropagation through stochastic nodes. The reparametrization trick, as it is called, can be explained as follows. Consider a latent variable $\mathbf{z}$ which is drawn from a distribution $q_\phi(\mathbf{z}|\mathbf{x})$, where ϕ are the parameters of the distribution, and $\mathbf{x}$ is some input data. The reparametrization trick introduces an auxiliary variable $\boldsymbol{\epsilon}$ and rewrites $\mathbf{z}$ as a deterministic function of $\boldsymbol{\epsilon}$ and ϕ. For instance, if $\mathbf{z}$ is normally distributed with mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$, we can express $\mathbf{x}$ as:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon} \tag{3.2}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ (a standard normal distribution), and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are functions of ϕ. This reparametrization allows for the computation of gradients of

the loss function with respect to ϕ despite the stochastic nature of $\mathbf{z}$. Since the generated noise now simply acts as a constant during differentiation.

To properly introduce a loss function for the VAE, we must again employ the maximum likelihood principle. This time, however, it will be impossible to optimize the log-likelihood directly. The combination of two objectives, data reconstruction and a requirement for regular latent space distribution, requires the introduction of simplifications. To continue further, we shall remind the reader of an important inequality. Jensen's inequality states that for a convex function f, a random variable X, and any probability distribution, the following inequality holds:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \tag{3.3}$$

where $\mathbb{E}[X]$ is the expected value of X. If f is a concave function, the inequality is reversed:

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)] \tag{3.4}$$

3.1.2 Evidence lower bound - VAE loss

First, we will define a Kullback–Leibler divergence. Kullback–Leibler divergence is a measure of the difference between two distributions. It is not a metric as it is not symmetric and does not fulfill triangle inequality. However, it proved to be tremendously useful in generative modeling. Given two probability distributions P and Q, we define the KL divergence as

$$KL(P||Q) = \int P(x) \log \frac{P(x)}{Q(x)}. \tag{3.5}$$

Notice we are using a continuous version of the KL divergence. This is analogous to the discreet one, only on continuous data distributions. By manipulating the logarithm argument, we can arrive at a formulation

$$KL(P||Q) = \int P(x) \log P(x) - \int P(x) \log Q(x) = H(P,Q) - H(P) \tag{3.6}$$

Where $H(P)$ is the entropy defined in (1.10) and $H(P,Q)$ is the cross-entropy defined in (1.11). The motivation behind using KL divergence comes from the fact that for two same distributions, KL divergence vanishes, while cross-entropy does not, giving us something closer to a notion of distance. A property that is also useful is the non-negativeness, where we will use that $\log x \leq x - 1$, or rather a version of it multiplied by -1, $-\log x \geq 1 - x$

$$\begin{aligned} KL(P||Q) = \int P(x) \log \frac{P(x)}{Q(x)} = -\int P(x) \log \frac{Q(x)}{P(x)} \geq \\ \geq \int P(x) \left(1 - \frac{Q(x)}{P(x)}\right) = \int P(x) - \int Q(x) = 0 \end{aligned} \tag{3.7}$$

This also gives a so-called Gibs inequality

$$H(P,Q) \geq H(P) \; \forall Q \tag{3.8}$$

ELBO derivation

Consider data $\mathbf{x}$ and a corresponding set of latent variables $\mathbf{z}$. The goal of variational inference is to approximate the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ with a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$. The Evidence Lower Bound (ELBO) is derived as follows: We try to maximize the log-likelihood of the observed data

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \tag{3.9}$$

While we introduce variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$:

$$\log p_\theta(\mathbf{x}) = \log \int \frac{p_\theta(\mathbf{x}, \mathbf{z}) q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\, d\mathbf{z}. \tag{3.10}$$

In the next step, we apply the Jensen inequality for the concave function (the logarithm)

$$\log p_\theta(\mathbf{x}) \geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} \tag{3.11}$$

Then, after simple rearrangement,

$$\begin{aligned} \log p_\theta(\mathbf{x}) &\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} \\ &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z})\, d\mathbf{z} - \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{x}|\mathbf{z})}{p_\theta(\mathbf{z})}\, d\mathbf{z} \end{aligned} \tag{3.12}$$

we get

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - KL[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})] \tag{3.13}$$

Here, $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ is the expected log-likelihood under the variational distribution, and $KL[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})]$ is the Kullback-Leibler divergence between the variational distribution and the prior over the latent variables.

