Bachelor Thesis

Machine Learning for Quantum Mechanical Problems

submitted in satisfaction of the requirements for the degree of
Bachelor of Science (BSc.)
of the TU Wien, Faculty of Physics

Bachelorarbeit

Maschinelles Lernen zur Lösung von Quanten-Mechanischen Problemen

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Bachelor of Science (BSc.)
eingereicht an der Technischen Universität Wien, Fakultät für Physik

von

Simon König

Matr.Nr.: 01234567

unter der Anleitung von

Univ.Prof. Mag.rer.nat. Dr.rer.nat. **Andreas Grüneis**

Institut für Theoretische Physik
Forschungsbereich
Technische Universität Wien
Wiedner Hauptstraße 8, 1040 Wien, Österreich

Wien, im Mai 2021

Abstract

The thesis centres around the analysis of quantum-mechanical environments. Atomic Energies and forces are determined for given atomic configurations using *Machine Learning (ML)* algorithms. In physics the computation of atomic energies and forces is an important field. This work focuses on building a model, based on *ML* methods, able to predict energies and forces for given atomic systems.
In order to use *Machine Learning* to solve problems, it is necessary to realise that these methods do not perform any complex computation describing the atomic environment. A *ML* model needs representative groundwork, so called, *Training Data* mirroring the system of interest. The model *learns* on the *Training Data* and is then able to make predictions on systems similar to the ones it has been trained on.

This work first discusses the computation of *Training Data* as foundation for the *Machine Learning* algorithm. In this case the *Training Data* consists of value pairs. One value pair maps an atomic configuration to its features, the atomic energies and forces. *Training Data* is generated using *Molecular Dynamics (MD) Simulations*, making use of *Density Functional Theory* as calculation tool.
The following part explains the *Machine Learning* algorithm and its specifications. The *ML* model processes information and yields predictions based on the *Gaussian Regression Process* algorithm, a stochastic mechanic centred around *Gaussian* probability distributions. The model is then created using the quantum-mechanically computed *Training Data* as foundation and a *Gaussian Process* algorithm as mechanic.

To optimise the prediction made by the model and thus improve resemblance of reality, the construction of the model is plugged into an error minimising feedback loop. This circular optimisation process tunes the model parameters to optimise results by reduction of the offset between model predictions and reality.
The refined results are presented and visualised in line with referring to future possibilities of *ML* mechanics describing atomic systems.

Contents

1 Introduction **6**

2 Theoretical Basis **7**

2.1 Many Body Problem of Quantum Theory . . . 7

2.1.1 Born Oppenheimer Approximation . . . 8

2.1.2 Hartree and Hartree-Fock methods . . . 9

2.2 Density Functional Theory . . . 10

2.2.1 Motivation . . . 10

2.2.2 Energy as Functional of Density . . . 10

2.3 Solutions for DFT . . . 11

3 Machine Learning **13**

3.1 Motivation . . . 13

3.1.1 From Density Functional Theory to Machine Learning . . . 13

3.1.2 Relationship of Atomic Systems and their Energy . . . 13

3.2 Gaussian Process Regression . . . 14

3.2.1 Prerequisites . . . 14

3.2.2 Building the Kernel . . . 14

3.2.3 Making Predictions . . . 15

3.3 Modifications of Gauss Process Regression for Quantum Mechanical use . . . 16

3.3.1 Total Energies and Forces . . . 16

3.3.2 Local Approximation - Cutoff Radius . . . 16

3.3.3 Sparsification - Representative Data . . . 17

3.3.4 Descriptors - Representing the Atomic Neighbourhood Environment . . . 18

4 Machine Learning Model Creation **19**

4.1 Observed Systems . . . 19

4.1.1 Water Molecule H2 . . . 19

4.1.2 Water Crystal . . . 19

4.2 Workflow . . . 20

4.2.1 Flow-Chart . . . 20

4.2.2 Get Data from Molecular Dynamics Simulation . . . 20

4.2.3 Split Data into Training and Validation Data . . . 21

4.2.4 Model Training . . . 21

4.2.5 Model Prediction . . . 24

4.2.6 Model Optimisation . . . 24

4.2.7 Water Crystal Model Creation . . . 26

5 Outcome and Analysis **27**

5.1 Water Molecule H2 . . . 27

5.1.1 Outcome Visualisation . . . 27

5.1.2 Model Parameter Analysis . . . 28

5.2 Hydrogen Crystal 30
5.2.1 Outcome Visualisation 30
5.2.2 Model Parameter Analysis 31

6 Summary and Outlook **33**
6.1 Model Improvement by Descriptor Adjustment 33
6.2 Larger Systems 33

Appendix **35**
A Preparation of Training Data 35
B Software 35
C Code 35
D Optimisation 35

Bibliography **36**

Chapter 1

Introduction

One of the main challenges in modern physics is the modelling of atomic environments and the determination of related quantum-mechanical features of interest, for example atomic energies and forces. These microscopic properties determine a lot of macroscopic material properties, like thermo- and electro-conductivity or elasticity. The Computation of quantum-mechanical environments is conducted using approximations, like the *Born-Oppenheimer-Approximation*, and *Self Consistent Methods*. In spite of these simplification, quantum-mechanical practices describing atomic environments, in this work the *Density Functional Theory (DFT)*, are not suited for large systems because of high computational cost. Theories, like the *DFT*, return correct results but lead to major computational cost.

This thesis deals with reducing the computational cost. The centre point for minimising runtime of analysis of atomic features is avoiding quantum-mechanical calculations and introducing a *Machine Learning (ML) Model* to make predications instead of computations. In general a *ML* model takes a set of expensively computed *Training Data* as input and is then able to predict quantum-mechanical properties based on the previously gained knowledge. In this thesis the practice of *learning* and *predicting* is examined on the example atomic systems of a *Hydrogen Molecule* H_2 and a *Hydrogen Crystal* containing 192 Atoms.

Chapter 2

Theoretical Basis

2.1 Many Body Problem of Quantum Theory

Solving the *Schrodinger Equation* is one of the fundamental problems of modern physics. The time independent equation is given in equation 2.1. The *Schrodinger Equation* is defined as Eigenvalue problem, demanding a set of *Wave-Functions* as solution. Applying the Hamiltonian on a set of *Eigen-Wave-Functions* yields an associated energy.

$$\hat{H} \left|\Psi\right\rangle = E \left|\Psi\right\rangle \tag{2.1}$$

- $\hat{H}$ ... Hamiltonian
- $\left|\Psi\right\rangle$ ... Basis-Wave-Function in Dirac Notation, eigenfunction of the *Schrodinger Equation*
- E ... Energy of the system, eigenvalue of the *Schrodinger Equation*

The *Hamiltonian* is the quantum-mechanical analogy to the classical concept of mechanical energy. Equation 2.2 states that the *Hamiltonian* is the sum of the *Kinetic Energy T* and the *Potential Energy V*, here denoted as Operators.

$$\hat{H} = \hat{T} + \hat{V} \tag{2.2}$$

Solving the *Eigenvalue Problem* analytically is possible for a small number of objects, for example one electron or an *Helium Atom.* Real problems relate to large systems with a large amount of objects. The solution of the *Schrodinger Equation* becomes unsolvable analytically and numeric methods come into place. The numerical method to solve the *Many Body Schrodinger Equation* in equation 2.3 used in this work is the *Density Functional Theory (DFT).*

In atomic units, a system of N_n nuclei and N_e electron is described by the *Many Body Schrodinger Hamiltonian* in [Lecture 4 18, p. 2]:

$$\hat{H} = -\frac{1}{2}\sum_{i}^{N_e} \nabla_i^2 - \sum_{i}^{N_e}\sum_{I}^{N_n} \frac{Z_I}{|\vec{r_i} - \vec{R_I}|} + \frac{1}{2}\sum_{i}^{N_e}\sum_{i \neq j}^{N_e} \frac{Z_I}{|\vec{r_i} - \vec{r_j}|} + \frac{1}{2}\sum_{I}^{N_n}\sum_{I \neq J}^{N_n} \frac{Z_I Z_J}{|\vec{R_i} - \vec{R_j}|} \tag{2.3}$$

- $\hat{H}$ ... Hamiltonian
- r_i, r_j ... position of i_{th}, j_{th} electron
- R_i, R_j ... position of i_{th}, j_{th} nucleus
- Z_i, Z_j ... charge of i_{th}, j_{th} nucleus

As precursors to the *DFT* two approximations are discussed int the following. The ideas of the presented methods offer an approach to finding a numerical solution to the *Many Body Schrodinger Equation* in equation 2.3 and lay the foundation for *DFT*.

The necessary approximations are:

- Born-Oppenheimer Approximation
- Hartree and Hartree-Fock Approximations

2.1.1 Born Oppenheimer Approximation

Born and Oppenheimer suggest, because of the significant mass difference of electron and nucleus to neglect the movement of the nuclues in respect to the electron.s The underlying idea is that the electrons, because of their mass being lower than that of the nucleus by a magnitude of 10^3, relax to an equilibrium state rapidly for any movement of the nucleus. Any movement of the nucleus does not affect the energy of the electrons as it is assumed the e^- move to equilibrium state instantly. Details to the idea of Born and Oppenheimer are to be found in [5, p. 272].

