PhD-FSTM-2023-109
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 25/10/2023 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN PHYSIQUE

by

Grégory Cordeiro Fonseca
Born on 30 May 1993 in Luxembourg

MACHINE LEARNING FORCE FIELDS UNDER THE MICROSCOPE: STABILITY, RELIABILITY AND PERFORMANCE ANALYSIS

UNIVERSITY OF LUXEMBOURG

Abstract

Faculty of Science, Technology and Medicine

Department of Physics and Materials Science

Doctor in Physics

Machine Learning Force Fields Under the Microscope: Stability, Reliability and Performance Analysis

by Cordeiro Fonseca Gregory

Machine Learning Force Fields (MLFF) are a crucial tool for bringing the accuracy of computationally expensive quantum mechanical calculations to practically feasible applications on molecules and materials. Over time, the sophistication of MLFF architectures has increased to match the complexity of systems of ever-growing sizes. This increase in complexity comes with a higher need for in-depth analytical tools to properly assess a Machine Learning (ML) model's quality. Even the most advanced models that showcase remarkably low overall prediction errors demonstrate highly heterogeneous predictive capabilities across the Configurational Space (CS) of a single system. In practice, these can significantly impact the reliability of a model as a high prediction error on a few key geometries can easily lead to e.g. the destabilisation of a molecular dynamics simulation.

In this work, we provide a cross-platform software package designed to give a detailed view into the performance and shortcomings of an MLFF model, complete with an easy-to-use graphical user interface. Entitled FFAST (Force Field Analysis Software and Tools), this actively developed software enables any user to gauge the quality of many state-of-the-art ML architectures and infer potential pitfalls in practical applications. Analytical tools are provided at any desired level of resolution, from average error metrics over entire datasets to assessments of prediction accuracies on an atom-by-atom basis.

To provide an optimal compromise between detailed analysis and simplicity, a novel approach is developed to determine a model's predictive capabilities across different regions of CS. This is achieved by employing methods from unsupervised learning to create clusters of qualitatively different configurations and calculate their respective prediction accuracies separately. This provides much-needed context to otherwise general error metrics and captures insightful details of the model's capacity to reproduce e.g. important out-of-equilibrium mechanisms rarely frequented in the reference dataset. Furthermore, inhomogeneous error curves across clusters also provide information on which regions are likely poorly represented in a model's training set.

The potential of the aforementioned methods as well as FFAST are showcased on example datasets of stachyose and docosahexaenoic acid (DHA) as well as a handful of smaller organic molecules. After successfully proceeding through a typical FFAST workflow with two state-of-the-art ML models (Nequip and MACE), it was quickly determined that carbons and oxygens near glycosidic bonds of the stachyose molecule have increased prediction errors. Furthermore, prediction errors on DHA rise as the molecule folds, with notably low accuracy on the carboxylic group at the edge of the molecule. Finally, the cluster prediction errors of a handful of small organic molecules were generated for three different ML models (sGDML, SchNet and GAP/SOAP). The latter showed that in all cases, prediction accuracies are highly heterogeneous across CS, hinting at a host of potential problems in real-world applications, where model stability is paramount.

Motivated by the newfound heterogeneities, an iterative training process is proposed that actively seeks out configuration clusters poorly represented in the training set. Assisted by clustering techniques, the training set is gradually extended until it equally reflects all relevant parts of CS of the reference dataset. It is found that models trained this way have enhanced stability in molecular dynamics, with performance comparable to significantly increasing the total number of training points. Furthermore, they demonstrate an up to two-fold decrease in the root mean squared errors for force predictions in the problematic regions of CS.

Finally, future avenues for research are briefly discussed in light of this work's findings. This includes the suggestion of a "divide and conquer" approach to subdivide the task of learning CS into smaller building blocks, providing a potentially reliable way to ensure reliability across all possible states of the system. As the complexity and sizes of the systems tackled by MLFFs have grown over recent years, a subdivision of the learning process into more manageable building blocks is of wide appeal.

Publications

Part of the thesis is based on the following publications:

1. Gregory Fonseca, Igor Poltavsky, and Alexandre Tkatchenko. Force Field Analysis Software and Tools (FFAST): Assessing Machine Learning Force Fields under the Microscope. 2023. arXiv: 2308.06871 [physics.chem-ph]

2. Gregory Fonseca, Igor Poltavsky, Valentin Vassilev-Galindo, and Alexandre Tkatchenko. "Improving Molecular Force Fields across Configurational Space by Combining Supervised and Unsupervised Machine Learning". In: J. Chem. Phys. 154.12 (Mar. 2021), p. 124102. ISSN: 0021-9606. DOI: 10.1063/5.0035530

3. Valentin Vassilev-Galindo, Gregory Fonseca, Igor Poltavsky, and Alexandre Tkatchenko. "Challenges for Machine Learning Force Fields in Reproducing Potential Energy Surfaces of Flexible Molecules". In: J. Chem. Phys. 154.9 (Mar. 2021), p. 094119. ISSN: 0021-9606. DOI: 10.1063/5.0038516

Contents

List of Abbreviations

BIGDML	**B**ravais-**I**nspired **G**radient **D**omain **M**achine **L**earning
BPNN	**B**ehler-**P**arrinello **N**eural **N**etwork
CCSD(T)	**C**oupled **C**luster **S**ingle-**D**ouble-(**T**riple)
CS	**C**onfigurational **S**pace
DaC	**D**ivide **a**nd **C**onquer
DFT	**D**ensity **F**unctional **T**heory
DHA	**D**ocosa**h**exaenoic **A**cid
DNN	**D**eep **N**eural **N**etwork
DTNN	**D**eep **T**ensor **N**eural **N**etwork
FF	**F**orce **F**ield
FFAST	**F**orce **F**ield **A**nalysis **S**oftware and **T**ools
GAP	**G**aussian **A**pproximation **P**otential
GDML	**G**radient **D**omain **M**achine **L**earning
GNN	**G**raph **N**eural **N**etwork
GP	**G**aussian **P**rocess
HF	**H**artree **F**ock
KRR	**K**ernel **R**idge **R**egression
LDA	**L**ocal-**D**ensity **A**pproximation
MAE	**M**ean **A**verage **E**rror
MBD	**M**any **B**ody **D**ispersion
MBTR	**M**any-**B**ody **T**ensor **R**epresentation
MD	**M**olecular **D**ynamics
ML	**M**achine **L**earning

MLFF	**M**achine **L**earning **F**orce **F**ield
MLP	**M**ulti-**L**ayered **P**erceptron
MSE	**M**ean **S**quared **E**rror
NED	**N**udged **E**lastic **B**and
NequIP	**N**eural **equ**ivariant **I**nteratomic **P**otentials
NN	**N**eural **N**etwork
PBE	**P**erdew–**B**urke-**E**rnzerhof
PES	**P**otential **E**nergy **S**urface
RMSE	**R**oot **M**ean **S**quared **E**rror
sGDML	**s**ymmetric **G**radient **D**omain **M**achine **L**earning
SOAP	**S**mooth **O**verlap of **A**tomic **P**ositions
TS	**T**katchenko-**S**cheffler (dispersion interaction)
vdW	**v**an **d**er **W**aals

Chapter 1

Introduction

All matter, ranging from molecules to crystals and solids to gases, are different manifestations of atoms interacting under the same physical laws at the microscopic scale. A complete understanding of these interactions and the means to replicate them *in silico* allow us to predict how particles move in tandem to compose matter with various structures and properties. This unlocks groundbreaking advancements in material design, novel pharmaceutics, energy solutions such as the development of new batteries, and more [1–14]. As such, atomistic simulations have developed into an integral part of scientific discovery.

Replicating the movement of atoms *in silico* at a nanoscopic scale requires accurately calculating the potential energy associated with every possible configurational state of the system(s) under study. The derivative of the potential energy surface (PES) gives rise to a force field (FF) that dictates the direction from every atom towards an energetically favorable state. The latter can then be used to run molecular dynamics and observe conformational changes, chemical reactions, protein folding, ligand binding, et cetera.

Any molecular or condensed system can be fully described by the electronic Schrödinger equation and have — in theory — its exact PES computed. In practice, exact solutions to Schrödinger's equation can only be found for the simplest of systems, thus approximations are needed for any real application. For instance, the universally applied Born-Oppenheimer approximation stipulates that the motion of electrons and nuclei can be treated separately due to electrons being much lighter and thus moving much faster. Many additional steps and approximations are necessary to find numerical solutions to said equations (especially for the electronic part), the development of which falls into the vast field of electronic structure theory. The methods that come from works in this field, dubbed *ab initio*, are the most accurate way to calculate a PES (and the associated force field). However, this comes at a significant computational cost with unfavorable scaling laws as the number of electrons in the system grows. Density Functional Theory (DFT) [15, 16] is among the cheapest *ab initio* methods and scales as $O(n^3)$ while Coupled Cluster Single-Double-(Triple) (CCSD(T)) [17, 18], widely regarded as the "golden standard" in terms of accuracy, suffers from $O(n^7)$ scaling, where n roughly corresponds to the size of the system.

Many applications of interest require large systems or long simulation times, such that using *ab initio* methods becomes unfeasible. The development of classical force fields has been one of the main alternatives providing more computationally efficient albeit much less reliable FFs. Here, the electrons are only implicitly considered in the potential energy, which consists of analytical terms modeling a molecule's (or any other system's) stretching and bending, change in torsion angles, pairwise van der Waals and typically non-polarizable electrostatic interactions. Each term is given as a predefined function whose parameters are fitted to experiments or reference *ab initio* calculations.

The main focus of this work is a second option to bypass the efficiency problems of *ab initio* methods: Machine Learning Force Fields (MLFF) [19–46]. Here, statistical models are trained on substantial amounts of *ab initio* data to accurately reproduce highly accurate energy and force calculations at a fraction of the cost. The early developments of MLFF were largely jumpstarted by the Behler-Parrinello Neural Networks (NN) in 2007 [30]. They use an invariant description of every atom and its environment and subsequently feed it to a NN of a single hidden layer to calculate atomic contributions to the total potential energy.

The initial target systems were simple and small, such as silicon crystals and water molecules. Over time, increasingly complex systems were tackled through systematic advancements in the MLFF architectures. For example, new descriptors were developed, such as the Smooth Overlap of Atomic Positions (SOAP) [47], Atom-centered symmetry functions [48] and Many-Body Tensor Representations (MBTR) [49]. Alternatively, kernel methods, such as the Gaussian Approximation Potential (GAP) [50] and Gradient Domain Machine Learning (GDML) [51], were proposed alongside NNs. In 2017, SchNet [40, 52, 53] presented itself as a novel NN architecture with defining features that are still present in many modern MLFFs [54]. Rather than relying on a predefined descriptor, SchNet iteratively learns the system representation through several interaction layers that update atomic embeddings in relation to their environment. This feature, coupled with the pairwise nature of the interaction layers, allows SchNet models to be highly accurate and transferable to different molecules.

Nowadays, many state-of-the-art MLFFs have adopted message-passing Graph Neural Network (GNN) [55, 56] approaches as a successor to SchNet's convolutional interaction layers. Beyond this, current research directions include the implementation of equivariant features [57–61] or the explicit inclusion of physical interactions in the network to better account for e.g. atoms that are far away from each other [62, 63]. All in all, the improvements made in the last decade have resulted in the ability of MLFF to reproduce the interactions in medium and even large-sized molecules and complex materials, raising atomistic simulations to a qualitatively new level.

Increasing the size and complexity of the systems available for MLFF comes with additional challenges. One of them is the models' quality measures. When systems are small, simple overall error metrics such as mean average errors (MAE) or root mean square errors (RMSE) are sufficient to determine a model's accuracy. The same is not true for highly flexible molecules (or similarly complex systems) with diverse states in Configurational Space (CS). MLFF stability and reliability are not guaranteed from small MAE and RMSE values, partly because single atoms or configurations have no tangible influence on overall metrics. Generally, performant ML models can easily lead to unphysical behaviors in practice when faced with out-of-equilibrium structures, rare events, or outliers in prediction accuracy profiles. This is especially relevant when using MLFFs to run dynamics for extended time scales: a high accuracy over a series of configurations is only relevant if a single, poorly predicted configuration in the past has not led the entire dynamics off course. One can thus compare an MLFF trajectory to the utilization of a self-driving car, where the car's ability to follow the road and speed limitations perfectly is no consolation if it fails to recognize even a single stop sign along the way.

Adoption of MLFFs at large hinges on models being trustable to adequately replace *ab initio* calculations in practical applications. Thus, the tools that provide the insight necessary to estimate MLFF reliability and stability must be as developed as the models themselves. However, the established ways to find flaws in ML models' predictions are limited, with most practices revolving around simple overall metrics such as MAE and RMSE on energies and forces. In this work, we delve deeper into how to properly assess a model's quality using various methods. Some tools created for this purpose serve as a foundation for future development in the field of analysis of MLFFs, mainly in the form of a novel interactive software dubbed FFAST (Force Field Analysis Software and Tools) [64]. The latter combines many of this work's elements into a user-friendly interface that allows any user to delve deep into the performance of their MLFFs or FFs of any kind.

The manuscript has the following structure. In chapter 2, the fundamental steps involved in creating an MLFF are introduced. This includes the basics behind energy calculations in electronic structure theory, a general overview of the different options to generate reference data for the training of MLFFs, a summary of the development of MLFFs over recent years, and a few examples of common FF analysis methods. Chapter 3 introduces the software package FFAST and serves as a way to show how different types of prediction errors can manifest when analyzing MLFFs in finer detail. Furthermore, it also suggests potential ways to improve the models in question in the future. This section also introduces the usage of clustering algorithms to analyze prediction errors across different CS regions. Chapter 4 offers further details on the theory behind clustering as well as a practical application on a handful of datasets of small organic molecules. It shows that prediction errors for different types of configurations can drastically vary for the same prediction model, sometimes leading to unreliable out-of-equilibrium behavior. As a consequence of this finding, chapter 5 suggests a way to counteract the inhomogeneous error distribution across clusters in CS through an iterative training set selection scheme. This leads to models with a "flattened" error curve that show better prediction errors for rare events and more reliable behavior in extrapolation regions. Finally, chapter 6 offers a perspective and suggestions for future work, followed by a summary of this work in chapter 7.

Chapter 2

Life cycle of a Machine Learning Force Field

2.1 Overview

A working MLFF model results from multiple interconnected steps such as data generation, architecture adjusting, training, verification, and practical applications to solve actual problems. Understanding the entire life cycle of an MLFF model is essential to ensure that its quality, reliability, accuracy, etc., is up to the required standards. It is useful to view the components involved in creating an MLFF — or any ML model – as a cycle rather than a linear succession of steps, as depicted in Figure 2.1. For MLFFs, we roughly identify four major ingredients: the choice of the suitable *ab initio* method to perform reference calculations, the choice of the sampling method to find representative data points across the CS of the system under study, the choice of the regression model (MLFF architecture in other words) to train on the data, and finally the analysis of the resulting model to determine its features, pitfalls and potential areas of improvement. The insights gained from analyzing the resulting model can lead to modification in previous steps to improve the model performance in the next iteration.

The choice of the *ab initio* method to perform reference calculations depends on many factors. A trivial consideration is that of computational costs, as there are

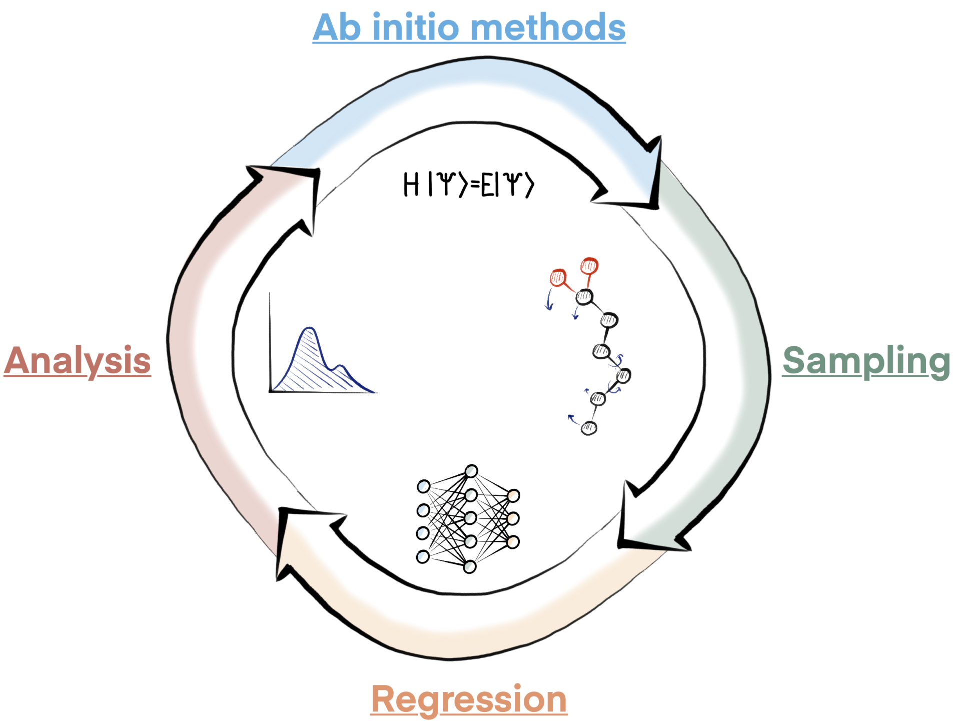

FIGURE 2.1: Illustration of the life cycle of a MLFF. The cycle includes roughly four components, the choice of *ab initio* methods, the choice of sampling methods, the choice of regression algorithm and finally the analysis. The last step enables identifying shortcomings in all previous steps to make informed changes in the next iteration of the cycle.

significant differences in efficiency between different approaches. Beyond this, it is vital to consider the physical and chemical phenomena essential for the particular system at hand to make the right choice for the level of theory. For instance, dispersion interactions are crucial for layered materials, as they govern the weak interlayer binding and stacking and thus determine the material's structural stability and properties. As such, explicitly including van der Waals interactions in the reference calculations is paramount for such systems. Another example is the employment of multireference calculations when excited electronic states are of interest.

Any MLFF can only learn the interactions reproduced by the reference *ab initio* method and included in the training data. Whenever the training data are missing some essential physics due to the wrong choice of the reference method, even the most advanced MLFF architectures are doomed to fail in practical applications.

Finding the right compromise between the required level of theory and acceptable efficiency in generating the training data is often a challenge. Multiple solutions have been proposed, including employing computationally cheap but less accurate computational chemistry methods for the initial sampling followed by highly accurate *ab initio* calculations for a selected subset of points, or iterative training of MLFFs using active learning techniques. Ultimately, the goal is to bring the high accuracy of reference quantum chemistry methods with minimal computational costs.

Sampling methods aim to generate the geometries that will be fed to the ML model during the training procedure. The best choices largely depend on the intended applications. The purpose of MLFFs is often to act as surrogate models in molecular dynamics to reduce computational cost and thus drastically increase simulation times or system sizes. In this case, an appropriate possibility of reference data generation is trajectories obtained by running a (short) molecular dynamics with an *ab initio* method of choice. Analysis of the obtained trajectory and/or the model trained on it can reveal flaws in the original data generation scheme. For example, the model might fail to accurately reproduce a rare event, such as an important reaction occurring in the long-time simulation. Another typical issue arises when the system has multiple disconnected local minima on its PES, but not all were sampled within the reference MD run. Multiple ways to combat these flaws have been developed. In the case of rare events, one can increase the prevalence of the particular reaction in the dataset by, e.g., sampling additional data points along its pathway, using methods such as Nudged Elastic Band or Umbrella Sampling [65–67]. For complex PESs, one can first employ a conformer search and then generate short MD trajectories starting from all relevant local PES minima.

Once the reference dataset is generated, one needs to decide on a regression model capable of learning the PES as well as the forces of the given system. Once again, the system in question can majorly affect the "best" choice. Here, the size, chemical composition, structural complexity, and importance of particular interatomic interactions are all significant. For instance, while state-of-the-art ML models based on message-passing algorithms demonstrate superior accuracy, efficiency, and scalability for modeling a wide range of molecules and solids, they might fail to describe an adsorption process whenever the surface-adsorbate distance exceeds the cutoff radius (an intrinsic model parameter). Conversely, global ML models — capable of reproducing all possible interactions without cutoffs — have limits in target system size due to computational limitations. For other applications such as periodic systems, the ML model needs to be capable of taking the particular translational symmetries into account.

Thus, the detailed analysis of an MLFF performance is a necessary element to enable an informed reconsideration and improvement of choices made in all previous ML model life cycle steps. The changes can be small, such as increasing the number of reference data in the training set or tuning the settings of the *ab initio* method and regression model. On the other hand, detailed analysis can reveal systematic errors requiring considerable modifications in the training methodology. This might motivate a switch to a different level of *ab initio* method, a revision of the ML architecture, or even a completely new direction for development. All in all, detailed analyses of MLFF performance lead to lasting advancements that expand the state-of-the-art and stimulate further research in the field of atomistic simulations.

The following subsections explore the current methods for each life-cycle step in more detail. As the main focus of this work, the development of regression models is given an additional spotlight in section 2.4.

2.2 Ab initio methods

One can distinguish two major branches in *ab initio* methods: (post) HF methods and DFT. The former describes electron interactions through a mean-field which is at the source of its main drawback: the inability to capture electron correlation, i.e. the interplay between moving electrons due to their Coulomb repulsion. This limitation is generally attenuated by additions such as **MP2** or **Coupled Cluster**, which recover some of the electron correlation through perturbation theory or additional excitation operators respectively. A rough overview of the method and its additions is described in section 2.2.1.

DFT instead rewrites Schrödinger's equation in terms of electron density rather than electron wavefunctions. This reduces complexity and while it includes electron correlation in a so-called exchange-correlation functional, its exact form is unknown and needs to be approximated. Approaches include expansions of the electron density and gradient (e.g. BLYP [68] and PBE [69]) as well as hybrid functionals that partially reinstate the HF exchange (e.g. B3LYP [68], PBE0 [70]). Furthermore, it is often necessary to add an additional functional to explicitly account for dispersion forces (e.g. vdW-DF [71, 72], D3/D4 [73, 74],TS [75], MBD [76]). DFT and some of the ways to deal with the exchange-correlation functional are briefly discussed in section 2.2.2.

Practically, many of these approaches are implemented and distributed in various software packages, such as Gaussian [77], VASP [78–80], Quantum Espresso [81, 82], ORCA [83–85], CP2K [86] and FHI-AIMS [87].

2.2.1 Hartree-Fock and Post Hartree-Fock

The complete Hamiltonian for an arbitrary system of nuclei and electrons can be written as a sum of kinetic terms T and Coulomb potential terms U for nuclei n and electrons e respectively:

$$\begin{aligned} H =& T_n(R) + T_e(r) + U_{en}(r,R) + U_{ee}(r) + U_{nn}(R) && (2.1)\\ =& -\sum_i \frac{\hbar^2}{2M_i}\nabla^2_{R_i} - \sum_i \frac{\hbar^2}{2m_e}\nabla^2_{r_i} - \sum_i\sum_j \frac{Z_i e^2}{4\pi\epsilon_0|R_i - r_j|} \\ &+ \frac{1}{2}\sum_i\sum_{j\neq i}\frac{e^2}{4\pi\epsilon_0|r_i - r_j|} + \frac{1}{2}\sum_i\sum_{j\neq i}\frac{Z_i Z_j e^2}{4\pi\epsilon_0|R_i - R_j|}, && (2.2)\end{aligned}$$

where M_i is the mass of nucleus i, Z_i is its atomic number, R_i its position, r_j the position of electron j and m_e its mass. In all applications, the Born-Oppenheimer approximation is assumed: that is, the movement of nuclei is much slower than that of electrons and so the electronic Hamiltonian can be solved while considering the position of every nucleus as a fixed parameter. As such, the fundamental equation to solve is:

$$E\Psi(r;R) = (T_e(r) + U_{en}(r;R) + U_{ee}(r) + U_{nn}(R))\Psi(r;R). \quad (2.3)$$

In the HF theory, the proposed ansatz for the wavefunction Ψ is a Slater determinant, which results in a linear combination of spin orbitals χ that satisfies the indistinguishability of electrons and the anti-symmetry of the wavefunction.

$$\Psi = \frac{1}{\sqrt{N!}}\begin{vmatrix} \chi_1(x_1) & \chi_2(x_1) & \cdots & \chi_N(x_1) \\ \chi_1(x_2) & \chi_2(x_2) & \cdots & \chi_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(x_N) & \chi_2(x_N) & \cdots & \chi_N(x_N) \end{vmatrix}, \quad (2.4)$$

where N is the total number of electrons in the system. Every spin-orbital χ depends on a single independent electron, and as such, writing the wave function in this form is equivalent to considering the Coulomb repulsion on every electron with respect to the average position of all others. This is why the HF theory is often referred to as a mean-field theory. In practice, these spin orbitals are approximated as a finite set of fixed functions in order to be computationally manageable, usually a linear combination of Gaussians. The equations are solved using the variational principle: since the ground state wavefunction is the one with the lowest energy, the latter can be varied (in this case, the coefficients of the linear combination of fixed functions) to minimize the energy and thus solve the system.

The **Hartree-Fock (HF)** method serves as a foundation for many other *ab initio* methods but can also be used on its own as a first-order approximation to evaluate the energies and forces of a molecular system. Due to its mean-field approach to describing electrons, the method cannot capture electron correlation, i.e. the interaction between moving electrons avoiding each other due to their Coulomb repulsion. This limitation makes standard HF rarely useful for actual applications, as the contribution of the electron correlation is too large to be ignored.

Finding ways to bypass this limitation within the same framework yields a variety of approaches to tackle the electron correlation problem, collectively called *post Hartree-Fock* methods. Those add a way to explicitly include electron correlation effects. The most popular examples of such approaches are the following:

- **Configuration Interaction** expresses the wavefunction of the system as a linear combination of multiple electronic configurations as opposed to the single determinant treated in the HF method, see Equation 2.5. Every term in the sum Φ_i^a is a Slater determinant with spin-orbital i excited to orbital a and is weighted by a coefficient c_i^a. The more configurations are included in the sum, the larger the proportion of the correlated electrons' motion is captured. However, the computational cost associated with this sum renders this method usable for small systems only.

$$\Psi = c_0\Phi_0 + \sum c_i^a \Phi_i^a + \sum c_{ij}^{ab}\Phi_{ij}^{ab} + \dots . \tag{2.5}$$

- **MPN**: In the Møller-Plesset Perturbation Theory, the electron correlation effects are taken into account via a perturbation added to the Hamiltonian of the system. The perturbation is the difference between the exact Hamiltonian and the HF Hamiltonian. The first term of the expansion is always zero; hence, the expansion is most commonly stopped at order two (**MP2**) as the best compromise between accuracy and efficiency.

- **Coupled Cluster** considers the true wavefunction to be the HF wavefunction Φ_0 (usually given by a Slater determinant) preceded by an exponential operator, as shown in Equation 2.6. This is similar to configuration interaction, with the exponent a sum of "cluster operators" T_i that correspond to excitations of either single electrons, double excitations, triples, etc... One of the most popular compromises is **CCSD(T)**, which includes single (S) and double (D) excitations and treats triple excitations perturbatively ((T)). This is often used as the golden standard and reference value for other methods to strive towards.

$$\Psi = e^{T_1+T_2+\cdots}\Phi_0. \tag{2.6}$$

2.2.2 Density Functional Theory

Aside from the approaches based on HF, another way to solve Schrödinger's equation is by rewriting the Hamiltonian as a functional of the electron density rather than the electron wavefunction. The core equation to solve is the same Hamiltonian under the Born-Oppenheimer approximation as introduced in Equation 2.3. However, it is rewritten to make the electron density $\rho(\mathbf{r})$ the key variable:

$$\rho(\mathbf{r}) = N \int d\mathbf{r}_2 ... \int \mathbf{r}_N \Psi^*(\mathbf{r}, \mathbf{r}_2, ..., \mathbf{r}_N)\Psi(\mathbf{r}, \mathbf{r}_2, ..., \mathbf{r}_N) \tag{2.7}$$

That is the core of DFT, tempering the complexity of the problem by reducing the dimensionality of the object of interest (3 rather than $3N$, where N is the number of electrons in the system). The equation above is — in principle — fully reversible in that once the ground state density is found, the corresponding ground state wavefunction can be determined. Thus, the expectation value of any observable can equally be calculated using this formalism as with a wavefunction-based description.

Most applications of DFT further alter the expression of the problem using the Kohn-Sham equations. There, a fictitious system of *non-interacting* electrons is introduced in such a way that it generates the same density as the full system of interacting electrons. The non-interacting electrons are subject to an external potential called the Kohn-Sham potential, which for molecular systems is at least the electron-nucleus Coulomb interaction. Note that similarly to HF methods, the Kohn-Sham wavefunction is assumed to be a Slater determinant of a set of orbitals χ, see Equation 2.4. Under this paradigm, the total energy can be expressed as:

$$E[\rho] = T_s[\rho] + E_{en}[\rho] + J[\rho] + E_{xc}[\rho], \tag{2.8}$$

$$T_s[\rho] = \sum_{i=1}^{N} \int d\mathbf{r}\chi_i^*(\mathbf{r})(-\frac{\hbar^2}{2m_e}\nabla^2)\chi_i(\mathbf{r}), \tag{2.9}$$

$$E_{en}[\rho] = \sum_i e^2 \int \frac{Z_i\rho(\mathbf{r})}{|R_i - \mathbf{r}|}d\mathbf{r}, \tag{2.10}$$

$$J[\rho] = \frac{e^2}{2}\int\int \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}d\mathbf{r}', \tag{2.11}$$

where e is the charge of an electron and m_e its mass, Z_i is the atomic number of nucleur i, ρ is the electron density of the system, T_s is the Kohn-Sham kinetic energy, E_{en} is the electron interaction with the external potential from the nuclei, $J[\rho]$ is the Coulomb energy of the electrons and E_{xc} is the exchange-correlation potential.

The latter accounts for the quantum mechanical effects not captured by the other terms of the Kohn-Sham formulation. It is often split into two terms: first, the exchange part E_x takes into account the effects due to the exchange symmetry of a wavefunction of indistinguishable particles. In this case, the particles in question are electrons and as such this interaction is often referred to as the Pauli repulsion. Secondly, the correlation part E_c accounts for the influence that electrons have on the movement of other electrons.

Unfortunately, the exact form of E_{xc} is unknown and as such needs to be approximated. Popular approaches include the following:

- **Local Density Approximation (LDA)** approximates exchange-correlation as a function of only the local electron density. In large, the most successful approximations following this idea are based on the homogeneous electron gas. For the exchange part, this leads to the analytically simple expression found in Equation 2.12. The correlation part is usually only available at the limits of low or high densities. Ultimately, this is an inexpensive method but it also proves insufficient to accurately describe most systems.

$$E_x^{LDA}[\rho] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3} \int \rho(\mathbf{r})^{4/3} d\mathbf{r}. \tag{2.12}$$

- **Generalized Gradient Approximation** goes beyond LDA by taking into account non-uniformities in the electron gas. This is done by expanding E_{xc} in terms of the electron density *and its gradient*. These functionals are generally more complex, but they correct some of LDA's tendency to underestimate the exchange energy and overestimate the correlation energy. Popular examples of GGA functionals include **BLYP (Becke-Lee-Yang-Parr)** and **PBE (Perdew-Burke-Ernzerhof)**.

- **Hybrid functionals** are, as the name indicates, a mixture of functionals. It takes advantage of the fact that HF methods have an explicit expression for the exchange interaction and utilises it in the DFT formalism. The correlation part E_c is then taken from a different source such as GGA approaches. Examples include **B3LYP (Becke, 3-parameter, Lee-Yang-Parr)** and **PBE0**, the latter of which combines the HF exchange term with the PBE functional mentioned above:

$$E_{xc}^{PBE0} = \frac{1}{4}E_x^{HF} + \frac{3}{4}E_x^{PBE} + E_c^{PBE}. \tag{2.13}$$

Additionally to approximations for E_{xc}, dispersion interactions also need to be explicitly accounted for. Dispersion arises from instantaneous fluctuations in the electron distribution around an atom which in turn induces a complementary fluctuation around neighbouring atoms. While this interaction is comparatively weak, it becomes particularly relevant over longer distances such as for large molecules, layered materials or molecular complexes. Thus, the consideration of Van der Waals functionals is necessary to properly account for dispersion forces. These terms are additional corrections to the energy calculated by DFT: popular choices include Grimme's DFT-D series [73], the Tkatchenko-Scheffler method [75] or Many-Body Dispersion (MBD) [76].

Similarly to HF, the Kohn-Sham equations are also solved self-consistently by varying parameters such as to minimize the total energy. The procedure includes initialization, calculating the effective potential using the guessed electron density, solving the equations — which give a new guess for the electron density — and repeating the process. Convergence is reached when the updates in the electron density are lower than a given threshold.

2.3 Sampling

Creating a reference dataset for ML potentials requires a meticulous exploration of the system's configurational space, which entails choosing specific molecular configurations or atomic geometries to represent the myriad of conditions the potential may face in real applications. This sampling process is pivotal, as the quality and diversity of the chosen configurations directly influence the trained model's versatility and efficacy in handling real-world scenarios.

A primary approach to this sampling is to select a set of configurations from available molecular dynamics (MD) trajectories. MD simulations track the time-evolution of atoms and molecules as they interact, based on the principles of classical mechanics combined with the accuracy of previously mentioned *ab initio* methods for energy and force calculations. By simulating the motion of atoms over time, MDs provide a window into the dynamical properties and behaviour of chemical systems at the atomic scale, making them rich sources of data. This dynamic nature means that MD trajectories can inherently capture a wide range of molecular configurations, spanning from stable to transition states. Therefore, extracting configurations from MD trajectories for a dataset offers a robust and diverse foundation, ensuring that the machine learning potential is trained on a set that reflects the true dynamical nature of molecular systems.

However, molecular dynamics also come with inherent limitations. First, MD inherently explores thermodynamically accessible states, which means rare but crucial configurations might be underrepresented or entirely missed. Furthermore, long-timescale events or slow dynamics, often critical in processes like protein folding or material phase transitions, might be computationally prohibitive to simulate directly with MD. Hence, relying solely on MD could lead to gaps in the dataset, potentially compromising the comprehensiveness of the trained machine learning model.

This section briefly explores the basics of molecular dynamics in the context of reference data creation as well as popular additions or alternatives when more sophisticated approaches are needed.