Practically, the first term can lead to the reconstruction loss (3.1). The second might seem unpractical, but if we choose the prior distribution $p_\theta(\mathbf{z})$ conveniently, we can analytically write the expression such that it is trivial to evaluate. More about particular example cases can be found in [50]. To illustrate one of them used in the following work, let us have a Gaussian prior of dimension J with $\boldsymbol{\mu} = \mathbf{0}$ (zero mean for all components) and $\boldsymbol{\sigma} = \mathbf{I}$ (identity matrix variance). In that case

$$-D_{KL}(q_\theta(\mathbf{z})||p_\phi(\mathbf{z})) = \frac{1}{2} \sum_{j=1}^{J} (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \tag{3.14}$$

The expression can be obtained easily by following the definition and integrating Gaussian distributions.

Intuitively, VAE keeps the reconstruction loss as in autoencoders but introduces an additional term, the Kullback-Leibler (KL) divergence
$\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$, which measures how much the learned latent variable distribution $q_\theta(\mathbf{z}|\mathbf{x})$ deviates from the prior $p_\theta(\mathbf{z})$. This regularization term ensures that the encodings are well-structured and interpretable in the latent space, leading to more controlled generation and better generalization. The overall objective function of a VAE, therefore, combines the reconstruction loss with the KL divergence term, balancing data fidelity with stochastic exploration. By optimizing

this objective, VAEs learn to compress data into a probabilistic latent space and, importantly, can generate new data by sampling from the prior $p_\theta(\mathbf{z})$ and decoding these samples, thus serving as a bridge between dimensionality reduction techniques and generative models.

3.2 VAEs as collective variable predictors

In molecular modeling, the reduced representation of a reaction is described by a set of so-called collective variables (CVs). These CVs serve as a guide to which regions of the chemical space to explore and focus on. A set of CVs should also describe the transition from reactants to products as precisely as possible.

For some reactions, the identification of CVs is relatively easy. If there is a particular bond breaking and the rest of the structure remains rigid, the bond length would likely constitute a reasonable choice for a CV. If all dihedral angles in a protein remain fixed except for a few, using these flexible dihedral angles likely describes the conformational change well. In many cases, however, obtaining the right CV poses a challenging task. In a complex crystal that overgoes a significant structural change, having the right CVs would mean carefully understanding the orchestration of events and encoding the logic in the projection constructed.

The idea of having machine-learned CVs has gained popularity in recent years. Since generating a large amount of data is not difficult, a clever machine learning tool that extracts insights could work well and automate the task. Recently, the task of identifying CVs has been partially automatized by multiple machine learning-based tools [51, 52, 53, 54, 55, 56]. The variational autoencoder approach has also been implemented in [57].

The paper [4] then implements approaches inspired by transfer learning [58]. The main idea is to build a variational autoencoder on atomic representations of a graph convolutional neural network (GCNN) [59, 60, 61] as inputs. The atomic representation in a GCNN is an intermediate neural network layer that stores a vector for every atom in the system. The vector should contain all the information necessary for the evaluation of energy and, after automatic differentiation, the evaluation of forces. It should respect all the invariances or equivariances of the architecture and is usually the last step before the energy is evaluated as a sum of atomic contributions. The reasons for such implementation can be attributed to

- Automatic invariance and possibly equivariance of collective variables with respect to rotation and translation. The invariance to the change in atom ordering can be addressed by the VAE design based on the specific problem.
- The paper demonstrates that the information extracted by pre-trained representations on forces and energies can significantly improve the ease of learning.
- When running the simulation with a GCNN as a neural network potential substituting the slow DFT-based approaches, we can get forces, energy, and collective variables in one pass, speeding up the workflow.