The Kinetic Energy Operator $\hat{T}$ of a *Center-of-Momentum-Frame* with one nucleus and N electrons is in equation 2.4. Mathematically the idea can be expressed by performing the limit M $\rightarrow \infty$.

$$\hat{T} = -\frac{\hbar^2}{2(M+Nm)}\nabla^2_{CM} - \frac{\hbar}{2\mu}\sum_i^N \nabla_i^2 - \frac{\hbar^2}{M}\sum_{i>j}^N \nabla_i\nabla_j \tag{2.4}$$

Equation 2.4 is reduced to equation 2.5 with the *Born-Oppenheimer Approximation.* Mathematically the idea can be expressed by performing the limit M $\rightarrow \infty$.

$$\hat{T} = -\frac{\hbar}{2m}\sum_i^N \nabla_i^2 \tag{2.5}$$

- $\hat{T}$ ... Kinetic Energy
- M ... Mass of nucleus
- m ... Mass of electrons
- N ... Number of electrons
- ∇_{CM} ... Nabla operator with respect to the centre of mass
- ∇_i, ∇_j Nabla operator with respect to the i_{th}, j_{th} electron
- $\mu = \frac{Mm}{M+m}$ ... Reduced mass

The concept is introduced for a system of one nucleus and N electrons and it then expanded to any arbitrary Hamiltonian describing N_n nuclei and N_e electrons in equation 2.3. The Born-Oppenheimer approximation is one of the main ideas building and instrumental to *Density Functional Theory.*

2.1.2 Hartree and Hartree-Fock methods

Hartree and in the following *Hartee & Fock* make an ansatz for the *Many Electron Wave Function.* The last term, describing the nuclei-nuclei interaction, of the *Many Body Hamiltonian* in equation 2.3 is assumed to be constant with the Born Oppenheimer Approximation. The *Many Body Hamiltonian* is reduced to equation 2.6.

$$\hat{H} = -\frac{1}{2}\sum_{i}^{N_e}\nabla_i^2 - \sum_{i}^{N_e}\sum_{I}^{N_n}\frac{Z_I}{|\vec{r_i} - \vec{R_I}|} + \frac{1}{2}\sum_{i}^{N_e}\sum_{i \neq j}^{N_e}\frac{Z_I}{|\vec{r_i} - \vec{r_j}|} \tag{2.6}$$

The first two terms of equation 2.3 are trivial since they are equivalent to the solution of the *One Body Schrodinger Equation.* Solving the eigenvalue problem for the first and the second term leads to a separable differential equation. The first term in equation 2.3 is the Kinetic Energy $\hat{T}_e$ for the i_{th} electron and the second the term is the Potential Energy $\hat{V}_en$ for the i_{th} electron in respect to the I_{th} nucleus. The solution for the eigenfunctions here is a product of the wave functions, solving the *One Body Schrodinger Equation.* [Lecture 4 18, p. 1]

The third term represents the Potential Energy $\hat{V}_ee$ and the denominator term entangles the coordinates of the i_{th} and the j_{th} electron.

$$\hat{V}_{ee} = \frac{1}{2}\sum_{i}^{N_e}\sum_{i \neq j}^{N_e}\frac{1}{|\vec{r_i} - \vec{r_j}|} \tag{2.7}$$

The differential equation is not separable for equation 2.7, this is where *Hartrees* approximation applies. The main idea is that the potential applying to any electron is identical for every electron of the system and is caused by all the other electrons and nuclei. An ansatz for V_{ee} is made in equation 2.8 with the potential being proportional to the proximity density of the wave functions ψ_i. This is called the *Mean Field Approximation.*

$$\hat{V}_{ee}(\vec{r_i}) = \sum_{i \neq j}^{N_e}\frac{|\psi_{ij}|^2}{|\vec{r_i} - \vec{r_j}|} \tag{2.8}$$

This approximation leads to a problem in computing the potential, thus also the Hamiltonian of the *One Body Schrodinger Equation.* The solution of the partial eigenvalue function in equation 2.9 is self dependent. The Potential V_{ee} on the left side of the equation depends on the solution of the equation - the wave functions ψ_i. [17, p. 77]

$$\hat{V}_{ee}(\vec{r_i})\psi_i = E\psi_i \tag{2.9}$$

The same problem occurs in *Density Functional Theory* and is solved by using a self-consistent method. This means looping through an algorithm by making an educated guess for the eigenfuctions then computing the potential and using the potential to compute new eigenfuctions.

A possible initial wave function to solve the *Partial Schrodinger Equation* in equation 2.9 is proposed by *Hartee.* The ansatz to initialise a *self-consistent* solving mechanism is a product of all the N *Single-Electron-Wave-Functions* in equation 2.10. [17, p. 78]

$$\psi_i(\vec{r}_1, ..., \vec{r}_N) = \psi_{i1}(\vec{r}_1)\psi_{i2}(\vec{r}_2)...\psi_{iN}(\vec{r}_N) \tag{2.10}$$

The idea was later improved by *Hartee and Fock*, by conforming to the *Pauli Principle.* Electrons are Fermions and have to apply to the *Parity Operator* $\hat{P}$. The Pauli Principle demands the wave functions to be anti-symmetric. The revised solution for the wave function is a linear combination of *Slater-Determinants*, where the entries of the *Slater Determinant* are product wave functions of the *Single-Electron-Wave-Functions.* Details in [Lecture 4 18, p. 2].

Hartree and Hartree-Fock solve the *Many Body Schrodinger equation* numerically using a self-consistent method. The same concept is used in *Density Functional Theory.* The *Self Consistent* algorithm is an iteratively repeated process.

1. Ansatz for eigen functions
2. Potentials are determined by computing the probability density of the eigenfunctions: $|\psi_{ij}|^2$
3. New eigenfunctions are computed with the given Potential

The algorithm is an iterative process and is repeated until a *Self-Consistent* solution is found.

2.2 Density Functional Theory

2.2.1 Motivation

The solution of the *Many Body Schrodinger Equation* has become feasible after applying *Hartree & Focks* and *Born & Oppenheimers* approximations. With Hartree the eigenfunctions of the N-body problem separate into a product of N orbitals for every spacial coordinate. One electron equals the observation of the 3 cartesian coordinates, N electrons equal the observation of $3N$ coordinates. The *Hartree & Fock* method provides a reduction in coordinates compared to 3^N orbitals necessary for solving the *Schrodinger Equation* exactly. For a real *Many Body Problem* the computation in $3N$ coordinates is expensive. An CO_2 Molecule for instance, consists of 22 electrons, the related *Schrodinger Equation* is a 66-dimensional problem.

$$CO_2 \cong 22 \text{ electrons} \tag{2.11}$$

The configuration of a lead unit cell contains about 100 Atoms. One Pb-Atom possesses 14 electrons, which counts up to 1400 electrons per unit cell. The eigenvalue problem has to be solved for 4200 dimensions.

$$Pb_{unitcell} \cong 1400 \text{ electrons} \tag{2.12}$$

Using *Quantum Chemistry* methods, like the *Møller–Plesset perturbation theory*, computation time sums up to about one year for the lead unit cell. *Density Functional Theory* reduces computation time significantly and the *Many Body Schrodinger Equation* for the lead unit cell can be solved in about 5 hours. Informations on this are derived from [19]

2.2.2 Energy as Functional of Density

In the article, related to the honour of his nobel price achievement, Walter Kohn writes:

"The basic lemma of HK. The ground-state density n(r) of a bound system of interacting electrons in some external potential v(r) determines this potential uniquely (Hohenberg and Kohn, 1964)" [1, p. 7]

This Lemma is the substantial concept underlying *Density Functional Theory.* Kohn proposes the ground state energy of any quantum system depends solely on the electron density $n(r)$. For any electron configuration the *Many Body Schrodinger Equation* is a problem in 3 coordinates. The Hamiltonian in Born-Oppenheimer Approximation is given in equation 2.13. [3, p. 6]

$$\hat{H} = \hat{T} + \hat{V}_{ext} + \hat{V}_{ee} \tag{2.13}$$

- $\hat{T}$ ... Kinetic Energy of the *free* electrons
- $\hat{V}_{ext}$... External Potential as functional of the electron density $\vec{n}(r)$
- $\hat{V}_{ee}$ ... Potential describing the electron-electron-interaction

The ideas of *DFT* proposed by *Kohn, Hohenberg & Sham* in 1964 rest on two fundamental theorems:

1. $\hat{V}_{ext}$ is, apart from a constant term, determined uniquely as a functional of the electron density of the system: $\vec{n}(r)$
2. The functional V_{ext} has its Minimum in respect to a variation of the electron density $\partial\vec{n}(r)$ for a density $\vec{n}_0(r)$ for a given external potential V_{ext}.

2.3 Solutions for DFT

A general formula for the energy based on the theorems by *Kohn, Hohenberg & Sham* in equation 2.14 as denoted by [3, p. 6]

$$E_0 = E_V(n_0) = T[n_0] + \int V_{ext}(r)[n_0](r)dr + J[n_0] + E_{NC}[n_0] \tag{2.14}$$

uses the terms:

- E_0 ... Ground State Energy
- n_0 Ground State Electronic Density
- $\hat{V}_{ext}[n_0]$ ... External Potential
- $J[n_0]$ ... Classical Coulomb Energy
- $E_{NC}[n_0]$ ... Non-classica electron electron Energy.

Taking a step back and having *Hartree* in mind, finding the ground state energy has to be done by a self consistent method. *Hartrees* idea is applied by the *DFT*, the electron density is determined by the sum over the density of the probability of presence:

$$n(r) = \sum_i^N |\psi_i(r)|^2$$

The *DFT* introduces the concept of the Exchange Correlation Functional given by equation 2.15 with $T_s[n]$ as the Kinetic Energy for the non-interacting electrons.

$$E_{XC}[n] = T[n] - T_s[n] + E_{NC}[n] \tag{2.15}$$

and the related potential by:

$$V_{XC}[n] = \frac{\delta E_{XC}[n]}{\delta n(r)} \tag{2.16}$$

With the *Exchange Correlation Energy* $E_{XC}[n]$, the total Energy of the system is reduced to equation 2.17 with

$$E_V(n_0) = T_s[n_0] + \int V_{ext}(r)[n_0](r)dr + J[n_0] + E_{XC}[n_0] \tag{2.17}$$

The equations 2.17, 2.16 and 2.15 lead to the *Kohn Sham Equations* in equation 2.18 used to determine the eigenfunctions $\psi(r)$. Information is taken from [16, p. 161].

$$\left[T_s + V(r) + \int \frac{n_0(r')}{|r - r'|}dr' + V_{XC}(n_0)\right] \psi_i = \epsilon_i \psi_i \tag{2.18}$$

The solution the Kohn Sham equations are the *Kohn-Sham-Orbitals* and have to be found in a *Self Consistent* fashion. The eigenvalue problem is a generalised *One Body Schrodinger Equation*, with ϵ_i as *Lagrange Multiplier* obtaining orthogonality for the *Kohn-Sham-Orbitals*.

