Dipl.-Ing. Ralf Meyer, BSc

# Machine Learning in Computational Chemistry

**DOCTORAL THESIS**

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to

**Graz University of Technology**

**Supervisor**

Assoc.Prof. Mag.phil. Dipl.-Ing. Dr.phil. Dr.techn. Andreas W. Hauser

Institute of Experimental Physics

Graz, April 2021

## AFFIDAVIT

# Abstract

Data-driven methods have been a cornerstone in the development computational chemistry algorithms since their inception, for example in the form of force fields and semi-empirical methods. The recent developments and renewed interest in the field of machine learning gave rise to tools allowing for the treatment of extremely large data sets and fitting of models with potentially millions of parameters. Open availability of these tools has led to a quick spread of machine learning-based algorithms throughout several other fields of research including computational chemistry.

After a short introductory overview of the many suggested applications of machine learning in computational chemistry, detailed investigations of four algorithms, with the common goal of facilitating the simulation of heterogeneous catalysis on bimetallic nanoparticles, are presented. The improvement of classical many-body potentials, typically used for the simulation of metallic systems, by replacing some of their contributions by flexible machine learning expressions is investigated. A previously proposed algorithm to accelerate the search for stable molecular geometries is extended to molecule-specific coordinate systems. Various machine learning methods are compared with respect to their performance as surrogate model in the calculation of reaction pathways and energy barriers. Finally, a methodological improvement of a machine learning-based approach to a fundamental unsolved problem in computational chemistry, the description of the kinetic energy of electrons as functional of their density, is investigated.

# Kurzfassung

Datengestützte Methoden bilden seit Anbeginn der Computerchemie einen Eckpfeiler der Methodenentwicklung, zum Beispiel in Form von Kraftfeldern und semi-empirischen Methoden. Die jüngsten Entwicklungen auf dem Gebiet des maschinellen Lernens haben Algorithmen und Programmpakete hervorgebracht, die die Behandlung extrem großer Datensätze und die Anpassung von Modellen mit potenziell Millionen von Parametern ermöglichen. Die offene Verfügbarkeit dieser Methoden hat zu einer schnellen Verbreitung von maschinellem Lernen in verschiedenen anderen Fachgebieten, einschließlich der computergestützten Chemie, geführt.

Nach einem kurzen einführenden Überblick über die vielen vorgeschlagenen Anwendungen von maschinellem Lernen in der Computerchemie werden detaillierte Untersuchungen von vier Algorithmen vorgestellt, deren gemeinsames Ziel es ist, die Simulation der heterogenen Katalyse an bimetallischen Nanopartikeln zu erleichtern. Es wird versucht klassische Vielkörperpotentiale, wie sie typischerweise für die Simulation von metallischen Systemen verwendet werden, durch den Einbau flexibler maschinell erlernter Ausdrücke zu verbessern. Ein zuvor vorgeschlagener Algorithmus zur Beschleunigung der Suche nach stabilen molekularen Geometrien wird auf molekülspezifische Koordinatensysteme erweitert. Verschiedene Methoden des maschinellen Lernens werden im Hinblick auf ihre Genauigkeit als Näherungsmodell bei der Berechnung von Reaktionswegen und Energiebarrieren verglichen. Abschließend wird eine methodische Verbesserung eines auf maschinellem Lernen basierenden Ansatzes für ein grundlegendes, ungelöstes Problem in der Computerchemie, der Beschreibung der kinetischen Energie von Elektronen als Funktional ihrer Dichte, untersucht.

# Contents

# 1. Introduction

In the introduction to his article on "Quantum Mechanics of Many-Electron Systems", published in 1929, Paul Dirac famously stated,[1]

> The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation.

Decades later the hunt for these approximate practical methods, which in modern days almost exclusively refers to computational algorithms, continues. These algorithms are typically tailored to specific use cases due to some inherent trade-offs. Highly accurate predictions for comparison with high resolution experiments are for example limited to molecules containing a few atoms due to the enormous computational effort required. For the simulation of large systems or long time scales it is therefore necessary to sacrifice some of the accuracy in order to reduce the computational effort.