The GCNN used in the paper is obtained with permission from the authors of [62] and is compared with fixed (not pre-trained) Atom-centered symmetry functions

(ACSF) [63, 64]. The VAE is not used as a generative model. Instead, the decoder is only used in the training loop as a way to force CVs to be as expressive of the reaction as possible. The scheme of training and evaluation can be seen in Figure 3.1 while the evaluation scheme is in Figure 3.2. The paper illustrates

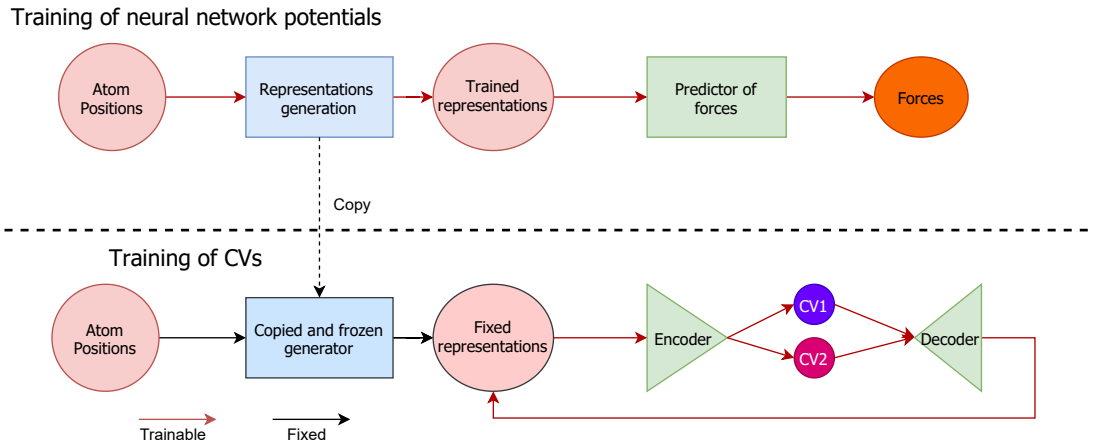

Figure 3.1: Training of our collective variables.

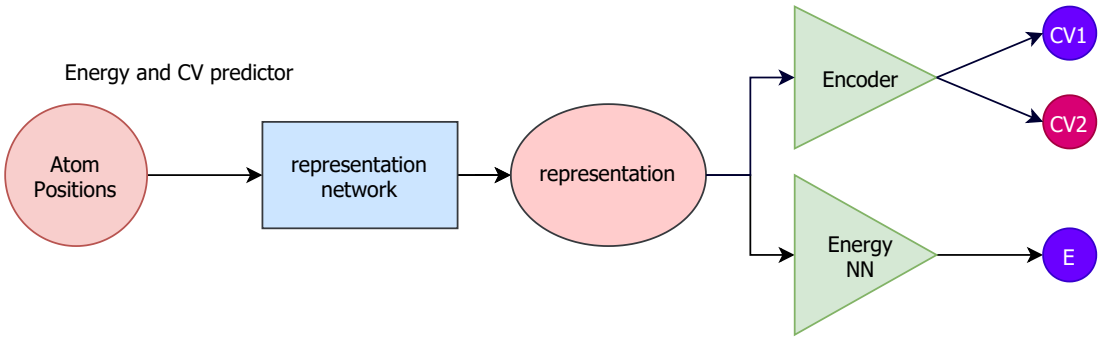

Figure 3.2: Single neural network for energy and collective variable prediction.

the approach of three reactions, ranging from a simple ket-enol transition to a complex formation in a zeolite crystal.

The approach proposed in the paper has several drawbacks. First and foremost, there is no objective function to evaluate the quality of the proposed CVs. The only way, and the approach presented in the paper, was to compare the training result with the expertly chosen collective variables. The lack of a more objective function describing the quality of a collective variable is described in the paper [5]. To understand the paper, let us first introduce differentiable simulations as a concept.

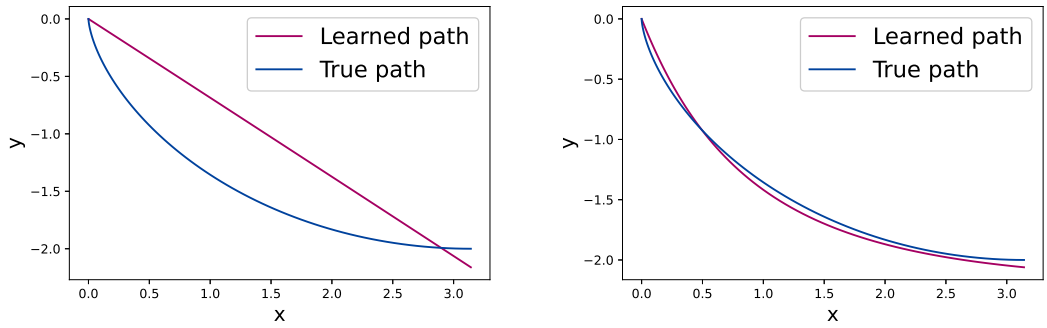

Figure 3.3: *left* Initial state. The path is initialized as an almost straight path. *right* After 200 iterations of differentiable simulations training, the path approximates the true path. The difference between the true curve and the one obtained by training is likely in the numerical scheme used to evaluate the integral.