Chapter 3

Machine Learning

3.1 Motivation

3.1.1 From Density Functional Theory to Machine Learning

The description of atomic systems based on *Density Functional Theory*, presented in chapter 2, results in significant reduction in computational cost compared to the exact solution of the *Many Body Schrodinger Equation*. In general quantum-mechanical systems are highly complex and diverse, in a way that exerting *DFT* transcends the computational resources. Further the solutions by *DFT* are limited to a precise system for which the calculations were conducted and need to be recalculated once the system changes. The motivation for utilising *Machine Learning* is the creation of a *Potential Energy Surface (PES)*, meaning a general description of a quantum-mechanical system inside specific boundaries. This description is not susceptible to small changes in configuration and describes the system as a whole. Details for *PES* can be found in [8].

Machine Learning tries to model a *PES*, describing the energy of the atoms of a system as function of their environment. This *atomic-neighbourhood-environment* reacts sensitive to small changes of the atomic environment, for example the change of an atomic species or atomic quantity. [10, p. 1051]

The *Machine Learning Algorithm* introduced in this work is not per se able to find quantum properties, like energies and forces, for atomic systems. The solutions of an *ML Model* are based on Training Data the model, upon which a model makes classifications. Training Data needs to be generated quantum-mechanically, in this work it is computed using *DFT*.

The stochastic tool, underlying the *ML Model* is the *Gaussian Process Regression (GAP)*. Note that *ML Models*, here *GAP* in specific, do not offer good extrapolations of problems. Valid output is generated for interpolation tasks. General informations on this are to be found in [2].

3.1.2 Relationship of Atomic Systems and their Energy

Predicting the quantum-mechanical properties of atomic systems makes use of a stochastic framework. Other possibilities include utilising *Neuronal Networks (NN)*. The stochastic tool used in this work is the *Gaussian Process Regression (GAP)*.

To make predictions through Regression, the atomic energies of a set of atoms are introduced as a function of their configuration, this is their euclidean geometry. The hallmark of the interatomic potential is that the energy consists of a sum of energy functionals lying inside a, later introduced *cutoff-radius* and negligible long-range terms. This definition is found in [10, p. 1051]

$$E = \sum_{\alpha} \sum_{i \in \alpha} \epsilon_i^{\alpha} + lrt \tag{3.1}$$

- E ... Total Energy of atomic system
- ϵ_i^{α} ... Local Energy Functionals within a *cutoff radius*
- α ... type of contribution to the Total Energy, descriptor type
- i ... counts instances of the Local Energy Contributions
- lrc ... *Long Range Contributions* outside a *cutoff radius*, include *Polarisability*, *Van-der-Walls Forces*, etc.

The contributions ϵ_i^{α} range over a α, each denoting a specific type of Energy Functional. These Energy Functionals take, so called, *descriptors* as arguments and describe the atomic energy as functional of the atomic configuration. *Descriptors* characterise the given atomic constellation by mapping the cartesian coordinates of the atomic environment to a descriptor feature space. This descriptor feature space connects an atomic energy to an atomic constellation. This value pair serves as input for the *ML Model*. Details are in [10].

3.2 Gaussian Process Regression

3.2.1 Prerequisites

Gaussian Processes are used to handle regression problems. The data input consists of N data points $\mathbf{X}_n, \mathbf{t}_n = \{x^{(n)}, t_n\}_{n=1}^{N}$. The list of N input points $\mathbf{X}_n$ exists in the space of an H-dimensional fixed set of basis functions $\{\phi(\mathbf{x})\}_{h=1}^{N}$ at the points $\{\mathbf{x}_n\}$ and is denoted by the Matrix

$$R_{nh} \equiv \phi_h(\mathbf{x}^{(n)})$$

the target value vector $\mathbf{y}_n$ is defined by

$$y_n \equiv \sum_h R_{nh} w_h$$

with the weights w_n being Gauss distributed with mean zero.

$$P(\mathbf{w}) = Normal(\mathbf{w}; 0; \sigma_w^2 \mathbf{I}).$$

With [2, p. 540] the covariance matrix of $\mathbf{y}$ is introduced as:

$$\mathbf{Q} = \langle \mathbf{y}\mathbf{y}^T \rangle = \langle \mathbf{R}\mathbf{w}\mathbf{w}^T\mathbf{R}^T \rangle = \mathbf{R}\langle \mathbf{w}\mathbf{w}^T \rangle \mathbf{R}^T = \sigma_w^2 \mathbf{R}\mathbf{R}^T$$

3.2.2 Building the Kernel

The main assumption is that the weights $\mathbf{w}$ and consequently the vector $\mathbf{y}$are Gauss distributed. This condition holds for any selection of points $\mathbf{X}_N$. The target values$t_n$are assumed to differ from the correlated function value y_n by a Gaussian noice of σ_ν^2. This determines the target value vector $\mathbf{t}$ to be Gaussian distributed.

$$P(\mathbf{w}) = Normal(\mathbf{w}; 0; \mathbf{Q} + \sigma_\nu^2 \mathbf{I})$$

The covariance matrix of $\mathbf{t}$ is introduces as $\mathbf{C}$.

$$\mathbf{C} = \mathbf{Q} + \sigma_\nu^2 \mathbf{I} = \sigma_w^2 \mathbf{R}\mathbf{R}^T + \sigma_\nu^2 \mathbf{I} \tag{3.2}$$

Every Data Set $\mathbf{X}$ living in an arbitrary basis $\mathbf{H}$ induces a Covariance Matrix, also *Kernel*. The Covariance Matrix is then extended by the other part of the Training Data Set, the target values $\mathbf{t}$. Note that in this work the Data Set $\mathbf{X}$ contains the atomic configurations, more specific the descriptor values of the atomic configurations. The target values $\mathbf{t}$ are populated by atomic energies calculated by *DFT*. Details on the creation of the *Kernel* are found in [2, p. 540]

In general the choice of the Covariance Matrix, that is the choice of the Set of H basis functions is not predetermined. The only constraint is that it must not be negative definite. In this work a *Squared Exponential Kernel* is used. [9, p. 83]

3.2.3 Making Predictions

The model makes predictions based on the *Gaussian Process Algorithm* using a Data Set consisting of N data points $\mathbf{X}_N$ and $\mathbf{t}_N$. The task is to make a prediction for value t_{N+1}. The Covariance Matrix $\mathbf{C}_{N+1}$ in equation 3.3 is the extended Kernel for the target vector $\mathbf{t}_{N+1}$.

$$\mathbf{C}_{N+1} \equiv \begin{bmatrix} \begin{bmatrix} \mathbf{C}_N \end{bmatrix} & \begin{bmatrix} \mathbf{k} \end{bmatrix} \\ \\ \begin{bmatrix} \mathbf{k}^T \end{bmatrix} & \begin{bmatrix} \kappa \end{bmatrix} \end{bmatrix} \tag{3.3}$$

(Ask Andreas if this is detailed enough or if couple of more steps should be included)
After some conversions detailed in [2, p. 543] the predicted $\hat{t}_{N+1}$ is given by

$$\hat{t}_{N+1} = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t}_N \tag{3.4}$$

and the error $\sigma^2_{\hat{t}_{N+1}}$ on the prediction

$$\sigma^2_{\hat{t}_{N+1}} = \kappa - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k} \tag{3.5}$$

with

- $\mathbf{C}_N^{-1}$ ... Inverse Kernel of Kernel in equation 3.2
- $\mathbf{k}$ ... Vector containing mapping t_{N+1} to $\mathbf{t}$, part of matrix $\mathbf{C}_{N+1}$, $\mathbf{k} \equiv \langle t_{N+1}, \mathbf{t} \rangle$
- $\mathbf{t}_N$ ... Target value vector of Training Data
- κ ... Intrinsic value related to unknown target t_{N+1}, needed to scale the error, part of matrix $\mathbf{C}_{N+1}$, $\kappa \equiv \langle t_{N+1}, t_{N+1} \rangle$

The power of the *Gaussian Process* lies within not having to invert the Matrix $\mathbf{C}_{N+1}$ to make predictions at configuration $\mathbf{x}^{N+1}$. Omitting the inversion of $\mathbf{C}_{N+1}$ allows for the creation of a model using an arbitrary large number H of Basis Functions. This is because the inversion of $\mathbf{C}_N$ always scales with N^3 independent of H. Raising the number of Basis Function may improve the accuracy of the model. This is because the inversion of $\mathbf{C}_N$, in equation 3.2always scales with N^3 independent of H.

3.3 Modifications of Gauss Process Regression for Quantum Mechanical use

The *Density Functional Theory* in chapter 2 and the *Gaussian Process Regression* in chapter 3.2 lay the foundation for building a model and creating an Interatomic Potential. Data generated by *DFT* is input into a *Gaussian Regression Machine* and an atomic system is modelled. The *Machine Learning* fundamentals need to be augmented and further optimised for quantum mechanical use.

3.3.1 Total Energies and Forces

The first extension concerns the computation of *Total Energies* and *Total Gradients* of those. In chapter 3.2 the *Gaussian Process Regression* was fit using per atom energies. This simplification does not hold, because just *Total Energies* of systems are known and generated by *DFT*. In equation 3.6 *Gaussian Noise* ν_E is added to the *Total Energy* of the system from equation 3.1. Procedural the computation of the Kernel is identical to equation 3.2 but summarising over all N local energies of the system. [10, p. 1053]

$$E = \sum_{\alpha} \sum_{i} \epsilon_i^{\alpha} + \nu_E \tag{3.6}$$

$$\langle E_N E_{N'} \rangle = \sigma_w^2 \sum_{i \in N} \sum_{j \in N'} C(\mathbf{d}_i, \mathbf{d}_j) + \sigma_E^2 \delta_{NN'} \tag{3.7}$$

Computing forces is straightforward. The *Total Forces* relating to the *Total Energies* are the Gradient with respect to any atom k of the system and the general coordinates ξ. *Gaussian Noise* is added and denoted by ν_ξ

$$\frac{\partial E}{\partial \xi_k} = \frac{\partial \sum_\alpha \sum_i \epsilon_i^\alpha}{\partial \xi_k} + \nu_\xi \tag{3.8}$$

The computation of the derivatives can be written as

$$\frac{\partial^2 \langle E_N E_{N'} \rangle}{\partial \xi_k \partial \chi_l} = \sigma_w^2 \sum_{i \in N} \sum_{j \in N'} \frac{\partial \mathbf{d}_i^T}{\partial \xi_k} (\nabla_{\mathbf{d}_i} C(\mathbf{d}_i, \mathbf{d}_j) \nabla_{\mathbf{d}_j}^T) \frac{\partial \mathbf{d}_j}{\partial \chi_l} + \sigma_E^2 \delta_{NN'} \delta_{\xi_k \chi_l} \tag{3.9}$$

Notice that for the all the equations 3.7, 3.8 and 3.9 a term representing *Gaussian Noise* is added. This is outlined in chapter 3.2.2 and is due to an assumed *Gaussian* error of the target values $\mathbf{t}$.