2.3.1 Basics of molecular dynamics

At the most basic level, a molecular system can be described by the positions of all N atoms of the system [88]:

$$\mathbf{R} = (\mathbf{r}_1, ..., \mathbf{r}_N). \tag{2.14}$$

Simply using Newton's equations of motion, we can determine that for every atom i:

$$\mathbf{a}_i = \frac{\mathbf{F}_i}{m_i} = -\frac{1}{m_i}\frac{\partial E}{\partial \mathbf{r}_i}, \tag{2.15}$$

where $\mathbf{a}_i$ is the acceleration of the atom, m_i its mass, $\mathbf{F}_i$ the forces acting on it and E the total potential energy of the system. From this, one can evolve the system iteratively through numeric integration. The two most commonly used integrators are Velocity Verlet and the Leapfrog method. The former is shown below:

$$\mathbf{x}(t+\delta t) = \mathbf{x}(t) + \mathbf{v}(t)\delta t + \frac{1}{2}\mathbf{a}(t)\delta t^2, \tag{2.16}$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \frac{\mathbf{a}(t) + \mathbf{a}(t+\delta t)}{2}\delta t, \tag{2.17}$$

where $\mathbf{x}(t)$, $\mathbf{v}(t)$ and $\mathbf{a}(t)$ are the position, velocity and acceleration vectors at time t and δt is a finite timestep. Performing multiple iterations of this sort (usually until the forces are below a given threshold) results in the minimization of the potential energy and thus a **geometry optimization**. This is usually the first step towards the creation of a dataset, as it creates a starting configuration of the system that is stable and at a (local) minimum.

2.3.2 Thermostat and Barostat

To extract thermodynamical properties from a system, the latter needs to be simulated within a given ensemble. This is also needed if we want to go beyond simple geometry optimization and push the system to explore CS.

The two important ensembles are the canonical ensemble (NVT), where the number of atoms, the volume and the temperature are kept constant, and the isothermal-isobaric ensemble (NPT) where the pressure is kept constant instead of the volume.

Keeping the temperature constant requires the introduction of a thermostat. Simply put, the thermostat defines the temperature as the time-averaged kinetic energy K, where $K = \frac{1}{2}\sum_i \mathbf{v}_i^2$ and $\langle K \rangle = \frac{3}{2}Nk_BT$. In its most simple form, the thermostat takes form as a simple scaling factor λ on the amplitude of the velocities of the system:

$$v_i^{new} = v_i^{old} \cdot \lambda. \tag{2.18}$$

At each time step, the average kinetic energy $\langle K \rangle$ and its associated temperature T is calculated over a given period and compared to the target temperature T_t. The scaling factor λ is adjusted such as to equalise the temperature:

$$\lambda = \sqrt{\frac{T_t}{T}}. \tag{2.19}$$

In practice, this simple thermostat is unable to properly reproduce the dynamics inside of a canonical ensemble as the kinetic energy is not allowed to fluctuate. There are better alternatives such as the stochastic Andersen thermostat [89] where random collisions act on the atoms of the systems to regulate the temperature. Another viable method is the Nosé-Hoover thermostat which incorporates a heat bath directly into the system. Finally, one of the most widely used approaches is the Langevin thermostat, where the equations of motions are directly altered to maintain the temperature:

$$\mathbf{a}_i = \frac{\mathbf{F}_i}{m_i} - \gamma_i \mathbf{v}_i + \frac{f_i}{m_i}. \tag{2.20}$$

Here γ_i is a friction coefficient and f_i is a stochastic force that simulates thermal kicks or collisions from the environment. In the Langevin framework, the environment is assumed to be a continuum of small particles that our (large) particles of interest are surrounded by.

Note that the NPT ensemble requires a barostat as well as a thermostat. Barostats work very similarly to the above, with the total pressure being kept constant (as opposed to the temperature) by varying the total volume (as opposed to the velocities).

2.3.3 Best practices and advanced sampling techniques

Simple molecular dynamics such as those described above tend to stay in or near equilibrium throughout the entirety of its runtime. This is natural, as the only effect counterbalancing energy minimization efforts are the thermostat and barostat. Thus, uncommon or even rare events are unlikely to be visited unless the dynamics run for prohibitively long times. Given the aforementioned computational costs attributed to our chosen *ab initio* methods that calculate the potential energy, this is not ideal.

Thus, especially for large and/or flexible molecules, it is often necessary to bias the dynamics such that they are more likely to move out of equilibrium states. One of the simplest methods is running temperature accelerated dynamics, where the temperature of the thermostat is raised significantly. This allows rare events such as overcoming energy barriers to occur more frequently, thus hastening the exploration of CS.

Another way to accomplish this goal is by performing **adaptive umbrella sampling**. The basic idea is to add an external bias B to the system resulting in a new, modified potential E':

$$E'(\mathbf{R}) = E(\mathbf{R}) + B(\mathbf{R}). \tag{2.21}$$

Every few timesteps, the external potential B is adjusted such as to make the states already visited less likely. It typically accomplishes this by calculating a histogram over the visited states and adding its logarithm as an additional term to the potential.

This method is very similar to that of **metadynmics**, where histograms are skipped but every (nth) visited timestep leads to the creation of an additional Gaussian term to the potential. This Gaussian is centred at the visited state $\mathbf{R}$ itself, thereby pushing the system away from it. Note that both umbrella sampling and metadynamics often act on collective variables rather than the atomic coordinates themselves, i.e. the external potential is given by $B = B(\mathrm{CV}(\mathbf{R}))$, where $\mathrm{CV}(\mathbf{R})$ is a collection of characteristic variables for the system (angles, dihedrals, distances...).

More specialized methods are available when dealing with specific systems, such as the **Nudged Elastic Band** [65] method for datasets containing the reaction path between two configurations of interest, or **Steered Molecular Dynamics** [90, 91] when dealing with e.g. the manual stretching of a molecule. Furthermore, machine learning approaches can be applied here as well through the usage of **active learning** [92, 93]. However, a complete discussion of the different methods is beyond the scope of this work.

2.4 Regression

Due to the high computational costs associated with *ab initio* methods, the idea of bypassing them through cheaper means is of high interest. ML provides one such way and its application in the space of chemical compounds has gained traction over the years. This new field of Quantum Machine Learning (QML) deals with building statistical relations between molecules (often in particular configurations) and corresponding properties of interest. These can include thermodynamic properties such as atomization energies or dipole moments [94–96], as well as directly the PES or FF of the system. In this work, the prediction of the latter is the focus.

Thus, the problem boils down to building a regression model capable of associating molecular geometries and their respective potential energy and/or atomic forces. These relations are learned by parametrizing regression models such as NNs to closely imitate the behaviour of given reference data. However, this is a highly non-trivial task in large due to the complexity of chemical interactions, as illustrated by the ongoing research on electronic structure theory and beyond. Furthermore, as the cost of creating reference data is high, the models utilised in this field have to be both accurate and data-efficient.

This section provides a short overview of the developments made in pursuit of these goals over the last decades. While not an extensive deep dive, it provides the necessary minimum to understand the main challenges as well as the major breakthroughs and ideas that defined the field throughout the years.

2.4.1 Early Neural Network Potentials

Some of the earliest applications of NNs on molecular systems trace back to around 1995 [97, 98]. The networks of that time were limited to shallow architectures of 1-2 hidden layers. The inputs consisted of one or a few internal coordinates representing a relevant part of the system. For example, the potential energy of medium-sized organic molecules (Tetrahydrobiopterin, 32 total atoms) with respect to two chosen torsional angles was learned using a small neural network with a topology of 2-3-3-1 [99].

In all cases, the input describing the molecular system provided to the early-generation neural network was simple representative variables such as specific distances, angles, dihedrals, etc. Notably, all those descriptors are *rotationally and translationally invariant*. That is, if the entire system is translated by an arbitrary vector or rotated in any direction, the input variables — and thus the predicted energy — do not change. Including these physical symmetries in the regression models is paramount, and as such, it is present in all models discussed in this chapter. On the other hand, permutational symmetry, which ensures that the potential energy does not change when swapping the order of two identical atoms, was not preserved with networks of that type. Furthermore, the reliance on handpicked internal coordinates means the networks were tailored to specific systems without any transferability.

To preserve the permutational symmetries of the original system, it is necessary to consider every atom equivalently. For instance, the total potential energy E computed for a system of N atoms can be given by $E = \sum_{i=1}^{N} E_i$ where E_i is the contribution to the total energy of atom i. Each atomic contribution $E_i = \mathrm{NN}_i(\mathbf{x}_i)$ is calculated based on a descriptor of the atom's environment $\mathbf{x}_i$, where the neural network NN is shared across the same atomic species. Moreover, the network itself must be permutationally invariant, i.e. $\mathrm{NN}_i(\mathbf{x}_i) = \mathrm{NN}_i(\hat{P}\mathbf{x}_i)$, where $\hat{P}$ are relevant permutations of atoms. This, along with the commutative nature of the summation over atomic contributions, satisfies the desired permutational invariance. One of the earliest implementations of this scheme is the Behler-Parrinello Neural Networks (BPNN) in 2007 [30], depicted in Figure 2.2.

As BPNNs are shallow and thus contain a limited amount of parameters compared to modern architectures, the energetic contribution E_i of an atom i can be expressed in a simple analytical formula:

$$E_i = f_a^2[w_{01}^2 + \sum_{j=1}^{3} w_{j1}^2 f_a^1(w_{0j}^1 + \sum_{\mu=1}^{2} w_{\mu j}^1 G_i^{\mu})], \tag{2.22}$$

where w_{ij}^k is the linear weight connecting the jth node in layer k with the ith node in layer $k-1$, w_{0j}^k is the bias and f_a^k is a non-linear activation function. As bare

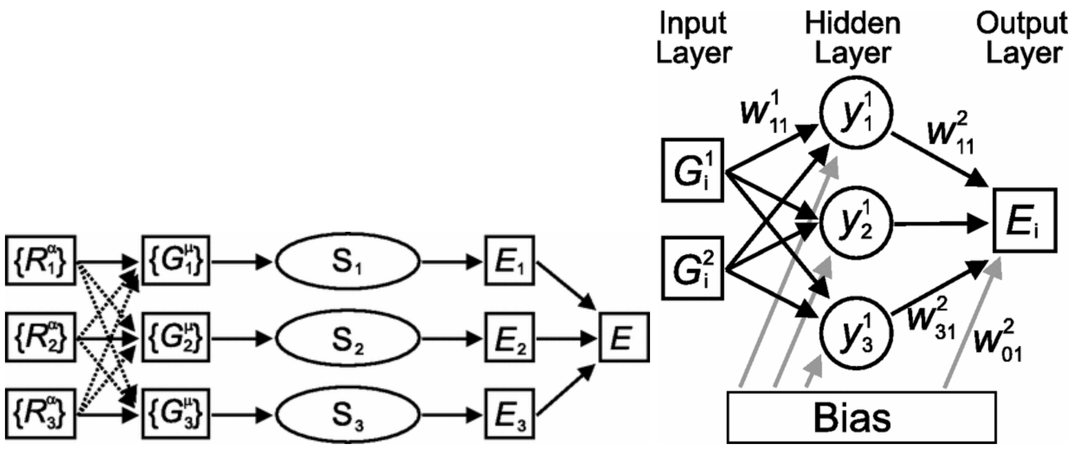

FIGURE 2.2: (Left) Illustration of a Behler-Parrinello Neural Network (BPNN) acting on a system composed of 3 atoms. Every atom's environment is expressed in terms of symmetry functions and fed to a subnet S. (Right) Structure of the atomic subnets S. Figures taken from their original publication [30].

NNs are fundamentally just sequences of linear combinations, non-linear activation functions are needed to allow NNs to fit non-linear functions of input variables. These functions are applied element-wise to a layer after the linear combination step. In the case of BPNNs, f_a^1 (the activation function of the hidden layer) is a hyperbolic tangent while f_a^2 (the activation function of the last layer) defaults to a linear function.

The descriptor G_i^μ of atom i and its environment are modelled by radial symmetry functions. They consists of one radial contribution G_i^1 and one angular contribution G_i^2:

$$G_i^1 = \sum_{j\neq i}^{\text{all}} e^{-\eta(R_{ij}-R_s)^2} f_c(R_{ij}), \tag{2.23}$$

$$G_i^2 = 2^{1-\zeta} \sum_{j,k\neq i}^{\text{all}} (1+\lambda \cos\theta_{ijk})^\zeta e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}), \tag{2.24}$$

$$f_c = \begin{cases} 0.5\left(\cos\frac{\pi R_{ij}}{R_c}+1\right), & \text{if } R_{ij} \leq R_c \\ 0, & \text{if } R_{ij} \geq R_c. \end{cases} \tag{2.25}$$

Here, R_{ij} is the distance between atoms i and j while θ_{ijk} is the angle between atoms i, j and k respectively. $\lambda \in \{-1, 1\}$, η, ζ and R_c are all hyperparameters chosen before training. R_c in particular, denotes a cutoff distance (set to 6 Å in the original paper) after which contributions of the environment are neglected. The descriptor once again satisfies rotational and translational invariance as well as permutational invariance due to it being applied equivalently to all atoms independently.

2.4.2 Kernel methods

In developing statistical methods to infer potential energies given molecular structures, one of the most complex and time-intensive tasks is the judicious construction of the descriptor. This is particularly important for shallow NNs (such as BPNN), where only a handful of parameters are dedicated to tuning the representation of the atomic environment before the output layer. Kernel Ridge Regression (KRR) or Gaussian Processes (GP) work very similarly to shallow NNs. In fact, they can equivalently be represented as a single-layer NN architecture. In this framework, predictions of the atomic energy contributions are given as a linear combination over all training points n:

$$E_i = \sum_n \alpha_n \kappa(\mathbf{x}_i, \mathbf{x}_n). \tag{2.26}$$

Here, $\kappa(\mathbf{x}_i, \mathbf{x}_n)$ is the kernel function providing the similarity between the representation of atoms i and n, and α_n is the linear weight that is optimized during the learning process.

In particular, as GPs provide a closed-form analytical solution, the weights $\boldsymbol{\alpha} = \{\alpha_n\}$ do not need to be trained iteratively like in NNs but are instead given as a function of the covariance matrix $\mathbf{C}$ of the training data:

$$\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}. \tag{2.27}$$

Here, $\mathbf{y} = \{y_n\}$ is the set of reference values as calculated by the chosen *ab initio* method, δ and σ are model hyperparameters. The *kernel matrix* $\mathbf{K}$ (or covariance matrix) is a symmetric matrix representing the similarity between every point in a given training set, calculated using the kernel function κ.

One of the earliest and most prominent implementations of this scheme on the prediction of PES is the Gaussian Approximation Potentials (GAP) method in 2010 [50]. In the equation above, $\mathbf{x}_i$ once again represents the descriptor of atom i in its environment: the most popular choice to be used with GAP is the Smooth Overlap of Atomic Positions (SOAP) [47]. The fundamental idea of the latter is to represent an atomic density neighbourhood function for each atom:

$$\rho(\mathbf{r}) = \sum_i w_{Z_i} \delta(\mathbf{r} - \mathbf{r}_i), \tag{2.28}$$

where $\sum_i$ is the summation over all neighbours (usually within a cutoff distance) and w_{Z_i} is a weight factor dependent on the atomic species Z_i of atom i. Then, this atomic density is expanded into a set of radial basis functions and spherical harmonics. The former provides radial information, while the latter describes angular information:

$$\rho(\mathbf{r}) = \sum_n \sum_l \sum_{m=-1}^{l} c_{nlm} g_n(r) Y_{lm}(\hat{\mathbf{r}}), \tag{2.29}$$

$$c_{nlm} = \langle g_n(r) Y_{lm} | \rho \rangle, \tag{2.30}$$

where $g_n(r)$ is a set of radial basis functions (assumed to be orthogonal) and Y_{lm} is the set of spherical harmonics. Typically, the suggested choice of $g_n(r)$ is cubic and higher order polynomials, though other choices have also been used [100, 101]. The radial part depends only on the distance between pairs of atoms and is, as such, rotationally and translationally invariant. The spherical harmonics, on the other hand, generally depend on the orientation of the reference frame. However, rotational invariance can be achieved by integrating the kernel over all possible rotations [102].

Another noteworthy kernel method in the space of learning molecular interactions is the Gradient Domain Machine Learning (GDML) model in 2016 [51]. The motivation behind this work is the potential a directly learning the forces as the primary observable. In previous ML potentials, the observable that is directly predicted is the energy, while the forces have to be analytically derived or numerically computed. While this satisfies energy conservation, there is a compromise between optimising the energy and force predictions of the model. However for applications in molecular dynamics, the forces are the fundamental property used to evolve the system and, as such, are arguably more important than the energy.

The GDML's paradigm revolves around learning the forces directly instead of energy contributions. It does so with an extended KRR model that computes all forces of the entire system in one go:

$$\mathbf{K}_{\mathrm{Hess}(\kappa)} + \lambda \mathbf{I}\boldsymbol{\alpha} = \nabla E = -\mathbf{F}. \tag{2.31}$$

Here, $\mathbf{K}_{\mathrm{Hess}(\kappa)}$ is the Kernel matrix where every element is a Hessian (matrix) of the similarity between two points, as given by the kernel function κ. Once again, λ is a regularization hyperparameter and $\boldsymbol{\alpha}$ is the set of learnable parameters. Under this scheme, a trained model predicts the forces for a given configuration as:

$$f(\mathbf{x}) = \sum_{i=1}^{M} \sum_{j=1}^{3N} (\boldsymbol{\alpha}_i)_j \frac{\partial}{\partial x_j} \nabla \kappa(\mathbf{x}, \mathbf{x}_i). \tag{2.32}$$

Energies can be reconstructed (up to a constant) by integrating the expression above. Through learning forces directly, GDML can substantially increase accuracy, making it more appropriate for running dynamics than if it instead only learned the PES.

GDML is an atomistic model and treats the system as a whole in a single pass. In particular, the descriptor used is the set of inverse pairwise distances between all unique pairs of atoms. As no cutoff function is used, the descriptor is global and has — in principle — all the information needed to reproduce even long-range interactions. However, while the translational and rotational invariances are still maintained, GDML is not permutationally invariant. Indeed, the ordering of the descriptor changes upon swapping the position of two atoms.

This issue can be alleviated by manually reintroducing symmetries into the model's predictions. In an extension to the model called symmetric GDML (sGDML) [43, 44], the descriptor for every geometry is effectively expanded into its set of all permutations. However, for the sake of computational speed, the permutations are limited to those statistically found to be present in the provided dataset. Nevertheless, the inclusion of permutational symmetries increases the performance of the model significantly for any system for which permutational symmetries are relevant.

Another consequence of the architecture of GDML is its lack of transferability. As the system is treated as a single object, a GDML model can only be used for systems with the same number of atoms as the original dataset. Otherwise, the descriptor size does not match the shape of the kernel matrix and the learned parameters $\boldsymbol{\alpha}$.

Furthermore, the GDML approach has also been extended to periodic systems with the introduction of Bravais-Inspiried GDML (BIGDML) [45]. There, the descriptor was altered to be suitable for periodic systems by making use of periodic boundary conditions, and the translational as well as Bravais symmetry groups in materials were explicitly leveraged. This modification to the framework led to accurate dynamics of the interaction of a molecular benzene with a 2D graphene layer as well as interstitial hydrogen diffusion in bulk Pd.

2.4.3 Deep Neural Networks

To overcome the mentioned limitations of the kernel models and early-generation NNs, most modern ML architectures to learn the PES or FF are deep neural networks (DNN). The idea of using more hidden layers was largely popularised after 2012 from the work done on image recognition [103]. This, along with continuous upgrades in both available software and hardware, led to the majority of modern NNs in cutting-edge applications involving deep architectures.

The main implication of MLFFs is that DNNs can learn the descriptor during the training process. While shallow networks or even kernel methods are very heavily dependent on the descriptor of choice, DNNs can allocate a large part of their learnable parameters to fine-tune the descriptor to more accurately represent the system and thus aid the final predictions.

The Deep Tensor Neural Network (DTNN) [104] was one of the first to implement this methodology on MLFFs, which was later refined by models such as SchNet [52, 53]. Its network, depicted in Figure 2.3, was influential in the field and motivated many future developments. First, it contains a substantial amount of hidden layers, including 3 interaction layers. The interaction layers are the fundamental building blocks of the network that allow the descriptor of the system to be learned.

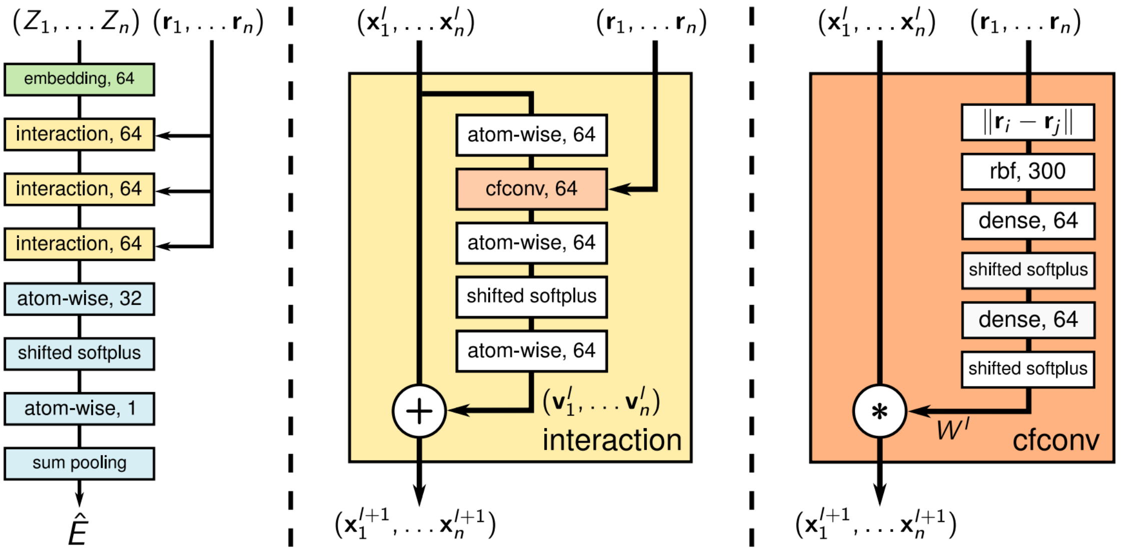
$(Z_1, \ldots Z_n)$
$(\mathbf{r}_1, \ldots \mathbf{r}_n)$
embedding, 64
interaction, 64
interaction, 64
interaction, 64
atom-wise, 32
shifted softplus
atom-wise, 1
sum pooling
$\hat{E}$
$(\mathbf{x}^l_1, \ldots \mathbf{x}^l_n)$
$(\mathbf{r}_1, \ldots \mathbf{r}_n)$
atom-wise, 64
cfconv, 64
atom-wise, 64
shifted softplus
atom-wise, 64
$(\mathbf{v}^l_1, \ldots \mathbf{v}^l_n)$
interaction
$(\mathbf{x}^{l+1}_1, \ldots \mathbf{x}^{l+1}_n)$
$(\mathbf{x}^l_1, \ldots \mathbf{x}^l_n)$
$(\mathbf{r}_1, \ldots \mathbf{r}_n)$
$\|\mathbf{r}_i - \mathbf{r}_j\|$
rbf, 300
dense, 64
shifted softplus
dense, 64
shifted softplus
W^l
cfconv
$(\mathbf{x}^{l+1}_1, \ldots \mathbf{x}^{l+1}_n)$

FIGURE 2.3: Illustration of the SchNet architecture and its three blocks (from left to right): an overview of the complete network, the interaction block, and the continuous-filter convolution block. The figure is taken from its original publication [52].

Secondly, the atom-wise layers within those interaction layers transform the representations **x** of every atom independently and allow the descriptor to be iteratively refined per atom. The representation of each atom is altered as a function of learnable parameters that are shared across atoms, thereby maintaining permutational invariance. Note that the interaction layer also features a skip connection, adding the original atomic representations to the transformed ones. This is a trick inspired by ResNet [105] and serves to stabilize the gradients during training.

Thirdly, the interaction layer includes a so-called continuous-filter convolution. This enables the atoms to interact with one another in a pairwise way by feeding in a rotationally and translationally invariant description of the original atom environments. The output of the layer is a filter that emphasizes the important parts of the radial environment for each atom, reminiscent of attention mechanisms in other fields of ML [106].

Note that SchNet provides a single scalar representing the total energy; forces are obtained by differentiating the entire network with respect to the input atomic positions. This feature requires all layers to be at least twice differentiable, putting some constraints on the choice of architecture and activation functions.

All in all, SchNet showcased remarkable accuracy while retaining all the required invariances. This was largely made possible by treating the entire network in an atom-wise manner, as was previously established by e.g. BPNN. Note that this also allows SchNet to be transferable: a network trained on any system can, in principle, be applied to any other system of arbitrary size, though the similarity of the systems largely determines the model's accuracy in such a scenario.

2.4.4 Message Passing Neural Networks

While SchNet did not qualify itself as a Graph Neural Network (GNN) at the time of its creation, its architecture can be viewed as such. Many modern MLFFs have adopted the GNN architecture and, more specifically, the message-passing kind. In essence, every atom is represented as a node in a graph connected to other nodes in their neighbourhood via edges (usually within a given cutoff distance).

Every node v is associated with its node attributes (or embedding) $\mathbf{h}_v$. In the simile with SchNet, this is equivalent to the atomic representations within each layer. Furthermore, edges can also have their own attributes $\mathbf{h}_e$, though this will not be discussed here for the sake of simplicity. Together, these attributes constitute the representation of the system and in principle include all the information necessary to properly predict the PES.

In the message-passing phase, information is exchanged between connected nodes and the embedding of each node is updated based on all the incoming messages. As multiple passes of this phase are done, long-range interactions can, in principle, be modelled despite the distance cutoff for edges. The network can be summarized by three learnable functions, which are each intrinsically independent networks. They are M_t for generating messages from embeddings, U_t for updating node embeddings and R for pooling the entire graph into a single output (such as the energy). In mathematical terms, a message passing pass evolving the system from step t to step $t+1$ can be summarized as follows:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, h_{e,vw}), \tag{2.33}$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}), \tag{2.34}$$

with v a node and $N(v)$ all its neighbouring nodes. The final aggregate value y after K message passes is given by:

$$y = R(h_v^K | v\, in\, G), \tag{2.35}$$

where G is the entire graph. A schematic depiction of this process for both molecules and crystalline materials can be found in Figure 2.4 [107]. Continuing the comparison with SchNet, U_t is the equivalent of the interaction layer, M_t is the convolution layer, and there are a total of 3 message passes (i.e. interaction blocks).

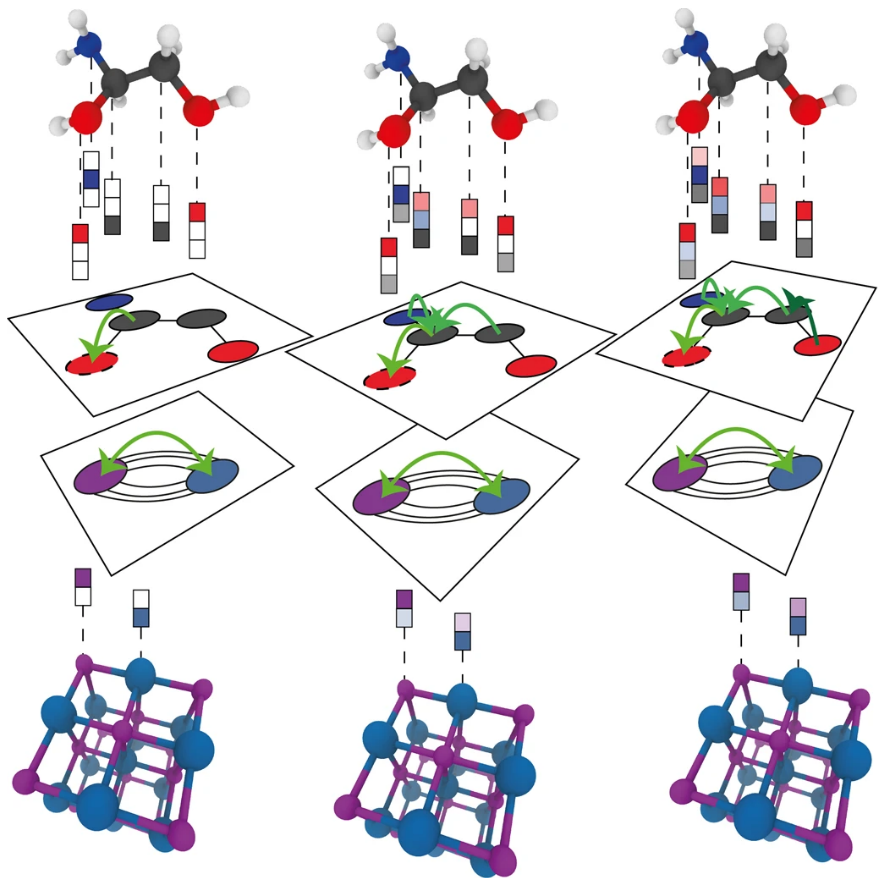

FIGURE 2.4: Schematic depiction of a message passing operation for both molecules (top) and crystalline materials (bottom). Figure taken from its original publication [107].

These GNNs provide a general framework for building MLFFs that allow atom descriptors to be learned through successive interactions with their environment. The efficacy and efficiency of these networks have made them ubiquitous in the field and allow for great flexibility when modelling interactions between atoms in the message-passing phase. Many models have been proposed within this paradigm, including GemNet [29], ForceNet [108], SphereNet [109] and more [28, 110–116].

2.4.5 Equivariant representations

Up to now, every feature, descriptor and interaction of atoms were invariant with respect to the symmetries of the original system, as is the potential energy that is ultimately predicted. However, considering only invariant features makes abstraction of the orientation between atoms. Including directional information in the modelling of atomic interactions inside the NN was shown to be more beneficial than increasing the depth of an invariant network [117].

Thus, equivariant representations to describe atomic features and interactions have been proposed recently alongside their invariant counterparts. That is, the equivariant features of the GNN undergo the same transformations as those applied to the original system used as input. An example of such an operation is the equivariant convolution as implemented in the e3nn package [61]:

$$h_v^{t+1} = \frac{1}{\sqrt{z}} \sum_{w \in N(v)} h_w \otimes MLP(||\mathbf{d}_{vw}||) Y(\mathbf{d}_{vw}/||\mathbf{d}_{vw}||), \tag{2.36}$$

where z is the average degree of nodes, $\mathbf{d}_{vw}$ is the distance vector between node v and node w, Y is their spherical harmonics and $\otimes$ is a tensor product with learnable parameters. Modern MLFFs utilise this feature to further enhance the accuracy of their predictions, such as Nequip [60], PaiNN [118], MACE [58, 59] and more [119–121]. The performance and data efficiency of these GNNs combined with equivariant representations make them the current state-of-the-art architectures for most ML applications in the field, be it for learning a PES, forces, or other properties.

2.4.6 Explicit physical interactions

Most architectures seen so far circumvent the scalability issues that arise from an increasing number of atoms by considering local neighbourhoods, usually within a given cutoff distance (5 – 6 Å). Beyond the cutoff, GNNs can theoretically model long-range interactions through successive messages from atom to atom. In practice, unless the cutoff distance is sufficiently large, there is no guarantee that long-range interactions are properly accounted for.

To bypass this, some architectures opt to include explicit physics, such as Coulomb interactions between atoms at large distances. One such example is

PhysNet [62], where the total potential energy prediction is given by the sum of atomic contributions plus a Coulomb term and a dispersion correction term.

$$E = \sum_{i=1}^{N} E_i + k_e \sum_{i=1}^{N} \sum_{j>1}^{N} \tilde{q}_i \tilde{q}_j \chi(r_{ij}) + E_{D3}. \tag{2.37}$$

Here, k_e is Coulomb's constant, E_{D3} is a dispersion correction term calculated according to DFT-D3 [73], and $\chi(r_{ij})$ is a function the smoothly interpolates the long-range $1/r_{ij}$ behaviour of Coulomb's law but damps short-range to avoid the $r_{ij} = 0$ singularity. $\tilde{q}_i$ and $\tilde{q}_j$ respectively are so-called corrected partial charges:

$$\tilde{q}_i = q_i - \frac{1}{N}(\sum_{j=1}^{N} q_j - Q), \tag{2.38}$$

where Q is the total charge of the entire system and q_i is the partial charge of atom i as predicted by the NN (additionally to its energy contribution). This correction ensures the conservation of the total charge.

Similarly, SpookyNet [63] adds a nuclear repulsion, an electronic, and a dispersion term to the total potential energy:

$$E = \sum_{i=1}^{N} E_i + E_{rep} + E_{ele} + E_{vdw}. \tag{2.39}$$

The nuclear repulsion is based on the Ziegler-Biersach-Littmark stopping potential [122, 123] and acts as a correction to short-range electrostatics which roughly follows a $1/r$ trend with a parametrized factor. The electronic term models long-range electrostatics using atomic partial charges predicted from atomic features by the network. The final term introduces van der Waals interactions using the two-body term of the D4 dispersion correction [124].

Furthermore, it considers additional degrees of freedom, such as the total charge or spin state of the system, to better model nonlocal effects (i.e. effects beyond a given cutoff). This is done by considering an additional layer in parallel to the local interaction layer during the message-passing phase. This layer models the nonlocal interactions using a self-attention mechanism, which gets added to the atomic embedding.

With these additions, both PhysNet and SpookyNet can predict long-range interactions beyond the cutoff radius involved in the message-passing steps and reproduce correct asymptotic behaviour. For instance, dissociation curves of a handful of diatomic molecules were computed by SpookyNet models with and without nonlocal interactions. Notable differences were found between the two predictions, with the nonlocal model showcasing long-range behaviour more consistent with reference calculations [add refs].

The evolution of ML architectures demonstrates that the most advanced ML approaches alone cannot reliably mimic *ab initio* calculations without extensive reference datasets unfeasible in practice. One needs to synergize ML models with physical laws and chemical intuition to achieve reliable performance with ML models.

2.5 Analysis

The development of MLFFs has reached a point where many systems that were recently thought out-of-reach for computational chemistry are now feasible to be learned by ML models. When new milestones are reached, the tools to analyse the new systems must be equally adequate. This is a notion that applies to MLFFs as well as other ML fields: with the advent of the new generation of large language models such as GPT4 [125], the number of papers dedicated to the evaluation of those models increased tenfold across the first half of 2023 [126]. Similarly, the evaluation and analysis of ML models applied to chemical systems need further development to support the new level that has been reached over the years.

In its simplest form, the base analysis of an MLFF model boils down to comparing its predictive outputs to the already existing reference dataset. That is, for a given configuration $\mathbf{x}$ and its corresponding forces $\mathbf{F}$, we compute the prediction $\mathbf{F}' = h(\mathbf{x})$ to compare against.