3.3 Differentiable Simulations

Simulations that are fully differentiable (we will call them DiffSim) have been developed for optimization, control, and learning of motion, [65, 66, 67, 68] but also for the learning and optimization of quantities of interest in molecular dynamics [69, 70, 71]. Differentiating through simulations comes naturally from the optimization of path-dependent quantities. As a toy problem, the paper [5] works with the Brachystochrone example, demonstrating a differentiable simulation to solve it. If the minimization of a loss function cannot be formulated separately for every point in the path, then optimization has to include the whole path leading up to it. While the results of DiffSims are often promising, it is well known [72] that naïvely backpropagated gradients may vanish or explode, thus not leading to a useful parameter update. How to control their behavior remains an open challenge. This problem of differentiable simulations is associated with the spectrum of the system's Jacobian [72, 73] and is closely connected to the chaotic nature of the simulated equations. Therefore, in order to employ path differentiation, one needs to find ways to produce well-behaved and controllable gradients.

The example problem we will be investigating is defined as such: Consider a mass sliding without friction along a curve $y = y(x)$ under the influence of gravity g. The objective is to determine the curve connecting point **A** to a lower point **B**, where the sliding duration is minimized. It is assumed that there's no friction or air resistance and that **B** is not vertically aligned under **A**. For easy analysis, **A** is set as the coordinate system's origin. The cycloid curve is the known solution to this, discovered by Leibniz, L'Hospital, Newton, and the Bernoulli brothers, as noted by [74]. There's also an approach where only the vertical displacement Δx is specified, solved by Lagrange, and later detailed in the contemporary language of variational formalism by [75]. Here, the solution is also a cycloid curve with certain fixed parameters. We set the horizontal Δx as π and seek a solution through differentiable simulations. A basic fully connected neural network $f(x)$ is used as the curve's derivative $f(x) = \frac{dy(x)}{dx}$, allowing us to derive $y(x)$ by path integration of the neural network. Subsequently, the time for the simulated path

is calculated numerically.

$$t = \int_0^l \frac{ds}{v(x)} = \int_0^\pi \sqrt{\frac{1 + \left(\frac{dy(x)}{dx}\right)^2}{-2gy(x)}} dx \tag{3.15}$$

and minimized. In the first integral, l represents the length of the curve. The formula comes from the conservation of kinetic energy and from a Pythagorean expression $ds^2 = dx^2 + dy^2$. For the path construction and backpropagation, we employ the *torchdiffeq* python package shipped with the paper [76]. The results for a short training are in Fig. 3.3.

3.4 Enahnced Sampling of Rare Events Using DiffSim

The main idea of the paper [5] is to define an objective function describing the quality of the biased dynamics. The reaction frequency occurring in finite time is selected and further modified to form a differentiable function. The final form of loss function describes the minimal distance of any point in a trajectory starting in the metastable state A to the metastable state B. The precise definition and formulation can be understood from the paper. Since the loss function is only defined in a single point where the distance is minimized, we leverage the DiffSim approach to optimize the path leading to such a point. The method itself is an iterative procedure and seeks to control the simulation by adding an additive term $B(\mathbf{x}, \theta)$ to the original potential function, such that the potential becomes

$$U(\mathbf{x}, \theta) = U_0(\mathbf{x}) + B(\mathbf{x}, \theta). \tag{3.16}$$

The procedure is stopped once the sampling becomes more even and the transition from metastable state A to B and back becomes sufficiently common. The properties of a reaction are then inferred from the form of $B(\mathbf{x}, \theta)$. The unique property of such a DiffSim tool is that it unites the collective variable selection and biasing step, creating a powerful end-to-end approach to reaction biasing. The results seen on a simple 2D toy potential are in Figure 3.4. The paper also investigates a simple two amino acid model of a protein. The alanine dipeptide serves as a standard benchmark for enhanced sampling methods and is, therefore, very well studied with ideal CVs known and usable as reference. The comparison of our method and commonly used benchmark is in the 3.5. We can see good agreement with the reference. The main difference and advantage of our method is that it did not require previous knowledge of the collective variables for the biased dynamics, only for postprocessing and comparison with the methods used. The results of transition path and reaction barrier magnitude are extensively employed in many workflows in material science, biochemistry, and computational drug discovery. Predicting a material property of the ability of a novel drug candidate to bind to the target is all hidden in the knowledge of reaction dynamics. The methods presented in this chapter promise to automatize the workflow and introduce machine learning as a tool that helps with laborious collective variable identification.