This trade-off is also reflected in the approach taken to derive the various methods. High accuracy algorithms are designed by starting from first-principles (*ab initio*), paired with a careful inspection of each introduced approximation with respect to the loss in accuracy and the reduction in computational effort. In algorithms designed for large systems and time scales these approximations have to be applied much more widely, and some of the computationally most challenging contributions are typically replaced by parameterized expressions. Since the parameters are fitted to high accuracy calculations or experimental results these algorithms are often referred to as empirical methods. One of the most widely used computational chemistry methods, density functional theory, straddles this boundary between *ab initio* and empirical methods, offering both options, namely functionals derived purely from physics-based arguments such as scaling laws, and others incorporating fit parameters that are tuned to a specific class of molecules.

A fairly recent development is the emergence of machine learning-based methods. Throughout this thesis the definition of machine learning given by Mitchell is used,[2]

> A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

These newly suggested methods forego the concept of physically motivated approximations and directly approximate a functional relationship between the input to computational chemistry algorithms and their output. This typically involves using highly flexible mathematical functions containing a large number of adjustable parameters and vast amounts of data to fit the parameters. The rise of machine learning-based methods is driven by a general renewed interest in the field of machine learning following increased public and private research funding due to breakthroughs in the areas of computer vision and natural language processing. This led to the development of a plethora of easily accessible tools which in turn facilitated the application of machine learning in many different fields of research, including computational chemistry.

The goal of this thesis is to identify combinations of computational chemistry methods and machine learning algorithms that yield a significant reduction of the computational effort, with a special focus on the simulation of heterogenous catalysis on bimetallic clusters. While most of the ground work in this regard has already been done by various research groups over the course of the last few years, there is still much room for improvement in these early implementations.

Based on several reviews on the topic,[3–13] Chapter 2 gives a rough overview of the broad range of suggested machine learning-based computational chemistry algorithms. While all of the presented concepts can be followed by simply considering machine learning algorithms as black box fitting tools, a short introduction and references to detailed explanations are given in Appendix A. The overview in Chapter 2 is intended to provide context for the detailed investigations of four specific applications presented in Chapters 3-6, corresponding to previously published standalone manuscripts. In Chapter 3 a machine learning-informed approach to the construction of many-body potentials for bimetallic nanoparticles is investigated. After replacing some of the contributions by neural network expression, the newly fitted functions are closely inspected to derive simple analytical formulas for an extended many-body potential. Chapter 4 shows that a previously suggested Gaussian process regression-based geometry optimization algorithm can be improved significantly by using molecule-specific coordinates instead of simple Cartesian coordinates. Since locating transition states is a crucial step for the computational description of catalysis, several machine learning models are compared with respect to their ability to accelerate a transition state search algorithm in Chapter 5. Chapter 6 covers a significantly more fundamental problem by applying the insights from gradient-based geometry optimization using machine learning to a toy model in the field of orbital-free density functional theory. The resulting improved model of the kinetic energy functional is shown to be significantly more robust especially during the iterative search for minimum energy densities. Finally, a conclusion and an outlook are provided in Chapter 7.

# 2. Machine learning in computational chemistry methods

The main goal of most computational chemistry calculations is to solve the time-independent Schrödinger equation,

$$H_{\text{mol}}(\mathbf{R},\mathbf{r})|\Psi(\mathbf{R},\mathbf{r})\rangle = E|\Psi(\mathbf{R},\mathbf{r})\rangle, \tag{2.1}$$

for the non-relativistic molecular Hamiltonian, which (in atomic units $\hbar = m_{\text{e}} = a_0 = e = 1$) is given by

$$H_{\text{mol}}(\mathbf{R},\mathbf{r}) = -\sum_a \frac{1}{2M_a}\nabla^2_{\mathbf{R}_a} - \sum_i \frac{1}{2}\nabla^2_{\mathbf{r}_i} + \sum_{a>b}\frac{Z_aZ_b}{|\mathbf{R}_a-\mathbf{R}_b|} - \sum_{a,i}\frac{Z_a}{|\mathbf{R}_a-\mathbf{r}_i|} + \sum_{i>j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}, \tag{2.2}$$

where $M_a$, $\mathbf{R}_a$, and $Z_a$ are the masses, positions and charges of the $N_{\text{nuc}}$ atomic nuclei, denoted by indices $a$ and $b$ and $\mathbf{r}_i$ are the positions of the $N_{\text{elec}}$ electrons denoted by indices $i$ and $j$.