The most commonly used metrics are the Mean Average Error (MAE) and the Root Mean Squared Error (RMSE), shown below:

$$\text{MAE} = \frac{1}{3N}\sum_{i=1}^{N}\sum_{a=1}^{3}|f'_{i,a} - f_{i,a}|, \tag{2.40}$$

$$\text{RMSE} = \sqrt{\frac{1}{3N}\sum_{i=1}^{N}\sum_{a=1}^{3}(f'_{i,a} - f_{i,a})^2}, \tag{2.41}$$

where $f_{i,a}$ is the force component of atom i and direction a. The same expressions can be used for the energy, where an MAE of less than 1 $kcal/mol$ is said to be of "chemical accuracy", i.e. where the prediction errors of the ML model itself is smaller than those of the reference dataset itself.

One substantial advantage to kernel methods is their innate ability to analytically compute the variance of their prediction and thus the degree of certainty at a given point. With $\mathbf{K}(\mathbf{X},\mathbf{X}) = \mathbf{X}^T\mathbf{X}$ the kernel matrix for a given training set $\mathbf{X}$ and $\mathbf{X}^*$ a new test point, the variance of the prediction y^* is given as [127]:

$$\text{var}(y^*) = \mathbf{K}(\mathbf{X}^*,\mathbf{X}^*) - \mathbf{K}(\mathbf{X}^*,\mathbf{X})\mathbf{K}(\mathbf{X},\mathbf{X})^{-1}\mathbf{K}(\mathbf{X},\mathbf{X}^*). \tag{2.42}$$

For NNs, such a variance is not intrinsically available. However, one can use the variance produced by a committee of models. In other words, rather than training a single model on a specific training set, one instead trains multiple models on different subsets of the reference dataset. This way, multiple independent predictions can be made with which to approximate a variance.

$$\text{var}(y^*) = \frac{1}{n}\sum_{i=1}^{n}(y_i^* - \mu^*), \tag{2.43}$$

$$\mu^* = \frac{1}{n}\sum_{i=1}^{n}y_i^*, \tag{2.44}$$

where n is the number of independent models created.

Other traditional methods to assess the quality of an ML model are also present in the field, such as cross-validation or learning curves to ensure the convergence of the model. Missing from the approaches above, however, is the physicality of the original problem. More sophisticated methods of analysis include reproducing properties or general behaviours when running a dynamic with the trained model. All in all, the ultimate test of stability and reliability is to use the model in its intended deployment environment. It is very common for MLFFs trained on molecular trajectories generated from molecular dynamics to rapidly break down when directly used in said dynamics. However, this is not always feasible and while a botched simulation indicates the existence of instability, finding its root cause is not trivial. Even the uncertainty measures described above are not always reliable: indeed, the fact that an ML model or a set of models is confident in its predictions does not mean that the predictions are correct. For instance, message-passing NNs without explicit long-range corrections cannot describe an adsorption process when the distance between the adsorbate and the substrate exceeds the cutoff radius. At the same time, such models would be confident that the adsorbate-substrate interaction is zero in such a case.

This work's main goal is to fill in some of the gaps in the current analysis tools to facilitate the task of finding out the quality and applicability of trained ML models. This is done via new methods and baseline tools for future model quality assessment which are explained, implemented and discussed throughout the following chapters.

Chapter 3

Force Fields under the Microscope with FFAST

Mastery of physico-chemical understanding at a molecular level is necessary for stimulating industrial and technological advancements, be it for creating novel materials, designing new drugs, and more. In these fields, computer simulations have already taken root as an indispensable tool for fast and competitive research and development. The popularity of MLFFs is steadily rising intending to extend the applicability of existing computational tools to larger systems and more accurate simulations. Pushing towards accurate modelling of complex systems, such as large peptides and proteins, requires us to bring the accuracy of *ab initio* methods to areas previously only accessible to classical FFs. While MLFFs have shown impressive predictive accuracy [19–47, 50, 52–54, 57–60, 62, 63] for various systems across many benchmark datasets [51, 128–133], there is much work to be done before one can trust MLFFs to reliably substitute *ab initio* calculations in any practical application of interest.

The previous section showed that MLFFs are born from many different, interlinked processes that each bear their own complexities. In particular, the common ways to analyse the performance of a model were touched on. It was highlighted that overall error metrics such as MAEs or RMSEs are particularly commonplace as a way to assess model quality. However, it is clear that for a model that relies on so many complex steps, from electronic structure theory to architectural choices, a single error value cannot fully describe the details of a model's predictive capabilities.

It only takes a single particularly problematic configuration for the model to cast the entire system into improbable or unphysical states, drastically affecting many if not all following simulation steps. This can occur even if the overall prediction error on said configuration is low, but a single atom's force calculations stray far from the ground truth. The larger the system at hand, the more likely that regions of extrapolations will be reached even in low-temperature simulations, thereby constantly putting the model at risk of breaking down if it is not stable as well as accurate.

For these reasons, the adoption of MLFFs in widespread industrial applications and development requires the ability to convincingly assess a model's applicability beyond simple numerical accuracies. There are, of course, no silver bullets to creating perfectly reliable ML models or even defining the applicability region of a model. Note that this problem is not unique to FFs, the quest for interpretability and transparency in neural networks for better assessment of real-world performance is a hot topic in the field of ML as a whole [134, 135]. Nevertheless, shedding light on MLFFs to avoid potential pitfalls and limitations requires us to look beyond usual error metrics [136, 137]. To take one major step towards this goal, we developed FFAST (Force Field Analysis Software and Tools) [64]: a cross-platform software package whose entire purpose is to delve deep into a model's performance and limitations. In particular, it focuses on gaining detailed insights at a glance through an easy-to-use graphical interface complete with intuitive 3D visualisations. Note that the focus of the software is MLFFs, but all the tools are generally applicable to molecular FFs of any kind (e.g. empirical mechanistic force fields).

3.1 Software overview

The analysis starts with the loading of a dataset (currently .xyz, .npz, and .db are supported). The software already offers ways to gain insights on the dataset itself, usual applications also involve loading an MLFF model of choice (currently sGDML, SchNet, Nequip, MACE, SpookyNet, and pre-predicted datasets are supported). For all supported models, **energy and force predictions** can be generated on the fly, though for larger datasets a **headless mode** (i.e. no graphical user interface) is provided to avoid excessive loading times by pre-computing predictions externally (e.g. on a high-performance computer).

Once loaded, here are some of the tools the interface makes available to the user:

- **Error timelines** showing the MAE throughout the dataset, useful for analysing time-ordered datasets such as trajectories. An adjustable smoothing factor allows one to remove noise and visualise general trends.
- **Error distributions** visualise energy and force predictions of any dataset/model combination of choice.
- **Cluster errors** as explored in section 4.4.
- **Error scatter** plots allow finding energy and force outliers quickly.
- **Atomic error distributions** allow visualising force prediction errors per atom type.

All plots can be interacted with: in particular, most plots allow the creation of **sub-datasets** through zooming. Those are subsets of loaded datasets that can be used independently in the software to focus on particular areas of a model and/or dataset. They can also be **saved in any format** supported by the software for external use.

One of the major advantages of using FFAST is the inbuilt **3D visualisation tool**. Any dataset or subset can have its molecules visualised on the fly in one or more interactive windows. The visualiser itself comes with a handful of features such as easier visualisation of flexible molecules through dynamic **alignment of geometries** along a chosen plane, **manual choosing of bonds** to be displayed and of course **animation** of a given trajectory. It also provides rudimentary information about a target geometry, such as **atomic distances, angles** or **dihedrals**. While by default each atom is coloured according to its element, this can be changed to show **atomic force prediction errors** or **displacement** either at a given timeframe or as an average throughout a dataset/subset. Finally, if a specific set of atoms is of particular interest, **atom-filtered sub-datasets** can also be created in this interface. Note that all features above can also be used on the underlying reference dataset via a dummy model. With that, FFAST acts on reference energies and forces rather than ML model predictions.

All components of the software are modular. Users comfortable with Python can add features and adjust existing ones to their workflow. The goal is to reduce the need to swap between software and specialised analysis scripts as much as possible. Note that while FFAST itself only depends on readily available Python packages, installation of ML models is subject to their own installation process. With all the tools combined, FFAST operates as a one-stop shop for beginners and experts alike who want to asses their MLFF performance with any desired level of detail.

3.2 Typical workflow

To showcase FFAST's features beyond a simple collection of analytical tools, this section presents what a typical use case of the software might look like. This does not serve as an all-inclusive exposition of features, but rather a map to guide the user through options available to them when starting to analyse a new dataset and/or model.

The first step is the loading of one or multiple ML models as well as a dataset of interest (often the respective reference dataset). In the case of a large dataset, all force and energy predictions would preferably be pre-computed on a supercomputer in headless mode, thus avoiding prolonged loading times during the utilisation of the software. As an expository example, this section uses a Nequip model trained on a stachyose dataset from the MD22 database. The Nequip model was trained on a total of 1000 training points and all hyperparameters were left unchanged from those recommended in the original paper and official documentation.

3.2.1 Prediction error overview

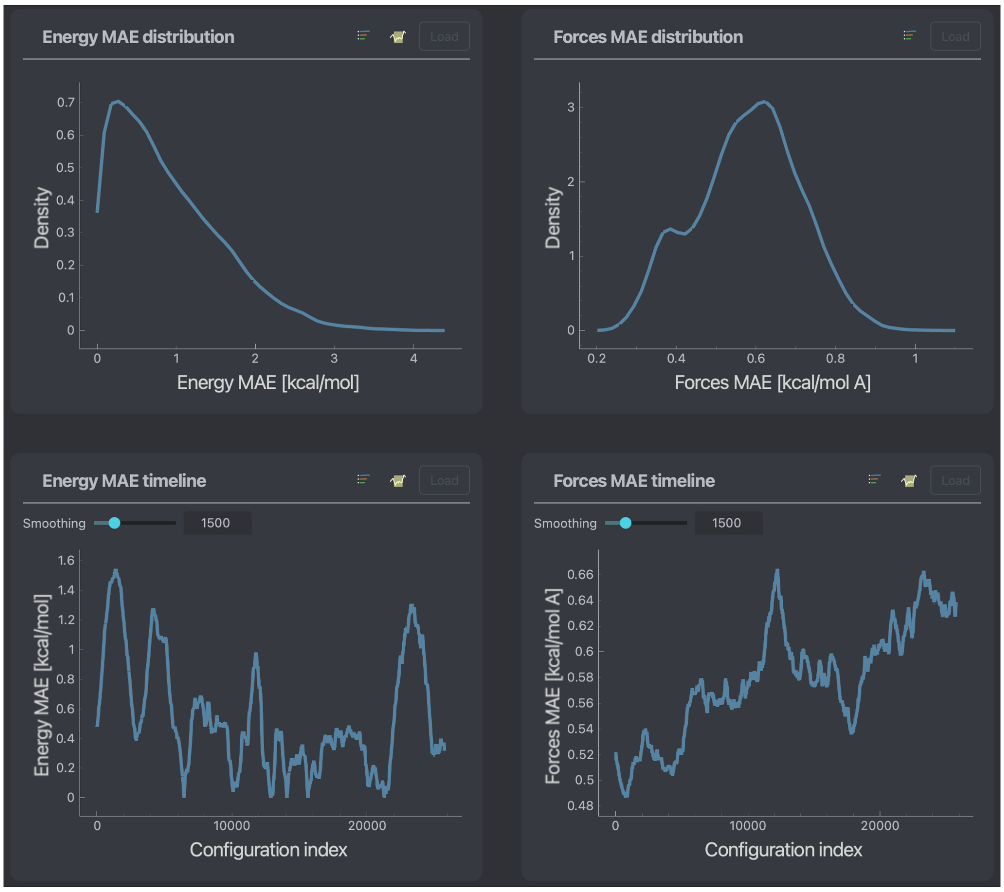

FIGURE 3.1: Example of a basic error screen in FFAST, showcasing timelines and distributions for energy and force prediction MAEs. Predictions performed by a Nequip model trained on 1000 points on a dataset of a stachyose trajectory.

The very first screen to greet the user when opening FFAST is a general overview of a model's performance through the visualisation of basic information, as shown in Figure 3.1. This includes a table of MAEs and RMSEs, MAE distributions and MAE timelines for both energies and forces. A glance at the distributions gives the user an immediate idea of the model's general performance. They provide a rough estimate of accuracy across the entire dataset along with rudimentary expectations for the variance within. One can also infer information from the shape of the distributions, such as deviations from a normal curve potentially indicating systematic errors on particular subsets.

The error timelines at the bottom show the prediction errors in the same order as the reference dataset. This is useful for time-ordered datasets such as trajectories taken from molecular dynamics, facilitating the observation of general trends or time snippets during which a given MLFF performed particularly poorly. These particular plots also provide a smoothing factor to separate general behaviour and noise: it does so by performing a sliding average of a given window size over the timeline.

Note that most plots here can be the source of sub-datasets. Practically, a user can toggle the respective button (yellow icon on the toolbar) and a new dataset will automatically be created that dynamically filters out all points not currently present in the zoomed-in area of the original plot. This can be particularly useful to analyse e.g. peaks in the error timeline.

3.2.2 Outlier detection

Finding outliers in a dataset is a difficult and loosely defined task. Average errors over entire datasets are helpful, but they don't provide real indications of the types of configurations an FF fails at. Yet this is important information to gather, as the dataset might contain vastly different geometries and perhaps even unphysical outliers that need to be identified. FFAST provides a series of tools to help with that, this section concentrating on two specific ones.

Correlation scatter plots like the one showcased in Figure 3.2 are a simple way to visualise large deviations from the norm on a point-by-point basis. This is done

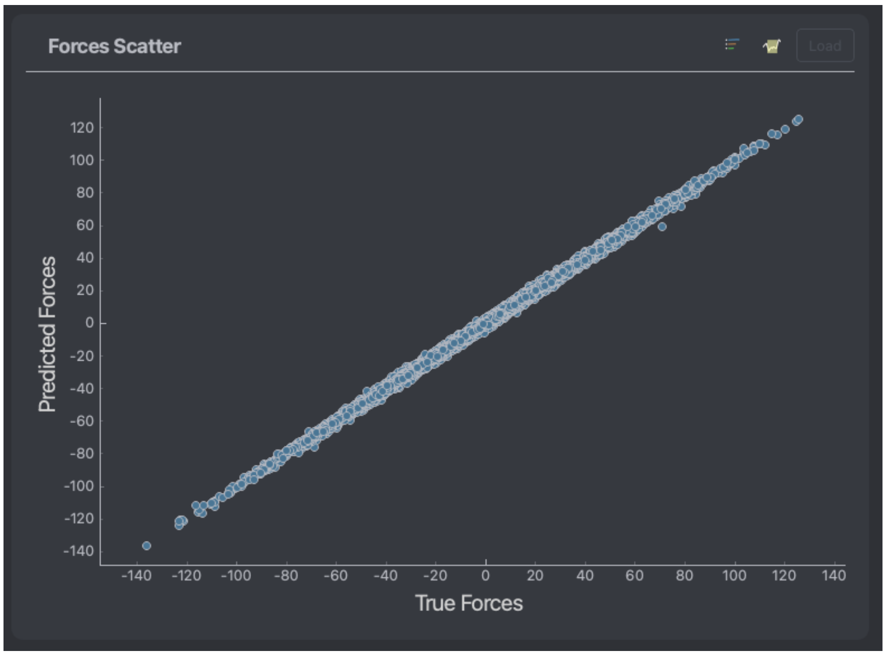

FIGURE 3.2: Example of a scatter plot in FFAST, showcasing predicted forces against their true value. Predictions performed by a Nequip model trained on 1000 points on a dataset of a stachyose trajectory.

by plotting predicted values against the real reference. In the case of forces, every dimension of every atomic force is treated as an independent point. For a perfect model, a linear correlation is expected. Thus, the width of the distributions gives a general indication of the stability of the model while points with strong deviations from the gross behaviour encourage further investigation.

An alternative way to visualise deviations from normal behaviour is through the cluster errors already introduced in section 4.4. In contrast to the scatter plots, the error distribution across clusters describes the performance of representative groups of configurations as opposed to singular points. In essence, these plots provide a middle-ground between error distributions and error scatters. The resolution is finer than that of error distributions by giving more weight to rarer types of geometries, yet broader than correlation scatters by forcing configurations into groups. The user can use this tool along with intuition on the chemistry of the system at hand to try and infer why certain clusters fail more than others as well as what this means for the applicability of the model.

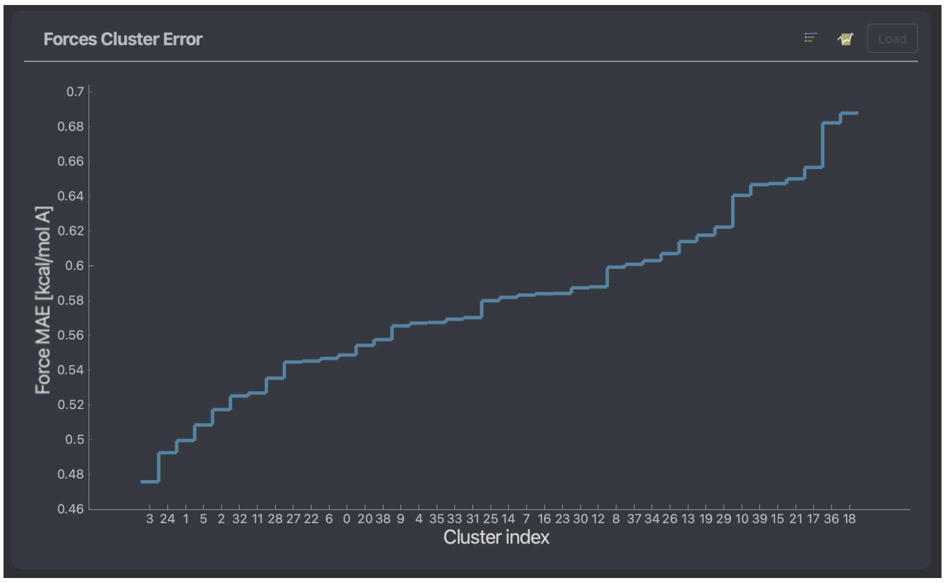

FIGURE 3.3: Example of a cluster error plot in FFAST, showcasing force prediction MAEs on different clusters. Predictions performed by a Nequip model trained on 1000 points on a dataset of a stachyose trajectory.

3.2.3 Atomic errors

The outlier detection methods described above focus on individual points or groups of points inside a dataset. Here, we instead turn our attention back to metrics spanning the entire dataset but we distinguish between different atom types. This becomes particularly important as chemical compositions and structural complexities of the system grow, as interactions between atoms strongly depend on atom environments.

For most ML model applications, predictions will vary across atom types. As such, FFAST provides a way to plot error distributions per atom type, with the user being able to select which elements to include as well as compare them to the overall error distribution. An example is given in Figure 3.4. This particular analysis can provide a window into the inner workings of the ML models, as various architectures treat the inclusion of atom types differently. As such, two models with similar overall performances can easily show noticeably different error distribution curves over atomic types. Furthermore, datasets might contain mechanisms with a distinct effect on atomic error distributions that are otherwise

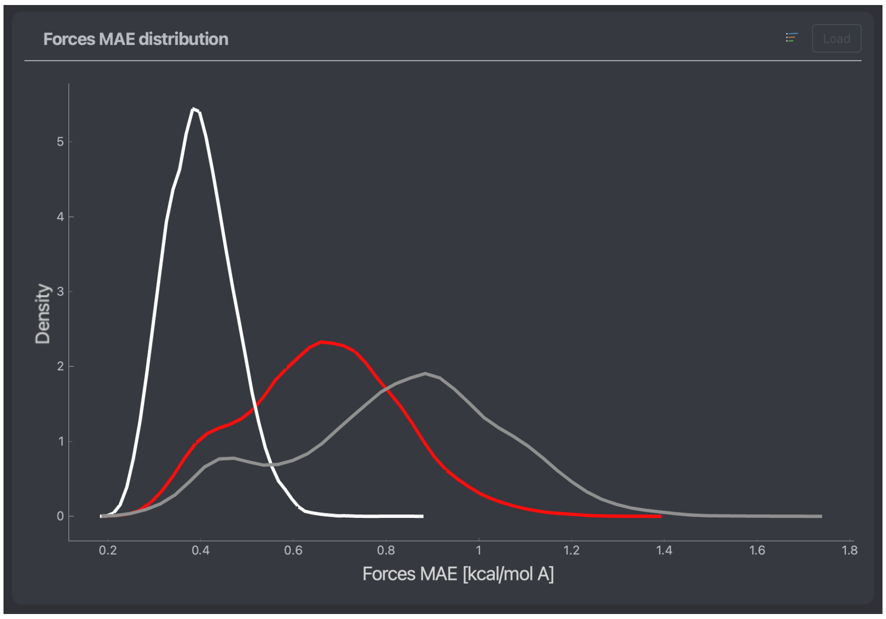

FIGURE 3.4: Example of an atomic error distribution plot in FFAST, showcasing force predictions for different atom types: Hydrogen (white), Oxygen (red), and Carbon (grey). Predictions performed by a Nequip model trained on 1000 points on a dataset of a stachyose trajectory.

overshadowed in the overall plot.

3.2.4 3D Visualisation

A large part of human intuition comes from abstracting concepts in physical space. Chemistry is a pioneer of this effect as notions like atoms and bonds themselves are an example of this. As such, it is no surprise that chemical intuition in large relies on the visualisation of the molecules in 3D space. FFAST allows any dataset and sub-dataset to be visualised in a separate window to accompany all analysis on the go. Furthermore, as most plots can generate subplots, the 3D visualiser can be used to qualitatively identify subregions such as problematic clusters and correlation plot outliers.

The visualiser comes with its own set of features, namely the ability to colour every atom by its respective force prediction error, see Figure 3.5 (right). When viewing small subsets, e.g. encompassing a reaction or pathway of interest, it might also be useful to colour the atoms by their displacement. This quality of life

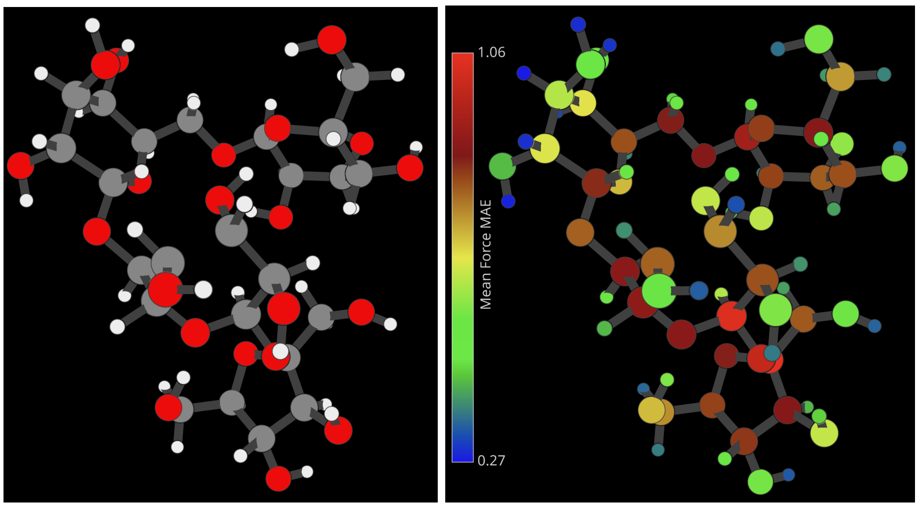

FIGURE 3.5: Example of a stachyose molecule in FFAST's 3D visualiser. Atoms are coloured by element (left) or mean average atomic force error (right). Predictions performed by a Nequip model trained on 1000 points on a dataset of a stachyose trajectory.

feature makes it easier to identify which atoms contribute to a movement without the need to watch the transformation. Furthermore, a vector field corresponding to the instantaneous forces applied to each atom can also be visualised to provide more insight into what is driving the movement at each timestep. Finally, if a particular set of atoms stood out in any of the previous analyses, the interface allows the user to select them in the 3D visualiser to create a sub-dataset containing these atoms only.

3.3 Applications

After demonstrating most of the main features present in FFAST, the following section will shift towards real applications to two state-of-the-art ML models: Nequip and MACE. To showcase the level of detail that FFAST is capable of, we analyse the ability of the models to reproduce the PESs and FFs of two flexible molecules from the MD22 dataset: stachyose and docosahexaenoic acid (DHA). Both architectures here are easily capable of reaching chemical accuracy on a test dataset for both molecules, but that does not guarantee stable long-time dynamics.

Thus, understanding the origins of the prediction errors in detail is paramount for future applications and improvements.

3.3.1 Stachyose

Stachyose ($C_{24}H_{42}O_{21}$) consists of a total of 87 atoms and appears as one of the most common tetrasaccharides in plants and as is such an important sugar [138]. This section focuses on the performance of a Nequip model on the dataset first introduced in section 3.2. Unlike the latter section, the focus here is on the analysis of the dataset and model at hand, as opposed to a general overview of a typical FFAST workflow.

First, there are noteworthy trends revealed by the basic error screen in Figure 3.1. According to the energy MAE distribution, the chosen model predicts almost all of the configurations inside the dataset with chemical accuracy (1 $kcal/mol$) or better. The table reveals that the overall energy MAE hovers around 0.89 $kcal/mol$. This behaviour is mimicked by the force MAE distribution and the corresponding overall MAE of 0.58 $kcal/(mol\text{Å})$. Unlike its energy counterpart, the force MAE distribution demonstrates a secondary peak, generally indicative that some distinct system components are not equally well reconstructed.

The force and energy MAE timelines show an oscillatory movement. This hints at the existence of a distinct set of visited configurations, meaning that the molecular dynamics at the heart of the dataset did not simply stay at a single equilibrium state but rather explored qualitatively different regions of CS.

We now shift our focus back to finding the source of the secondary peak in the error distribution. To this end, the atomic error plot in Figure 3.4 is particularly useful as it shows a clear difference between hydrogen and other elements. Naturally, as hydrogens make up almost half of the entire molecule, overall metrics such as the force distribution are highly impacted by this deceptively good accuracy. While this explains to a large extent the double-peak behaviour, this does not explain why the same trend is seen for carbon and oxygen to a lesser extent.

Stachyose is composed of a total of four carbohydrate rings: three 6-membered ones (pyranoses) and one 5-membered (furanose). Figure 3.5 shows the molecular structure of stachyose, with atom colours indicating the chemical element (left) or the respective atomic force prediction error averaged over all reference configurations (right). From the latter visualisation, one can derive that not all carbons are created equal: those involved in or neighbour to a glycosidic bond (connecting two rings) are notably more difficult for the model to predict than the others (by a factor of 18%). Similarly, oxygens inside those bonds also demonstrate a higher error than e.g. oxygens on side chains (by a factor of 56%). Table 3.1 summarises the prediction errors by atom types. In practice, a model's performance is largely tied to its weakest link rather than its average accuracy. As such, one could argue that this model's MAE and RMSE are closer to those presented by the worst predicted atoms: no longer chemical accuracy. This reflection warrants the consideration to differentiate not simply between atomic elements, but also include information on their local chemical environment, as has been done for employing empirical force fields on chemically diverse systems [139].

	H	C	C_b	C_r	C_s	O	O_b	O_r	O_s	All
MAE	0.40	0.83	0.92	0.78	0.77	0.66	0.89	0.82	0.57	0.58
RMSE	0.55	1.10	1.23	1.04	1.02	0.90	1.20	1.10	0.76	0.82

TABLE 3.1: Overall force MAE and RMSE in $kcal/(mol\,\text{Å})$ for different atom types in the stachyose dataset. Subscripts b, r, and s correspond to atoms touching a glycosidic bond, inside the rest of the ring, or in a side chain, respectively. Predictions performed by a Nequip model trained on 1000 points on a dataset of a stachyose trajectory.

Finally, the above process can be repeated for a different model, e.g. a MACE model trained on the same training set. However, almost equivalent trends are observed for this model as well, reinforcing the idea that the variance between atom types is a consequence of the different chemical environments rather than an artefact due to the choice of model. Nevertheless, Figure 3.6 shows the atomic force distributions as computed by the MACE model. Note that here, the different types of carbon atoms are more finely visible in the distribution.

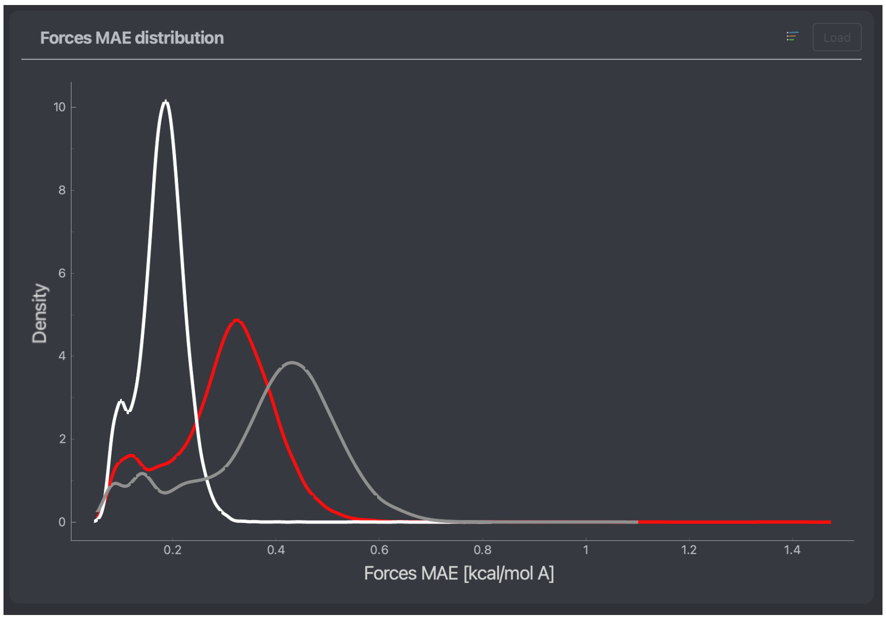

FIGURE 3.6: Atomic error distribution of force prediction for different atom types: Hydrogen (white), Oxygen (red), and Carbon (grey). Predictions performed by a MACE model trained on 1000 points on a dataset of a stachyose trajectory.

3.3.2 Docosahexaenoic acid

In this section, we pivot to a different molecule from the MD22 dataset: that of docosahexaenoic acid (DHA, $C_{22}H_{32}O_2$). This molecule is a fatty acid with a total of 56 atoms and includes a long, flexible hydrocarbon tail of 22 atoms connected to a carboxylic head. Due to its flexibility, it is natural to expect that many different states exist from unfolded to folded even at ambient conditions. In order to trust an MLFF to handle such a molecule well in practice, it is crucial to verify how it handles the various geometries.

FFAST provides a simple but effective tool to help track folding and unfolding processes inside of a dataset. This does not require the loading of any ML model but simply computes the gyration radius (gyradius) as shown in Figure 3.7. We can conclude that throughout the trajectory, the molecule goes through a total of six extended and compact states. The figure also conveniently shows the potential energy at every step, allowing us to assess that it correlates well with the gyradius past an equilibration period of 10k simulation steps.

However, the correlation is imperfect at a few particular places (e.g. at 40k steps). Despite the molecule's compact state, the potential energy is increased. This is a non-trivial effect demanding further investigation. Interestingly, it will later be seen that these particular sets of configurations also tend to demonstrate the largest energy MAE as well as sizeable force MAEs for both ML models used in this subsection.

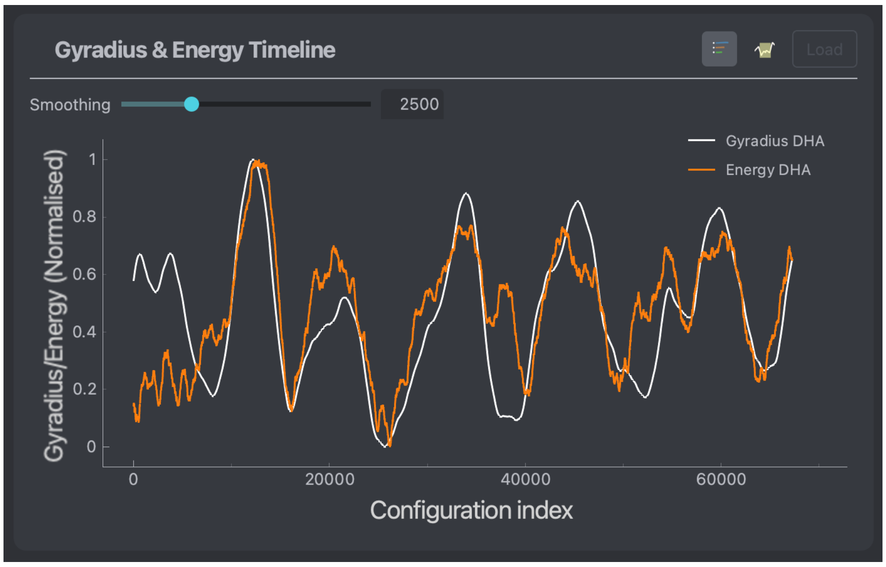

FIGURE 3.7: Gyradius (white) and potential energy (orange) throughout the DHA dataset. All values are averaged over 2500 neighbouring points to smooth out noise.

Before moving to the ML model performances, it is fruitful to ensure the chosen training sets are adequate. This is particularly important due to the flexibility of the molecule. A swift analysis can be done using FFAST: in particular, we compare the distributions of gyradius, forces and potential energies of the full dataset to those of the training set (1000 points). Figure 3.8 reveals that the training set is well-representative of the dataset in all three cases. While this does not ensure the training set is perfectly adequate (see chapter 5), it still reveals that there are no glaring pitfalls.

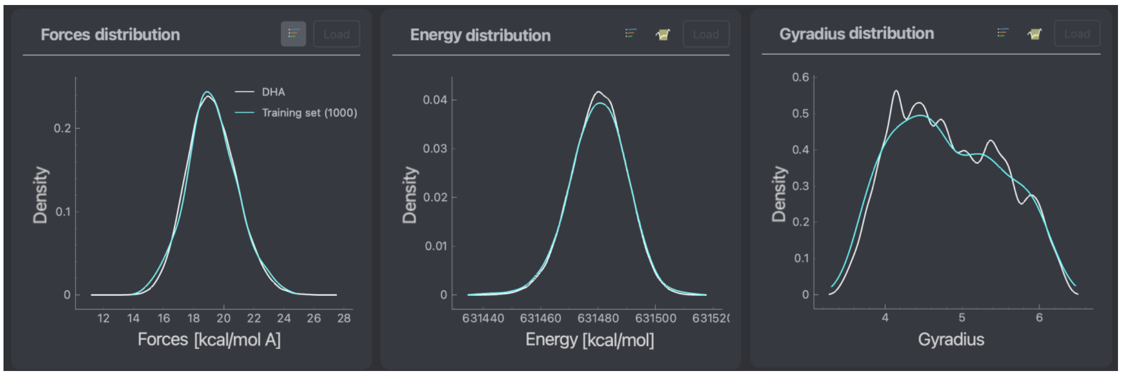

FIGURE 3.8: Force distribution (left), energy distribution (middle) and gyradius distribution (right) comparing an entire DHA dataset (white) to a selected training subset of 1000 points (cyan). Training set selection performed using the sGDML package.