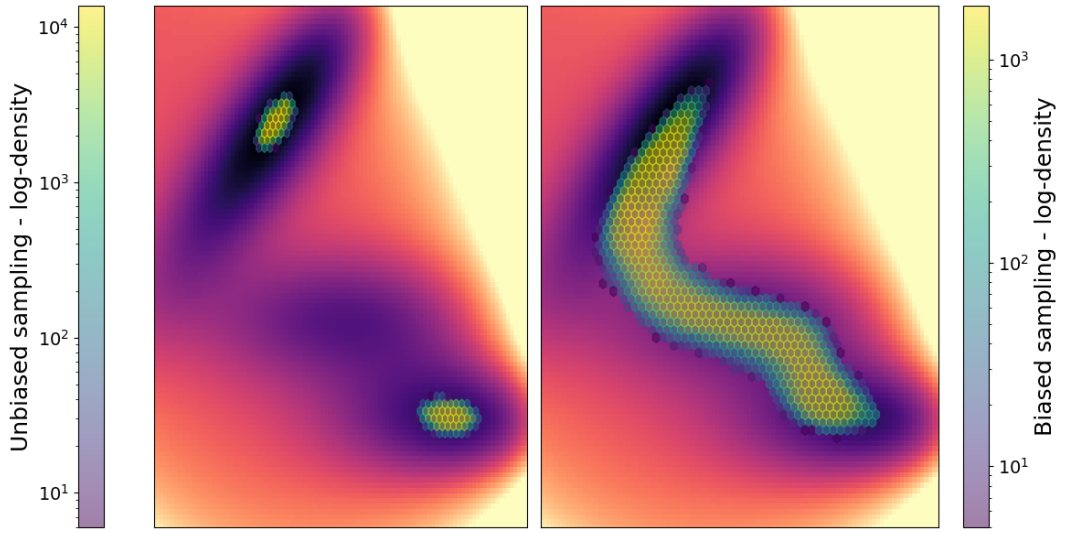

Figure 3.4: Log-density of simulated points before (*left*) and after the training (*right*) of bias function by differentiable simulations. The right plot shows how well all important regions are sampled after training. The background of the Figure is the $U_{\mathrm{MB}}(x, y)$, the underlying Muller-Brown potential described in the Appendix of [5]

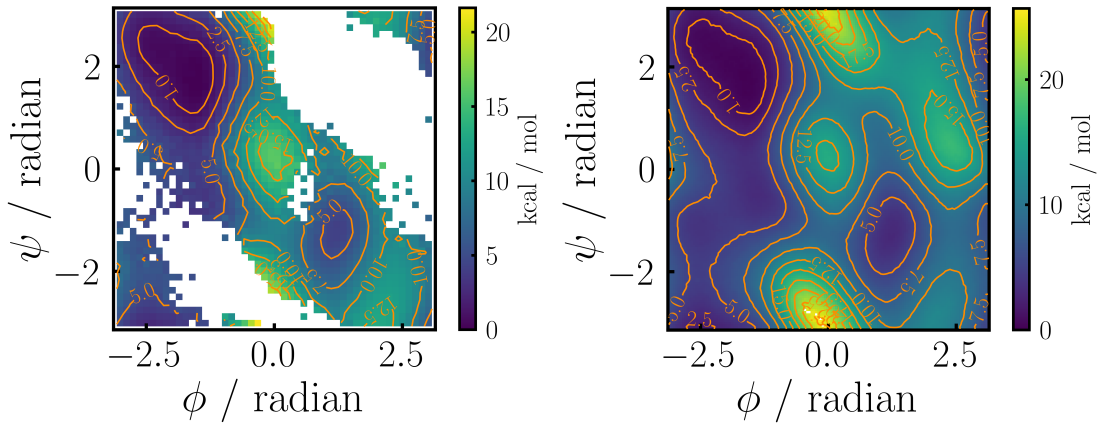

Figure 3.5: *left:* Directly reweighted PMF from simulations with the bias potential projected on the Ramachandran plane. White regions are without sufficient sampling. *right:* PMF of the ϕ-ψ-plane obtained from an eABF simulation.