A first step towards solving Equation 2.1 is the so-called Born-Oppenheimer approximation.[14] Motivated by the large difference in mass between nucleons and electrons, the nuclear and electronic degrees of freedom are separated by basis expansion of the total wave function,

$$\Psi(\mathbf{R},\mathbf{r}) = \sum_i \chi_i(\mathbf{R})\psi_i(\mathbf{R},\mathbf{r}), \tag{2.3}$$

where the $\chi_i(\mathbf{R})$ are the expansion coefficients and the basis functions $\psi_i(\mathbf{R},\mathbf{r})$ are typically chosen to be the eigenfunctions of the electronic Hamiltonian,

$$H_{\text{elec}}(\mathbf{R},\mathbf{r}) = -\sum_i \frac{1}{2}\nabla^2_{\mathbf{r}_i} + \sum_{a>b}\frac{Z_aZ_b}{|\mathbf{R}_a-\mathbf{R}_b|} - \sum_{a,i}\frac{Z_a}{|\mathbf{R}_a-\mathbf{r}_i|} + \sum_{i>j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}. \tag{2.4}$$

Inserting the expanded wave function in the molecular Schrödinger equation and multiplying by $\langle\psi_j(\mathbf{R},\mathbf{r})|$ from the left gives

$$\langle\psi_j(\mathbf{R},\mathbf{r})|\left(-\sum_a\frac{1}{2M_a}\nabla^2_{\mathbf{R}_a} + H_{\text{elec}}\right)\sum_i\chi_i(\mathbf{R})|\psi_i(\mathbf{R},\mathbf{r})\rangle = \langle\psi_j(\mathbf{R},\mathbf{r})|E\sum_i\chi_i(\mathbf{R})|\psi_i(\mathbf{R},\mathbf{r})\rangle. \tag{2.5}$$

Using the fact that the basis functions are solutions to the electronic Schrödinger equation $H_{\text{elec}}|\psi_i(\mathbf{R},\mathbf{r})\rangle = E_i|\psi_i(\mathbf{R},\mathbf{r})\rangle$ and orthonormal $\langle\psi_j(\mathbf{R},\mathbf{r})|\psi_i(\mathbf{R},\mathbf{r})\rangle = \delta_{ij}$ yields

$$\begin{aligned}-\sum_a\frac{1}{2M_a}\sum_i\Big[2\langle\psi_j(\mathbf{R},\mathbf{r})|\nabla_{\mathbf{R}_a}|\psi_i(\mathbf{R},\mathbf{r})\rangle\left(\nabla_{\mathbf{R}_a}\chi_i(\mathbf{R})\right) + \langle\psi_j(\mathbf{R},\mathbf{r})|\nabla^2_{\mathbf{R}_a}|\psi_i(\mathbf{R},\mathbf{r})\rangle\chi_i(\mathbf{R})\Big]\\ + \left(-\sum_a\frac{1}{2M_a}\nabla^2_{\mathbf{R}_a} + E_j\right)\chi_j(\mathbf{R}) = E\chi_j(\mathbf{R}).\end{aligned} \tag{2.6}$$

In the Born-Oppenheimer approximation the non-adiabatic coupling terms are neglected, $\langle\psi_j(\mathbf{R},\mathbf{r})|\nabla_{\mathbf{R}_a}|\psi_i(\mathbf{R},\mathbf{r})\rangle\nabla_{\mathbf{R}_a} = \langle\psi_j(\mathbf{R},\mathbf{r})|\nabla^2_{\mathbf{R}_a}|\psi_i(\mathbf{R},\mathbf{r})\rangle = 0$, simplifying the nuclear Schrödinger equation to

$$\left(-\sum_a \frac{1}{2M_a}\nabla^2_{\mathbf{R}_a} + E_j(\mathbf{R})\right)\chi_j(\mathbf{R}) = E\chi_j(\mathbf{R}). \tag{2.7}$$

This leads to the picture of nuclei moving on the so-called potential energy surface (PES), the hyperplane of solutions to the electronic Schrödinger equation which assigns a potential energy value $E_j(\mathbf{R})$ to each $3N_{\text{nuc}}$-dimensional molecular geometry. Machine learning-based algorithms for solving the nuclear Schrödinger equation have been suggested by several groups. A review of these approaches was given by Sergei Manzhos.[11] The remainder of this thesis, however, is focused on solving the electronic Schrödinger equation and the classical motion of the nuclei on the resulting PES.

## 2.1. Ab initio computational chemistry

Methods of solving the electronic Schrödinger equation without empirical parameters are commonly referred to as *ab initio* methods. What follows is a quick review of some of these algorithms intended to provide context for recently suggested machine learning-based modifications. The explanations of standard computational chemistry methods presented in this section closely follow the corresponding chapters in "Introduction to Computational Chemistry" by Frank Jensen.[15]

### 2.1.1. Wave function-based methods

The Hartree-Fock (HF) method represents the basis for all wave function-based *ab initio* methods. Following the variational principle, the energy of a trial wave function is minimized to obtain the ground state energy $E_0$.