With preliminary trust in the training set, two state-of-the-art models (Nequip and MACE) were trained and compared. Below, a comparison of the performance of both models is done. First and foremost, Figure 3.9 shows the basic error analysis for both models at the same time. It is immediately clear that the Nequip model (orange) has worse overall accuracy than the MACE model (red), with respective force prediction MAEs of 0.20 $kcal/(mol\ \text{Å})$ and 0.33 $kcal/(mol\ \text{Å})$.

Beyond overall errors, one can observe that the error distribution peaks for the MACE model are narrower, both for forces and energies. A quick look at atomic errors (not shown here, see Figure 3.4 for an example on stachyose) reveals that this decrease in prediction error does not depend on atomic types: MACE performs about 50% better for hydrogens, oxygens and carbons alike. Thus, one can conclude that the MACE architecture is indeed better able to reconstruct the PES of this particular molecule.

Nevertheless, both models show synchronised peaks at key points of the trajectory. As both models present the same qualitative behaviour, it is unlikely that this effect is caused by artefacts of the models themselves but is rather due to fundamental changes in geometrical composition. The peaks show a loosely

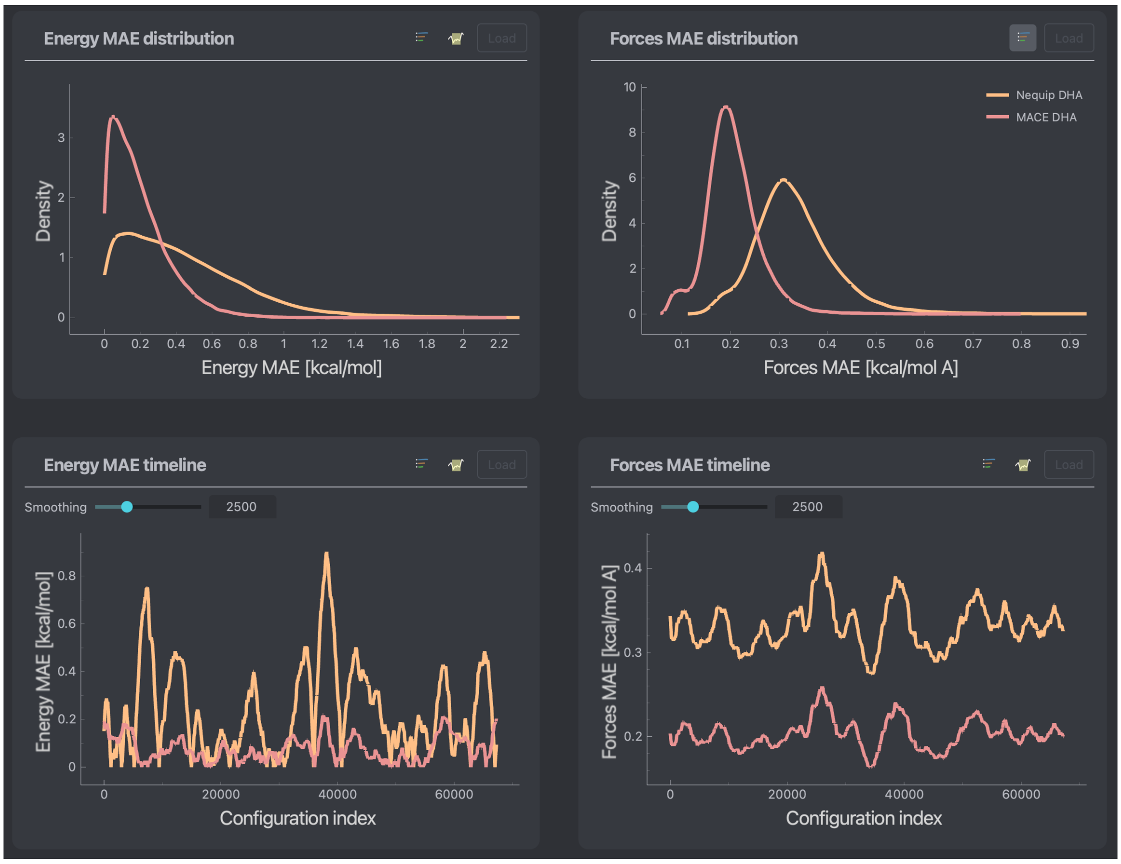

FIGURE 3.9: Basic error screen in FFAST, showcasing timelines and distributions for energy and force prediction MAEs. Predictions performed by a Nequip model (orange) and MACE model (red) trained on 1000 identical points of a DHA dataset.

periodic behaviour and upon further inspection using the 3D visualiser, it is revealed that the valleys (low error) contain extended geometries while the peaks (high error) correspond to folded geometries. An example for each configuration is shown in Figure 3.10.

Similarly, the force prediction cluster errors tell a similar story. Overall, the curves are relatively flat: there is less than a factor of 2 difference between the worst and best cluster, see Figure 3.11. Nevertheless, the trend that associates high errors with coiled configurations continues as indicated by the 3D visualisation of an example taken from the lowest-error cluster and highest-error cluster respectively, see Figure 3.10.

Additional information can be extracted from the 3D visualiser when showing the average force prediction error per atom, see Figure 3.12. Unsurprisingly, it is

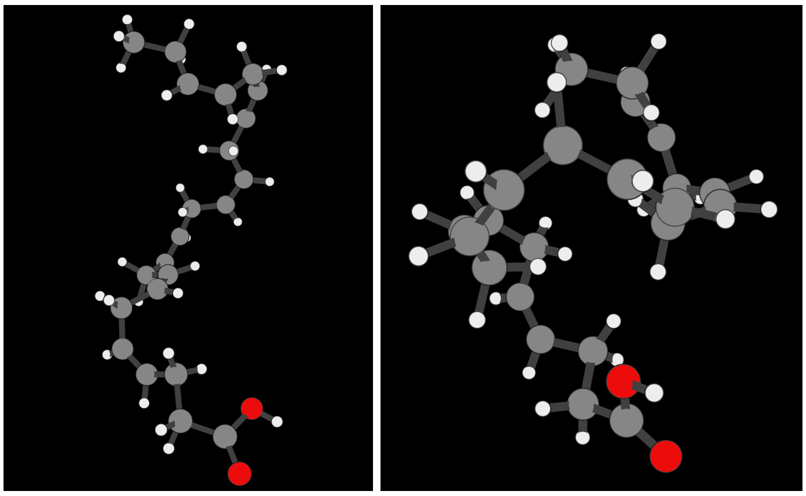
FIGURE 3.10: Example of an extended DHA molecule (left) and a folded DHA molecule (right), found in low-error valleys and high-error peaks respectively. Extended geometry picked from the lowest force prediction error cluster and folded geometry picked from the highest error cluster.

clear the hydrogens are very well predicted compared to the rest of the chain. Thus, as was the case for the stachyose molecule in section 3.3.1, a large merit of the low overall prediction errors is attributed solely to 32 hydrogen atoms in the molecule. Beyond the hydrogen, there is also a notable difference between carbons near the carboxylic head (left side of the figure) and carbons in the middle of the chain or at the tail end of it. Note that while the figure below shows errors as predicted by the MACE model, the same behaviour is observed for Nequip.

The above observations of non-uniform force reconstruction throughout the chain can easily be explained by applying physical and ML intuition to the results shown in FFAST. Any carbon atom involved in or touching the carboxylic head is exposed to a significantly more complex chemical local environment. The descriptor space for a more complex neighbourhood is likely to have more varying possible states, which is exasperated in coiled-up configurations. This makes the task of learning this part of the molecule a lot more challenging for our ML models, thus the larger prediction errors.

Once again, it is worth pointing out a possible future direction to combat this issue. The carbons near the carboxylic head and inside the main chain are wildly

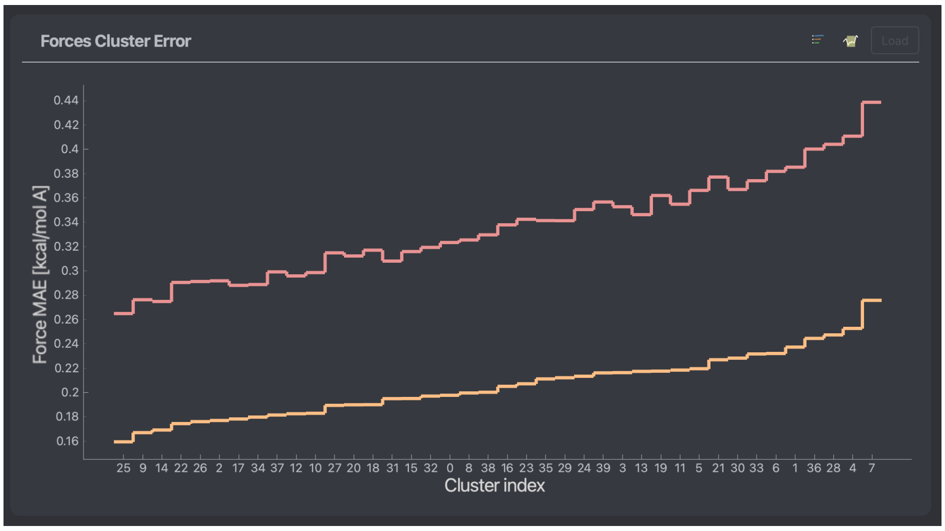

FIGURE 3.11: Force prediction MAEs on different clusters of DHA.Predictions performed by a Nequip model (orange) and MACE model (red) trained on 1000 identical points of a DHA dataset.

different and could benefit from being treated as different atom "types". The usefulness of decomposing a molecule into atomic environments rather than elements would likely also rise for larger, more challenging systems.

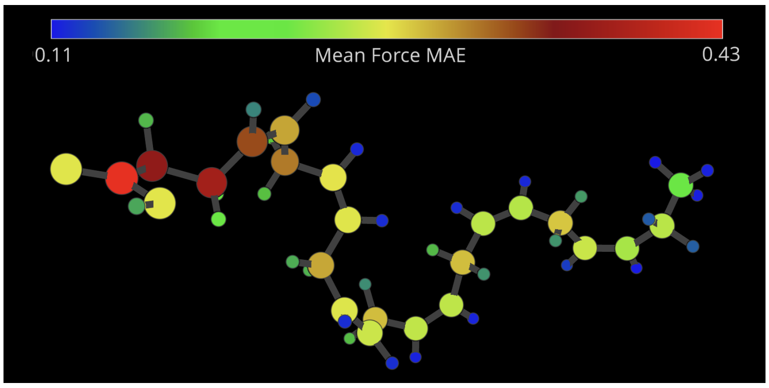

FIGURE 3.12: 3D visualisation of DHA with atoms coloured by mean average atomic force error. Predictions performed by a MACE model trained on 1000 points on a dataset of a DHA trajectory.

All things considered, the analysis performed in this section leads one to believe that both the Nequip and MACE models would perform very similarly in practical applications. All general trends were identical and the models mostly differed in the scale of the errors, with a difference of less than 0.25 $kcal/mol$ in energies and 0.20 $kcal/(mol\,\text{Å})$ in forces.

3.4 Conclusion

As more and more complex systems are tackled by the MLFF community, it becomes a necessity to develop tools able to provide insightful analysis of the created models. Even highly effective MLFF models can easily present unexpectedly high levels of heterogeneity in predictions across CS. This effect is further explored in the following section (4), where clustering algorithms are used to separate different regions of CS. This serves to both analyse models but also re-emphasise the necessity of such analytical tools by studying the similarly skewed prediction patterns of a variety of state-of-the-art MLFF models.

These inhomogeneities are easily missed when only looking at overall error metrics such as MAE and RMSE and instead require the systematic analysis of predictive behaviour across various parts of CS. In this chapter, FFAST was introduced as a way to break the ice in this domain and provide experts and non-experts with a baseline of tools to gain in-depth insight into the performance of an FF. Many of the features were showcased in the example of two medium-sized flexible organic molecules: DHA and stachyose.

This section covered the detailed analysis of the prediction of two state-of-the-art models (Nequip and MACE), though FFAST supports many more modern MLFFs. Throughout the showcase of a general FFAST workflow, it was found that hydrogens are (numerically) much better predicted than other atoms. This has a large influence on the overall MAEs and RMSEs due to both molecules being hydrogen-rich systems. On the other hand, forces on carbons and oxygens are unevenly reproduced, with atoms surrounded by chemically or structurally complex environments presenting a notable increase in prediction error. For DHA, the prediction errors of the main chain steadily rise as the molecule undergoes folding, with the largest source of errors found in the carboxylic head.

FFAST was also shown to provide a way to analyse reference data. It was used to quickly find the number of folding and unfolding processes in the DHA trajectory and it was found that these processes generally correlate with the molecule's potential energy. Overall, FFAST offers a detailed analysis of both the dataset as well as the model, allowing field-specific knowledge of the system at hand to be used to both assess the model's quality and enhance it after identification of the main pitfalls.

Chapter 4

Study of inhomogeneous predictions across configurational space

More and more branches of science increasingly incorporate a computational side to their workflow, from simple scripting and analysis to wide-scale simulations generating copious amounts of data, ripe for interpretation. The field of computational chemistry is one of the many fields to have become a data-generating behemoth. As computational resources grow, so do the durations of molecular dynamics and their system sizes. The advent of powerful statistical models further pushes it further by leveraging large amounts of generated information to train data-hungry predictive models.

The usefulness of data, however, is limited by its interpretability. Before using the bounteous heaps of information provided by e.g. a molecular dynamic, one first needs to inquire about the content of the data, what it is potentially missing and whether it is representative of all prospective use cases. For small simulations and systems, intuition tends to be a powerful tool to answer those very questions. Such trajectories can be viewed in real time and a scientist with enough chemical knowledge can quickly assess quality, scope and limitations.

Long simulations or simulations of large/flexible systems tend to be significantly more challenging to analyse. Besides, it is often necessary to know in advance what mechanisms or configurations one is looking for to find them. However, identifying unexpected chemical phenomena or outright mistakes hidden in millions of rows of data is not feasible without additional help. Prior knowledge of the system at hand and the smart identification of the important collective variables is often an effective way to start gaining insight into a dataset and the model trained on it. This allows visualising prediction errors for qualitatively different parts of e.g. a transition between states, thereby revealing inhomogeneities in the predictive power of a model.

However, many systems are too complex to be reasonably summarised by a few collective variables. Furthermore, the identification of such parameters can often be highly non-trivial. Clustering is one of the tools one can utilise when faced with this challenge. Its primary task is breaking down a large corpus of data into smaller, digestible groups without knowing in advance what those groups should look like. In principle, this does not require any prior knowledge of the system at hand or even the content of the dataset, thus being a much simpler and more "automatic" procedure with a large number of potential applications.

In this chapter, the idea of identifying different regions of CS and their considerations for analysing the performance of MLFFs is explored. In section 4.1 we introduce the basics of clustering before showing instances of specific implementations in section 4.4.1. This is followed up by a brief discussion of descriptors and metrics in section 4.3 before arriving at the work done on MLFFs using clustering methods (section 4.4). Finally, section 4.5 discusses more chemically inspired ways to split up a dataset when prior knowledge about the system is available.

4.1 Basics

The fundamental idea of clustering algorithms is to group unlabelled data points such that similarity between points of the same group is maximised. The concept of similarity is broad and necessitates additional definitions, but in most cases, it boils down to choosing a distance d in a given space. More specifically, we are given a dataset $X = \boldsymbol{x}_i$ with $i \in [1, N]$ comprising a total of N samples $\boldsymbol{x}_i$. We want to find a mapping $c_i = f(\mathbf{x}_i)$ where $c_i \in [1, K]$ are the cluster labels with K the total number of clusters. The function f is chosen such that it minimises the total intra-cluster distances across an entire dataset:

$$\text{Minimise: } \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\delta(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j))d(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{4.1}$$

This expression is never solved analytically, instead, cluster algorithms rely on heuristic approaches that simplify the problem in various ways. In this work, every data point $\boldsymbol{x}_i$ represents a molecular configuration. The specific implementation of how to represent molecular configurations (a.k.a. the descriptor) is its own can of worms and will be touched on in section 4.3. For now, we simply assume that every data point $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iM}]$ where M is the total number of features. This set of features, combined with an appropriate distance in the descriptor space, should enable the clustering method to clump all configurations with qualitatively similar geometries together. If successful, the dataset X, originally a sea of uncategorised molecules, will now be organised into smaller, more interpretable chunks as depicted in Figure 4.1.

4.2 Clustering methods

4.2.1 KMeans

KMeans is by far one of the most popular clustering algorithms, largely owed to its efficiency and simplicity. The algorithm requires the user to define the number of clusters K in advance. Once chosen, the dataset will iteratively create K disjoint clusters C_a, each defined by their respective cluster centroid μ_a. The metric that is minimised is the overall intra-cluster sum of distances ϕ:

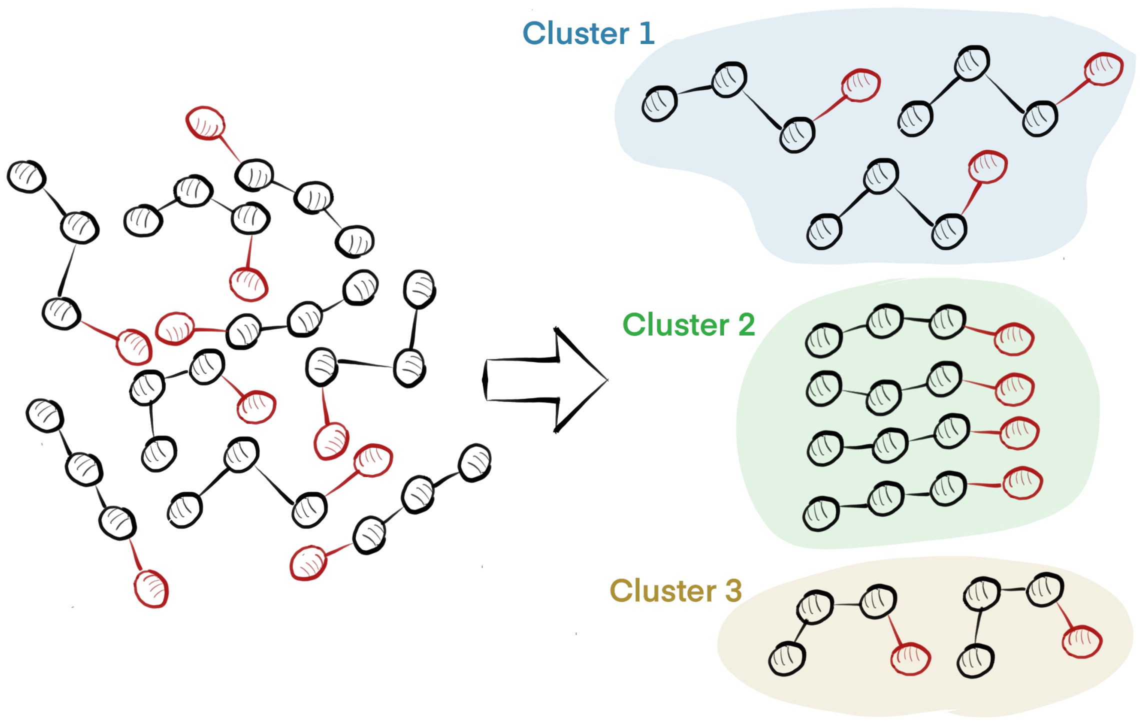

FIGURE 4.1: Illustration of a clustering algorithm applied on a set of unlabelled molecular configurations (left). Resulting clusters represent groups of similar configurations (right).

$$\phi = \sum_{i=1}^{N} \min_{a \in [1,K]} \left(|\boldsymbol{x}_i - \boldsymbol{\mu}_a|^2 \right). \tag{4.2}$$

In this scheme, every point $\boldsymbol{x}$ is attributed to the cluster C_a whose centroid $\boldsymbol{\mu}_a$ is the closest. KMeans uses an iterative approach to reach convergence. Below we outline the most common implementation, known as Lloyd's algorithm, though others also exist [140, 141].

1. Choose initial centroids μ_a in the same space as the data points
2. Associate each point to its closest centroid
3. Recalculate centroid positions as the centre of mass of all its associated points
4. Repeat steps 2 and 3 until convergence

KMeans is guaranteed to converge since every step reduces ϕ (or keeps it the same). If the cluster labels don't change after a step or a maximum amount of steps is reached, the algorithm is stopped.

The outcome of KMeans is highly dependent on the chosen parameter K but also the initial centroid positions μ_a. This is particularly important because KMeans is prone to falling into local minima during its minimisation process. Hence, it is not unusual to do multiple runs with different initialisations to ensure the results are consistent. Furthermore, smart initialisation procedures are often used such as k-means++ [142]. There, the centroids are chosen to ensure faster convergence with better results:

1. Set centroid $\boldsymbol{\mu}_1$ to be equal to a random point $\boldsymbol{x}_i$ in the dataset
2. For an unchosen $\boldsymbol{x}_j$, calculate its distance $D(\boldsymbol{x}_j)$ to the nearest already chosen centroid $\boldsymbol{\mu}_a$
3. Choose a new centroid $\boldsymbol{\mu}_b$ at a data point $\boldsymbol{x}_j$ randomly, but with a probability distribution proportional to $D(\boldsymbol{x}_j)^2$
4. Repeat steps 2 and 3 until K centroids are chosen

Finally, it is important to note that the minimisation of a criterion such as intra-cluster distances inherently favours the creation of similarly sized clusters. In other words, KMeans is well-suited for approximately homogeneous data but suboptimal for identifying rare features or finding outliers in a varied dataset. This point is especially important for applications in the molecular field, as datasets tend to be highly inhomogeneous. Moreover, part of the motivation for using clustering methods in the first place is to spot rare chemical phenomena or configurations. If the latter is of high priority, it is likely other clustering algorithms are more appropriate.

4.2.2 Agglomerative clustering

Unlike KMeans and other similar algorithms, agglomerative clustering does not rely on centroids to define a cluster. Instead, it is entirely described by its members. The base algorithm is very straightforward and intuitive: at first, every single data point is considered to be its own independent cluster. Then, an iterative process starts where the two "most similar" clusters are merged at every step. The final number of clusters K is once again predetermined and the iterations are stopped once only K clusters are left. A schematic representation of the algorithm is shown in Figure 4.2.

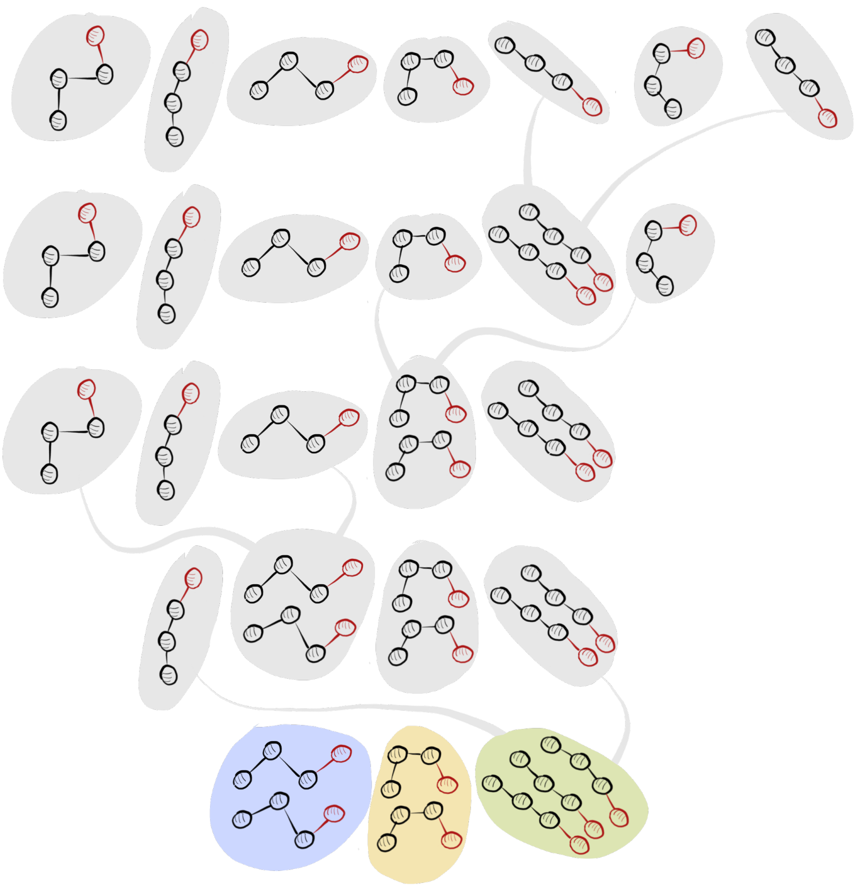

FIGURE 4.2: Schematic representation of the agglomerative clustering algorithm.

The key notion that needs to be defined is the "similarity" between clusters which is given by the **linkage** $l(C_a, C_b)$ where $a, b \in [1, K]$. For a given set of pairwise distances between every pair of points between two clusters, given by the **metric** $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the linkage provides the distance between these two clusters. The metric itself is a simple distance between two points, usually chosen to be the Euclidean distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2$. The linkage has a variety of options:

- **Single linkage**: $l(C_a, C_b) = \min_{i,j} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with $i \in C_a$ and $j \in C_b$.
  Here, the cluster distance is given by the pairwise distance of the closest pair of points from respective clusters. This performs well when non-elliptical cluster shapes are desired but struggles with noisy data or outliers.
- **Complete linkage**: $l(C_a, C_b) = \max_{i,j} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with $i \in C_a$ and $j \in C_b$.
  Here, the cluster distance is given by the pairwise distance of the furthest pair of points from respective clusters. The tendency here is the creation of compact clusters and as such it favours well-separated data.
- **Average linkage**: $l(C_a, C_b) = 2/(|C_a| * (|C_b| - 1)) \sum_{i \in C_a} \sum_{j \in C_b} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$,
  with $|C_a|$ the number of points in cluster C_a. Here, the cluster distance is given by the average pairwise distance of all pairs of points from respective clusters. This choice tends to produce more evenly sized clusters.
- **Ward linkage**
  Ward linkage chooses the pair of clusters to merge such that it minimises the total within-cluster variance. Just like average linkage, it tends to create evenly sized clusters that are compact. Note that this method assumes equal variances across clusters and suffers if this assumption is very wrong.

With the right choice of metric and linkage, agglomerative clustering is not bound to create clusters of similar sizes, as is the case with KMeans. For applications in molecular datasets, this can be a great advantage as it allows the creation of small, independent clusters containing rare events or outliers in the dataset. However, this does come at the price of higher computational times and higher sensitivity to noise.

4.2.3 DBScan

DBScan (Density-Based Spatial Clustering of Applications with Noise) is a popular density-based clustering algorithm. Unlike KMeans, it does not utilise centroids to define clusters but instead relies on detecting regions of high and low density. In essence, the method creates clusters that encompass an area of high density, all separated by regions of low density.

DBScan requires the user to predefine two main parameters: the maximum distance ϵ and the minimum number of samples m. The former dictates how far two points can be while remaining part of the same cluster and the latter determines the minimum amount of points that need to exist inside a region of radius ϵ to be considered a cluster. Every point is determined to be one of three types, see Figure 4.3:

- **Core points** are those that have at least $m - 1$ other points in a region of radius ϵ around them
- **Border points** are those that fall within the region of a core point (same cluster) but are not core points themselves
- **Noise points** don't fall into either category above and remain unlabelled (outliers/noise)

Like most clustering algorithms, the implementation is iterative. One starts by selecting a random point and identifying it as core, border or noise. Then, a new point within the neighbourhood is selected and identified and the loop continues. This repeats until no unassigned points remain within ϵ of any already-assigned point: this completes the creation of the first cluster (so long as it includes at least m points). Then, the algorithm jumps to another random unassigned point and begins anew. The algorithm stops when all points have been visited once.

Unlike KMeans, DBScan does not care about the shape of the cluster and can easily deal with non-elliptical data. It can be closely compared to agglomerative

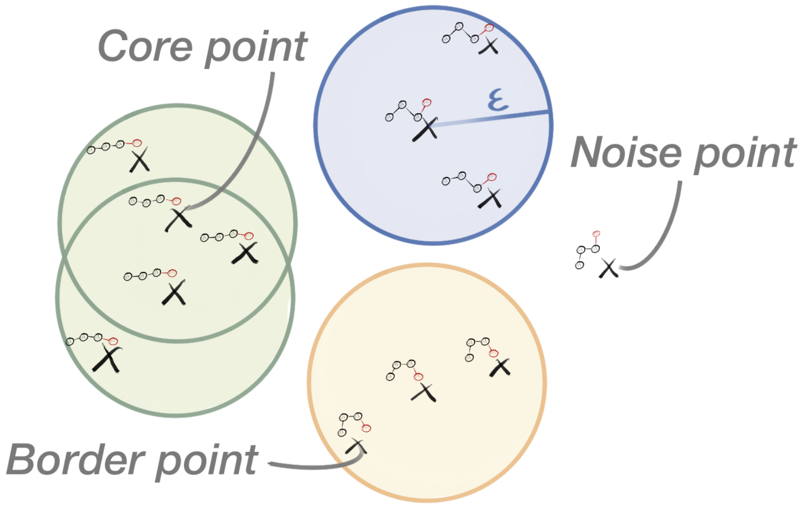

FIGURE 4.3: Schematic representation of the DBScan algorithm, including maximum distance ϵ and an example of a core point, border point and noise point. The drawing uses $m = 3$ minimum samples.

clustering with the single linkage but is better able to separate noise and outliers from main clusters. On the downside, the results are highly dependent on the choice of parameters ϵ and m which can require meticulous tuning to get expected results.

4.3 Descriptors

For all clustering applications, it is assumed that every point in the dataset is represented as a single vector of a given size. However, as we are dealing with molecular configurations, our dataset originally is a tensor of size $(NxMx3)$, where N is the number of samples or configurations and M is the number of atoms. Furthermore, we also know the chemical element of every atom. This section deals with finding a function f to convert the $Mx3$ Cartesian matrix representing atomic positions $P = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_M]$ into a single representative vector $\mathbf{x}$:

$$\mathbf{x} = f(P). \tag{4.3}$$

The importance of choosing function f wisely is not to be underestimated, as different representations can lead to wildly different results both in clustering and prediction tasks for MLFFs. The descriptor should encompass changes in local atomic environments along with long-range changes in molecular configurations. As such, changes in bond distances, angles and dihedrals should have a noticeable effect on the representation as well as global changes in geometry.

One could technically flatten the matrix P into a vector of length $M * 3$, though this runs into the first immediately obvious problem: a translation or rotation of the molecule entirely changes the descriptor $\mathbf{x}$. Naturally, this is undesirable as neither of these transformations change the configuration of the molecule nor does it have any influence on its energy. In other words, our function f has to be invariant with respect to any arbitrary rotation R and translation T in $\mathbb{R}^3$:

$$f(RP + T) = f(P) = \mathbf{x}. \tag{4.4}$$

The most obvious way to comply with these demands is to rely on distances. For example, for a given distance metric d (usually Euclidean), the descriptor could be given as a set of pairwise distances between atoms. Generally, information about the atoms' elements is also taken into consideration, which then results in descriptors such as the widely used Coulomb matrix:

$$M_{ij}^{Coulomb} = \begin{cases} Z_i^{2.4} & \text{for } i = j \\ Z_i Z_j / R_{ij} & \text{for } i \neq j, \end{cases} \tag{4.5}$$

where Z_i is the nuclear charge of atom i. Usually, the lower or upper triangular part of this MxM matrix is flattened to a translationally and rotationally invariant vector describing a molecular structure. Interatomic distances do not change under any translation or rotation of the original space, and they don't cause any loss of information. On the contrary, this description is over-complete, as can easily be seen when observing the dimensionality of the vector $\mathbf{x}$. The problem originally started with $3M - 6$ degrees of freedom when accounting for symmetries. However, there are a total of $M(M-1)/2$ unique pairs of atoms that constitute the new descriptor

x.

In other words, the size of the descriptor scales rapidly with the number of atoms, even when just using distances. This problem is further exasperated when three-body (angles) or four-body (dihedrals) components need to be explicitly added for accurate representations, which quickly results in unmanageable sizes leading to a common problem dubbed the curse of dimensionality [143]. For our clustering purposes, where distances between points (each described by a descriptor $\mathbf{x_i}$) is the fundamental metric everything is based on. One of the direct consequences of high dimensional space is the reduction in information that distances contain. As the number of dimensions grows, pairwise distances between points tend to increase and present a much narrower distribution relative to the breadth of possible values. This effect is showcased in Figure 4.4 using a toy model. In other words, distances tend to become less descriptive for large descriptor sizes, and thus, the quality of clustering algorithms suffers. Note that this applies to Euclidean distances as well as most other common choices.

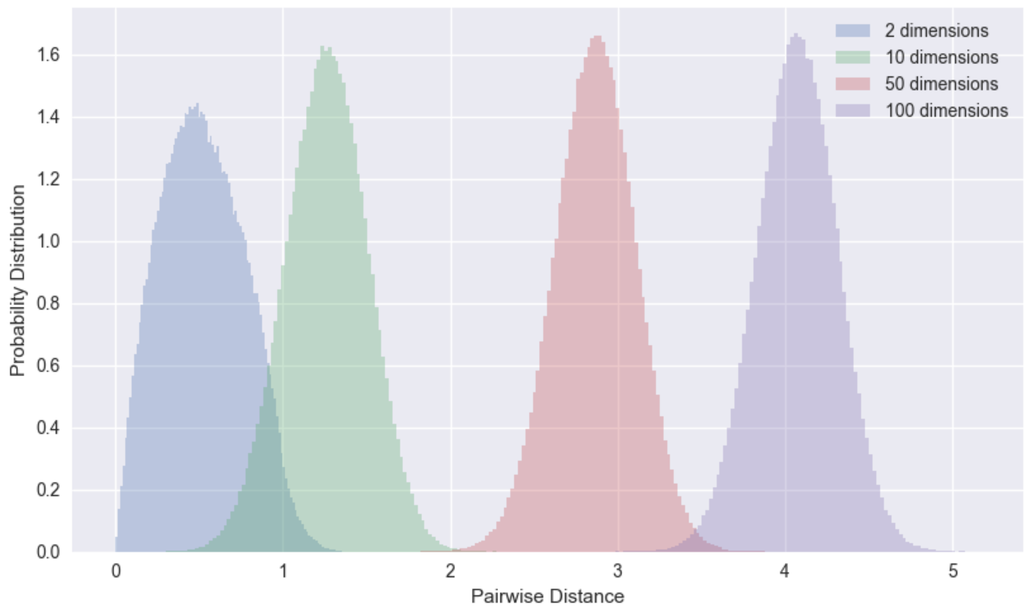

FIGURE 4.4: Probability distribution of pairwise distances of 1000 random points in an n dimensional box, for $n = 2$ (blue), $n = 10$ (green), $n = 50$ (red), $n = 100$ (purple)

Another noteworthy downside of the Coulomb matrix is that —on its own— it is not permutationally invariant. That is, while swapping the order of two atoms of the same element leaves the structure and energy of a system untouched, exchanging two rows (and columns) of the Coulomb matrix has a significant impact on this descriptor. In general, this permutational invariance is a desired trait that the remaining descriptors in this section all maintain. For this particular descriptor, this feature can be re-introduced e.g. by sorting the values or by using the (sorted) eigenvalues of the matrix instead.