Conclusion

Throughout this dissertation, we explored the synergy between geometry, Hamiltonian mechanics, rare event sampling, and applications to physics and chemistry. In all of the discussed papers written during the course of the PhD studies, we employed the Hamiltonian formalism and consciously considered the underlying physics. We introduce new approaches that connect physics and machine learning, investigate their weaknesses and strengths, and find multiple interesting directions for future research that are worth exploring.

Firstly, we explored a novel way of learning Hamiltonian equations by employing neural networks in the process. We studied different systems and various scenarios on how to enforce the Jacobi identity. We proposed a new way to identify dissipative dynamics without assuming anything more than a GENERIC structure of an evolution. The methods and findings are proposed to be used in the emerging field of model-based control, where insights and correct representation strategies are necessary. Our contribution focuses on the representation learning for the non-symplectic systems and the ability to detect dissipation with almost no prior knowledge about the system.

We contributed to the automated collective variable search efforts, which are crucial in simplifying the high-dimensional complexity inherent in chemical reactions. The application of variational autoencoders in combination with the use of a pre-trained graph convolutional model promises a new approach in this context. By integrating graph convolutional neural network (GCNN) representations, we were able to impose essential physical constraints and exploit pre-trained knowledge, leading to more robust and interpretable models. The proposed method was then subsequently employed in further work that aimed to connect developed ML potentials and ML-guided CV search to create an efficient synergized framework.

Furthermore, we pushed the frontiers of differentiable simulations, highlighting their potential in optimizing path-dependent outcomes and explaining phenomena in molecular dynamics. We first illustrate the phenomenon of differentiable simulations on simple and well-known problems in physics and then apply it to a more specific domain. While acknowledging the challenges, particularly related to gradient behaviors and the chaotic nature of molecular systems, we underscored the importance of pursuing more controlled and stable methods in these simulations. The proposed modifications to the framework address the gradient issue and push the boundaries of what is known on the path to solving this complex problem.

Despite these promising advancements, several challenges and open questions remain. The objective evaluation of CVs, which is critical for validating the effectiveness of machine learning models in capturing chemical reaction nuances, still lacks a standardized methodology. This gap signifies a crucial area for future research, where developing metrics or benchmarks could profoundly enhance model comparison and drive the field toward more reliable and interpretable solutions.

Moreover, while differentiable simulations hold immense promise, their current limitations necessitate a focused research effort toward stabilizing gradient behaviors and ensuring that these methods can consistently contribute valuable

insights into molecular dynamics studies.

In conclusion, this work serves as a testament to the transformative power of interdisciplinary approaches, melding machine learning's computational approach with the knowledge required to fully understand geometry and intricate chemical systems. As we constantly see new, exciting progress in this area, the potential for discovery and innovation attracts more talent and resources, promising to unravel the fundamental mysteries too complex for humans alone. With the world needing breakthrough technologies more than ever before to overcome mounting challenges, progress in the realms of material science, biotechnology, and robotics is a necessity rather than a luxury.

Bibliography

[1] A. Einstein. Die grundlage der allgemeinen relativitätstheorie. *Annalen der Physik*, 49, 1916. In German.

[2] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[3] M. Šípka, M. Pavelka, O. Esen, and M. Grmela. Direct poisson neural networks: Learning non-symplectic mechanical systems. *Journal of Physics A: Mathematical and Theoretical*, 56(49), 2023.

[4] M. Šípka, A. Erlebach, and L. Grajciar. Understanding chemical reactions via variational autoencoder and atomic representations. *arXiv*, (2202.00817), 3 2022.

[5] M. Šípka, J. C. B. Dietschreit, L. Grajciar, and R. Gómez-Bombarelli. Differentiable simulations for enhanced sampling of rare events. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[6] T. M. Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[7] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[8] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[9] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[10] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[11] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[12] L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–215, 2001.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[15] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad. Machine learning at the energy and intensity frontiers of particle physics. *Nature*, 560(7716):41–48, 2018.

[16] J. Carrasquilla and R. G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, May 2017.