The first approximation in Hartree-Fock theory is due to the fact that the electronic wave function $\psi$ is represented by a single Slater determinant,

$$\Phi = \frac{1}{\sqrt{N_{\text{elec}}!}}\begin{vmatrix} \phi_1(1) & \phi_2(1) & \dots & \phi_{N_{\text{elec}}}(1) \\ \phi_1(2) & \phi_2(2) & \dots & \phi_{N_{\text{elec}}}(2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(N_{\text{elec}}) & \phi_2(N_{\text{elec}}) & \dots & \phi_{N_{\text{elec}}}(N_{\text{elec}}) \end{vmatrix}, \tag{2.8}$$

where the so-called molecular orbitals $\phi_i(k)$ are orthonormal singe-electron wave functions ($\langle\phi_i|\phi_j\rangle = \delta_{ij}$) occupied by electron $k$, and the determinant is used to build an anti-symmetric wave function. Recently, several groups have suggested neural network-based representations of the wave function[16–21] and machine learning algorithms for the optimization of wave function parameters.[22–24]

The energy of a Slater determinant is given by

$$E = \langle\Phi|H_{\text{elec}}|\Phi\rangle = \sum_i^{N_{\text{elec}}}\langle\phi_i|h_i|\phi_i\rangle + \frac{1}{2}\sum_{ij}^{N_{\text{elec}}}\left(\langle\phi_j|J_i|\phi_j\rangle - \langle\phi_j|K_i|\phi_j\rangle\right) + V_{\text{nn}}, \tag{2.9}$$

where the nuclear repulsion,

$$V_{\mathrm{nn}} = \langle\Phi| \sum_{a>b}^{N_{\mathrm{nuc}}} \frac{Z_a Z_b}{|\mathbf{R}_a - \mathbf{R}_b|} |\Phi\rangle = \langle\Phi|\Phi\rangle \sum_{a>b}^{N_{\mathrm{nuc}}} \frac{Z_a Z_b}{|\mathbf{R}_a - \mathbf{R}_b|} = \sum_{a>b}^{N_{\mathrm{nuc}}} \frac{Z_a Z_b}{|\mathbf{R}_a - \mathbf{R}_b|}, \tag{2.10}$$

is an additive scalar, the core Hamiltonian,

$$h_i = -\frac{1}{2}\nabla_i^2 - \sum_{a}^{N_{\mathrm{nuc}}} \frac{Z_a}{|\mathbf{R}_a - \mathbf{r}_i|}, \tag{2.11}$$

is the sum of both one-electron operators, and the Coulomb operator,

$$J_i|\phi_j(2)\rangle = \langle\phi_i(1)|\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}|\phi_i(1)\rangle|\phi_j(2)\rangle, \tag{2.12}$$

and the exchange operator,

$$K_i|\phi_j(2)\rangle = \langle\phi_i(1)|\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}|\phi_j(1)\rangle|\phi_i(2)\rangle, \tag{2.13}$$

are two-electron operators.

Variation of the energy with respect to the molecular orbitals under the constraint of orthonormality yields the Hartree-Fock equations,

$$F_i\phi_i = \varepsilon_i\phi_i, \tag{2.14}$$

where the $\varepsilon_i$ are referred to as molecular orbital energies and the Fock operator is defined as

$$F_i = h_i + \sum_{j}^{N_{\mathrm{elec}}} (J_j - K_j). \tag{2.15}$$

In a final approximation the molecular orbitals are expanded using a linear combination of atomic orbitals (LCAO),

$$\phi_i = \sum_{j}^{N_{\mathrm{basis}}} c_{ij}\varphi_j. \tag{2.16}$$

Combining Equation 2.14 and Equation 2.16 yields the Roothaan-Hall equations,

$$FC = SC\varepsilon, \tag{2.17}$$

where $F$ is the Fock matrix $F_{\alpha\beta} = \langle\varphi_\alpha|F|\varphi_\beta\rangle$ in the space of atomic orbitals, $C$ is the matrix of basis set expansion coefficients, $S$ is the overlap matrix $S_{\alpha\beta} = \langle\varphi_\alpha|\varphi_\beta\rangle$, and $\varepsilon$ is a diagonal matrix containing the molecular orbital energies.