Here, we want to recall a discussion in section 2.4 outlining the particular need for intricate descriptors, especially when the goal is to train a shallow NN or kernel method. Clustering algorithms share the same line of thought: the descriptor is not learned or refined by the model and as such its original choice is paramount. If a descriptor doesn't fully describe the system and its symmetries — or is too large and redundant — it will have direct negative consequences on the quality of the resulting clusters.

Among others, the section introduces SOAP as an example of an elaborate descriptor tailored towards the extensive description of molecular systems, see Equation 2.29. The idea revolved around local atomic neighbourhoods, split into a radial part (containing pairwise information) and a many-body part. The former was represented using a predefined set of radial basis functions such as high-order polynomials while the latter was expanded into a set of spherical harmonics. Importantly, all the choices kept the final symmetries of the original system in mind.

More descriptors have been added over time, such as the Many-Body Tensor Representation (MBTR) [49]. MBTR builds on the idea of many-body expansion and the final vector describing the system is a concatenation of $k = 1, 2, 3, 4$ subvectors representing k-body terms. Each subvector is a sum of histograms running over a pre-defined grid of a relevant metric, see Equation 4.6. For example, the pairwise $k = 2$ terms can be represented by the distribution over possible distances x from 0.5 Å to 10 Å.

$$f_k(x, \mathbf{z}) = \sum_{\boldsymbol{i}=1}^{N_a} w_k(\boldsymbol{i}) D(x, g_k(\boldsymbol{i})) \prod_{j=1}^{k} C_{z_j, Z_i}. \tag{4.6}$$

Here, x is the value of the relevant metric g_k at which the distribution function f_k is calculated. The metric g_k depends on the chosen parameter k, where typical choices for $k = 1, 2, 3, 4$ are atomic numbers, (inverse) distances, angles and dihedrals respectively. Furthermore, f_k is calculated separately for every unique set of chemical elements $\mathbf{z}$ of size k. The weighting factor $w_k(\boldsymbol{i})$ ensures convergence of the sum, e.g. for pairwise terms it is usually an exponential decay over distances. The probability distribution D is a smoothing factor (akin to SOAP) to make the descriptor differentiable and less sparse, often chosen to be Gaussian. Finally, N_a is the total number of atoms in the system, Z_i is the element of atom i in the system and $C_{i,j}$ is an element correlation matrix describing the "likeness" of chemical elements. By default, $C_{i,j} = \delta(Z_i, Z_j)$.

MBTR is translationally and rotationally invariant thanks to its choices of metrics g_k, each individually invariant. Permutational invariance is achieved by summing over atomic contributions and the separation of the terms based on unique sets of chemical elements $\mathbf{z}$ treated.

The list of descriptors presented here as well as section 2.4 are not exhaustive and represent a fraction of the work done in the field. Moreover, new descriptors will assuredly be developed in the future, as the growing system sizes and complexity that modern MLFFs are treating will likely necessitate it. Any application in unsupervised learning approaches, such as clustering, will naturally benefit from these developments as well.

4.4 Cluster errors in MLFF

It has been made clear over the recent years that MLFFs are indeed able to capture molecular interactions for a large variety of systems. Amidst those successes, it is important to be reminded of the inescapable downside of ML: the quality of the result is entirely at the mercy of the availability of prime data. Human intuition and reasoning are necessary to understand what is otherwise a meaningless series of numbers, and as such it is up to that human input to distinguish between relevant data, noise, outliers and downright mistakes. In the field of empirical force fields, all parameters are tuned such as to fit what is known or believed to be true intuitively and experimentally. In the domain of MLFF, some of this insight can be injected into e.g. the architecture of the model itself and its parameters, but that is highly dependent on the method at hand.

For instance, many datasets are born from simple molecular dynamic simulations. A trajectory is extracted from that simulation representing a series of data points, each associating a set of Cartesian coordinates for every atom to energy and forces. Due to how these dynamics operate, the majority of the points sampled should be found in or near equilibrium, as represented in Figure 4.5. As statistical models try to minimize an overall error, rare (out-of-equilibrium) configurations can be very poorly predicted without impacting the training significantly. Hence, one can expect such ML models to be unreliable or unpredictable for deployment in any MD simulation where out-of-equilibrium configurations are deemed important. This effect will always have to be considered when the data distributions in the reference dataset differ from those of the target simulation. Examples include studying phase transitions and reaction rates based on data generated from stable phases, studying quantum effects (such as proton transport) while learning on classically generated trajectories, etc.

This section focuses on the aforementioned “unevenness” of machine learning models across configurational space. First, a method based on clustering algorithms is explored to detect this effect without the need to run dynamics. As opposed to most standard approaches, where only average errors across entire datasets are calculated, we instead compute the prediction error of a model on separate regions of the reference data. This approach allows delving into what types of configurations are problematic as well as which are potentially overtrained. Furthermore, this can simultaneously act as a way to detect outliers inside of the dataset itself, discovering e.g. rare events our out-of-equilibrium processes inside a trajectory.

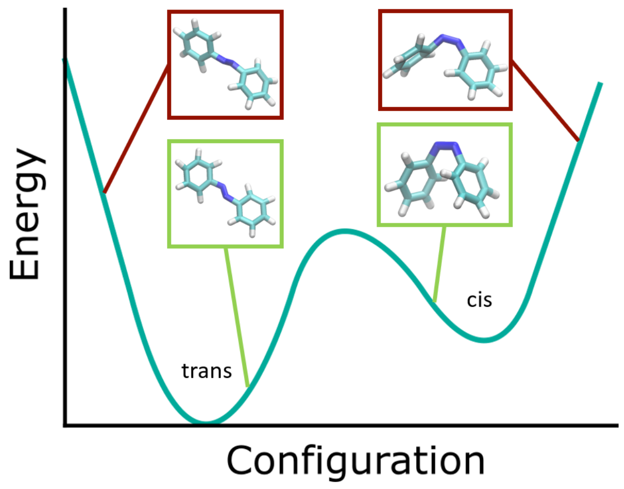

FIGURE 4.5: Hypothetical potential of a salicylic acid molecule presenting two stable energy minima for isomers trans and cis respectively. Configuration in red are naturally poorly sampled whereas those in green are found in abundance, thereby biasing MLFFs.

4.4.1 Methods

This section provides an in-depth view of the error analysis method proposed here. This method's goal is to provide a way to assess an MLFF model's ability to produce equally reliable predictions across all parts of CS of a given dataset, as well as serving as an outlier detection tool. Note that while we apply this method for MLFFs, this method can in principle, easily be extended to any regression problem of choice. A visual representation of the method can be found in Figure 4.6

As a first step, the molecular configurations inside the dataset of interest are represented using a translationally and rotationally invariant descriptor. The choice throughout this section is a pairwise inverse distance descriptor. The reasoning behind this choice lies in the primary application of the method: analysing the error curves of sGDML models. Using the same descriptor as the

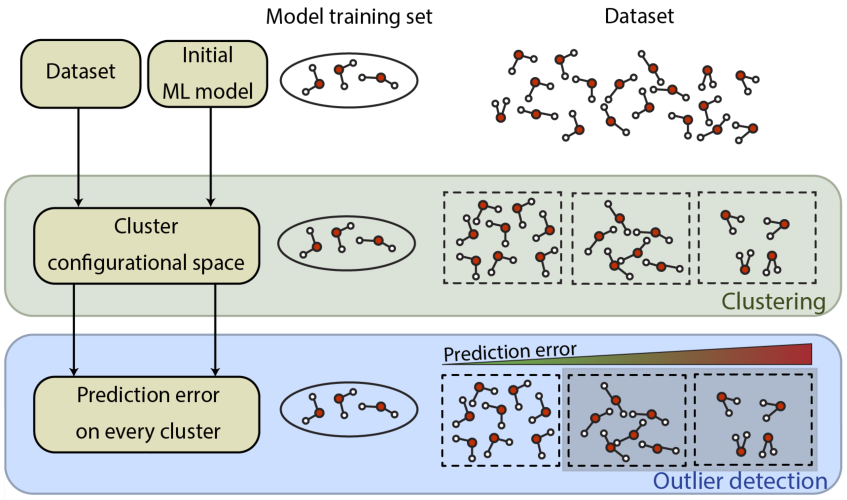

FIGURE 4.6: Overview of the cluster error method. Clusters of a given dataset are generated and prediction errors on MLFFs are calculated on each of them separately.

MLFF to be analysed ensures that configurations with widely different inputs for the model are also well separated in our clustering methods.

Once the descriptor is generated, the clustering step can begin. In this section, we limit the choice of clustering to a single, generally effective method. It consists of first separating the entire dataset into 10 clusters based on the descriptor of geometrical information. This initial step roughly distinguishes between varying configurations throughout the dataset and is in particular meant to identify geometrical outliers. As such, an agglomerative approach is chosen for its ability to create clusters of different sizes. This avoids merging rare but geometrically unique points with otherwise commonly found samples. However, agglomerative clustering is computationally taxing particularly due to the memory requirement that scales as $O(n^3)$, with n the number of samples. As such, for large datasets, only a "small" amount of initial points were able to be fed to the algorithm (usually at least 20000) and unlabelled points were afterwards assigned to their respective closest existing cluster (using the same metric and linkage). To make this step more reliable and less likely to misclassify points, it is preferable if the initial clusters are at least somewhat compact. Thus, a complete linkage was chosen despite it lessening the ability to detect outlier clusters of arbitrary shape.

The metric of choice is Euclidean, which is not a natural choice for the descriptor space. This, combined with the fact that the descriptor space is overcomplete, means that clusters produced based on those choices are prone to containing large variations in potential energy. We bypass this problem with a second round of clustering, this time distinguishing between different energy levels using KMeans with kmeans++ initialisation. Since the potential energy is a simple one-dimensional value, KMeans performs well without any concern for computational costs. In practice, all of the 10 clusters created above are further split into 5 smaller clusters using this method, for a total of 50 final clusters.

Once the splitting of the dataset through geometrical and energetical considerations is complete, one can select an MLFF model of choice and compute its prediction error on each individual cluster. Here, the RMSE was chosen due to its ability to emphasise large differences. When lining up the clusters in ascending order, we can identify outliers and provide the general geometries associated with them.

Note that all methods explained in this section are available in the open-source MLFF software package developed for this purpose [144]. This comes with a slew of options, such as changing the clustering sequences and parameters and more.

4.4.2 Practical application of the cluster error on salicylic acid

This section describes a practical use-case of the outlier detection using an example of a salicylic acid molecule. The dataset in question contains 320000 configurations and is part of the MD17 collection [51]. The level of theory of the reference calculation is done using PBE+TS.

The clustering procedure was performed as outlined in section 4.4.1 resulting in a total of 50 distinct clusters containing qualitatively different configurations. Performing this step using the MLFF software boils down to:

```
python run.py cluster -d <dataset_file>
```

This command provides the cluster indices but a MLFF model is required to perform a proper analysis. For this tutorial, we use an sGDML model trained on 1000 points of this reference dataset using the default training scheme implemented in the package. All 320000 configurations have their forces predicted and grouped by cluster before the cluster force prediction RMSEs are calculated. The simple command to perform this analysis using our package is found below and the resulting image can be seen in Figure 4.7:

```
python run.py cluster_error -d <dataset_file> -i <model_file>
```

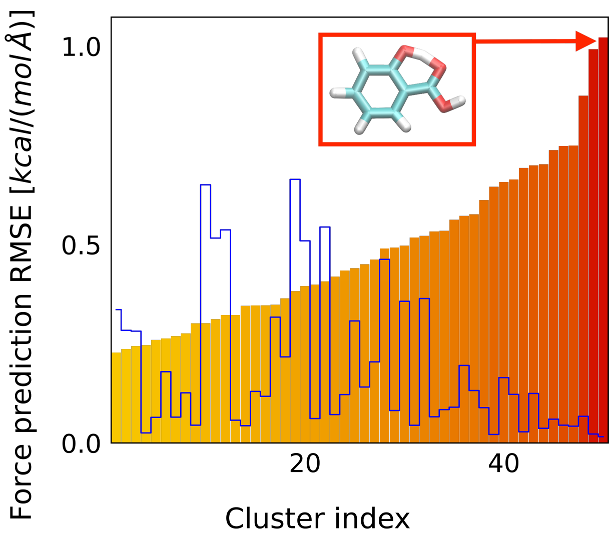

FIGURE 4.7: Force Prediction RMSE (coloured bars) on a total of 50 clusters of an sGDML model trained on 1000 reference data points of a salicylic acid dataset. Clusters are ordered by ascending error. Relative cluster sizes are indicated (solid blue line, arbitrary units). A single representative structure of the worst predicted cluster is shown (red box).

Here, it is important to emphasise that every single cluster corresponds to a set of configurations that are qualitatively different to that of other clusters. In other words, a cluster is a look into a limited region of CS. The poorly predicted clusters thus represent general types of geometry that the model struggles with in particular. One example of such a geometry is shown in Figure 4.7 (red box): one can observe that this particular geometry clearly involves the sharing of a hydrogen atom between the hydroxyl and carboxyl groups of the molecule. This event is quite rare inside of the dataset and it is not obvious that this type of exchange would be included in a trajectory generated this way. While simple visualisation or human analysis of the trajectory might easily miss the inclusion of such an appearance, this method makes it significantly easier to detect interesting phenomena. Note that all the geometries inside of the last cluster presented a similarly shared hydrogen, hence its small population due to the scarcity of this attribute in the dataset (a few hundred among 320000 configurations).

4.4.3 Application on a variety of small organic molecules

A procedure equivalent to the one explored in section 4.4.2 was applied to a variety of state-of-the-art MLFF models as well as small-to-medium-sized datasets of organic molecules. The models in question were sGDML, GAP along with its SOAP descriptor, and SchNet. All of the reference datasets used are part of the MD17 collection, namely uracil, salicylic acid, toluene and ethanol. All MLFF models used identical training datasets generated by the sGDML training set selection scheme. The clustering procedure was performed identically to the previous sections for all four datasets, thus resulting in 50 uniquely different set of configurations per trajectory. The cluster errors are shown in Figure 4.8.

One can immediately see that for all model and dataset combinations, a significant disparity between high error clusters and overall RMSE (black line) is present. In the more extreme cases, this disparity is a factor of up to 3. As such, a single overall error is a very insignificant metric to properly quantify how well an ML model might perform in real tasks since out-of-equilibrium configurations are substantially less stable than “trivial” equilibrium ones. This goes against the idea of replacing *ab initio* calculations with equivalent machine-learned forces, considering potentially important regions of CS present accuracies that seriously differ from reference calculations.

We identify two main possible reasons to explain this occurrence. As a first,

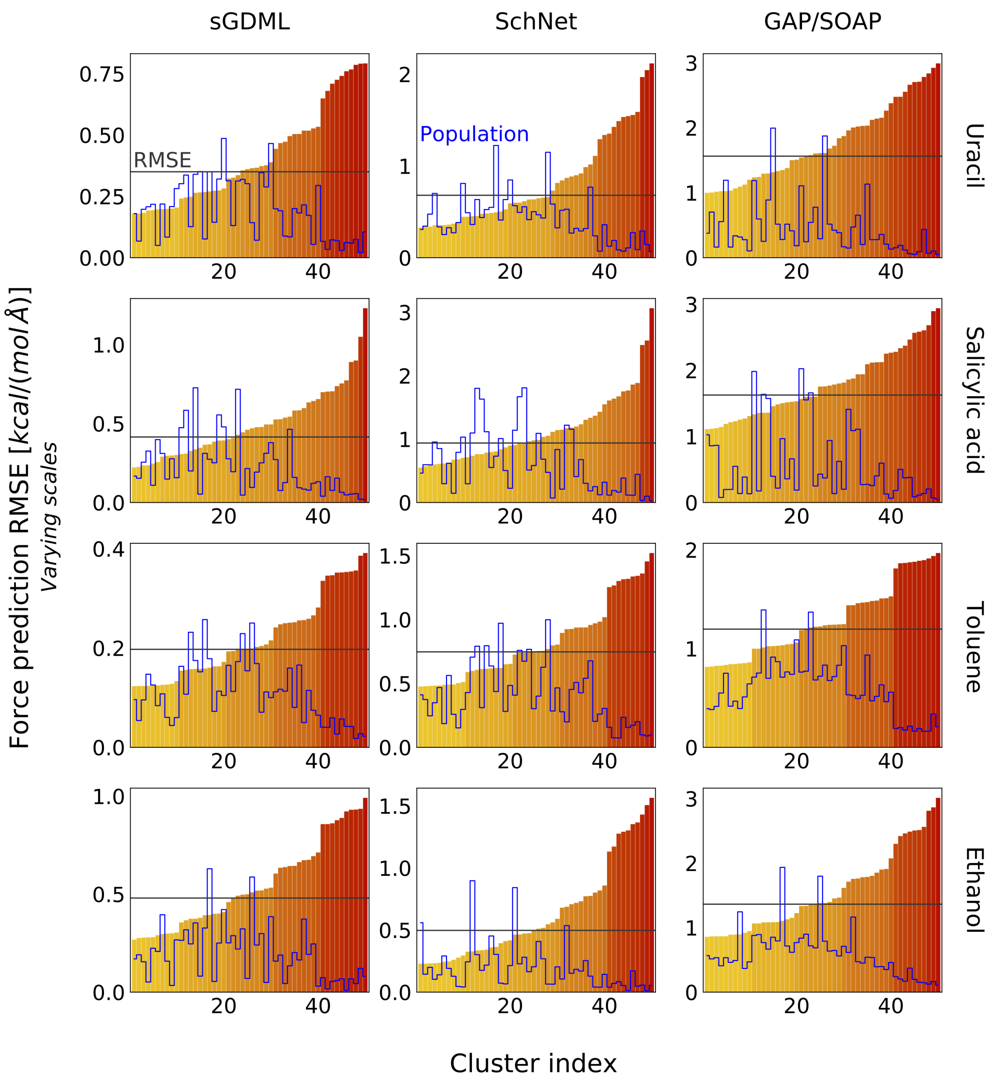

FIGURE 4.8: Force Prediction RMSE (coloured bars) on a total of 50 clusters of sGDML, SchNet and GAP/SOAP models trained on identical 1000 reference data points of uracil, salicylic acid, toluene and ethanol datasets. Clusters are ordered by ascending error. Relative cluster sizes are indicated (solid blue line, arbitrary units). Overall RMSE is also shown (black horizontal line).

clusters with high prediction errors might correspond to regions with large fluctuations in geometry. Those are naturally less well represented in a training set, thus impeding accurate learning through sheer lack of information. Of course, many simple applications might remain unaffected by a poor performance on these outliers due to being constrained to given regions of CS. However, such unpredictable fluctuations in target configurations can greatly affect final results when studying e.g. chemical reactions, configurational changes or stabilities.

A less trivial reason considers the nature of the configurations contained in those clusters. As showcased in the example of section 4.4.2, where the most poorly predicted cluster involved a shared proton between the hydroxyl and carboxyl group, some regions of CS might represent physically or chemically relevant phenomena otherwise missing in the majority of the sampled dataset. In the case of the shared proton, the reference dynamics would require proper accounting of nuclear quantum effects to produce adequate sampling of this effect. Thus, in our case, this exchange happens in a minuscule portion of the entire trajectory (a few hundred out of 320000).

Overall, this points towards the fact that none of the models above would be capable of properly describing a proton-sharing effect. Naturally, this is simply representative of a larger problem not just limited to this particular chemical phenomenon, dataset or choice of models. It is non-trivial to know which configurations of a given molecule can stump the MLFF model in real applications, hence the necessity to automate the process of finding weak points. Furthermore, it is fruitful to consider ways to circumvent this issue, a territory that is explored in chapter 5.

4.5 Challenges for reproducing potential energy surfaces of flexible molecules

The previous chapters focused on the usage of clustering algorithms to separate predictive patterns across different regions of CS and/or different mechanisms inside of a dataset. The key idea was that proper separation of the system's CS provides information that is otherwise hidden in average metrics or distributions.

In this section, which is a brief run-down of a collaborative paper of the same name [145], ML models are also analysed based on their abilities to accurately reproduce different areas of CS. However, the techniques used to segregate said areas do not rely on clustering algorithms but are instead a result of prior knowledge of the system at hand as well as the construction of the datasets themselves. This allows chemical insight and intuition to guide the analysis when the relevant configurational states and/or the transitions between them are known beforehand.

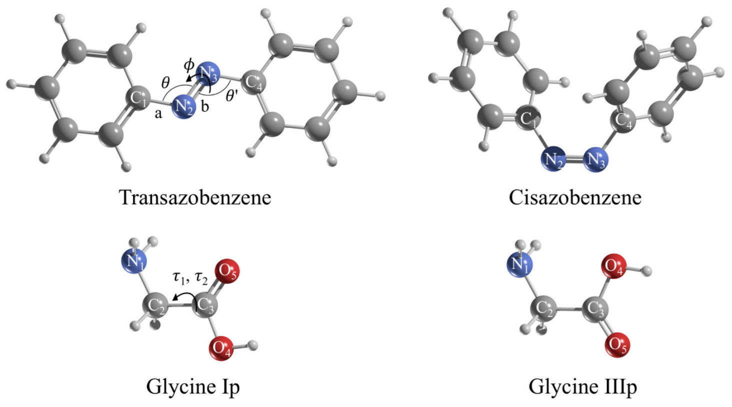

FIGURE 4.9: Optimized geometries at the minima of both isomeric states for glycine and azobenzene. The main degrees of freedom are also shown. For azobenzene, they are the bonds a and b as well as the corresponding bending angles θ and θ' and torsional angle ϕ. For glycine, they are the torsional angles τ_1 and τ_2 [145].

The target datasets of this section are of two molecules — glycine and azobenzene — presenting two distinct isomeric states as shown in Figure 4.9. As both isomeric states are known a priori, the datasets in this section were created to include ample information about both states as well as their transition. Furthermore, as the mechanisms are known, it is clear which states represent equilibrium configurations and which ones are part of a transition, thereby providing “clusters” from the get-go.

For glycine, a transition was created using the string [146] and Nudged Elastic Band (NEB) [65] method providing similar pathways. The mechanism can be described by almost equal rotations of torsional angles τ_1 and τ_2. As the minimum energy path was less than 3 $kcal/mol$ the transformation could be observed in standard constant-temperature MD simulations of 500 K with a timestep of 1 fs using the PBE+TS method. Thus, the dataset is a total of 5000 configurations directly sampled from the MD simulation.

For azobenzene, neither NEB nor the string methods can converge to reasonable transition paths between cis and trans. Thus, the transition pathway was instead constructed manually through a combination of a rotation path (via torsional angle ϕ), an inversion path (via bending of θ and θ') and a path combining both rotation and inversion. Every path was manually constructed via linear interpolation of the relevant angles and consisted of 15 intermediate geometries linking the starting and ending minima. For each intermediate geometry, separate MD simulations were run (using PBE+TS) to provide sufficient sampling of the transition paths. First, this includes a long run at 300 K and a time step of 1 fs to provide a basis for the dataset. Then, an additional high-temperature (750 K) run simulates the cooling-down process of the respective transition states back to equilibrium states, which requires high kinetic energies. Finally, a low-temperature (50 K) simulation with a short time step (0.025 fs) starting from configurations at the middle of both transitions was run to include slow changes of the relevant degrees of freedom. All in all, the rotation and inversion datasets contain 26455 and 25528 data points respectively.

4.5.1 MLFF accuracy on transition paths

A variety of ML models were trained on the reference datasets of both glycine and azobenzene, namely BPNN, SchNet, GAP/SOAP and sGDML. All the models were trained on up to 1000 points chosen by the sGDML training point selection method. In all cases, models were created using a 5-fold cross-validation task on a subset consisting 5 times larger than the final training set size (i.e. 5000 configurations for a model training set size of 1000). Thus, for each training set size, 5 different models were trained on different training points. The respective training sets were selected such as to preserve the energy distribution of the whole dataset. For each model, the remaining 4 folds of the subset were used as a test set.

The performance of the different models on the various transition paths can be seen in Figure 4.10. Note that only the best model out of the aforementioned cross-validation task is shown. For sGDML, two different descriptors were considered: the default inverse pairwise distances [1/r] as well as an extended descriptor that also takes into account bonded angles and dihedrals in the form $D_\Theta = (1 - e^{-\Theta})^2$ and $D_\Phi = 1 + \cos\Phi$ respectively. Only the best of both descriptors is shown for every figure.

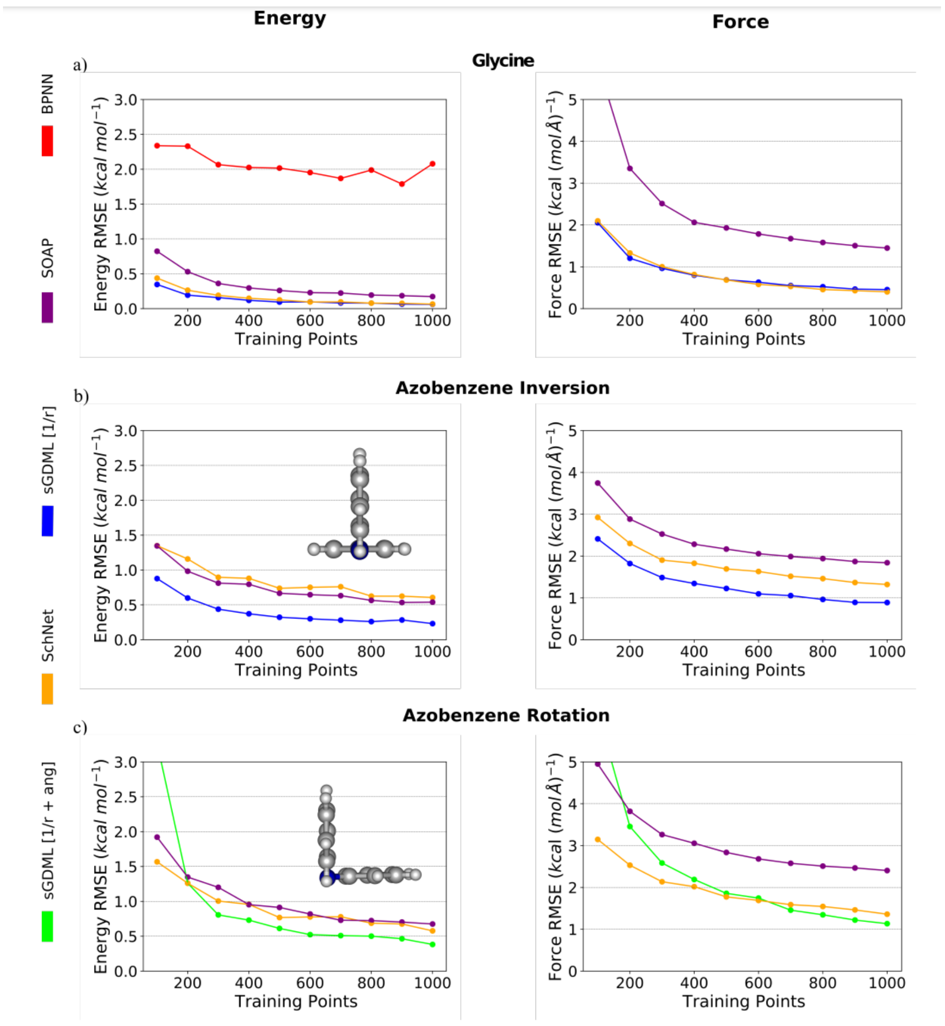

FIGURE 4.10: RMSE of energy *kcal*/*mol* and force *kcal*/(*mol* Å) predictions as a function of training set size for a) glycine, b) azobenzene inversion and c) azobenzene rotation datasets. For rotation, an extended sGDML descriptor [1/r+ang] is shown along the default descriptor [1/r]. Models with errors above 5.0 *kcal*/(*mol* Å) or 3.0 *kcal*/*mol* are omitted [145].

One observes that for glycine, all models besides BPNN obtain chemical accuracy. For sGDML and SchNet, errors below 1 $kcal/mol$ and 1 $kcal/(mol\,\text{Å})$ are reached at 300 and 400 training points respectively. For GAP/SOAP, the force errors remain at around 1.5 $kcal/(mol\,\text{Å})$ even at 1000 training points, though the energy is equally well predicted.

For azobenzene, the different models show very different behaviour. BPNN is not able to meaningfully reproduce this dataset and GAP/SOAP reaches chemical accuracy in energies at 400 training points for the rotation mechanism but only 200 training points for the inversion mechanism. Force prediction accuracies are significantly worse, with the rotation mechanism in particular remaining above 2.4 $kcal/(mol\,\text{Å})$ at 1000 training points.

Both transition mechanisms are very well predicted by both sGDML and SchNet. One can however notice that sGDML is more reliable at predicting forces (force RMSE of 1.1 $kcal/(mol\,\text{Å})$ as opposed to 1.4 $kcal/(mol\,\text{Å})$ at 1000 training points). On the flip side, SchNet can reproduce inversion and rotation equally well with a single descriptor, as opposed to sGDML where significant differences can be found when using either descriptor for the respective mechanisms.

In part, the different predictions across models and mechanisms can be attributed to different long-range interactions. For example, SOAP largely learns local information (within a given cutoff radius). While this is efficient, it is unable to reproduce long-range interactions, unlike the global descriptor of sGDML or the self-interaction layers of SchNet. Thus, it particularly fails to accurately reproduce the forces in the inversion mechanism.

4.5.2 Challenges for ML models in flexible molecules

While it is clear that GAP/SOAP, sGDML and SchNet are viable methods to learn the PES of azobenzene at chemical accuracy, the variance between their predictions concerning the specific mechanism deserves further thought. Below we briefly discuss the two main sources of this effect: imperfections of the training set selection and intrinsic limitations of the descriptors.

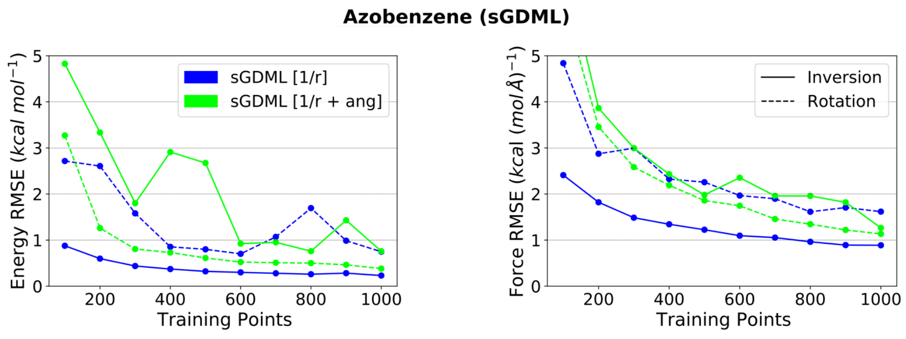

FIGURE 4.11: RMSE of energy ($kcal/mol$) and force ($kcal/(mol\,\text{Å})$) predictions as a function of training set size for azobenzene inversion and rotation datasets. An extended sGDML descriptor [1/r+ang] is shown along the default descriptor [1/r] [145].

Figure 4.11 shows the force and energy prediction error curves as a function of training set size on the inversion and rotation mechanisms of azobenzene. The model of choice is sGDML, showcasing both the default and the extended descriptor for both mechanisms. One can indeed see that while the inversion mechanism is much better predicted using the default sGDML descriptor, an extension using angles and dihedrals is beneficial for the rotation mechanism.

One can also notice that there are considerable oscillations in the energy prediction errors, namely for the inversion mechanism using the extended descriptor and the rotation mechanism using the default descriptor. This is a crucial point, as it shows the potential sensitivity of a model to the particular selection of training points. Note that this effect would be even more noticeable if the training points were selected randomly, if cross-validation had not been performed, or if the molecule was larger and more flexible (thereby increasing the size of CS).

The training set dependency is further influenced by how sGDML specifically treats energy calculations. Regularization of the model is done via two hyperparameters, with the main one being the width of the kernel function which is optimised during the validation phase of the model training. As sGDML directly learns forces, energies are recovered up to a constant and thus a second hyperparameter is required to learn the optimal energy shift. However, energies of flexible molecules such as azobenzene are highly degenerate. Thus, although our training set follows the general energy distribution, different parts of the PES are unequally represented. As such, the particular selection of training points often results in an energy shift that is suboptimal for the dataset as a whole. This is why, when comparing the best and the worst models of the rotation mechanisms using the extended descriptor, the average difference in energy RMSE is 0.7 $kcal/mol$ while that of the forces is only 0.2 $kcal/(mol\,\text{Å})$.

Finally, it is interesting to subdivide the pathway of a mechanism into different phases or clusters. In figure 4.12, the inversion mechanism is subdivided into 17 different clusters, each corresponding to a different value of the dihedral angle Φ. Specifically, cluster index 0 corresponds to an interval between 0° and 10° and cluster index 17 corresponds to an interval between 170° and 180°.

Two SchNet models were trained for this purpose, the first one consisting of 1000 training points of the rotation dataset and the second one of 4000 training points of the combination of rotation and inversion datasets. One can see that the prediction errors on close-to-equilibrium configurations (left and right) are up to four times larger than the in-between states. This effect is observed for both training set sizes and is thus unlikely to be due to insufficient sampling, but rather due to intrinsically more complex interactions in those geometries.