3.3.2 Local Approximation - Cutoff Radius

To reduce computational cost of energies and their derivatives compact support is needed. In order to achieve this, a *Cutoff-Radius d* is introduced. The regarded local environment of every atom of a system is geometrically limited to the *Cutoff-Radius* $r_c ut$. The compact support for the local energies is provided by adding a cutoff function f_{cut}

$$\epsilon(\mathbf{d}_i, w) = f_{cut} \sum_h w_h \phi_h(\mathbf{d}_i) \tag{3.10}$$

with the cutoff function, used by the *GAP* and *QUIP* software, referenced in Appendix 6.2

$$f_{cut} = \begin{cases} 1, & r \leq r_{cut} - d \\ \frac{\left[cos\left(\pi \frac{r - r_{cut} + d}{d}\right) + 1)\right]}{2}, & r_{cut} - d < r \leq r_{cut} \\ 0, & r > r_{cut} \end{cases} \tag{3.11}$$

The *Cutoff-Radius* r_cut may be changed, depending on the required precision of the model. The parameter d can also be set manually, if needed, but is typically set to 1 Å as the typical atomic length scale. Note that a larger r_cut may lead to a better fit but an increase in computation time. The concept is explained in [10].

3.3.3 Sparsification - Representative Data

Further reduction of computational cost is achieved by applying a common *Machine Learning* concept, *Sparsification.* This practice tries to lower the dimension N of the Covariance Matrix by selecting a set of *Sparse Points*, the *Sparse Configuration. Sparsification* projects all of the input data onto a subset of *Sparse Configurations*, if done correctly, with very little loss of information. One can think of the*Sparse Configuration* as resemblance of the *Training Data Set*, by selection of a highly representative set of points. Similarly to the the *cutoff-radius* in chapter 3.3.2, the size of the *sparse set* is variable. A small number of *sparse points* reduces the computational cost of the model significantly but may also result in a loss of precision in prediction.

Let the number of *Sparse Points* be M and the number of *Data Points* in the original data set N. The Covariance Matrix C_{NN} is computed as detailed in chapter 3.2 for the original input configurations $\mathbf{x}_N$. [7, Appendix]

$$\mathbf{C}_{NN} = \langle E_N E_{N'} \rangle \tag{3.12}$$

Similarly, the Covariance Matrix C_{MM} is built for the set of *Sparse Configurations* $\mathbf{x}_M$

$$\mathbf{C}_{MM} = \langle E_M E_{M'} \rangle \tag{3.13}$$

and the Covariace Matrix C_{MN} mapping the *Sparse Configuration* to the original data configuration:

$$\mathbf{C}_{NM} = \langle E_N E_M \rangle \tag{3.14}$$

Putting it all together, the predicted value $t_{N+1}(x_{N+1})$ at a new atomic configuration x_{N+1} is written as

$$t_{N+1}(x_{N+1}) = \mathbf{k}^T(\mathbf{C}_{MM} + \mathbf{C}_{MN}\Lambda^{-1}_{MM}\mathbf{C}_{NM})^{-1}\mathbf{C}_{MN}\Lambda^{-1}_{MM}\mathbf{t} \tag{3.15}$$

with the final constituents

- $\mathbf{k}$ ... vector containing mapping of t_{N+1} to $\mathbf{t}$, $\mathbf{k} \equiv \langle t_{N+1}, \mathbf{t}_M \rangle$
- $\mathbf{C}_{MM}$ ... Covariance matrix of *Sparse Configuration*
- $\mathbf{C}_{MN}$ ... Covariance matrix of *Sparse Configuration* and *Original Configuration*
- $\mathbf{t}$ ... Target Values of *Original Configuration*
- Λ^{-1}_{MM} ... Diagonal Matrix, containing the *Gaussian Noise* terms σ^2_E for the energies and σ^2_ζ for the forces respectively.

The final form of the *Gaussian Process* in equation 3.15 combines the computational cost saving mechanisms derailed in this chapter. Note that computational cost for each prediction scales with the number M of *Sparse Points* since multiplying $\mathbf{t}$ from the right in equation 3.15 has been precomputed at the training stage of the model, explained in [10, p. 1054]. The effects of *Sparsification* and a *Gauss* distributed Kernel allow for a fast *Machine Learning* model.

3.3.4 Descriptors - Representing the Atomic Neighbourhood Environment

The *Training Data* for the *Machine Learning* model introduced in 3.2 consists of the N data points $\mathbf{X}_n, \mathbf{t}_n = \{x^{(n)}, t_n\}_{n=1}^{N}$. The set $\mathbf{X}_n$ mirrors the raw atom positions of the quantum mechanical system, simulated by *Density Functional Theory.* The set $\mathbf{t}_n$ are the target values for the *ML* process, these are atomic energies and forces.

Outlined in chapter 3.1, the *Machine Learning* model uses energy functionals for its calculations, taking *Descriptor* values as arguments. Descriptors map the atomic configuration in cartesian coordinates to a feature space that is used as input data for building the model. The transformed data in the *Descriptor* space is denoted by $\mathbf{X}_n$. An appropriate transformation of the raw input data to a descriptive feature space determines the success of the *ML* model. [10, p. 1055]

In general a descriptor is a function applied to an atom configuration and converting it to a feature space. Possible descriptions of an atomic neighbourhood environment take interatomic distances or angles into account. More complex descriptors include representing an atomic environment by *Neighbourhood Density Functions.*

In this work a simple *2-body-distance* descriptor, in equation 3.16 is utilised. This descriptor characterises atomic neighbourhood environments by the euclidean distance between atoms.

$$\mathbf{d}_{ij} = |\mathbf{r}_i - \mathbf{r}_j| \tag{3.16}$$

- $\mathbf{d}_{ij}$... interatomic distance
- $\mathbf{r}_i, \mathbf{r}_j$ ... position of the *ith*, *jth* atom

Chapter 4

Machine Learning Model Creation

4.1 Observed Systems

4.1.1 Water Molecule H2

The first observed and analysed system consists of two *Hydrogen* atoms forming a molecule. H_2 is pictured in cartesian coordinates in figure 4.1. The atom configurations stem from a *Molecular Dynamics (MD) Simulation* computed using *Density Functional Theory*. The *Hydrogen* atoms in the simulation are fixed in the *x-y-plane* and oscillate in *z-direction*.

With the system only changing in distance between the two hydrogen atoms, building a *Machine Learning* model using a two body distance abbreviation, presented in chapter 3.3.4, is sound. The atoms oscillate in one dimension with no changes in angle between them, making a two-body descriptor taking only the distance between them into account suitable.

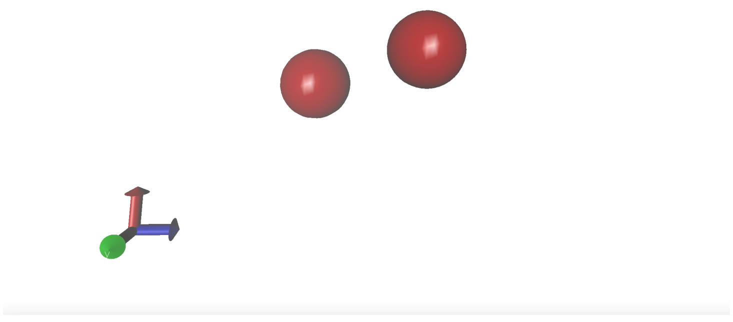

Fig. 4.1: H_2 Molecule

4.1.2 Water Crystal

Second, a a larger more complex system of a *Hydrogen Crystal*, containing 192 Atoms is introduced. Again the atom configurations are the output of a *Molecular Dynamics (MD) Simulation* computed quantum-mechanically making use of the Approximations of *DFT*. For simplicity reasons a *MD* is run for a bulk of atoms of just one type. The way the software, to create the *ML* model, is built this simplicity restriction could be lifted and the system could be updated to include multiple atom species.

The Hydrogen *H* in form of a crystal is depicted in figure 4.2. The *MD* run to generate the atomic configurations starts at a state of atoms out of ground state. The simulation generates training data for *Hydrogen* atoms relaxing towards an equilibrium state. The atomic configurations yielded each time-step serve as as basis of the building process of the *Machine Learning* model.

To describe this system, again a pairwise distance descriptor is used. The two body descriptor oversimplifies the problem and there is a loss of information describing a 192-Atom *Hydrogen Crystal.* This is done to compare results of the same *Machine Learning* algorithm for two systems of differing complexity.

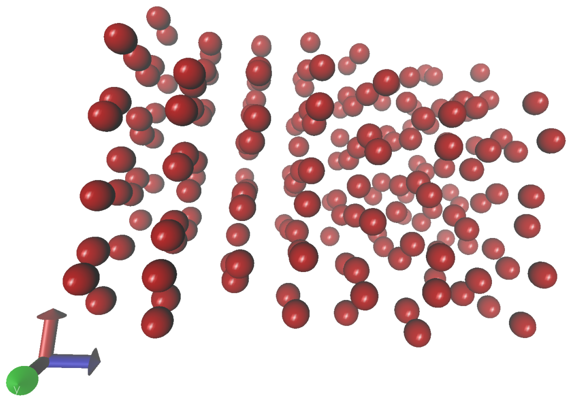

Fig. 4.2: Hydrogen Crystal

4.2 Workflow

4.2.1 Flow-Chart

Creating the *ML* model is not a linear process, but a circular one. As can be seen in the flowchart 4.3 there are several steps building the model and turning the process into a result optimising feedback loop. The depicted Flowchart lays the foundation for creating and tuning the model and its steps are derived in the following chapters.