[17] Christopher J. S. and Andrew M. V. Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. *The Astronomical Journal*, 155, 2017.

[18] M. H. S. Segler, M. Preuss, and M. P. Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, Mar 2018.

[19] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and Ch. Kim. Machine learning in materials informatics: recent applications and prospects. *npj Computational Materials*, 3(1):54, Dec 2017.

[20] J. Kirkpatrick, B. McMorrow, D. H. P. Turban, A. L. Gaunt, J. S. Spencer, A. G. D. G. Matthews, A. Obika, L. Thiry, M. Fortunato, D. Pfau, L. R. Castellanos, S. Petersen, A. W. R. Nelson, P. Kohli, P. Mori-Sánchez, D. Hassabis, and A. J. Cohen. Pushing the frontiers of density functionals by solving the fractional electron problem. *Science*, 374(6573):1385–1389, 2021.

[21] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[22] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2924–2932, Cambridge, MA, USA, 2014. MIT Press.

[23] M. Telgarsk. Neural networks and rational functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3387–3393. PMLR, 06–11 Aug 2017.

[24] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] M. Pavelka, V. Klika, and M. Grmela. *Multiscale Thermo-Dynamics*. de Gruyter, Berlin, 2018.

[27] M. Fecko. *Differential Geometry and Lie Groups for Physicists*. Cambridge University Press, 2006.

[28] E. Dierkes and K. Flaßkamp. Learning Hamiltonian Systems considering System Symmetries in Neural Networks. *IFAC-PapersOnLine*, 54(19):210–216, 2021.

[29] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In H Wallach, H Larochelle, A Beygelzimer, F d Alché-Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[30] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. 2020.

[31] P. Jin, Z. Zhang, I. G. Kevrekidis, and G. E. Karniadakis. Learning poisson systems and trajectories of autonomous systems via poisson neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2022.

[32] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Texts in Applied Mathematics. Springer New York, 2013.

[33] B. Hernández-Bermejo. New solutions of the Jacobi equations for three-dimensional Poisson structures. *J. Math. Phys.*, 42(10):4984–4996, 2001.

[34] B. Hernández-Bermejo. One solution of the 3D Jacobi identities allows determining an infinity of them. *Phys. Lett. A*, 287(5-6):371–378, 2001.

[35] B. Hernández-Bermejo. New solution family of the Jacobi equations; characterization, invariants, and global Darboux analysis. *J. Math. Phys.*, 48(2):022903, 11, 2007.

[36] T. M. Mitchell. The need for biases in learning generalizations. Technical report, Rutgers University, New Brunswick, NJ, 1980.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[38] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K. Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017.

[39] X. Fu, Z. Wu, W. Wang, T. Xie, S. Keten, R. Gomez-Bombarelli, and T. S. Jaakkola. Forces are not enough: Benchmark and critical evaluation for machine learning force fields with molecular simulations. *Transactions on Machine Learning Research*, 2023. Survey Certification.

[40] P. Guha and A. G. Choudhury. First integrals and Hamiltonian structure for a system of ordinary differential equations occurring in magnetohydrodynamics. *AIP Conference Proceedings*, 1582(1):116–123, 02 2014.

[41] B. K. Shivamoggi. Current-sheet formation near a hyperbolic magnetic neutral line in the presence of a plasma flow with a uniform shear-strain rate: An exact solution. *Physics Letters A*, 258(2):131–134, 1999.

[42] G. Schneider. Automating drug discovery. *Nature Reviews Drug Discovery*, 17(2):97–113, 2018.

[43] A Erlebach, M. Šípka, I. Saha, P. Nachtigall, Ch. J. Heard, and L. Grajciar. A reactive neural network framework for water-loaded acidic zeolites, 2023.

[44] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.

[45] M. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. Oxford Graduate Texts. OUP Oxford, 2010.

[46] E. Weinan, W. Ren, and E. Vanden-Eijnden. String method for the study of rare events. *Physical Review B*, 66(5):052301, 2002.

[47] A. Barducci, G. Bussi, and M. Parrinello. Well-Tempered Metadynamics: A Smoothly Converging and Tunable Free-Energy Method. *Physical Review Letters*, 100(2):020603, 1 2008.

[48] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[49] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[50] D. P Kingma and M. Welling. Auto-encoding variational bayes. *ArXiv*, 2014.