4.6 Conclusions

In this chapter, we delved into the inhomogeneities of prediction accuracies when analysed on different regions of CS. Specifically, we leveraged unsupervised ML techniques as well as chemical insights to create different clusters of a given dataset, each representing a group of qualitatively different configurations or mechanisms. The clustering procedures presented are general enough to automatically extract

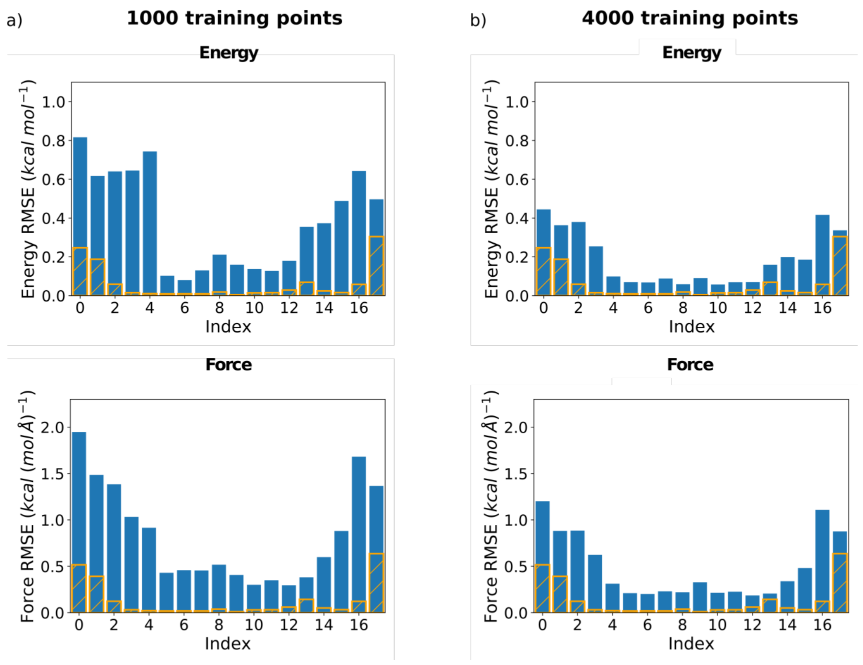

FIGURE 4.12: RMSE of energy ($kcal/mol$) and force ($kcal/(mol\,Å)$) predictions for different clusters of the azobenzene inversion dataset. In a) the best SchNet model trained on 1000 points of the rotation dataset and b) the best Schnet model trained on 4000 points of the rotation and inversion dataset. Clusters correspond to different intervals of the dihedral angle Φ (e.g. cluster 0 includes angles Φ from $0°$ to $10°$). Relative population of each cluster is indicated (orange, arbitrary units).

small subsets of geometries representing nontrivial mechanisms from a large pool of data, even when no prior knowledge of the latter is available.

It was found that state-of-the-art models applied to small to medium-sized organic molecules present non-uniform error distributions across the different regions of CS created through the clustering methods. This confirms that analytical tools such as this one are required to properly assess a model's performance, as overall error metrics are unable to capture these nuances. In practice, these inhomogeneities are linked to instabilities in e.g. molecular dynamics, where the inaccurate reproduction of key mechanisms or geometries can lead to very erroneous results.

A similar story is told when observing the prediction accuracies for different mechanisms of transition paths of flexible molecules, as was done for glycine and azobenzene in this work. This study showed that not only do the prediction errors vary highly from mechanism to mechanism (and model to model), but that different descriptor choices can have varying efficacy for different parts of CS. In particular, it was found that an extension of the default inverse distances descriptor using angles and dihedrals is beneficial for predicting the rotation mechanism with an sGDML model.

Finally, it was highlighted that the choice of training points is crucial to the final performance of a given model. For azobenzene, sGDML models showed oscillatory prediction errors across growing training set sizes. It was also found that for many organic molecules observed in this chapter, some important clusters represent rare configurations that are unlikely to be well represented in a training set, even when the latter reproduces the overall energy distribution. Thus, this work motivates giving the training set selection further thought to potentially enhance a model's stability and prediction errors curve across CS. This idea is further explored in the next chapter.

Chapter 5

Improving model predictions across configurational space

The analyses performed in the previous chapter showed that despite MLFFs' remarkable accuracy, there are tangible downsides to the data-driven nature of the models. Ensuring stability, performance and efficiency hinges on the ability to first provide adequate initial data. While ML approaches themselves are unbiased by nature, the ways to generate reference data highly influence what is ultimately learned. For example, a lot of training data is extracted from molecular dynamics trajectories. An ML model of choice is then trained to reproduce this data accurately: in particular, they minimise the *overall* prediction error across a (usually random) selection of points. This however teaches the model to focus on the common configurations (i.e. close-to-equilibrium) at the expense of rarer configurations (out-of-equilibrium) that don't majorly impact overall error metrics.

A major bias towards well-frequented regions of CS renders ML models unreliable and unpredictable in long or out-of-equilibrium simulations. Specifically, a single mispredicted configuration can send the trajectory into unrecoverable extrapolation regions. This becomes likely to occur for long simulations or examples such as computing reaction rates, exploring nuclear quantum effects such as proton transports or simulating phase transitions. In all those specialised cases, it is important to not only sample pivotal configurations but also ensure they have a meaningful contribution to the model's loss function.

An overall error across a randomly selected training set does not fulfil those conditions.

This chapter largely focuses on the problems discussed above by flattening the error curve across regions of CS. In other words, the goal is to make sure MLFFs are equally reliable for common structures as well as rare (but important) events regardless of their prevalence in the reference dataset. The methods proposed are completely general and can be applied to any ML model of choice, thus enhancing its stability for any use case.

5.1 Optimised training set selection

5.1.1 Method

In order to accomplish the goal of providing reliable MLFFs across all geometries inside a dataset, we propose a novel method that optimises the training set of ML models. With proper application, this leads to "fairer" FFs with significantly reduced bias and thus a more constant accuracy across the reference dataset. The method is available online as open-source software [144] focused on MLFFs, but the approach is generally applicable to any ML model with uneven data generation. A schematic overview of the procedure is given in Figure 5.1. Section 5.1.2 covers practical procedures and implementation details at the hand of an example of salicylic acid.

While standard training techniques simply calculate averages across batches of an entire dataset, we follow the principles introduced in section 4.4.1 and introduce prediction errors across different clusters (representing distinct areas of CS). In particular, section 4.4.3 proposes two reasons to explain particularly high prediction errors in specific clusters. First, the cluster might contain physical and chemical properties that deviate from the norm. Secondly, the cluster might

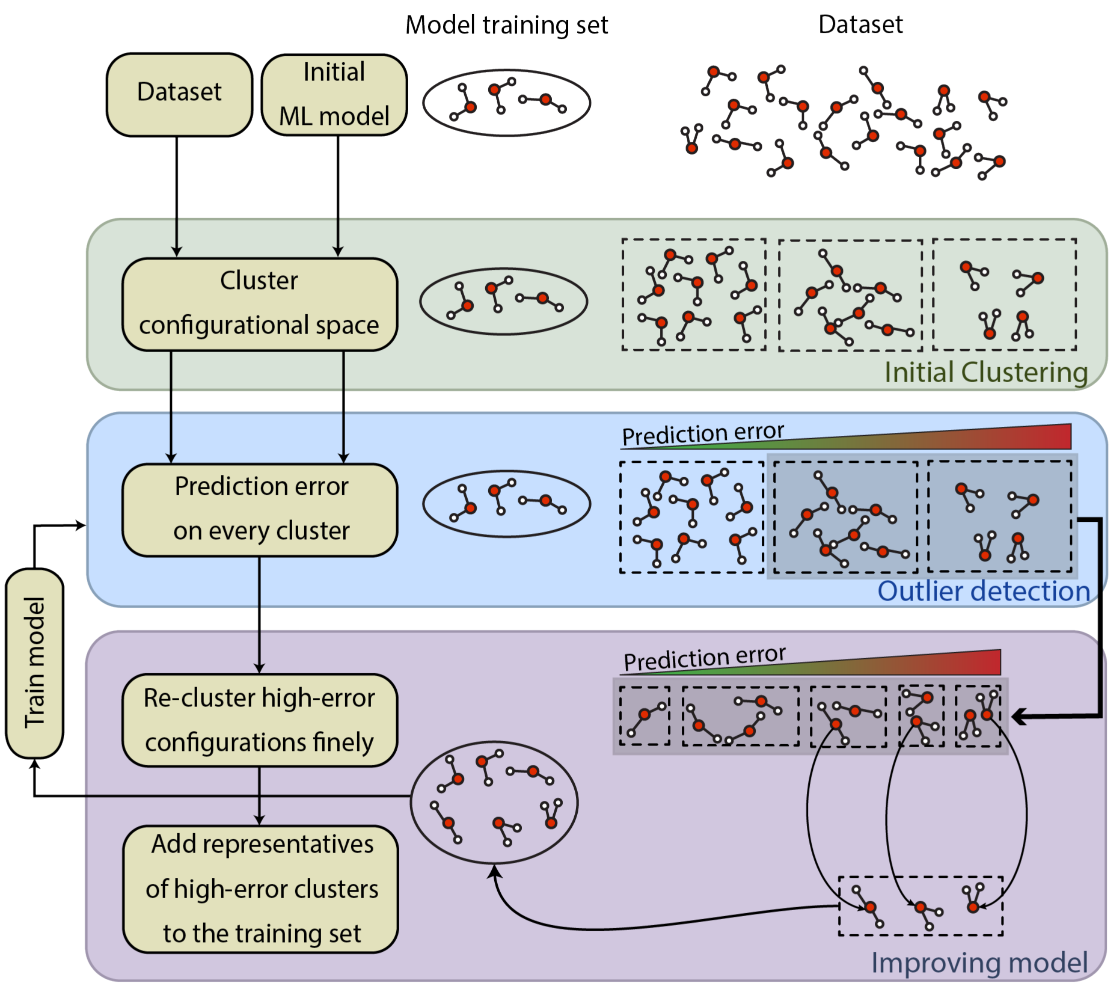
Model training set
Dataset
Dataset
Initial ML model
Cluster configurational space
Initial Clustering
Prediction error
Prediction error on every cluster
Outlier detection
Train model
Prediction error
Re-cluster high-error configurations finely
Add representatives of high-error clusters to the training set
Improving model

FIGURE 5.1: Overview of improved training set selection method. Clusters of a given dataset are generated and prediction errors on MLFFs are calculated on each of them separately. Problematic clusters are re-clustered finely and problematic configurations are added to an initial training set. The procedure repeats several times.

represent a region that is poorly represented in the dataset. The latter is what this section aims to address.

To do that, a model with a reduced training set is trained and its cluster prediction errors are computed. Then, all poorly predicted areas of a model are merged for a larger amount of new clusters to be created. This essentially provides a fine grid of problematic regions, enabling us to pick out well-predicted configurations that the initial broader clustering scheme had mislabelled. Furthermore, the poorly predicted clusters are represented on a fine scale, thereby minimising the risk of missing important nuances in out-of-equilibrium configurations. From the numerous clusters, the most poorly predicted ones are once again set aside and representative geometries are extracted from each. Every extracted geometry is then added to the initial training set, thereby improving performance specifically on the difficult sets of geometries. The process is repeated with the new model until a final model is achieved with an optimised training set, allowing the model to produce similarly accurate results across all data.

5.1.2 Implementation details

Below is a list of all the primary parameters of the improved training set selection scheme:

1. The number of initial clusters and algorithm choices
2. The number of fine clusters and algorithm choices on the iterative step
3. The number of initial training points
4. The number of training points extracted and added at every step
5. The total number of iterations (or number of total training points)

The first set of parameters is implemented exactly as described in section 4.4.1: 10 agglomerative clusters on inverse distances followed up by 5 KMeans clusters on energies for a total of 50 initial clusters. The number of fine clusters is purposely set to a larger number ($100-200$), as the goal is to create a fine grid rather than general configurations. Also, distinguishing between energies is unnecessary as clusters are small enough for energetic values within single groups to be more consistent. In general, it was empirically determined that a good number of fine clusters is twice the number of training points extracted at every step (parameter 4).

Parameters 3-5 are largely interdependent, as the model's planned final size influences the number of initial training points as well as the number of iterations and step size. For the initial number of training points, the resulting model must be at least marginally able to give acceptable predictions. If the initial model's predictions are too far from the truth, the prediction errors inside the clusters cannot be trusted to be a relevant metric for determining valuable training points. For our purposes, it was found that choosing an initial training set of around 20% of the final model's was high enough to provide adequate initial results while still providing enough headroom for improvement. Similarly, a generally good number of the steps was found to be 8 as it constitutes a good compromise: if the number is too low, too many of the initial model's problematic structures are added to the training set, thereby preventing the addition of other problematic structures. If the number is too high, the training time is significantly increased with diminishing benefits. A rundown of some parameter choices and their effects on the final model is explored at the end of section 5.5 with a salicylic acid example.

Specifically, for a run with an initial training set size of 200 with 8 steps of 100 points each, the simple command in the MLFF package boils down to (parameters not present in the command are provided through a parameter file with argument `-p`:

```
python run.py train -d <dataset_file> -n 8 -i 200 -s 100
```

5.2 Application to salicylic acid and a proton exchange mechanism

5.2.1 Flattening the error curve

This procedure was applied on a salicylic acid dataset, as introduced in section 4.4.2. The parameters were set to the defaults introduced above: 200 initial points with iterative steps of 100 points each. In Figure 5.2, we show the cluster error for the force predictions on the improved models for a total of 400, 700, and 1000 training points (i.e. 2, 5 and 8 iterative steps). This is compared to a model trained with the default sGDML procedure to select training points.

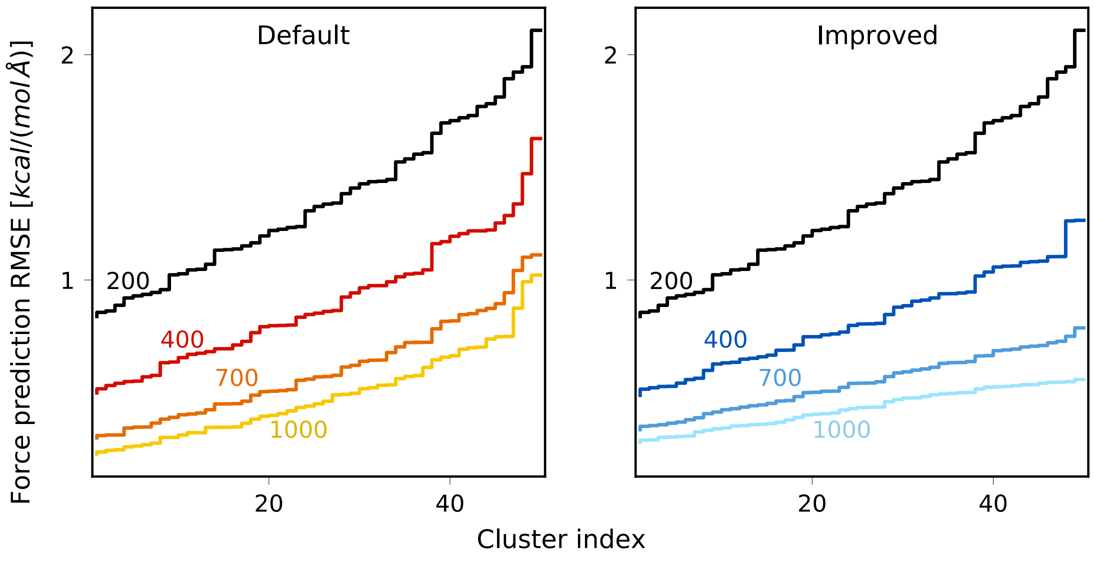

FIGURE 5.2: Force prediction RMSE on the 50 initial clusters (x-axis) of a salicylic acid dataset ordered by ascending error. Every colour corresponds to a different final training set size (or number of iterations) of an sGDML model. We compare two cases: default sGDML training method (left) and the proposed improved training set selection (right).

The differences between the default and improved models are clear to see throughout the training curves: the right-side region (i.e. high-error clusters) flattens significantly more for the improved models over time, indicating a greater ability to ensure a more equally distributed accuracy throughout CS. Note that we concentrate on the force predictions due to forces being a direct actor in the equations of motion, as such their errors are directly tied to potential deviations from an expected reference trajectory. If one were to compute e.g. mean energies at low temperatures or similar properties that are well described by equilibrium configurations, the default models would be expected to perform as well as the improved ones. However, processes involving either broad regions of the PES or underrepresented regions of CS would be noticeably better represented by the models.

5.2.2 Proton exchange

In order to illustrate this effect, we turn our attention to the mechanism pointed out in section 4.4.2: the proton sharing between the carboxyl and hydroxyl group in salicylic acid. The cluster errors determined some of the worst predicted clusters involved almost exclusively in this very mechanism, leading us to conclude that default models would be significantly less adequate at describing this effect. To showcase this, we created an artificial dataset where the hydrogen in question was linearly moved between the two chemical groups. The reference energies for this new dataset were calculated and compared to predictions of four models of 1000 training points: a default and improved sGDML model (see Figure 5.2), and a default and improved SchNet model. The parameters for the training of the models are shown at the start of section 5.3. The settings for the calculations of the reference energies were equivalent to those of the original dataset's generation (PBE+TS, light). The results are shown in Figure 5.3.

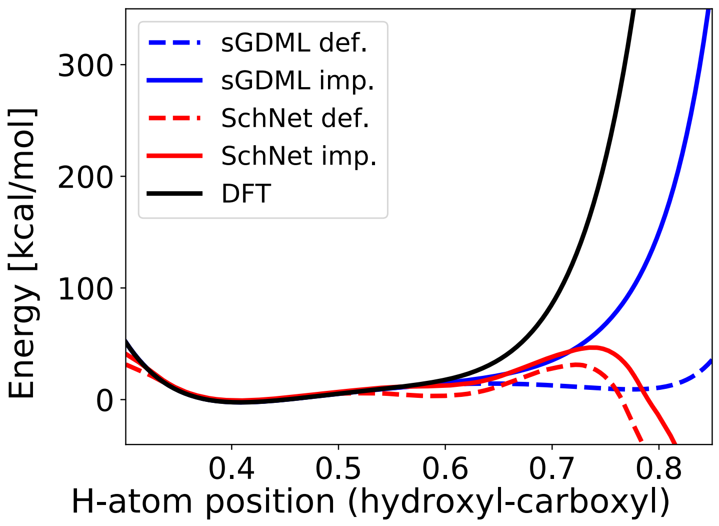

FIGURE 5.3: Energies (y-axis) of a salicylic acid molecule with a shared hydrogen between the hydroxyl and carboxyl group (x-axis, 0 = on hydroxyl oxygen, 1 = on carboxyl oxygen). The re-calculated reference values (black, solid) are compared to a default sGDML model (blue, dashed), an improved sGDML model (blue, solid), a default SchNet model (red, dashed) and an improved SchNet model (red, solid).

It is important to know that all values beyond 0.45 are considered to be an extrapolation since a negligible amount of such data points are found inside the reference dataset that all the models were trained on (as well as values below 0.38). As such, this plot is not a measure of accuracy as much as stability: a model's capacity to not stray too far from the truth when facing completely new configurations is one of the necessary factors to ensure reliable results in out-of-equilibrium situations.

We observe that the default SchNet model immediately deviates from the expected behaviour. In order to move the hydrogen from the hydroxyl group to the carboxyl group, the model's predictions present a shallow second minimum, followed by a weak energy barrier towards an unphysical steep descent in energy. On the flip side, the improved SchNet model shows no shallow additional minimum and the right-side energy barrier starts earlier and is more pronounced than its default counterpart. On the side of the sGDML models, the default model once again presents a second minimum, except significantly closer to the carboxyl group. Conversely, the improved training set provides the other sGDML model with enough information to qualitatively reproduce the reference energy curve, albeit still with an appreciably delayed onset for the steep energy increase. Nevertheless, in both cases, the improved models show more reasonable behaviour across the extrapolation region. This could have a significant impact in real applications, as both default models cause the existence of a metastable state with a shared proton. This would, for example, lead to qualitatively wrong results from imaginary time path-integral MD simulations, while the removal of this artefact by improving the training set renders such simulations reliable.

5.3 Application to a variety of small organic molecules

In this section, the improved training method was applied to a variety of small organic molecules from the MD17 dataset [51]. The default models and reference datasets are exactly the same as those already introduced in section 4.4.3. A comparison of all the models to their improved counterparts is shown in Figure 5.4

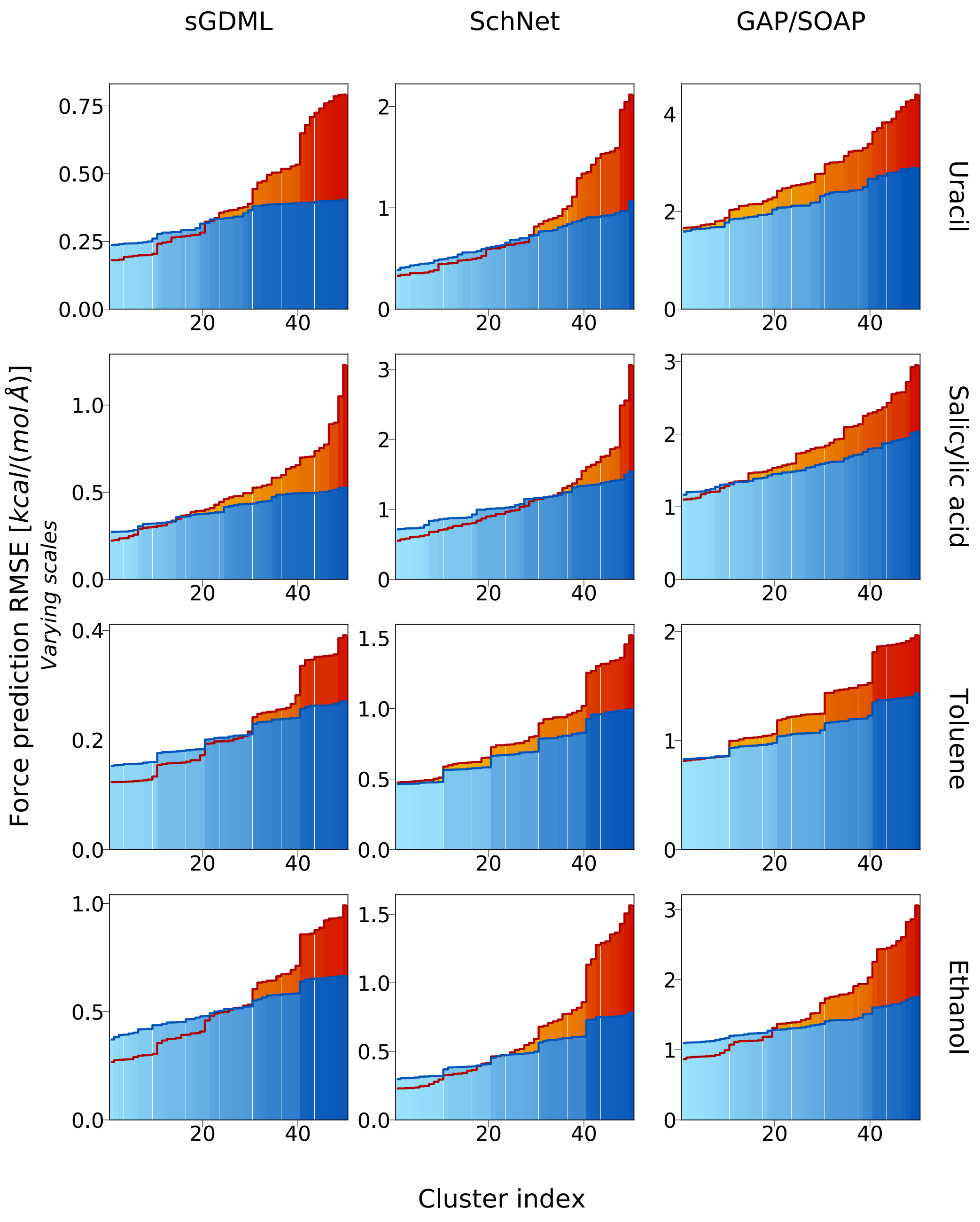

FIGURE 5.4: Force Prediction RMSE on a total of 50 clusters of sGDML, SchNet and GAP/SOAP models trained on identical 1000 reference data points of uracil, salicylic acid, toluene and ethanol datasets. Clusters are ordered by ascending error. The default models (orange bars) are compared to the models with improved training sets (blue bars).

One can observe that for all combinations, there is a significant difference between the improved and default learning curves. This highlights the generality of the method, as it proves effective for various choices of datasets and models. Here, we also want to highlight the stability of our method thanks to the reliance on clusters. Being able to group up similar configurations allows us to asses prediction errors fairly reliably without the need to compute every single point. Indeed, the GAP/SOAP models in Figure 5.4 were trained in such a way that only 1% of every cluster's points was predicted at every step. This applies both to initial clusters as well as fine clusters. Nevertheless, there is still a distinct flattening of prediction errors across the different regions.

Finally, it is important to be reminded this flattening and the resulting model stability is the ultimate goal of the method. For a fixed number of training points, this explicit inclusion of rarer configurations comes at the expense of common configurations whose influence on overall metrics is greater. Thus, this work did not aim to improve overall RMSE; on the contrary, one would be led to believe that overall metrics should worsen after this procedure. Nevertheless, while the differences are quite small, the overall RMSE actually sees a decrease for the majority of the molecules in this study, see Table 5.1.

TABLE 5.1: Overall force prediction RMSE for sGDML, SchNet and GAP/SOAP models, comparing default and improved training methods. All numbers are given in $kcal/(mol\ \text{Å})$

model	uracil	salicylic acid	toluene	ethanol
def. sGDML	0.38	0.44	0.21	0.51
imp. sGDML	0.32	0.39	0.20	0.50
def. SchNet	0.77	0.99	0.78	0.57
imp. SchNet	0.65	1.03	0.67	0.47
def. GAP/SOAP	2.71	1.80	1.32	1.62
imp. GAP/SOAP	2.17	1.54	1.09	1.36

5.4 Application to dynamics of a flexible molecule

5.4.1 Flattening the error curve

Up to now, the scope of all applications was molecules of limited sizes. Here, we want to extend the applicability and concentrate on a significantly larger molecule. Specifically, the candidate molecule here is an alanine tetrapeptide, a molecule large enough to start showing the beginning of secondary structure motifs observed in peptides and proteins. The reference dataset is a trajectory born from an *ab initio* molecular dynamics. Specifically, the FHI-aims [87] software was used at a PBE+MBD level of theory. A timestep of 1 fs was used and a global Langevin thermostat was set at 500 K and a friction coefficient of 2 fs. A total of 80k data points were generated with a coverage of at least three energy minima throughout the trajectory.

It is fruitful to recall that the improved training method relies on an initial model with somewhat acceptable accuracy to base its first iteration process on. Fulfilling this requirement on a larger, more flexible molecule such as this one necessitates a higher number of initial training points (as well as final), as the PES this time is higher-dimensional and markedly more complex. It is for this reason that we exclude sGDML and GAP/SOAP for this subsection to instead solely focus on SchNet, as neural networks can train on much larger datasets than kernel-based methods. The initial training set size was set to 2000 for this system, accompanied by 8 iterative steps with a step size of 500 points.

The comparative results between the default and improved models are once again shown in the form of cluster errors, see Figure 5.5. It is important to keep in mind that our training selection method only considers improvements in force predictions. As such, the aim is to flatten the force prediction error curve, which succeeds by lowering the error factor between the best and worst predicted clusters from 2 to 1.1. Furthermore, this does not come at the cost of reduced accuracy on common clusters as almost every single cluster sees an improvement. On the energy plot (left) one can see an unexpected consequence of the training method: a small consistent improvement in energy predictions. The RMSE on energies only drops from 0.54 to 0.49 $kcal/mol$ (from 0.89 to 0.80 $kcal/(mol\text{Å})$ for forces), but the flattening of the error curve can nonetheless have tangible benefits in real applications. This will be further explored at the end of this section.

The energy prediction improvement, while small, merits further thought. In order to reduce errors on energies and forces at the same time without altering the model fundamentally or tweaking the dataset, one would be required to use a mixed loss function, where both energies and forces have an impact on the minimised error at the same time. Training on both energies and forces at the same time is commonplace among NNs, but it comes with a downside: one can only obtain an "optimal" model for either forces or energies. A mixed loss function is in some ways less efficient due to splitting its focus into two competing functions whose optimal set of parameters can wildly differ. Turning our focus instead on the training points themselves does not introduce this dichotomy and exceptionally allows the improvement of both metrics at the same time.

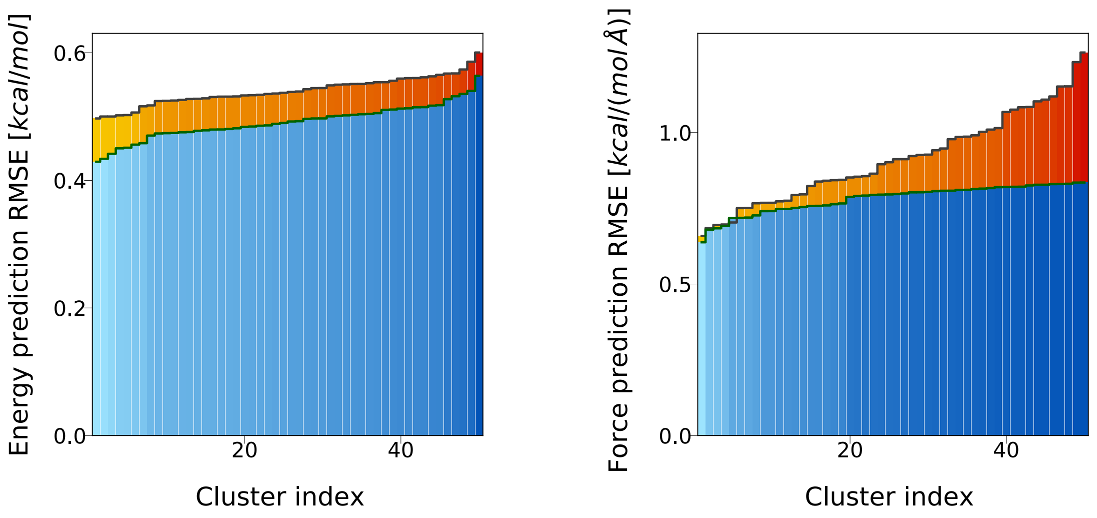

FIGURE 5.5: Prediction RMSE for energies (left) and forces (right) of SchNet models on different clusters: comparing default models (orange bars) and improved models (blue bars) of 6000 training points each.

In order to put the all improvements into perspective, Figure 5.6 compares the improved model of 6000 training points to default models of varying training set sizes: 3000, 6000, and 9000. Once again, both energies and forces are shown left and right respectively. One can see that both in terms of general improvements and flattening of the error curve, the improved model with 6000 training points seems to mimic the behaviour one would expect from an increase in training set size. This effect shows that the default training sets contain unnecessary redundancy, thereby reducing their effective size compared to more carefully selected points.

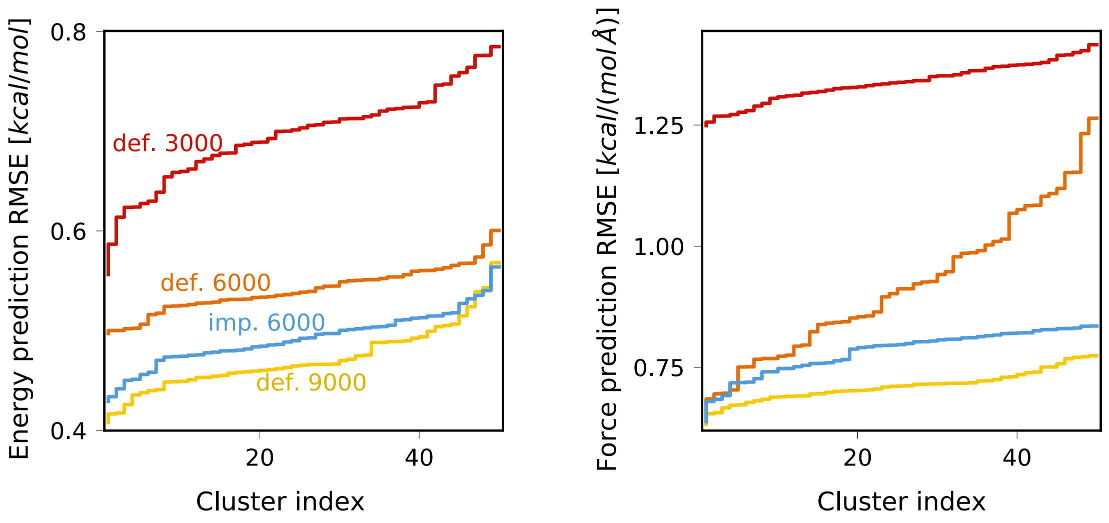

FIGURE 5.6: Prediction RMSE for energies (left) and forces (right) of SchNet models on different clusters: comparing a defaults models of 3000, 6000 and 9000 training points (red, orange and yellow solid line) to an improved model of 6000 training points (solid blue line).

TABLE 5.2: Overall force and energy RMSEs for default SchNet models of various training set sizes and an improved SchNet model of 6000 training points.

RMSE	def. 3000	def. 6000	imp. 6000	def. 9000
Forces [$kcal/(mol$ Å$)$]	1.34	0.89	0.79	0.71
Energy [$kcal/mol$]	0.70	0.54	0.48	0.46

Figure 5.6 also broadly highlights different model behaviours based on the completeness of their training set. Roughly, we can identify three different states:

- (3000 points) The training set lacks information altogether and is unable to reconstruct a reliable MLFF with good accuracy across the entire CS. Here, selecting the same amount of training points with the iterative approach would likely not have a strong effect on the outcome, as the initial model would struggle to identify problematic regions of CS

- (6000 points) The training set contains enough configurations to accurately learn the PES. However, the forces still show high variance in their accuracy across different clusters, hinting that a more suitable choice of training points could be preferable. This is the case for all the previously seen cases (see section 5.3) where a refocusing of the training set provides tangible benefits. In the case of the tetrapeptide, the improved model seems to mimic the expected performance of a model of 7.5-8k points, making it significantly more data-efficient.
- (9000 points) The training set starts being overloaded with data. This leaves little room for improvement as the training set likely contains most of the relevant configurations needed to describe all of CS. As such, a meticulous choice of training points becomes less impactful, although not necessarily useless.

5.4.2 Dynamics of a flexible molecule

Finally, we want to compare the usability of the improved and default models by running constant-temperature molecular dynamics simulations using those models. We use a Langevin thermostat with a friction coefficient of 100 fs and a time step of 0.5 fs. As the molecule at hand is both large and flexible, it is necessary to generate long trajectories to obtain well-converged mean energies. As such, all trajectories have a total runtime of 0.9 nanoseconds or almost 2 million steps. Naturally, simulations of this size would be very difficult with *ab initio* methods due to their prohibitive computational cost, so MLFFs are the only viable alternative, making this a realistic practical application. Note that while the improved training method comes with a slight computational overhead by training a model more than once, time spent running simulations easily overshadows the time spent on training.

A first trajectory was generated at 300 K. There, both the default and improved models converge without hiccups and the resulting average total energies differ by only 0.5 *kcal*/*mol*. As this easily lies within the accuracy of the ML models themselves, one can conclude that both simulations give effectively identical results. In order to move the ML model outside its zone of comfort, a second set of trajectories was computed at 400 K. This drastically changes the situation and the average total energy curves over time present stark differences between models, see Figure 5.7.