4.2.2 Get Data from Molecular Dynamics Simulation

Beginning at the top of the flowchart the first step on the way to creating a model is choosing which system is going to be modelled and then run a *Molecular Dynamics Simulation*, briefly explained in chapter 4.1. The simulation uses *DFT* to yield atomic configurations for each time step, these serve as input to the model. Details on how DFT solves quantum-mechanical problems are derived in chapter 2. The following chapter explain the workflow to go along with the flowchart on the example of the system containing the H_2 *Molecule.*
The *MD* outputs a a set of atomic constellations with a corresponding energy per constellation and an associated force for each atom of the system per constellation. The data then needs to be transformed from an *XMLl* file coming out of the *MD* to a certain file format, namely an *XYZ* file so it can be interpreted correctly. Details are derived in Appendix 6.2

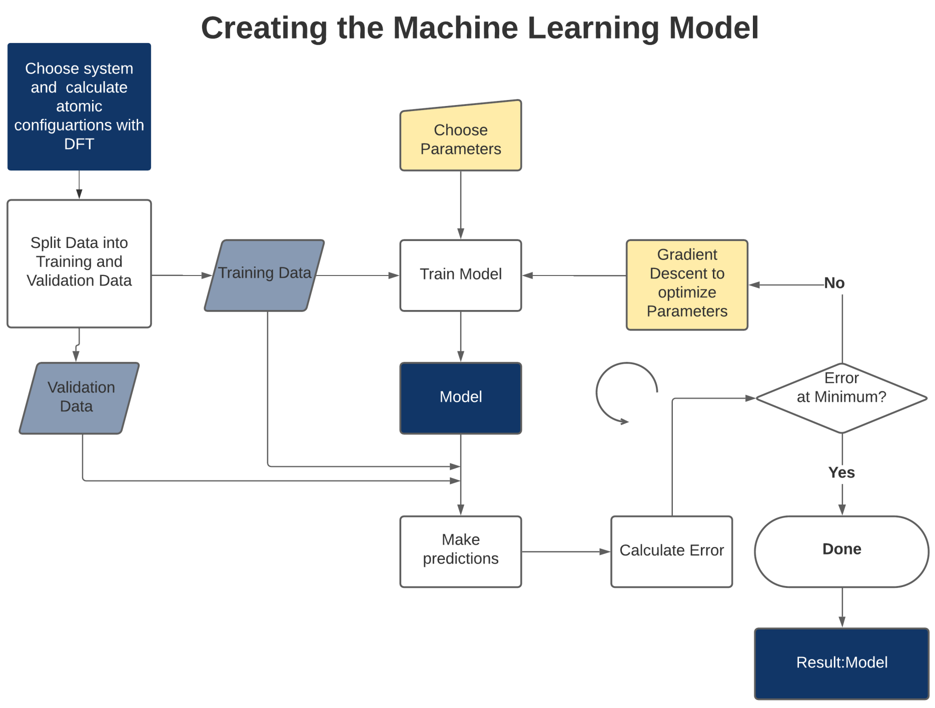
Creating the Machine Learning Model
Choose system and calculate atomic configuartions with DFT
Split Data into Training and Validation Data
Training Data
Validation Data
Choose Parameters
Train Model
Model
Gradient Descent to optimize Parameters
No
Error at Minimum?
Yes
Make predictions
Calculate Error
Done
Result:Model

Fig. 4.3: Flowchart

4.2.3 Split Data into Training and Validation Data

It is good practice in any *ML-process* to split up the input data into two parts. [14, p. 17] One is going to be the training data used, directly used in the model building process and the other part is saved to evaluate and improve the model, that is validation data. Once the model has gone through the first iteration of its optimisation loop the validation data is used to see the quality of the model. This is done by letting the model make prediction on the validation data it has not been trained on. Saving a part of the input data for validation is done to prevent the model from overfitting, by using the validation data to refine the model. Overfitting means the model is highly accurate on reproducing the results of the training data but fails to correctly classify new input data.

In this case the split up between the training and validation data is done randomly and the ratio of the fit is set to 0.8. An 0.8 ratio delegates 80 percent of the input data for the model to training data and marks 20 percent as validation data. The ratio is by choice, but the way the code is written for this work it is possible to change split percentages and construct the model with a different split.

4.2.4 Model Training

Following the split into training data and validation data, the latter is put aside and the training data is input to the *ML Process* which uses Gaussian Process Regression as underlying tool. Details are derived in chapter 3. The algorithm works like a black box where pairs of atomic configurations and the associated energies and forces are input to the model. In addition a set of parameters is demanded due to the algorithmic of the Gaussian Process.

There are parameters true to the nature of the Gaussian Process and further descriptor specific parameters. The chosen descriptor here is the *2-body-distance*. The black box model is trained, meaning in builds instructions by internally setting certain variables to certain values, which then enables the model to predict atom energies and forces for given atomic constellations.

The required parameters by the Gaussian Process represent the expected variance σ of the training data. [7, Appendix] Intuitively this can be thought of as intrinsic noise of the training data. Mathematical details can be found in chapter 3.2.

- σ_E ... variance of training data energies
- σ_F ... variance of training data force
- σ_S ... variance of training data functional derivatives of energies, this is the virial stress [10, p. 1053]
- σ_H ... variance of training data functional second derivatives of energies, this is the hessian [10, p. 1053]

The 4 σ parameters are the diagonal elements of a *variance* matrix.

The *2-body-distance* descriptor requires the following parameters to be set, details can be found in chapter 3.3:

- *Cutoff Radius* ... sets geometrical distance for each atom up to which its environment is considered by the model, in Ångstrom
- *Covariance type* ... sets how the kernel is built using the descriptor transformations of the input data as basis functions
- θ ... goes into the computation of the covariance type as denominator. Determines the *size* of the classification bins by resembling the geometrical scale of the input data.
- δ ... puts weight on the kernel for the chosen descriptors if there is more than one. The input settings determine the choice of δ.
- *Sparse Method* ... sets the *Sparsification* method. The sparse data is a representative selection all the input data
- Number of *Sparse Points* .... size of the *Sparse Set*, defines the magnitude of the reduction on the sparse point set

Of the needed parameters an educated choice is made for all but σ_E σ_F and θ, those are later optimised using a Gradient Descent Method. The predetermined parameters are set to the following values:

- *Cutoff Radius* $:=$ 4 Ångstrom
- *Covariance Type* $:=$ Squared Exponential [7, Appendix 2.1.], see equation 4.1
- $\delta := 1$, can be set to any value due to just one descriptor being used, hence no no weighting between different descriptors is needed
- Sparse method $:=$ Uniform, this means the sparse points are chosen based upon a selection of bins all of the same size. The size of the bins is determined by a uniform grid. [11, p. 2]

- Number of sparse points $:= 20$, for the H_2 System this is redundant and means all the atoms, but for the 192 Atom $H - Crystal$ the sparse representation is relevant
- $\sigma_S := 0.0$
- $\sigma_H := 0.0$

$$C(x_n, x_m) = \delta^2 exp\left(-\frac{1}{2}\frac{(x_n - x_m)^2}{\theta^2}\right) \tag{4.1}$$

- $C(x_n, x_m)$ ... kernel, covariance matrix between two configurations
- δ ... weighting between different descriptor kernels
- x_n ...descriptor value of $n_t h$ configuration
- x_m ... descriptor value of $m_t h$ configuration
- θ ... hyperparameter, weighs the scale of input configurations

Note that the variance of the virial stress and the hessian in the training data are set to zero because no further deviation for the derivation is expected. This is due to the *2-body-distance* descriptor operating in one dimension. Choosing the fixed parameters is done based upon knowledge of the analysed atomic system. Varying the "fixed" parameters will not result in a recognisable change of the model. Thus not further optimising them is reasonable in respect to its cost.

The most dependencies on the quality of the model rely on σ_E σ_F and θ. It turns out that applying an optimisation algorithm to them returns the most precise results and converges fast to an optimum in fit. Still a guess needs to be made for σ_E σ_F and θ to then pass to the first iteration of the cyclic optimisation process. A first guess is made:

- $\sigma_E := 0.004\ eV$
- $\sigma_F := 0.08\ eV/Ångstrom$
- $\theta := 1.0$

Note that it makes sense to select the variance of force to be one order higher as the variance of energy. This is because the calculation of forces includes derivatives of energies. The variance of forces respects the error in forces and energies alike. The noise of energy values is propagated. Following the flowchart in 4.2.1 the model can enter the optimisation process with the input of

1. Training data consisting of atomic configurations with their corresponding energies and forces, and a
2. Set of fixed and variable parameters

4.2.5 Model Prediction

Using the model to make predictions requires any set of atomic configurations as input and gives a set of energies and forces as output. If the output values resemble the values of a quantum-mechanical computation using, for example, *Density Functional Theory*, then the model is correct. In detail the prediction yields one energy per atom configuration and forces for each atom for each spatial coordinate per atom configuration.

Going back to the flowchart in 4.2.1 the prediction is performed on the training data and additionally on the validation data. This is done to see how the model fares on input it has not been trained on. The mathematical details on how the prediction is performed based on the input data and the chosen parameters is derailed in chapter 3.

4.2.6 Model Optimisation

The model is trained and able to make predictions. To optimise the quality of the model a measure of goodness is needed. The error, in this case the *Root Mean Squared Error (RMSE)*, is used as measure of inverse goodness. The *RMSE* is a measurement of how fare the predicted values are off compared to the real values. A perfect model would always predict values mirroring real values, barring any intrinsic inaccuracies in the data, leading to an *RMSE* of zero. Then *Root Mean squared error* is a highly simple measure of error as it just takes the root of the mean of the squared error of two values, as in equation 4.2. Further computational details are explained in [21].