[51] M. M. Sultan and V. S. Pande. Automated design of collective variables using supervised machine learning. *The Journal of Chemical Physics*, 149(9):94106, 2018.

[52] D. Mendels, G. Piccini, and M. Parrinello. Collective Variables from Local Fluctuations. *Journal of Physical Chemistry Letters*, 9(11):2776–2781, 6 2018.

[53] Ch. Wehmeyer and F. Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, 148(24):241703, 2018.

[54] Y. Wang, J. M. L. Ribeiro, and P. Tiwary. Past–future information bottleneck for sampling molecular reaction coordinate simultaneously with thermodynamics and kinetics. *Nature Communications 2019 10:1*, 10(1):1–8, 8 2019.

[55] L. Bonati, V. Rizzi, and M. Parrinello. Data-Driven Collective Variables for Enhanced Sampling. *The journal of physical chemistry letters*, 11(8):2998–3004, 4 2020.

[56] L. Sun, J. Vandermause, S. Batzner, Y. Xie, D. Clark, W. Chen, and B. Kozinsky. Multitask Machine Learning of Collective Variables for Enhanced Sampling of Rare Events. *Journal of Chemical Theory and Computation*, 18(4):2341–2353, 2022.

[57] J. M. L. Ribeiro, P. Bravo, Y. Wang, and P. Tiwary. Reweighted autoencoded variational Bayes for enhanced sampling (RAVE). *The Journal of Chemical Physics*, 149(7):072301, 05 2018.

[58] B. Neyshabur, H. Sedghi, and Ch. Zhang. What is being transferred in transfer learning?, 2021.

[59] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, NIPS'15, page 2224–2232, 2015.

[60] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen. A graph-convolutional neural network model for the prediction of chemical reactivity. *Chem. Sci.*, 10:370–377, 2019.

[61] A. Kensert, R. Bouwmeester, K. Efthymiadis, P. Van Broeck, G. Desmet, and D. Cabooter. Graph convolutional networks for improved prediction and interpretability of chromatographic retention data. *Anal. Chem.*, 93(47):15633–15641, 2021.

[62] I. Saha, A. Erlebach, P. Nachtigall, Ch. J. Heard, and L. Grajciar. Reactive neural network potential for aluminosilicate zeolites and water: Quantifying the effect of si/al ratio on proton solvation and water diffusion in h-fau. *ChemRxiv*, 2022.

[63] J. Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.*, 134(7):074106, 2011.

[64] J. Behler and M. Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.

[65] J. Degrave, M. Hermans, J. Dambre, and F. Wyffels. A Differentiable Physics Engine for Deep Learning in Robotics. *arXiv*, (1611.01652), 11 2016.

[66] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-End Differentiable Physics for Learning and Control. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[67] Y. Hu, T. Li, L. Anderson, J. Ragan-Kelley, and F. Durand. Taichi. *ACM Transactions on Graphics*, 38(6):1–16, 12 2019.

[68] Y. Hu, L. Anderson, T. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. DiffTaichi: Differentiable Programming for Physical Simulation. In *International Conference on Learning Representations*, 2020.

[69] W. Wang, S. Axelrod, and R. Gómez-Bombarelli. Differentiable Molecular Simulations for Control and Learning. *arXiv*, (2003.00868), 2 2020.

[70] J. Ingraham, A. Riesselman, Ch. Sander, and D. Marks. Learning Protein Structure with a Differentiable Simulator. In *International Conference on Learning Representations*, 2019.

[71] J. G. Greener and D. T. Jones. Differentiable molecular simulation can learn all the parameters in a coarse-grained force field for proteins. *PLOS ONE*, 16(9):e0256990, 9 2021.

[72] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman. Gradients are Not All You Need. *arXiv*, (2111.05803), 11 2021.

[73] C. L. Galimberti, L. Furieri, L. Xu, and G. Ferrari-Trecate. Hamiltonian Deep Neural Networks Guaranteeing Non-vanishing Gradients by Design. *arXiv*, (2105.13205), 5 2021.

[74] C. Boyer and Merzbach U. *A History of Mathematics, Second Edition*. Wiley, 2 edition, 3 1991.

[75] S. Mertens and S. Mingramm. Brachistochrones with loose ends. *European Journal of Physics*, 29(6):1191–1199, 11 2008.

[76] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 2018-December, 2018.