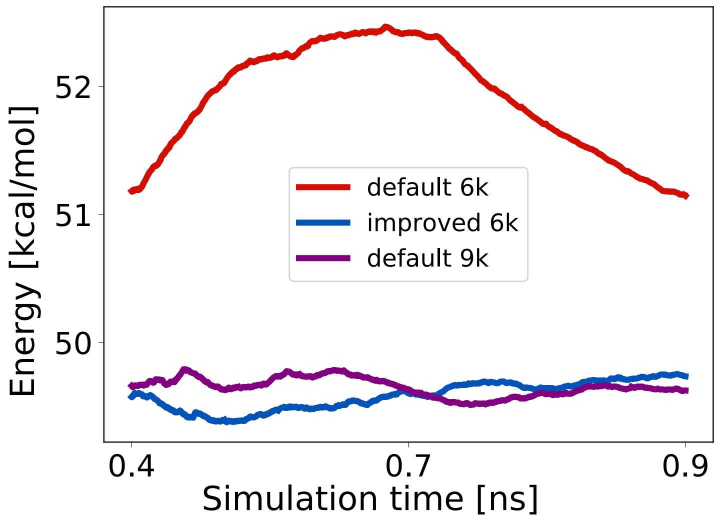

FIGURE 5.7: Average total energy over time throughout a molecular dynamics simulation of alanine tetrapeptide. Three SchNet models are compared: a default model of 6000 training points (red), a default model of 9000 training points (purple) and an improved model of 6000 points (blue). All simulations were performed at 400 K with a timestep of 0.5 fs. Energy zero-point selected as the lowest energy in the original reference dataset.

The 6k default model is unable to reproduce the dynamics at 400 K. The increase followed by a sudden monotonic decay hints that the model is unreliable and its results untrustable. This is likely due to the model hitting one or more rare configurations, or even reaching the model's extrapolation regime at some point. We have shown in section 5.2 that the default training sets do not provide enough context to qualitatively reproduce correct behaviour when deviating from the dataset's CS. Conversely, the improved 6k model has no stability issues and behaves almost identically to the 9k model, despite having 50% fewer training points.

In principle, the reference dataset should contain all information necessary to simulate dynamics at 400 K, as it was generated at 500 K. Nevertheless, practical applications using MLFFs need to remain at much lower temperatures due to the dataset's limited size and the inability of any sampling method to cover all of available CS. As such, increasing the applicability range of a default ML model would require creating larger reference datasets at potentially higher temperatures (requiring even more data points). Instead, the developed training scheme provides better predictions, boosted reliability and increased applicability range with a comparatively negligible increase in computational cost.

5.5 Details of parameter tuning

5.5.1 Number of iterative steps

In section 5.1.2, parameter choices for the improved learning technique were briefly summarised. In this section, we will touch on some of the parameters and their effects on the resulting model. The most important parameter is the step size (or the number of steps), assuming a fixed number of initial training points and a fixed number of final training points. The effects of this parameter are the easiest to understand, as a higher amount of steps leads to a better resolution on gradual improvements for problematic configurations. In principle, a very large amount of steps would be ideal, but this comes at the cost of significantly increasing the training time. Thus, the choice of 8 as our number of steps throughout this work is based on a realistic choice that minimises inconvenient (and expensive) training durations while providing most of the benefits.

Figure 5.8 shows the effect that the parameter has on the resulting model. Here, the uracil dataset with an sGDML model was chosen because it highlights the effect of the improved training method very well (see Figure 5.4). All sGDML models were trained using a total of 1000 training points. It is clear to see that a higher number of steps provides a strictly better model, with particularly increased performance on the poorly predicted clusters. This makes sense, as large step sizes (low number of steps) tend to overshoot and include too many instances of at-the-time problematic configurations, thereby neglecting other regions of CS.

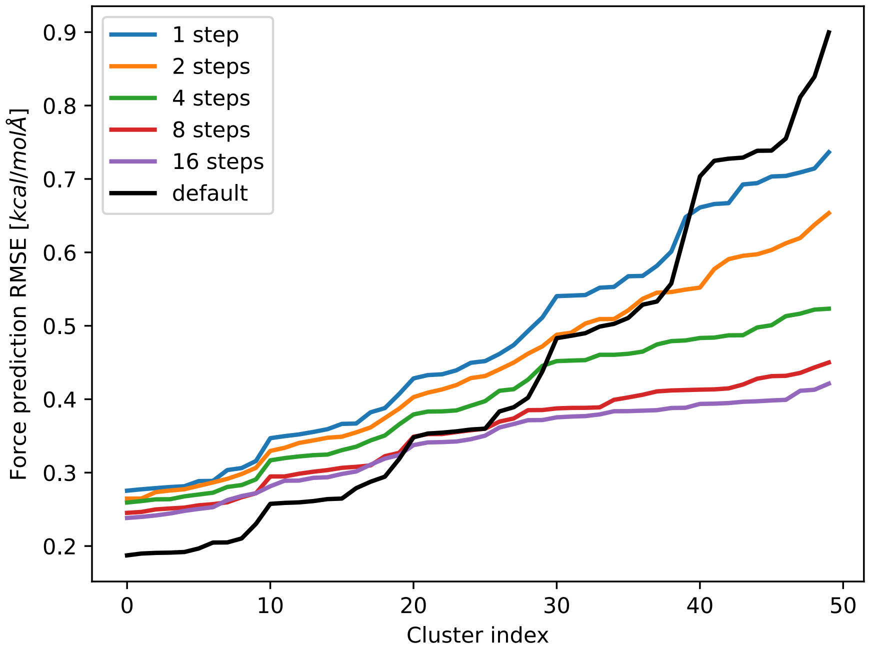

FIGURE 5.8: Force prediction RMSE across 50 clusters of a uracil molecule for improved sGDML models of 1000 training points and a varying number of iterative steps (coloured lines). A default sGDML model of 1000 training points is shown for comparison (black line).

5.5.2 Number of fine clusters

The next parameter of interest is the number of fine clusters created during the iterative step. We previously described this particular parameter as a way to increase the fineness of the grid through which CS is filtered during the iterative process. To explore its influence, several improved sGDML models were trained on uracil but for various amounts of fine clusters, see Figure 5.9. Note that the step size is fixed at 100 and all models contain a total of 1000 training points.

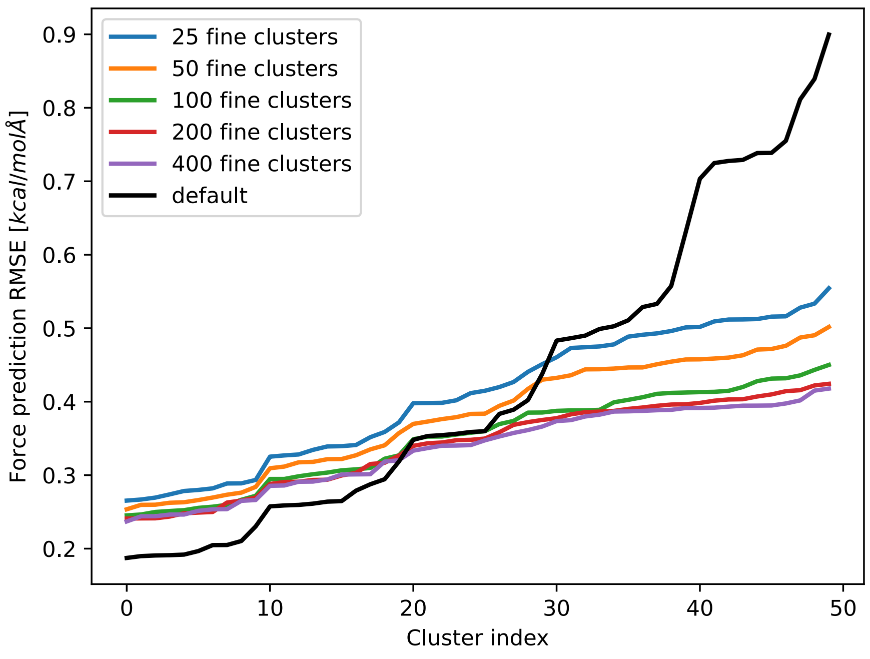

FIGURE 5.9: Force prediction RMSE across 50 clusters of a uracil molecule for improved sGDML models of 1000 training points and a varying number of fine clusters (coloured lines). A default sGDML model of 1000 training points is shown for comparison (black line).

Similarly to the number of steps, it is generally true that a higher number of clusters provides better results, although the difference in performance is less significant. Furthermore, there are much stronger diminishing returns on this effect: there is practically no difference between the models using 200 and 400 fine clusters. On the flip side, this particular parameter does not affect computational time and thus, at first glance, has no downside to be increased.

As such, it is natural to question why the creation of fine clusters is necessary in the first place: after all, this would maximise the resolution and thus (seemingly) optimise the model even further. As a first, proceeding this way would render it impossible to only predict a fraction of every cluster (by virtue of their inexistence), as was done for GAP/SOAP models in section 5.3. As a reminder, only 1% of every cluster needed to be computed, hence reducing the computational cost by two orders of magnitude.

Furthermore, the goal is not to minimise the prediction error on the worst *configurations*, but rather the worst *types of configurations*. This distinction is important because single configurations — or even a handful of them — can be complex and rare and thus hard to predict without being significant for the dataset. Singular outliers and anomalies should not be catered towards as they are a "waste" of training points unless they are representative of a non-negligible region of CS. This is what clusters allow us to extract: not singular points, but representative groups. This last argument also explains why it is not advised to increase the number of fine clusters past the point they offer noticeable improvements, as it shifts more power towards singular configurations as opposed to representative groups.

5.5.3 Number of initial training points

The last parameter this section will focus on is the number of initial training points. Broadly speaking, this parameter allows the user to choose how many "usual" training points to sacrifice to make room for targeted ones. To get a deeper look, we created several improved sGDML models trained on uracil but for various amounts of initial training points, see Figure 5.10. Note that the step size is fixed at 100(or close to it) and all models contain a total of 1000 training points.

On the left side of the plot, one can observe that increasing the number of initial points shifts the curve closer to that of a default training scheme. This is of course not surprising, as the parameter is essentially a measure of the mixing factor between an improved and default training set. On the right side of the plot, a similar story is observed. This time, however, the improvements associated with optimising more points inside the training set hit heavily diminishing returns after

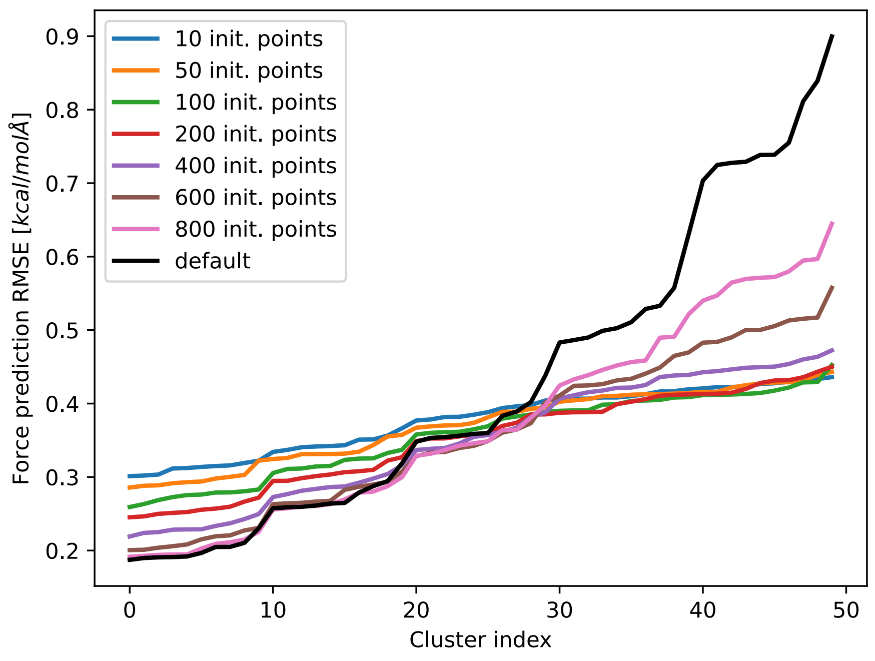

FIGURE 5.10: Force prediction RMSE across 50 clusters of a uracil molecule for improved sGDML models of 1000 training points and a varying number of initial training points (coloured lines). A default sGDML model of 1000 training points is shown for comparison (black line).

about 200 initial training points. This indicates that while it is in general beneficial for the flattening of the error curve to minimise the number of initial training points, there is no reason to go below a certain threshold as it only makes the model slightly less accurate on the common configurations.

The reason for this observation is that if the initial model is too unreliable, the first few iterative steps are likely to select suboptimal points to add to the training set. As such, the added benefit of having a couple more iterations is outweighed by the fact that these iterations are largely ineffective. Note that this plot is limited to the uracil molecule, which is a small, simple organic molecule that does not require a lot of training points to predict accurately. All the effects observed here would be exasperated for significantly larger or more complex systems, where a reasonable amount of initial training points is required for the initial model to have any worthwhile predictive qualities.

5.6 Conclusions

The analysis provided by FFAST — and in particular the analysis of prediction errors across configurational clusters — showed that MLFFs tend to not equally learn all of CS. In particular, small clusters or clusters representing rare mechanisms inside of a large dataset need to be explicitly accounted for when building a training set to ensure their adequate representation.

This section introduced an iterative approach to building such a training set, providing the ML model with the data needed to learn all of CS equally well. The method utilises several clustering methods, namely agglomerative clustering to distinguish geometrical features and KMeans to separate different energy values. Using this, representative configurations from all the important regions of CS are added into the training set, thereby "flattening" the prediction error curve across clusters of the resulting model.

This effect was showcased on a handful of small to medium-sized organic molecules as well as three MLFFs of choice: sGDML, SchNet and GAP/SOAP. It was shown that the models utilising the improved training point selection scheme present more homogeneous error profiles across configurational clusters. Furthermore, the improved models performed quantitatively and qualitatively better in the extrapolation regime, as showcased with a proton exchange mechanism in salicylic acid. There, the improved sGDML and SchNet models both produced values closer to the reference calculations than their default counterparts. Finally, SchNet models trained an alanine tetrapeptide molecule were used to run molecular dynamics, which revealed that the improved training scheme boosted the model's stability to rival that of a default model up to 50% more training points.

Chapter 6

Perspective

Over the years, many innovative changes have been brought to the field of MLFFs, including deep NNs, kernel methods, message-passing architectures, equivariant interactions between atomic environments, explicit physical correction terms, and more. Altogether, MLFFs have been successfully applied to various systems of ever-growing sizes. While in the beginning, only simple crystals and small organic molecules were feasible, nanoscale systems such as molecular crystals, proteins, crystals with defects, etc., are now within reach. The increasing size and complexity of the systems under study bring new challenges and require reconsidering some default approaches.

One of the challenges is the reliable measures of the quality of MLFF. As demonstrated in this thesis, overall error metrics such as MAE and RMSE do not fully reflect the ability of MLFFs to reconstruct the PES of large enough complex systems. Moreover, the MLFF's reliability for long-time simulations is non-obvious, even in more sophisticated analyses. Yet, it can significantly impact the outcome of ML simulations and their usage in practice (such as computing thermodynamic properties, reaction rates, etc.). The toolbox developed in this work aims to help resolve this issue and identify its roots, but new developments are likely required, especially for transferable ML models aiming at large-scale applications while trained on much smaller reference systems.

Importantly, ML potentials are uniquely positioned compared to many other fields employing ML models, as the functions that create the training data are fully known. This enables the combining of ideas and solutions from the fields of ML and quantum chemistry/physics, going beyond the limits of each particular area. Steps in this direction have already been taken, and it has been shown that smartly incorporating physical and chemical knowledge directly into the construction of the ML model can positively affect both general accuracy and asymptotic behaviour. Improvements such as those will continue to be important when facing the challenging problems of multiscale modelling. Even for the current systems, the error analyses provided within this work with the help of FFAST and error flattening clearly show that all the current state-of-the-art MLFFs have flaws, which can and should be addressed. We found a significant inhomogeneity in force reconstruction, even for atoms of the same type (same chemical element) when the system under study possesses chemically diverse atomic neighbourhoods.

In previous sections, the role of MLFFs in simulations was likened to the software controlling a self-driving car. In both cases, the object in question (car or molecule) is being moved to follow an expected path as closely as possible. For the car, it is obvious that it needs to be equally reliable for all locations at any time of the day, as a single accident is not excused by an otherwise accurate following of the road. For molecular systems, “accidents” are instead the evolution into unstable or physically incorrect states, such as hydrogens spontaneously breaking away from their bonded pair or atoms closely overlapping. These cases mostly happen when the model moves into a configuration too far from the reference dataset it was trained on, thus entering an extrapolation regime. To minimise the chances of this happening, it is crucial to ensure that none of the potentially visitable states of the system requires sheer extrapolation.

This hints at the need to introduce a “divide and conquer” (DaC) approach when building MLFF models. There, stabilization can be achieved by employing elementary blocks to simplify the learning task at the cost of increasing the structural complexity of the system. This idea is not new and has been used in empirical force fields for decades: the limitations of employing specific functional forms result in the need to parametrize the same chemical elements differently based on their local environment. The same reasoning applies to MLFFs. While they can fit their functional forms in broad ranges, the growing requirements for high accuracy, as well as the complexity of interactions featured in large-scale systems with multiple chemical elements, might make the flexibility of ML architectures insufficient when all atoms belonging to the same chemical elements are considered as identical. For instance, the interaction partners for carbon atoms forming a graphene substrate might significantly deviate from those for carbons in molecules adsorbed on said substrate. Careful consideration of local environments to distinguish between qualitatively different types of atoms (within the same chemical element) could positively affect the accuracy and transferability of MLFF models.

Another potential benefit from splitting up the training process is reaping greater benefits from symmetries belonging to local PES minima. Symmetries are paramount for ML models, simplifying the learning task and decreasing the required number of training points. When the CS of the system under consideration includes multiple local minima or even phase transitions, the total number of symmetries is limited only to a subset shared by all the configurations at once. On the contrary, if a model could instead focus on a single equilibrium state and its surroundings, all the local symmetries present in this particular geometrical composition could be utilized. Thus, splitting the learning task into regions with similar local symmetries would lead to better accuracy, data efficiency, and reliability.

The methods developed during this Ph.D. as well as the FFAST package provide some basis for implementing a divide-and-conquer idea into new MLFF models. For example, datasets in section 4.5 were split into different regions of CS using chemical intuition of the system as well as knowledge about the dataset's creation. In particular for azobenzene, which can be found in two distinct isomeric states cis and trans, it is straightforward to split the training task into two separate models to handle each isomeric state. As a proof of concept, this methodology was applied to an azobenzene dataset containing 5000 cis and 5000 trans configurations, with both subsets being trajectories of MD runs at the level of PBE+TS with a time step of 1 fs and a constant temperature of 300 K. Two separate models GDML (cis and trans) models were trained on 500 points of their respective subset and tested on the 4500 remaining points each. It was found that the two 500-point models (henceforth the DaC model) had a 15% lower force prediction MAE than a single model trained on the same 1000 training points. More importantly, the cluster error curves (see section 4.4.1) showed a more homogeneous distribution, hinting at an increase in overall stability.

It is important to discuss the potential implementation difficulties of the method above. In practical applications, the outlined DaC approach requires the classification of a new configuration before assigning it for prediction to the corresponding model. In the case of azobenzene, it is trivial to distinguish between cis and trans by inspection of the central dihedral angle, but more complex systems might produce more ambiguous cases. Furthermore, reproducing a transition from cis and trans requires a swap from one model to the other. In an MD, this would create a discontinuity in the forces and a possible jump in potential energy, which is likely to become the cause of unstable behaviour. To address this in the simple case of azobenzene, the predictions at a transition state can be given as a linear combination of the two models that allows a continuous switch from one model to the other as a function of the dihedral angle.

The same approach was attempted on other datasets, all small to large-sized molecules taken from MD17 [43] or MD22 [132]. However, rather than relying on chemical intuition, clustering algorithms were used to separate CS. Here, agglomerative clustering on inverse pairwise distances with single linkage was used. In the case of 2 clusters, the azobenzene dataset is perfectly split into cis and trans using this approach as well, producing results identical to those above. For small or rigid molecules, the two clusters are not different enough to benefit from the split and as such the DaC model provides significantly worse results than a single model. On the large, flexible molecule of alanine tetrapeptide, the two approaches performed very similarly: the force prediction MAE of a DaC model composed of two 1000-point SchNet models was comparable to a single 2000-point model. This is expected, as the method is only beneficial for molecules that are large enough to be found in qualitatively different states with distinct behaviour. In the case of the tetrapeptide, the two clusters largely distinguished between folded and unfolded states, for which the atomic interactions are very dissimilar.

DaC models using a higher number of clusters all performed worse than their 2-cluster counterparts (keeping the total number of training points constant). This is because MLFFs are usually trained in a low-data regime, hence why data efficiency holds such importance in the field. ML model accuracy as a function of the number of training points is roughly an exponential decay, thus if the number of training points grows to a certain size, this naive DaC might still outperform a single model. A more sophisticated approach mimicking the iterative training process outlined in section 5.1.2 was also attempted: first, the dataset is split into two clusters and one model is trained for each, as outlined before. Then, the cluster whose respective model shows the highest cluster force prediction MAE (weighted by the number of configurations in the cluster) is further split into 2 sub-datasets and new models are trained on each. For a given number of clusters, this method performed strictly better than the previous approach for more than 3 clusters. However, this does come with an increase in training time as every split results involves discarding one fully trained model.

The classification of configurations during practical applications is more difficult when the dataset was originally split using clustering rather than a priori chemical knowledge. To this end, a classification model must be trained on the cluster labels to accurately identify which model needs to be used for a new, previously unseen configuration. For all cases tried in this work, this task proved to be simple to achieve (with 98-99% accuracy) using a bare-bones modification of a

PointNet architecture [112] to take into account chemical species. As PointNet is designed to act on clouds of many thousands of points, this method is not expected to be the limiting factor for DaC. Furthermore, ML classifiers provide probability distributions over clusters, which can directly be used to create a linear combination of models and largely alleviate the discontinuity problems.

Another useful addition to the DaC approaches outlined above is the sharing of general training points. In this approach, each model is trained in part on configurations of its respective clusters, but also on a general set of configurations shared with other models. In principle, this greatly increases the data efficiency of this particular DaC approach by making models "specialise" into a particular type of configuration rather than learning it exclusively. However, when applied to the small to medium-sized molecules in this work, this approach was found to have negligible benefits. This is likely due to the clusters not being distinguishable enough for molecules of this size, making the shared training points too similar to those extracted from clusters. Additional work — in particular on large and flexible molecules — needs to be done to assess the use cases of this method.

All in all, it was found that the simple DaC approach briefly explored in this work favours large or flexible molecules, especially if a priori chemical knowledge can be used to assist the clustering. This justifies the timely need to delve deeper into this possible avenue, as the system sizes handled by MLFFs have reached a state where the presence of many qualitatively different regions of CS is expected. However, an increase in system size also limits the performance of the current clustering algorithms. As outlined in section 4.3, clustering methods are highly reliant on the choice of descriptor. In this preliminary work, a simple pairwise inverse distance descriptor was used, but future applications will need to opt for more elaborate descriptors. Large systems tend to need a large descriptor to be fully represented, making distance metrics in this space unreliable due to the sheer number of dimensions. As most clustering algorithms rely on a notion of distance (or density), cluster quality is likely to decrease for larger systems.

To guarantee that clustering-based DaC can practically be utilised on systems large enough to benefit from the approach, new clustering approaches need to be developed at the same level as MLFF architectures themselves. Using the physically and chemically inspired methods explored in section 2.4 to enhance modulable clustering methods such as self-organizing maps [147] or deep clustering [148] is likely the key to the creation of meaningful subdivisions for any dataset. Furthermore, incorporating symmetry considerations into these novel architectures to ensure that the choice of clusters maximises the presence of local symmetries would further enhance the DaC approach and widen its applicability.

Finally, it is worth noting that the DaC approach described in this work is fully independent of the choice of MLFF. The many state-of-the-art architectures developed in the field all provide their advantages and disadvantages as well as unique parameters to tweak to increase the performance on a particular system. In a DaC scheme, each of these considerations can be applied locally as well, thereby optimising the choice of MLFF architecture and the varying of its hyperparameters for each cluster rather than a whole system. This can be particularly useful for models with explicit physical interactions incorporated into the network, as they are likely to present a different set of optimal parameters for different regions of CS, for instance in long-range interactions of folded or unfolded configurations.

Chapter 7

Summary

MLFFs have continuously improved over the years, reaching remarkable accuracy across various chemical systems of ever-increasing sizes. However, with great sizes comes great complexity, which makes it harder to define and assess a given model's quality. Even a model with state-of-the-art overall accuracy can demonstrate highly heterogeneous predictive capabilities across the CS of a single system. Furthermore, the prediction error profile across different atoms equally needs to be assessed, as even atoms of the same element can show notable differences if they are, e.g., in different functional groups.

This work proposed to push current practices in MLFF analysis to a more detailed and insightful standard. The novel Force Field Software and Analysis Tools (FFAST) lies at the forefront of this effort, providing detailed breakdowns of ML models' performance. The tool is in active development and aims to be accessible and flexible to provide the groundwork for future developments on top of current capabilities. A large list of provided features was showcased using practical examples.

To illustrate some of FFAST's facets, two datasets of medium-sized organic molecules were chosen: docosahexaenoic acid (DHA) and stachyose. Furthermore, two state-of-the-art ML models were trained on the datasets to exemplify a general FFAST workflow. Through FFAST's interactive plots and 3D visualization tools, it was quickly concluded that both models and datasets presented hydrogens with noticeably lower prediction errors than any other atom type. As hydrogens are abundant in both systems, this has a large impact on the models' overall MAE and RMSE, which do not reflect the stability and performance in the main chain of the molecules. Conversely, carbons and oxygens were tied to very uneven error profiles, with individual atoms showing higher prediction errors than other atoms of the same type if their environment was chemically or structurally more diverse. For example, prediction errors on DHA were larger if the molecule was in a folded state and the primary source of the errors was the atoms in the carboxylic group.

Beyond the analysis of MLFF predictions, FFAST was also shown to provide insight into the reference data itself. In the case of DHA, the number of folding and unfolding processes was easily found within its underlying MD trajectory and the transformations were correlated with the system's potential energy. Furthermore, energy and force distributions for different subsets of the reference data were independently analysed to e.g. ensure that the training set is representative of the validation and test sets.

One particular analytical tool provided by FFAST is the visualisation of a cluster prediction error curve. This is a new form of analysis developed for this work that leverages unsupervised ML to split a dataset into qualitatively different regions of CS. This procedure was shown to be able to extract small subsets of configurations representing nontrivial mechanisms that could easily be overlooked inside a large pool of data. These clusters were used to show that several state-of-the-art models present non-uniform error distributions across CS. This further emphasises the need for tools such as FFAST, as such nuances are not captured by overall error metrics such as MAEs or error distributions.

Furthermore, similar splittings of a dataset were performed using chemical intuition and prior knowledge of the system instead of clustering algorithms. It was shown that in the case of ML models trained on an azobenzene dataset, different parts of the transition from the molecule's cis isomeric state to its trans state are unequally well reproduced. It was also determined that the optimal choice of descriptors varies throughout the mechanism. Finally, additional analysis revealed that the choice of training set had a substantial effect on the final models. This is explained by the fact that rare mechanisms can easily be poorly represented in a randomly selected training set, leading to inadequate prediction accuracy.

To address the aforementioned issue, this work suggested a novel training points selection method. It relies on the clustering techniques previously explored to create an approach that iteratively refines the training data to better incorporate all regions of CS available in the reference dataset. It was shown that this approach leads to a "flattening" of the prediction error curve across qualitatively different types of configurations as represented by clusters, significantly reducing the variance in prediction errors across different clusters.

To test the improvements provided by the new training method in extrapolation regimes, a new dataset containing an extended version of the proton exchange mechanism subtly contained within a salicylic acid dataset was created. The improved SchNet and sGDML models trained on the original dataset were tested and compared to the default training procedures. The improved models were shown to have better asymptotic behaviour when leaving the regions contained in their original reference dataset. Furthermore, improved models were also shown to be more stable in dynamics as exemplified by a model trained on an alanine tetrapeptide.

Overall, this work showcased that after many advances made in the field of MLFF in recent years, there are still open problems to be addressed. Those require improvements on existing methods and a particular emphasis on developing tools to detect and understand the shortcomings of current MLFFs in practice. The insight provided by a detailed understanding of the strengths, limitations, and pitfalls of ML models, which can be obtained using FFAST or similar software packages, is the key to pushing forward the state-of-the-art.

Bibliography

[1] Andy C. Lee, Janelle L. Harris, Kum K. Khanna, and Ji-Hong Hong. "A Comprehensive Review on Current Advances in Peptide Drug Development and Design". In: *Int. J. Mol. Sci.* 20.10 (2019), p. 2383. ISSN: 1422-0067. DOI: 10.3390/ijms20102383.

[2] Saber Naserifar, Yalu Chen, Soonho Kwon, Hai Xiao, and William A. Goddard III. "Artificial Intelligence and QM/MM with a Polarizable Reactive Force Field for Next-Generation Electrocatalysts". In: *Matter* 4.1 (Jan. 2021), pp. 195–216. ISSN: 2590-2393. DOI: 10.1016/j.matt.2020.11.010.

[3] Tânia F. G. G. Cova and Alberto A. C. C. Pais. "Deep Learning for Deep Chemistry: Optimizing the Prediction of Chemical Patterns". In: *Front. Chem.* 7 (2019). ISSN: 2296-2646.

[4] Peter L. Freddolino, Anton S. Arkhipov, Steven B. Larson, Alexander McPherson, and Klaus Schulten. "Molecular Dynamics Simulations of the Complete Satellite Tobacco Mosaic Virus." In: *Struct.* 14.3 (Mar. 2006), pp. 437–449. ISSN: 0969-2126. DOI: 10.1016/j.str.2005.11.014.

[5] Gongpu Zhao et al. "Mature HIV-1 Capsid Structure by Cryo-Electron Microscopy and All-Atom Molecular Dynamics". In: *Nature* 497.7451 (May 2013), pp. 643–646. ISSN: 1476-4687. DOI: 10.1038/nature12162.

[6] Robin Winter, Floriane Montanari, Andreas Steffen, Hans Briem, Frank Noé, and Djork-Arné Clevert. "Efficient Multi-Objective Molecular Optimization

in a Continuous Latent Space". In: *Chem. Sci.* 10.34 (2019), pp. 8016–8024. ISSN: 2041-6520. DOI: 10.1039/C9SC01928F.

[7] Joao Ramos, Jayaraman Muthukumaran, Filipe Freire, João Paquete-Ferreira, Ana R. Otrelo-Cardoso, Dmitri Svergun, Alejandro Panjkovich, and Teresa Santos-Silva. "Shedding Light on the Interaction of Human Anti-Apoptotic Bcl-2 Protein with Ligands through Biophysical and in Silico Studies". In: *Int. J. Mol. Sci.* 20.4 (2019), p. 860. ISSN: 1422-0067. DOI: 10.3390/ijms20040860.

[8] Rachael A. Mansbach, Timothy Travers, Benjamin H. McMahon, Jeanne M. Fair, and S. Gnanakaran. "Snails In Silico: A Review of Computational Studies on the Conopeptides". In: *Mar. Drugs* 17.3 (2019), p. 145. ISSN: 1660-3397. DOI: 10.3390/md17030145.

[9] Zhifeng Jing, Chengwen Liu, Sara Y. Cheng, Rui Qi, Brandon D. Walker, Jean-Philip Piquemal, and Pengyu Ren. "Polarizable Force Fields for Biomolecular Simulations: Recent Advances and Applications". In: *Annu. Rev. Biophys.* 48.1 (2019), pp. 371–394. ISSN: 1936-122X. DOI: 10.1146/annurev-biophys-070317-033349.

[10] Sean Ekins. "The Next Era: Deep Learning in Pharmaceutical Research". In: *Pharm. Res.* 33.11 (Nov. 2016), pp. 2594–2603. ISSN: 1573-904X. DOI: 10.1007/s11095-016-2029-7.

[11] Eric Smalley. "AI-powered Drug Discovery Captures Pharma Interest". In: *Nat. Biotechnol.* 35.7 (July 2017), pp. 604–605. ISSN: 1546-1696. DOI: 10.1038/nbt0717-604.

[12] Daniel C. Elton, Zois Boukouvalas, Mark S. Butrico, Mark D. Fuge, and Peter W. Chung. "Applying Machine Learning Techniques to Predict the Properties of Energetic Materials". In: *Sci. Rep.* 8.1 (June 2018), p. 9059. ISSN: 2045-2322. DOI: 10.1038/s41598-018-27344-x.

[13] Daniel C. Elton, Zois Boukouvalas, Mark D. Fuge, and Peter W. Chung. "Deep Learning for Molecular Design—a Review of the State of the Art". In: *Mol. Syst. Des. Eng.* 4.4 (2019), pp. 828–849. DOI: 10.1039/C9ME00039A.

[14] Felix A. Faber, Alexander Lindmaa, O. Anatole von Lilienfeld, and Rickard Armiento. "Machine Learning Energies of 2 Million Elpasolite $(AB{C}_{2}{D}_{6})$ Crystals". In: *Phys. Rev. Lett.* 117.13 (Sept. 2016), p. 135502. DOI: 10.1103/PhysRevLett.117.135502.

[15] David S. Sholl and Janice A. Steckel. *Density Functional Theory: A Practical Introduction*. John Wiley & Sons, Inc., 2009. ISBN: 978-0-470-37317-0.

[16] R. O. Jones. "Density Functional Theory: Its Origins, Rise to Prominence, and Future". In: *Rev. Mod. Phys.* 87.3 (Aug. 2015), pp. 897–923. DOI: 10.1103/RevModPhys.87.897.

[17] Herrman G. Kümmel. "A Biography of the Coupled Cluster Method". In: *Int. J. Mod. Phys. B* 17.28 (Nov. 2003), pp. 5311–5325. ISSN: 0217-9792. DOI: 10.1142/S0217979203020442.

[18] Isaiah Shavitt and Rodney J. Bartlett. *Many-Body Methods in Chemistry and Physics: MBPT and Coupled-Cluster Theory*. Cambridge Molecular Science. Cambridge: Cambridge University Press, 2009. ISBN: 978-0-521-81832-2. DOI: 10.1017/CBO9780511596834.

[19] Jörg Behler. "Perspective: Machine Learning Potentials for Atomistic Simulations". In: *J. Chem. Phys.* 145.17 (Nov. 2016), p. 170901. ISSN: 0021-9606. DOI: 10.1063/1.4966192.

[20] J. S. Smith, O. Isayev, and A. E. Roitberg. "ANI-1: An Extensible Neural Network Potential with DFT Accuracy at Force Field Computational Cost". In: *Chem. Sci.* 8.4 (2017), pp. 3192–3203. ISSN: 2041-6520. DOI: 10.1039/C6SC05720A.