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(\hat{X}_i - X_i)^2}{N}} \tag{4.2}$$

The denotations in 4.2 are

- $\hat{X}$ ... predicted value
- X ... real value
- N ... number of real and predicted value pairs

Further computational details are explained in [21].

In general a model is considered valid if it performs well but not perfect on the training data. A perfect resemblance of the training data would lead to overfitting and to inaccurate predictions of new, unknown input data. The validation data comes into place as it is unknown input data for the model. For both, the training atomic configurations and validation atomic configurations, the associated energies and forces are known. In this case *real* means calculated by *DFT*, which is assumed to mirror the features of a real system precisely bewaring numerical errors.

To measure the quality of the fit, the trained model is used to make calculations on the training data and validation data and output the related energies and forces. The output is then compared to the real values of energies and forces by calculating the *RMSE* between them. This is done by calculating the *RMSE* for every real value, predicted value tuple and then added up. The summarised error is then proportional to the difference between the real energies and forces and predicted energies and forces.

A small *RMSE* means predicted and real values lie close to each other. This is a good fit and the model works correctly. To get a well performing model the training, predicting and calculating the error procedure is plugged into a cycling optimisation algorithm. Since the atomic configurations cannot be changed the goal of the optimisation is to minimise the *RMSE* by tuning the model parameters, namely σ_E σ_F and θ. The optimisation is done by a Gradient Descent Method. The *Nelder-Mead Downhill Simplex Algorithm* is used to minimise a function, to which σ_E σ_F and θ are passed as independent arguments and and the *RMSE* is output as dependent variable. The dependent variable is the one to be minimised. Details on the *Nelder-Mead Downhill Simplex Algortithm*, can be found in [12].

For this work, the built in python package *scipy* is used to minimise the *RMSE* . An example function call for the optimisation of H_2 can be seen in the code snippet below. Details can be found in Appendix 6.2.

```python
import scipy.optimize
initial_guess = [1,0.004,0.08]
result = scipy.optimize.minimize(RMSE_train_val,initial_guess,method='
    Nelder-Mead',
                                 options={'fatol':10e-5,'maxiter':100})
```

The following arguments are passed to the optimisation algorithm:

- *RMSE_train_val* is the function to be minimsed
- *initial_guess* for the parameters σ_E σ_F and θ
- *fatol* is an exit condition for the algorithm, once the *RMSE* converges to a minimum with and converges with *fatol*, the process is stopped.
- *maxiter* is an exit condition for the algorithm. If the optimisation process exceeds the number passed to *maxiter* it is terminated

Notice that the contribution of the *RMSE* of every real value, predicted value pair of energies and forces need to be scaled and cannot simply be added up and passed on to the minimiser. This is due to the ratio between the energies and forces for each configuration. Each atomic constellation hold energies for the whole system at a specific point wheres forces are attached to every atom of the constellation for every spacial coordinate at a specific point. Referring to the H_2 *Molecule* of the *MD* simulation this means one energy for each time step compared to 6 forces, two atoms and three cartesian coordinates.

To scale the *RMSE* contributions correctly and to provide easy comparability to different atomic systems a per atom *RMSE* is introduced. In General, for any system, this is done by dividing both, the energies and the forces by the number of atoms ind the configurations. Additionally the Forces need to be divided by the number of spatial coordinates, for cartesian systems this means dividing by three. The scheme is shown in equation 4.3. To prevent the optimisation process from overweighting the importance of either energies of forces, the *RMSE* needs to be scaled correctly.

$$RMSE = \frac{RMSE_{Energy}}{n_{Atoms}} + \frac{RMSE_{Force}}{n_{Atoms} * n_{Coordinates}} \tag{4.3}$$

The optimisation is terminated by one of the exit conditions and returns a value triple of σ_E σ_F and θ. For this value triple the *RMSE* is minimal for the given system and the model is optimised.

4.2.7 Water Crystal Model Creation

Having introduced the optimisation process on hand of the simple case of it can also be applied to any other system. Staying true to a single atom species but just a more complex system, the workflow as shown in the flowchart in 4.3 is applied on a system of a *Hydrogen Crystal* containing 192 atoms. A visualisation is depicted in figure 4.2.

The workflow is identical to the one used for the H_2 *Molecule*. First, a *Molecular Dynamics Simulation* using the approximations of the *DFT* generates atomic configurations with associated energies and forces. These configuration-value pairs are then split up into training and validation data. Second, training data is then used for building the model. The training data and the validation data is used to optimise the model by minimising the *RMSE* and optimising parameters. Notice that the scaling of the RMSE summands of energies and forces differs significantly from the scaling done for the H_2 *Molecule*. The optimisation process returns an optimised parameters set for σ_E σ_F and θ different to the ones found for the Molecule. The same model creating algorithm returns differentiating models for specific system.

Chapter 5

Outcome and Analysis

In chapter 4 a *Machine Learning* model able to predict atomic energies and forces was created. A model is of high quality if it resembles but also abstracts reality. The better the abstraction that is a model, the more accurate it mirrors reality. The *ML* model has been trained on real values and has been optimised by minimising variation from reality.

5.1 Water Molecule H2

5.1.1 Outcome Visualisation

Visualising the quality of a model can be done by plotting input against output, hence real values of energies and forces against predicted values of energies and forces. The following graphs show real values on the *x-axis* and predicted values on the y-axis. To get a cleaner look on the relationship between real and predicted values a help line is drawn which exactly matches real values and predicted values. If a dot, representing a predicted value lies directly on the drawn line then the prediction coincides exactly with a real value. In this case the *RMSE*, discussed in subsection:4.2.6., for this single value is zero. The further away a dot is drawn from the line, the higher the magnitude of the *RMSE* gets for this single value and the prediction is less accurate.

In this chapter the outcome of the *ML* model is analysed for the H_2 *Molecule*. In figure 5.1 the energies before the optimisation process are plotted for training data and validation data respectively. The figures 5.2 show the forces of the training data and validation data with the initial guess for the parameters σ_E σ_F and θ.

Fixed and variable parameters are needed in addition to the training data for creating a model, referring to chapter 4.2.4. This set of params can be see in table 5.1 together with the corresponding *RMSE*.

	θ	σ_E	σ_F	*RMSE*
Before Optimisation	1	0.008	0.04	0.001276
After Optimisation	0.325703	0.007972	0.009353	0.0002264

Tab. 5.1: Parameters and *RMSE* before and after Optimisation for H_2 Molecule

The figures 5.1 and 5.2 depict the situation before the optimisation process, hence do not match reality yet. The initial guess for the parameters σ_E σ_F and θ and the corresponding *RMSE* in table 5.1 change after the model is handed to the optimisation process, explained in chapter 4.2.6. The second line of table 5.1 shows the tuned parameters together with the corresponding *RMSE*.

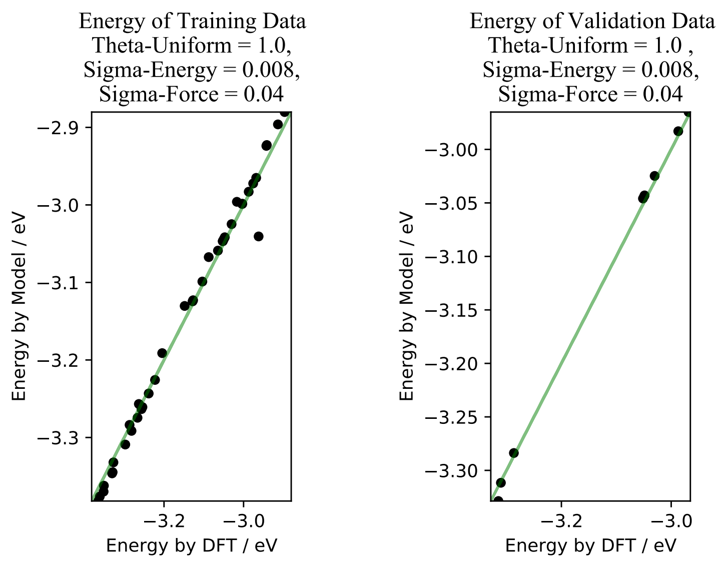

Fig. 5.1: Energies of H_2 Molecule before Optimisation

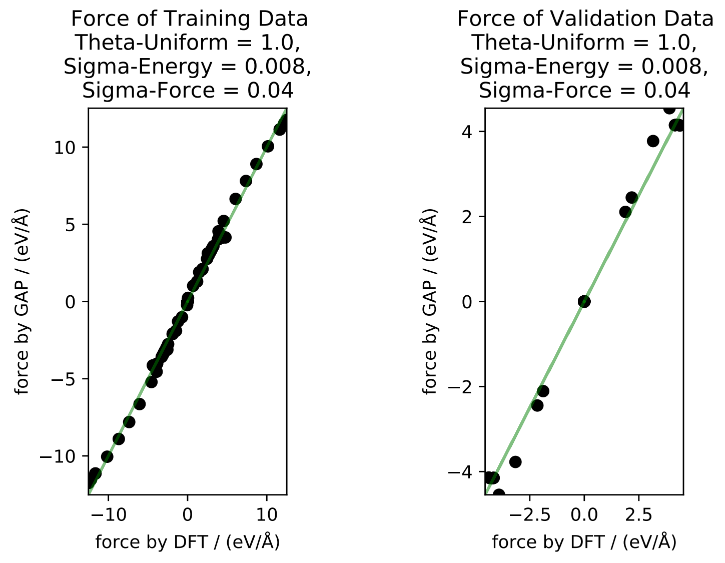

Fig. 5.2: Forces of H_2 Molecule before Optimisation

The tuned parameters and minimised *RMSE* link to the figures 5.1 and 5.2. The energy of the training data and validation data is shown in figure 5.3. Respectively the force of the training data and the validation data can be seen in figure 5.4.

The optimised model leads to a smaller *RMSE*, see table 5.1 and the visual comparison of the graphs in figures 5.1 and 5.2 to the graphs in figures 5.3 and 5.4 shows dots lying closer to the optimal drawn line for the optimised model.

5.1.2 Model Parameter Analysis

Performing the optimisation and tuning the parameters σ_E σ_F and θ leads to an almost exact mirroring of real energies and forces by the model. For the H_2 *Molecule* the *Gradient Descent* algorithm, introduced in chapter 4.2.6, terminated after 67 iterations and 137 function evaluations.

The almost excact reproduction or real energies and forces is due to the elementariness of the examined system. The considered H_2 *Molecule* oscillates along the z-coordinate, hence is moving one-dimensionally. The descriptor underlying the *ML* model takes *2-body distances* into account. Actual atomic configurations are transformed to a feature space.The *2-body distances* descriptor feature space allows an exact representation of the H_2 *Molecule* oscillating in the distance between the Hydrogen Atoms. One to one mapping of the geometric atomic configuration to the feature space of the *ML* model leads to a small *RMSE* and classifications of high quality by the model.

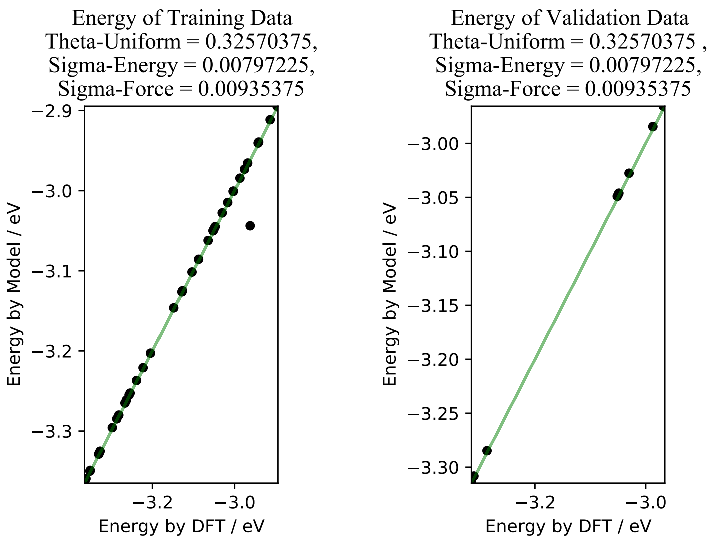

Fig. 5.3: Energies of H_2 Molecule after Optimisation

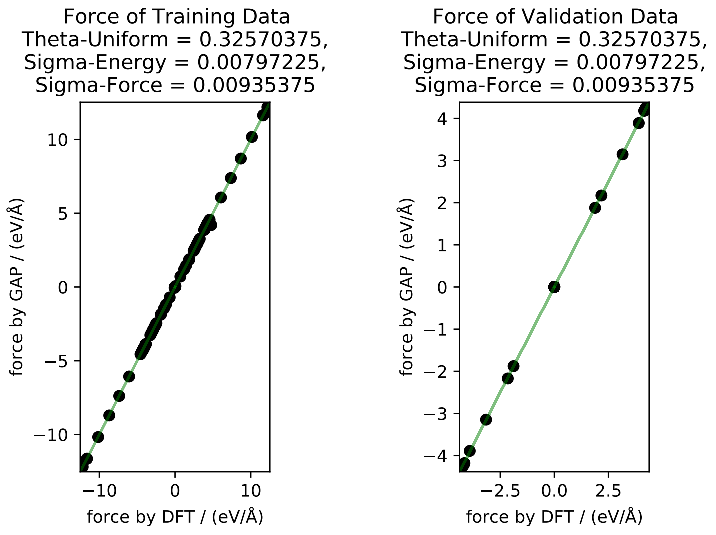

Fig. 5.4: Forces of H_2 Molecule after Optimisation

5.2 Hydrogen Crystal

5.2.1 Outcome Visualisation

Similarly to the H_2 *Molecule* the *Hydrogen Crystal* is examined. The system is more sophisticated due to the greater number of atoms, the Hydrogen Crystal consists of 192 atoms compared to just two for H_2, but more so due to the complexity of the arrangement of the atomic constellation. The *Molecular Dynamics Simulation (MD)* starts at an unordered state. During the simulation the Hydrogen Atoms relax towards an equilibrium state.

To train and refine the model the same set of parameters is for the H_2 *Molecule* and the *Hydrogen Crystal* alike. In this chapter the goodness of the *ML* model of the *Hydrogen Crystal* is analysed. A visualisation on how the model fares to reproduce the quantum-mechanically calculated values, before the start of the optimisation process can be found in figure 5.5 for the energies of the training data and the validation data. The forces of the training data and for the validation data can be found in figure 5.6. The green diagonal line serves as target line for the predicted values. If on the line the predicted values match the real values. By looking at the figures 5.5 and 5.6 and checking the positions of the dots, representing the classifications of the model, one can see that the initial model does not mirror reality. Additionally, the values of the parameters and the *RMSE* can be seen in table 5.2.

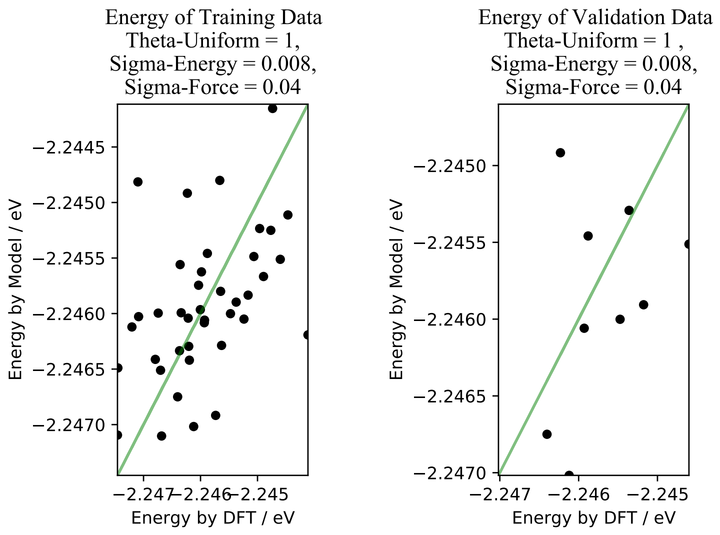

Fig. 5.5: Energies of Hydrogen Crystal before Optimisation

	θ	σ_E	σ_F	$RMSE$
Before Optimisation	1	0.008	0.04	0.000743
After Optimisation	0.384155	0.065248	0.052590	0.000259

Tab. 5.2: Parameters and RMSE before and after Optimisation for *Hydrogen Crystal*

Starting out with the same set of parameters for the H_2 *Molecule*, in table 5.1 and the *Hydrogen Crystal*, in table 5.2, σ_E σ_F and θ change significantly after the optimisation process is run. The values for the variable parameters vary from the ones found in table 5.1. A visualisation of the

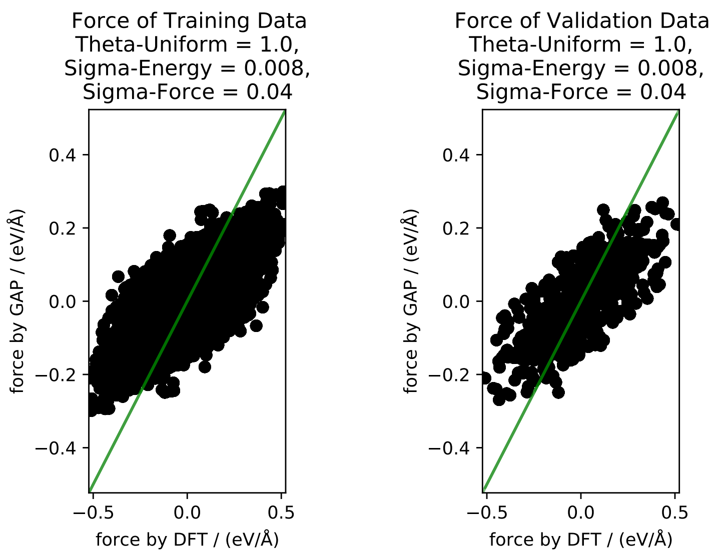

Fig. 5.6: Forces of Hydrogen Crystal before Optimisation

predictions of the optimised values for the energies can be found in figure 5.7 for the training data and the validation data. Predictions on forces of the training data and the validation data are shown in figure 5.8.

5.2.2 Model Parameter Analysis

Two main conclusions can be drawn from the optimisation process of the model of the *Hydrogen Crystal*. First, the *RMSE* error is reduced significantly throughout the optimisation process. Looking at the plots before the appliance of the optimising *Gradient Descent* algorithm in figures 5.5 and 5.6 and after in figures 5.7 and 5.8, one can see that the dots resembling the predicted values of energies and forces in respect to their real values have moved drastically towards the optimal green line. The reduction of the *RMSE* can be seen in table 5.2. The optimisation process terminated after 46 iterations and 104 function evaluations.

Second, the plots in figures 5.7 and 5.8, together with the *RMSE* in table 5.2 show the relatively low quality of the model compared to the H_2 *molecule*. The predictions for the *Hydrogen Crystal* feature a larger variance in respect to the corresponding real values than the predictions of the model for the H_2 *Molecule*. Looking at the predictions of the model in the figures 5.7 and 5.8 one cannot see the that predicted and real values coincide on every occasion, but a strong trend is visible.

The reason the model does not classify energies and forces for the *Hydrogen Crystal* as well as it does for the H_2 *Molecule* is the choice of descriptor. Both systems are represented by a *2-body-distance* descriptor. A one dimensional distance suffices the requirements of the oscillating H_2 *Molecule* but does not for a 192 Atom *Hydrogen Crystal* relaxing to an equilibrium state. The descriptor fails to precisely map the complex 192 Atom constellation to a feature space serving as input for the *ML* model. The loss of information by choice of a *2-body-distance* descriptor determines the worse prediction performance of the model compared to the precise predictions of the model for the H_2 *Molecule*. The interatomic interference of the larger system exceeds the sphere of the simple *2-body-distance* descriptor.

Fine tuning the parameters leads to the best achievable results, visualised in the figures 5.7 and 5.8 with the values off the optimal predictions but a visible slope and a trend towards correct classification.