[21] Christian Devereux, Justin S. Smith, Kate K. Huddleston, Kipton Barros, Roman Zubatyuk, Olexandr Isayev, and Adrian E. Roitberg. "Extending the Applicability of the ANI Deep Learning Molecular Potential to Sulfur and Halogens". In: *J. Chem. Theory Comput.* 16.7 (July 2020), pp. 4192–4202. ISSN: 1549-9618. DOI: 10.1021/acs.jctc.0c00121.

[22] Volker L. Deringer, Noam Bernstein, Albert P. Bartók, Matthew J. Cliffe, Rachel N. Kerber, Lauren E. Marbella, Clare P. Grey, Stephen R. Elliott, and Gábor Csányi. "Realistic Atomistic Structure of Amorphous Silicon from Machine-Learning-Driven Molecular Dynamics". In: *J. Phys. Chem. Lett.* 9.11 (June 2018), pp. 2879–2885. DOI: 10.1021/acs.jpclett.8b00902.

[23] Kevin Ryczko, Kyle Mills, Iryna Luchak, Christa Homenick, and Isaac Tamblyn. "Convolutional Neural Networks for Atomistic Systems". In: *Comput. Mater. Sci.* 149 (June 2018), pp. 134–142. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2018.03.005.

[24] Anders S. Christensen, Lars A. Bratholm, Felix A. Faber, and O. Anatole von Lilienfeld. "FCHL Revisited: Faster and More Accurate Quantum Machine Learning". In: *J. Chem. Phys.* 152.4 (Jan. 2020), p. 044107. ISSN: 0021-9606. DOI: 10.1063/1.5126701.

[25] Jiang Wang, Simon Olsson, Christoph Wehmeyer, Adrià Pérez, Nicholas E. Charron, Gianni de Fabritiis, Frank Noé, and Cecilia Clementi. "Machine Learning of Coarse-Grained Molecular Dynamics Force Fields". In: *ACS Cent. Sci.* 5.5 (May 2019), pp. 755–767. ISSN: 2374-7943. DOI: 10.1021/acscentsci.8b00913.

[26] Tsz Wai Ko, Jonas A. Finkler, Stefan Goedecker, and Jörg Behler. "General-Purpose Machine Learning Potentials Capturing Nonlocal Charge Transfer". In: *Acc. Chem. Res.* 54.4 (Feb. 2021), pp. 808–817. ISSN: 0001-4842. DOI: 10.1021/acs.accounts.0c00689.

[27] Emir Kocer, Tsz Wai Ko, and Jörg Behler. "Neural Network Potentials: A Concise Overview of Methods". In: *Annu. Rev. Phys. Chem.* 73.1 (Apr. 2022), pp. 163–186. ISSN: 0066-426X. DOI: 10.1146/annurev-physchem-082720-034254.

[28] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. *Directional Message Passing for Molecular Graphs*. 2022. arXiv: 2003.03123 [cs.LG].

[29] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. *GemNet: Universal Directional Graph Neural Networks for Molecules*. 2022. arXiv: 2106.08903 [physics.comp-ph].

[30] Jörg Behler and Michele Parrinello. "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces". In: *Phys. Rev. Lett.* 98.14 (Apr. 2007), p. 146401. DOI: 10.1103/PhysRevLett.98.146401.

[31] Volker L. Deringer, Albert P. Bartók, Noam Bernstein, David M. Wilkins, Michele Ceriotti, and Gábor Csányi. "Gaussian Process Regression for Materials and Molecules". In: *Chem. Rev.* 121.16 (Aug. 2021), pp. 10073–10141. ISSN: 0009-2665. DOI: 10.1021/acs.chemrev.1c00022.

[32] Albert P. Bartók, James Kermode, Noam Bernstein, and Gábor Csányi. "Machine Learning a General-Purpose Interatomic Potential for Silicon". In: *Phys. Rev. X* 8.4 (Dec. 2018), p. 041048. DOI: 10.1103/PhysRevX.8.041048.

[33] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. *Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds*. 2018. arXiv: 1802.08219 [cs.LG].

[34] Alireza Khorshidi and Andrew A. Peterson. "Amp: A Modular Approach to Machine Learning in Atomistic Simulations". In: *Comput. Phys. Commun.* 207 (Oct. 2016), pp. 310–324. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2016.05.010.

[35] Zhenwei Li, James R. Kermode, and Alessandro De Vita. "Molecular Dynamics with On-the-Fly Machine Learning of Quantum-Mechanical Forces". In: *Phys. Rev. Lett.* 114.9 (Mar. 2015), p. 096405. DOI: 10.1103/PhysRevLett.114.096405.

[36] John A. Keith, Valentin Vassilev-Galindo, Bingqing Cheng, Stefan Chmiela, Michael Gastegger, Klaus-Robert Müller, and Alexandre Tkatchenko. "Combining Machine Learning and Computational Chemistry for Predictive Insights Into Chemical Systems". In: *Chem. Rev.* 121.16 (Aug. 2021), pp. 9816–9872. ISSN: 0009-2665. DOI: 10.1021/acs.chemrev.1c00107.

[37] Tsz Wai Ko, Jonas A. Finkler, Stefan Goedecker, and Jörg Behler. "A Fourth-Generation High-Dimensional Neural Network Potential with Accurate Electrostatics Including Non-Local Charge Transfer". In: *Nat. Commun.* 12.1 (Jan. 2021), p. 398. ISSN: 2041-1723. DOI: 10.1038/s41467-020-20427-2.

[38] Kun Yao, John E. Herr, David W. Toth, Ryker Mckintyre, and John Parkhill. "The TensorMol-0.1 Model Chemistry: A Neural Network Augmented with Long-Range Physics". In: *Chem. Sci.* 9.8 (2018), pp. 2261–2269. ISSN: 2041-6520. DOI: 10.1039/C7SC04934J.

[39] Andrea Grisafi and Michele Ceriotti. "Incorporating Long-Range Physics in Atomic-Scale Machine Learning". In: *J. Chem. Phys.* 151.20 (Nov. 2019), p. 204105. ISSN: 0021-9606. DOI: 10.1063/1.5128375.

[40] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, and K.-R. Müller. "SchNetPack: A Deep Learning Toolbox For Atomistic Systems". In: *J. Chem. Theory Comput.* 15.1 (Jan. 2019), pp. 448–455. ISSN: 1549-9618. DOI: 10.1021/acs.jctc.8b00908.

[41] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. "Fast and Accurate Modeling of Molecular

Atomization Energies with Machine Learning". In: *Phys. Rev. Lett.* 108.5 (Jan. 2012), p. 058301. DOI: 10.1103/PhysRevLett.108.058301.

[42] Albert P. Bartók, Sandip De, Carl Poelking, Noam Bernstein, James R. Kermode, Gábor Csányi, and Michele Ceriotti. "Machine Learning Unifies the Modeling of Materials and Molecules". In: *Sci. Adv.* 3.12 (2017), p. 1701816. DOI: 10.1126/sciadv.1701816.

[43] Stefan Chmiela, Huziel E. Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. "Towards Exact Molecular Dynamics Simulations with Machine-Learned Force Fields". In: *Nat. Commun.* 9.1 (Sept. 2018), p. 3887. ISSN: 2041-1723. DOI: 10.1038/s41467-018-06169-2.

[44] Stefan Chmiela, Huziel E. Sauceda, Igor Poltavsky, Klaus-Robert Müller, and Alexandre Tkatchenko. "sGDML: Constructing Accurate and Data Efficient Molecular Force Fields Using Machine Learning". In: *Comput. Phys. Commun.* 240 (July 2019), pp. 38–45. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2019.02.007.

[45] Huziel E. Sauceda, Luis E. Gálvez-González, Stefan Chmiela, Lauro Oliver Paz-Borbón, Klaus-Robert Müller, and Alexandre Tkatchenko. "BIGDML—Towards Accurate Quantum Machine Learning Force Fields for Materials". In: *Nat. Commun.* 13.1 (June 2022), p. 3733. ISSN: 2041-1723. DOI: 10.1038/s41467-022-31093-x.

[46] Zhuoran Qiao, Matthew Welborn, Animashree Anandkumar, Frederick R. Manby, and Thomas F. Miller III. "OrbNet: Deep Learning for Quantum Chemistry Using Symmetry-Adapted Atomic-Orbital Features". In: *J. Chem. Phys.* 153.12 (Sept. 2020), p. 124111. ISSN: 0021-9606. DOI: 10.1063/5.0021955.

[47] Albert P. Bartók, Risi Kondor, and Gábor Csányi. "On Representing Chemical Environments". In: *Phys. Rev. B* 87.18 (May 2013), p. 184115. DOI: 10.1103/PhysRevB.87.184115.

[48] Jörg Behler. “Atom-Centered Symmetry Functions for Constructing High-Dimensional Neural Network Potentials”. In: *J. Chem. Phys.* 134.7 (Feb. 2011), p. 074106. ISSN: 0021-9606. DOI: 10.1063/1.3553717.

[49] Haoyan Huo and Matthias Rupp. “Unified Representation of Molecules and Crystals for Machine Learning”. In: *Mach. learn.: sci. technol.* 3.4 (Nov. 2022), p. 045017. DOI: 10.1088/2632-2153/aca005.

[50] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. “Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons”. In: *Phys. Rev. Lett.* 104.13 (Apr. 2010), p. 136403. DOI: 10.1103/PhysRevLett.104.136403.

[51] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. “Machine Learning of Accurate Energy-Conserving Molecular Force Fields”. In: *Sci. Adv.* 3.5 (May 2017), p. 1603015. DOI: 10.1126/sciadv.1603015.

[52] Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. *SchNet: A Continuous-Filter Convolutional Neural Network for Modeling Quantum Interactions*. 2017. arXiv: 1706.08566 [stat.ML].

[53] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. “SchNet – A Deep Learning Architecture for Molecules and Materials”. In: *J. Chem. Phys.* 148.24 (Mar. 2018), p. 241722. ISSN: 0021-9606. DOI: 10.1063/1.5019779.

[54] Oliver T. Unke, Stefan Chmiela, Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko, and Klaus-Robert Müller. “Machine Learning Force Fields”. In: *Chem. Rev.* 121.16 (Aug. 2021), pp. 10142–10186. ISSN: 0009-2665. DOI: 10.1021/acs.chemrev.0c01111.

[55] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric Deep Learning: Going beyond Euclidean Data”. In: *IEEE Signal Process. Mag.* 34.4 (July 2017), pp. 18–42. DOI: 10.1109/msp.2017.2693418.

[56] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. *Graph Neural Networks: A Review of Methods and Applications*. 2021. arXiv: 1812.08434 [cs.LG].

[57] J. Thorben Frank, Oliver T. Unke, and Klaus-Robert Müller. *So3krates: Equivariant Attention for Interactions on Arbitrary Length-Scales in Molecular Systems*. 2023. arXiv: 2205.14276 [cs.LG].

[58] Ilyes Batatia, David Peter Kovacs, Gregor N. C. Simm, Christoph Ortner, and Gabor Csanyi. “MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields”. In: *NeurIPS*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022.

[59] Ilyes Batatia, Simon Batzner, Dávid Péter Kovács, Albert Musaelian, Gregor N. C. Simm, Ralf Drautz, Christoph Ortner, Boris Kozinsky, and Gábor Csányi. *The Design Space of E(3)-Equivariant Atom-Centered Interatomic Potentials*. 2022. arXiv: 2205.06643 [stat.ML].

[60] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. “E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials”. In: *Nat. Commun.* 13.1 (May 2022), p. 2453. ISSN: 2041-1723. DOI: 10.1038/s41467-022-29939-5.

[61] Mario Geiger and Tess Smidt. *E3nn: Euclidean Neural Networks*. 2022. arXiv: 2207.09453 [cs.LG].

[62] Oliver T. Unke and Markus Meuwly. "PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges". In: *J. Chem. Theory Comput.* 15.6 (June 2019), pp. 3678–3693. ISSN: 1549-9618. DOI: 10.1021/acs.jctc.9b00181.

[63] Oliver T. Unke, Stefan Chmiela, Michael Gastegger, Kristof T. Schütt, Huziel E. Sauceda, and Klaus-Robert Müller. "SpookyNet: Learning Force Fields with Electronic Degrees of Freedom and Nonlocal Effects". In: *Nat. Commun.* 12.1 (Dec. 2021), p. 7273. ISSN: 2041-1723. DOI: 10.1038/s41467-021-27504-0.

[64] Gregory Fonseca, Igor Poltavsky, and Alexandre Tkatchenko. *Force Field Analysis Software and Tools (FFAST): Assessing Machine Learning Force Fields under the Microscope*. 2023. arXiv: 2308.06871 [physics.chem-ph].

[65] Bruce J Berne, Giovanni Ciccotti, and David F Coker. *Classical and Quantum Dynamics in Condensed Phase Simulations*. WORLD SCIENTIFIC, June 1998. ISBN: 978-981-02-3498-0. DOI: 10.1142/3816.

[66] Johannes Kästner. "Umbrella Sampling". In: *WIREs Comput. Mol. Sci.* 1.6 (Nov. 2011), pp. 932–942. ISSN: 1759-0876. DOI: 10.1002/wcms.66.

[67] Graeme Henkelman and Hannes Jónsson. "Improved Tangent Estimate in the Nudged Elastic Band Method for Finding Minimum Energy Paths and Saddle Points". In: *J. Chem. Phys.* 113.22 (Dec. 2000), pp. 9978–9985. ISSN: 0021-9606. DOI: 10.1063/1.1323224.

[68] Julien Toulouse. *Review of Approximations for the Exchange-Correlation Energy in Density-Functional Theory*. 2022. arXiv: 2103.02645 [physics.chem-ph].

[69] Matthias Ernzerhof and Gustavo E. Scuseria. "Assessment of the Perdew–Burke–Ernzerhof Exchange-Correlation Functional". In: *J. Chem. Phys.* 110.11 (Mar. 1999), pp. 5029–5036. ISSN: 0021-9606. DOI: 10.1063/1.478401.

[70] Jing Yang, Liang Z. Tan, and Andrew M. Rappe. "Hybrid Functional Pseudopotentials". In: *Phys. Rev. B* 97.8 (Feb. 2018), p. 085130. DOI: 10.1103/PhysRevB.97.085130.

[71] H. Rydberg, M. Dion, N. Jacobson, E. Schröder, P. Hyldgaard, S. I. Simak, D. C. Langreth, and B. I. Lundqvist. "Van Der Waals Density Functional for Layered Structures". In: *Phys. Rev. Lett.* 91.12 (Sept. 2003), p. 126402. DOI: 10.1103/PhysRevLett.91.126402.

[72] M. Dion, H. Rydberg, E. Schröder, D. C. Langreth, and B. I. Lundqvist. "Van Der Waals Density Functional for General Geometries". In: *Phys. Rev. Lett.* 92.24 (June 2004), p. 246401. DOI: 10.1103/PhysRevLett.92.246401.

[73] Stefan Grimme, Jens Antony, Stephan Ehrlich, and Helge Krieg. "A Consistent and Accurate Ab Initio Parametrization of Density Functional Dispersion Correction (DFT-D) for the 94 Elements H-Pu". In: *J. Chem. Phys.* 132.15 (Apr. 2010), p. 154104. ISSN: 0021-9606. DOI: 10.1063/1.3382344.

[74] Eike Caldeweyher, Christoph Bannwarth, and Stefan Grimme. "Extension of the D3 Dispersion Coefficient Model". In: *J. Chem. Phys.* 147.3 (July 2017), p. 034112. ISSN: 0021-9606. DOI: 10.1063/1.4993215.

[75] Alexandre Tkatchenko and Matthias Scheffler. "Accurate Molecular Van Der Waals Interactions from Ground-State Electron Density and Free-Atom Reference Data". In: *Phys. Rev. Lett.* 102.7 (Feb. 2009), p. 073005. DOI: 10.1103/PhysRevLett.102.073005.

[76] Alexandre Tkatchenko, Robert A. DiStasio, Roberto Car, and Matthias Scheffler. "Accurate and Efficient Method for Many-Body van Der Waals Interactions". In: *Phys. Rev. Lett.* 108.23 (June 2012), p. 236402. DOI: 10.1103/PhysRevLett.108.236402.

[77] M. J. Frisch et al. *Gaussian 16 Revision*. 2016.

[78] G. Kresse and J. Hafner. “Ab Initio Molecular Dynamics for Liquid Metals”. In: *Phys. Rev. B* 47.1 (Jan. 1993), pp. 558–561. DOI: 10.1103/PhysRevB.47.558.

[79] G. Kresse and J. Furthmüller. “Efficiency of Ab-Initio Total Energy Calculations for Metals and Semiconductors Using a Plane-Wave Basis Set”. In: *Comput. Mater. Sci.* 6.1 (July 1996), pp. 15–50. ISSN: 0927-0256. DOI: 10.1016/0927-0256(96)00008-0.

[80] G. Kresse and J. Furthmüller. “Efficient Iterative Schemes for Ab Initio Total-Energy Calculations Using a Plane-Wave Basis Set”. In: *Phys. Rev. B* 54.16 (Oct. 1996), pp. 11169–11186. DOI: 10.1103/PhysRevB.54.11169.

[81] Paolo Giannozzi et al. “QUANTUM ESPRESSO: A Modular and Open-Source Software Project for Quantum Simulations of Materials.” In: *J. Condens. Matter Phys.* 21.39 (Sept. 2009), p. 395502. ISSN: 1361-648X 0953-8984. DOI: 10.1088/0953-8984/21/39/395502.

[82] P. Giannozzi et al. “Advanced Capabilities for Materials Modelling with Quantum ESPRESSO.” In: *J. Condens. Matter Phys.* 29.46 (Nov. 2017), p. 465901. ISSN: 1361-648X 0953-8984. DOI: 10.1088/1361-648X/aa8f79.

[83] Frank Neese. “The ORCA Program System”. In: *WIREs Comput. Mol. Sci.* 2.1 (Jan. 2012), pp. 73–78. ISSN: 1759-0876. DOI: 10.1002/wcms.81.

[84] Frank Neese. “Software Update: The ORCA Program System, Version 4.0”. In: *WIREs Comput. Mol. Sci.* 8.1 (Jan. 2018), e1327. ISSN: 1759-0876. DOI: 10.1002/wcms.1327.

[85] Frank Neese. “Software Update: The ORCA Program System—Version 5.0”. In: *WIREs Comput. Mol. Sci.* 12.5 (Sept. 2022), e1606. ISSN: 1759-0876. DOI: 10.1002/wcms.1606.

[86] Thomas D. Kühne et al. “CP2K: An Electronic Structure and Molecular Dynamics Software Package - Quickstep: Efficient and Accurate Electronic

Structure Calculations". In: *J. Chem. Phys.* 152.19 (May 2020), p. 194103. DOI: 10.1063/5.0007045.

[87] *Who We Are - FHI-aims*. https://fhi-aims.org/who-we-are.

[88] Scott A. Hollingsworth and Ron O. Dror. "Molecular Dynamics Simulation for All." In: *Neuron* 99.6 (Sept. 2018), pp. 1129–1143. ISSN: 1097-4199 0896-6273. DOI: 10.1016/j.neuron.2018.08.011.

[89] Hans C. Andersen. "Molecular Dynamics Simulations at Constant Pressure and/or Temperature". In: *J. Chem. Phys.* 72.4 (July 2008), pp. 2384–2393. ISSN: 0021-9606. DOI: 10.1063/1.439486.

[90] Sergei Izrailev, Sergey Stepaniants, Barry Isralewitz, Dorina Kosztin, Hui Lu, Ferenc Molnar, Willy Wriggers, and Klaus Schulten. "Steered Molecular Dynamics". In: *Computational Molecular Dynamics: Challenges, Methods, Ideas*. Ed. by Peter Deuflhard, Jan Hermans, Benedict Leimkuhler, Alan E. Mark, Sebastian Reich, and Robert D. Skeel. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 39–65. ISBN: 978-3-642-58360-5.

[91] Jagdish Suresh Patel, Anna Berteotti, Simone Ronsisvalle, Walter Rocchia, and Andrea Cavalli. "Steered Molecular Dynamics Simulations for Studying Protein–Ligand Interaction in Cyclin-Dependent Kinase 5". In: *J. Chem. Inf. Model.* 54.2 (Feb. 2014), pp. 470–480. ISSN: 1549-9596. DOI: 10.1021/ci4003574.

[92] Justin S. Smith, Ben Nebgen, Nicholas Lubbers, Olexandr Isayev, and Adrian E. Roitberg. "Less Is More: Sampling Chemical Space with Active Learning". In: *J. Chem. Phys.* 148.24 (May 2018), p. 241733. ISSN: 0021-9606. DOI: 10.1063/1.5023802.

[93] Nicholas J. Browning, Raghunathan Ramakrishnan, O. Anatole von Lilienfeld, and Ursula Roethlisberger. "Genetic Optimization of Training Sets for Improved Machine Learning Models of Molecular Properties." In: *J.*

Phys. Chem. Lett. 8.7 (Apr. 2017), pp. 1351–1359. ISSN: 1948-7185. DOI: 10.1021/acs.jpclett.7b00038.

[94] Logan Ward, Ben Blaiszik, Ian Foster, Rajeev S. Assary, Badri Narayanan, and Larry Curtiss. “Machine Learning Prediction of Accurate Atomization Energies of Organic Molecules from Low-Fidelity Quantum Chemical Calculations”. In: *MRS Commun.* 9.3 (Sept. 2019), pp. 891–899. DOI: 10.1557/mrc.2019.107.

[95] Florbela Pereira and João Aires-de-Sousa. “Machine Learning for the Prediction of Molecular Dipole Moments Obtained by Density Functional Theory”. In: *J. Cheminf.* 10.1 (Aug. 2018), p. 43. ISSN: 1758-2946. DOI: 10.1186/s13321-018-0296-5.

[96] Meeri Kim. “A Machine Learning Model to Predict Molecular Dipole Moments”. In: *Scilight* 2020.28 (July 2020), p. 281104. ISSN: 2572-7907. DOI: 10.1063/10.0001609.

[97] Thomas B. Blank, Steven D. Brown, August W. Calhoun, and Douglas J. Doren. “Neural Network Models of Potential Energy Surfaces”. In: *J. Chem. Phys.* 103.10 (Sept. 1995), pp. 4129–4137. ISSN: 0021-9606. DOI: 10.1063/1.469597.

[98] Frederico V. Prudente and J.J. Soares Neto. “The Fitting of Potential Energy Surfaces Using Neural Networks. Application to the Study of the Photodissociation Processes”. In: *Chem. Phys. Lett.* 287.5 (May 1998), pp. 585–589. ISSN: 0009-2614. DOI: 10.1016/S0009-2614(98)00207-3.

[99] E. Tafeit, W. Estelberger, R. Horejsi, R. Moeller, K. Oettl, K. Vrecko, and G. Reibnegger. “Neural Networks as a Tool for Compact Representation of Ab Initio Molecular Potential Energy Surfaces.” In: *J. Mol. Graph.* 14.1 (Feb. 1996), pp. 12–18. ISSN: 0263-7855. DOI: 10.1016/0263-7855(95)00087-9.

[100] Lauri Himanen, Marc O. J. Jäger, Eiaki V. Morooka, Filippo Federici Canova, Yashasvi S. Ranawat, David Z. Gao, Patrick Rinke, and Adam S. Foster. "DScribe: Library of Descriptors for Machine Learning in Materials Science". In: *Comput. Phys. Commun.* 247 (2020), p. 106949. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2019.106949.

[101] Jarno Laakso, Lauri Himanen, Henrietta Homm, Eiaki V Morooka, Marc OJ Jäger, Milica Todorović, and Patrick Rinke. "Updates to the DScribe Library: New Descriptors and Derivatives". In: *J. Chem. Phys.* 158.23 (2023).

[102] James P. Darby, James R. Kermode, and Gábor Csányi. "Compressing Local Atomic Neighbourhood Descriptors". In: *npj Comput. Mater.* 8.1 (Aug. 2022), p. 166. ISSN: 2057-3960. DOI: 10.1038/s41524-022-00847-y.

[103] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[104] Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R. Müller, and Alexandre Tkatchenko. "Quantum-Chemical Insights from Deep Tensor Neural Networks". In: *Nat. Commun.* 8.1 (Jan. 2017), p. 13890. ISSN: 2041-1723. DOI: 10.1038/ncomms13890.

[105] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[106] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[107] Patrick Reiser et al. "Graph Neural Networks for Materials Science and Chemistry". In: *Communications Materials* 3.1 (Nov. 2022), p. 93. ISSN: 2662-4443. DOI: 10.1038/s43246-022-00315-6.

[108] Weihua Hu, Muhammed Shuaibi, Abhishek Das, Siddharth Goyal, Anuroop Sriram, Jure Leskovec, Devi Parikh, and C. Lawrence Zitnick. *ForceNet: A Graph Neural Network for Large-Scale Quantum Calculations*. 2021. arXiv: 2103.01436 [cs.LG].

[109] Yi Liu, Limei Wang, Meng Liu, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. *Spherical Message Passing for 3D Graph Networks*. 2022. arXiv: 2102.05013 [cs.LG].

[110] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG].

[111] Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montúfar, Pietro Liò, and Michael Bronstein. *Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks*. 2021. arXiv: 2103.03212 [cs.LG].

[112] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV].

[113] Shuo Zhang, Yang Liu, and Lei Xie. *Molecular Mechanics-Driven Graph Neural Network with Multiplex Graph for Molecular Structures*. 2020. arXiv: 2011.07457 [cs.LG].

[114] Kamal Choudhary and Brian DeCost. "Atomistic Line Graph Neural Network for Improved Materials Property Predictions". In: *npj Comput. Mater.* 7.1 (Nov. 2021), p. 185. ISSN: 2057-3960. DOI: 10.1038/s41524-021-00650-1.

[115] C. Li, W. Wei, J. Li, J. Yao, X. Zeng, and Z. Lv. "3DMol-Net: Learn 3D Molecular Representation Using Adaptive Graph Convolutional Network Based on Rotation Invariance". In: *IEEE J. Biomed. Health Inform.* 26.10 (Oct. 2022), pp. 5044–5054. ISSN: 2168-2208. DOI: 10.1109/JBHI.2021.3089162.

[116] Yeji Kim, Yoonho Jeong, Jihoo Kim, Eok Kyun Lee, Won June Kim, and Insung S. Choi. “MolNet: A Chemically Intuitive Graph Neural Network for Prediction of Molecular Properties”. In: *Chem. Asian J.* 17.16 (Aug. 2022), e202200269. ISSN: 1861-4728. DOI: 10.1002/asia.202200269.

[117] Benjamin Kurt Miller, Mario Geiger, Tess E. Smidt, and Frank Noé. *Relevance of Rotationally Equivariant Convolutions for Predicting Molecular Properties*. 2020. arXiv: 2008.08461 [cs.LG].

[118] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. *Equivariant Message Passing for the Prediction of Tensorial Properties and Molecular Spectra*. 2021. arXiv: 2102.03150 [cs.LG].

[119] Jigyasa Nigam, Michael J. Willatt, and Michele Ceriotti. “Equivariant Representations for Molecular Hamiltonians and N-center Atomic-Scale Properties”. In: *J. Chem. Phys.* 156.1 (Jan. 2022), p. 014115. ISSN: 0021-9606. DOI: 10.1063/5.0072784.

[120] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. *Clebsch-Gordan Nets: A Fully Fourier Space Spherical Convolutional Neural Network*. 2018. arXiv: 1806.09231 [stat.ML].

[121] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. *E(n) Equivariant Graph Neural Networks*. 2022. arXiv: 2102.09844 [cs.LG].

[122] A.N. Zinoviev and K. Nordlund. “Comparison of Repulsive Interatomic Potentials Calculated with an All-Electron DFT Approach with Experimental Data”. In: *The 27th International Conference on Atomic Collisions in Solids* 406 (Sept. 2017), pp. 511–517. ISSN: 0168-583X. DOI: 10.1016/j.nimb.2017.03.047.

[123] James F. Ziegler, M.D. Ziegler, and J.P. Biersack. “SRIM – The Stopping and Range of Ions in Matter (2010)”. In: *19th International Conference on Ion Beam Analysis* 268.11 (June 2010), pp. 1818–1823. ISSN: 0168-583X. DOI: 10.1016/j.nimb.2010.02.091.

[124] Eike Caldeweyher, Sebastian Ehlert, Andreas Hansen, Hagen Neugebauer, Sebastian Spicher, Christoph Bannwarth, and Stefan Grimme. “A Generally Applicable Atomic-Charge Dependent London Dispersion Correction.” In: *J. Chem. Phys.* 150.15 (Apr. 2019), p. 154122. ISSN: 1089-7690 0021-9606. DOI: 10.1063/1.5090222.

[125] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[126] Yupeng Chang et al. *A Survey on Evaluation of Large Language Models*. 2023. arXiv: 2307.03109 [cs.CL].

[127] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005. ISBN: 978-0-262-25683-4. DOI: 10.7551/mitpress/3206.001.0001.

[128] Johannes Hoja, Leonardo Medrano Sandonas, Brian G. Ernst, Alvaro Vazquez-Mayagoitia, Robert A. DiStasio Jr., and Alexandre Tkatchenko. “QM7-X, a Comprehensive Dataset of Quantum-Mechanical Properties Spanning the Chemical Space of Small Organic Molecules”. In: *Sci. Data* 8.1 (Feb. 2021), p. 43. ISSN: 2052-4463. DOI: 10.1038/s41597-021-00812-2.

[129] Lorenz C. Blum and Jean-Louis Reymond. “970 Million Druglike Small Molecules for Virtual Screening in the Chemical Universe Database GDB-13”. In: *J. Am. Chem. Soc.* 131.25 (July 2009), pp. 8732–8733. ISSN: 0002-7863. DOI: 10.1021/ja902302h.

[130] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. “Quantum Chemistry Structures and Properties of 134 Kilo Molecules”. In: *Sci. Data* 1.1 (Aug. 2014), p. 140022. ISSN: 2052-4463. DOI: 10.1038/sdata.2014.22.

[131] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. “Enumeration of 166 Billion Organic Small Molecules

in the Chemical Universe Database GDB-17". In: *J. Chem. Inf. Model.* 52.11 (Nov. 2012), pp. 2864–2875. ISSN: 1549-9596. DOI: 10.1021/ci300415d.

[132] Stefan Chmiela, Valentin Vassilev-Galindo, Oliver T. Unke, Adil Kabylda, Huziel E. Sauceda, Alexandre Tkatchenko, and Klaus-Robert Müller. "Accurate Global Machine Learning Force Fields for Molecules with Hundreds of Atoms". In: *Sci. Adv.* 9.2 (Jan. 2023), adf0873. DOI: 10.1126/sciadv.adf0873.

[133] Justin S. Smith, Roman Zubatyuk, Benjamin Nebgen, Nicholas Lubbers, Kipton Barros, Adrian E. Roitberg, Olexandr Isayev, and Sergei Tretiak. "The ANI-1ccx and ANI-1x Data Sets, Coupled-Cluster and Density Functional Theory Properties for Molecules". In: *Sci. Data* 7.1 (May 2020), p. 134. ISSN: 2052-4463. DOI: 10.1038/s41597-020-0473-z.

[134] Yu Zhang, Peter Tino, Ales Leonardis, and Ke Tang. "A Survey on Neural Network Interpretability". In: *IEEE Trans. Emerg. Top. Comput. Intell.* 5.5 (Oct. 2021), pp. 726–742. DOI: 10.1109/tetci.2021.3100641.

[135] Rabia Saleem, Bo Yuan, Fatih Kurugollu, Ashiq Anjum, and Lu Liu. "Explaining Deep Neural Networks: A Survey on the Global Interpretation Methods". In: *Neurocomputing* 513 (Nov. 2022), pp. 165–180. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.09.129.

[136] Gregory Fonseca, Igor Poltavsky, Valentin Vassilev-Galindo, and Alexandre Tkatchenko. "Improving Molecular Force Fields across Configurational Space by Combining Supervised and Unsupervised Machine Learning". In: *J. Chem. Phys.* 154.12 (Mar. 2021), p. 124102. ISSN: 0021-9606. DOI: 10.1063/5.0035530.

[137] Xiang Fu, Zhenghao Wu, Wujie Wang, Tian Xie, Sinan Keten, Rafael Gomez-Bombarelli, and Tommi Jaakkola. *Forces Are Not Enough: Benchmark and Critical Evaluation for Machine Learning Force Fields with Molecular Simulations*. 2022. arXiv: 2210.07237 [physics.comp-ph].

[138] G. Avigad and P.M. Dey. “4 - Carbohydrate Metabolism: Storage Carbohydrates”. In: *Plant Biochemistry*. Ed. by P.M. Dey and J.B. Harborne. London: Academic Press, Jan. 1997, pp. 143–204. ISBN: 978-0-12-214674-9. DOI: 10.1016/B978-012214674-9/50005-9.

[139] Carl S. Ewig et al. “Derivation of Class II Force Fields. VIII. Derivation of a General Quantum Mechanical Force Field for Organic Compounds”. In: *J. Comput. Chem.* 22.15 (Nov. 2001), pp. 1782–1800. ISSN: 0192-8651. DOI: 10.1002/jcc.1131.

[140] Laurence Morissette and Sylvain Chartier. “The K-Means Clustering Technique: General Considerations and Implementation in Mathematica”. In: 2013.

[141] Noam Slonim, Ehud Aharoni, and Koby Crammer. *Hartigan's K-means versus Lloyd's K-means: Is It Time for a Change?* Aug. 2013, p. 1684.

[142] David Arthur and Sergei Vassilvitskii. “K-Means++: The Advantages of Careful Seeding”. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 978-0-89871-624-5.

[143] Eamonn Keogh and Abdullah Mueen. “Curse of Dimensionality”. In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2017, pp. 314–315. ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_192.

[144] fonsecag. *Machine Learning Force Fields*. May 2022.

[145] Valentin Vassilev-Galindo, Gregory Fonseca, Igor Poltavsky, and Alexandre Tkatchenko. “Challenges for Machine Learning Force Fields in Reproducing Potential Energy Surfaces of Flexible Molecules”. In: *J. Chem. Phys.* 154.9 (Mar. 2021), p. 094119. ISSN: 0021-9606. DOI: 10.1063/5.0038516.

[146] Weinan E, Weiqing Ren, and Eric Vanden-Eijnden. "Simplified and Improved String Method for Computing the Minimum Energy Paths in Barrier-Crossing Events". In: *J. Chem. Phys.* 126.16 (Apr. 2007), p. 164103. ISSN: 0021-9606. DOI: 10.1063/1.2720838.

[147] N. Aras, B.J. Oommen, and İ.K. Altınel. "The Kohonen Network Incorporating Explicit Statistics and Its Application to the Travelling Salesman Problem". In: *Neural Netw* 12.9 (Nov. 1999), pp. 1273–1284. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(99)00063-5.

[148] Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S. Yu, and Lifang He. *Deep Clustering: A Comprehensive Survey*. 2022. arXiv: 2210.04142 [cs.LG].