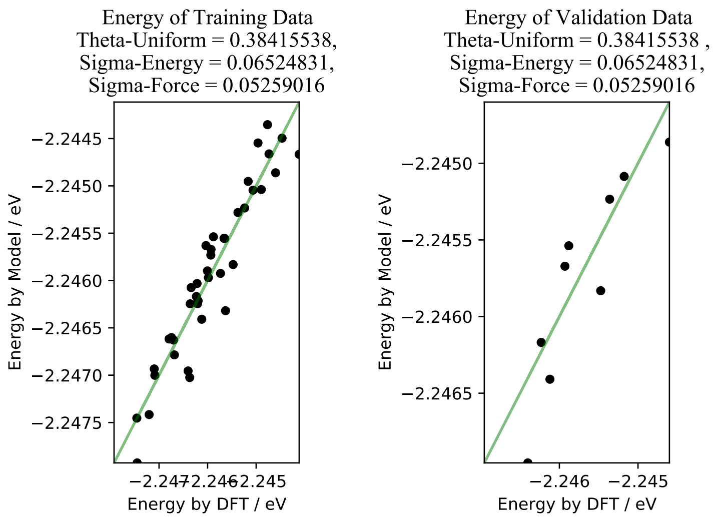

Fig. 5.7: Energies of Hydrogen Crystal for Training Data and Validation Data after Optimisation

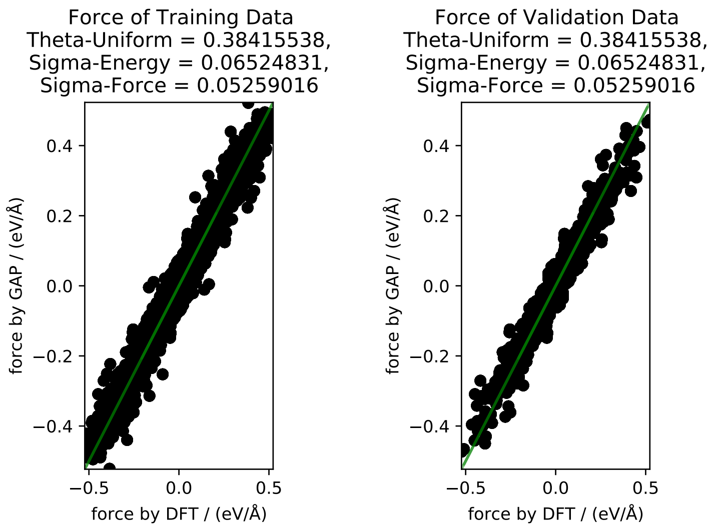

Fig. 5.8: Forces of Hydrogen Crystal for Training Data and Validation Data after Optimisation

Chapter 6

Summary and Outlook

6.1 Model Improvement by Descriptor Adjustment

Analysing the choice of parameters in chapter 5 the optimal selection of θ, σ_E and σ_F leads to great results for the *H_2 Molecule* and to satisfying results for the *Hydrogen Crystal* system. Using the same descriptor as feature space for both atomic structures the deficiencies are outlined modelling the 192 Atom *Hydrogen Crystal*.

While, as pointed out in chapter 5.1.2, the *2-body-distance* suffices the *H_2 Molecule* oscillating in one dimension, it does fail to express the interactions present in the *Hydrogen Crystal*. One possible solution can be to expand the feature space used to describe the atomic system by adding or changing descriptors. Other descriptors may be used to resemble the atomic configurations more accurately in case of larger systems, as is the *Hydrogen Crystal*. A descriptor taking angles between objects into account would be a potential expansion of the feature space. This would upgrade the geometrically mapped feature space from one dimension to three dimensions. That is the case for a *3-body angle descriptor* taking into account angles between 3 atoms. The software utilised for this work in Appendix 6.2 offers a *3-body angle descriptor* amongst others. Making use of more complex descriptors could have positive impact to the modelling process and result in an improved fit.

6.2 Larger Systems

Predicting energies and forces built on appropriate descriptor feature spaces and parameters opens up *Machine Learning* solutions for larger and more complex systems than the *H_2 Molecule* in chapter 4.1.1 and the *Hydrogen Crystal* in chapter 4.1.2. The suboptimal choice of the *2-body-distance* descriptor for a 192 Atom problem leads to classifications trending in the right direction, although not precise results. The correctly reproduced trend shows the possibilities of using *Machine Learning* algorithms for the prognosis of the behaviour of quantum-mechanical systems. As pointed out in chapter 6.1 a fitting descriptor choice can lead to more exact precision in predictions.

Larger, more diverse systems may require combinations of different descriptors than the *2-body-distance* descriptor utilised in this work. Gaining a model of high quality demands a sound choice of descriptors and refined parameter values. Additionally, the goodness of the model depends on the quality of the Input Data, computed with *DFT*. The input data, providing the model with training data and validation data, is generated prior to building the model. The importance of the input data becomes significant for diverse systems, possibly containing multiple atomic species.

Note that the creation of the model does not require any quantum-mechanical calculations and depends solely on the input data. This is pointed out in chapter 4.2.2. Skipping quantum-mechanical calculations, for example by *DFT*, and making prognoses based on a trained *Machine*

Learning model reduces runtime and cost of computation. This makes *Machine Learning* essential for the challenge of describing complex atomic systems.

Appendix

A Preparation of Training Data

The atomic environments used in this work are yielded by a *Molecular Dynamics (MD) Simulation*. The data is computed using *Density Functional Theory (DFT)* and each time-step of the *MD* simulation corresponds to one atomic configuration.

The *MD Simulation* output is primordially written to the *XML* file. The atomic configurations with their related energies and forces need to be extracted from the *XML* file using a shell script. The script can be found in (TODO: Simon Repository) The machine learning algorithm demands input data in format of an *XYZ* file. In respect to the needed output file format, the shell script transforms the data by extracting it from the *XML* file and writing it to an *XYZ* file.

B Software

The *Machine Learning* software packages used in this work are *GAP* ,*QUIP* by *Albert Bartok-Partay, Noam Bernstein, Gabor Csanyi* and *James Kermode*. [4] The software is mostly constructed in *Fortran*, with a *Python* user interface. The source code can be found in [20]. Also the python package *Atomic Simulation Environment (ASE)* is used. [4]

C Code

Preparing the *Training Data*, performing a *Split*, training the *ML* model, optimising the model parameters and generating plots is done using modules and methods written in *Python*. The entirety of the code is stored in (TODO: Simon Repository).

D Optimisation

To optimise the *Machine Learning* model its parameters are optimised. This is done by minimising the *Root Mean Squared Error (RMSE)* between the predictions of the model and computations by *DFT*. The minimisation is done using a python module, that is *scipy.py* [22]. The minimisation performs a *Gradient Descent* method, namely the *Nelder-Mead-Algorithm (NM)*. The *NM* method is a *Downhill Simplex* algorithm for minimising non-linear functions in respect to multiple parameters.

The process, introduced by *John Nelder* and *Roger Mead* in 1965, does not implicitly compute derivations of the function to be minimised. Such an algorithm is needed, since the *ML training* and *prediction* process does not offer a function on which algebraic operations can be performed on. The creation of the model is a *Black-Box* for the optimisation algorithm. Specifics on the *Nelder-Mead* algorithm can be found in [12]

Bibliography

[1]W. Kohn, *Nobel lecture: electronic structure of matter—wave functions and density functionals*, 1999.

[2]D. J. MacKay, *Information theory, inference, and learning algorithms* (Cambridge University Press, 2003).

[3]J. Hutter, *Dichtefunktionaltheorie*, 2004.

[4]*Gnu general public license, version 2*, `https://github.com/libAtoms/QUIP`, Last retrieved 2021-31-01, 2006.

[5]M. Hjorth-Jensen, *Computational physics*, 2008.

[6]J. F. Trevor Hastie Robert Tibshirani, *The elements of statistical learning, second edition*, Springer Series in Statistics (Springer, 2008).

[7]A. P. Bartók, M. C. Payne, R. Kondor, and G. Csanyi, "Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons", (2009).

[8]J. C. C. J. Espinosa-Garcia M. Monge-Palacios, "Constructing potential energy surfaces for polyatomic systems: recent progress and new problems", (2012).

[9]D. K. Duvenaud, *Automatic model construction with gaussian processes* (2014).

[10]B. A. P. and S. G., "Gaussian approximation potentials: a brief tutorial introduction", J. Quantum Chem, 115, 1051–1057, `10.1002/qua.24927` (2014).

[11]C. G. Vladimir Blinovsky, "Asymptotic enumeration of sparse uniform linear hypergraphs with given degrees", (2016).

[12]W. H. Charles Audet, *Derivative-free and blackbox optimisation* (Springer, 2017).

[13]A. Géron, *Hands-on machine learning with scikit-learn and tensorflow* (O´Reilly Media, Inc. 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2017).

[14]A. C. Müller and S. Guido, *Introduction to machine learning with python* (O´Reilly Media, Inc. 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2017).

[15]J. Toulouse, *Introduction to density-functional theory*, 2019.

[16]S. B. Paschen and P. Mohn, *Festkörperphysik ii*, 2020.

[17]M. Pitschmann, *Atom-, kern-, und teilchenphysik i* (2020).

[18]T. Hande, *Lecture notes on principles of density functional theory*, `http://www.physics.metu.edu.tr/~hande/teaching/741.html`, accessed: 05.12.2020.

[19]*Lecture 3: from many-body to single-particle: quantum modeling of molecules*, `https://ocw.mit.edu/courses/materials-science-and-engineering/3-021j-introduction-to-modeling-and-simulation-spring-2012/part-ii-lectures-videos-and-notes/lecture-3/`, accessed: 12.04.2021.

[20]*Libatoms git repository*, `https://github.com/libAtoms`, accessed: 19.04.2021.

[21]*Root-mean-squared-error an overview*, `https://www.sciencedirect.com/topics/engeneering/root-mean-squared-error`, accessed: 07.04.2021.

[22]E. Jones, T. Oliphant, P. Peterson, et al., *SciPy: open source scientific tools for Python*, 2001–.