Equation 2.17 has to be solved iteratively since the two-electron interactions in the Fock operator depend on the set of occupied orbitals and thereby the coefficient matrix $C$. Reasonable starting points for the iteration are given by the extended Hückel theory[25] or the superposition of atomic densities approach.[26] In the master's thesis of Johannes Cartus[27] we investigated the computational savings achieved by a neural network-based initial guess.

The restriction to a single Slater-determinant in Hartree-Fock results in a mean-field approximation, where each single electron only interacts with the averaged density of the remaining electrons. Methods for calculating the resulting deviation from the exact energy, the correlation energy,

$$E^{\mathrm{c}} = E^{\mathrm{exact}} - E^{\mathrm{HF}}, \tag{2.18}$$

are referred to as post-Hartree-Fock methods.

In the configuration interaction (CI) approach the wave function is represented as a linear combination of Slater determinants,

$$\psi_{\mathrm{CI}} = a_0\Phi_{\mathrm{HF}} + \sum_{\mathrm{S}} a_{\mathrm{S}}\Phi_{\mathrm{S}} + \sum_{\mathrm{D}} a_{\mathrm{D}}\Phi_{\mathrm{D}} + \sum_{\mathrm{T}} a_{\mathrm{T}}\Phi_{\mathrm{T}} + \dots \tag{2.19}$$

where the additional Slater determinants are constructed from the Hartree-Fock ground state $\Phi_{\mathrm{HF}}$ by single (S), double (D), triple (T), etc. excitation. The expansion coefficients can be calculated by diagonalizing the so-called CI matrix. Due to the combinatorial nature of the approach the number of possible determinants grows quickly with the number of excited electrons restricting calculations for all but the smallest systems to single and double excitations (CISD). However, only a small percentage of these configurations contributes significantly to the ground state wave function. Jeremy Coe suggested on the fly training of a neural network to predict the important configurations in an iterative CI calculation.[28,29] A similar approach for determining the active space for a multi-configuration self-consistent field calculation, where the molecular orbital coefficients and the CI expansion coefficients are optimized simultaneously, was suggested by Jeong et al.[30]

The Møller-Plesset (MP) method uses many-body perturbation theory to recover the correlation energy. The molecular Hamiltonian is split in an unperturbed part built from the sum of Fock operators,

$$H_0 = \sum_{i}^{N_{\mathrm{elec}}} F_i, \tag{2.20}$$

and a "small" perturbation operator consisting of the difference between the full electron-electron interaction and the averaged interaction

$$H_1 = V_{ee} - 2\langle V_{ee}\rangle. \tag{2.21}$$

The factor two stems from the fact that Fock operators as defined in Equation 2.15 count the mean-field interaction twice. This double counting is corrected by the first-order perturbation term already contained in the Hartree-Fock method (see Equation 2.9). Therefore, the lowest order correction containing correlation energy is given by the second-order Møller-Plesset (MP2) method.

To overcome the lack of size consistency in truncated CI methods, the coupled cluster (CC) approach uses a different expansion in the basis of excited Slater determinants generated from a Hartree-Fock reference using an excitation operator $T$. Using Equation 2.19, the operator $T$ can be defined as $\psi_{\mathrm{CI}} = (a_0 + T)\Phi_{\mathrm{HF}}$. The coupled cluster wave function is given by

$$\psi_{\mathrm{CC}} = e^{T}\Phi_{\mathrm{HF}} = \sum_{k=0}^{\infty} \frac{1}{k!} T^k \Phi_{\mathrm{HF}}, \tag{2.22}$$

where the (implicit) expansion coefficients are referred to as amplitudes. This exponential approach yields significantly higher accuracy even for truncated expansions such as the CCSD method, using only single and double excitations $T = T_S + T_D$, as the correlation energy due to certain contributions is accounted for to infinite order. The downside of the more sophisticated wave function is that the amplitudes have to be calculated by iteratively solving the so-called coupled cluster equations.

Townsend and Vogiatzis showed that the number of iterations needed to solve these coupled cluster equations can be reduced significantly by using machine learning to predict the CCSD amplitudes from the results of the reference calculation.[31]

The CCSD(T) method, in which the contribution due to triple excitations is calculated using perturbation theory, is often referred to as the "gold standard" of computational chemistry and serves as reference for many investigations of machine learning models.

Both the error due to the limited number of basis functions (basis set approximation) and the mean-field approximation can be reduced in a systematic increase in basis set size and correlation contributions. Several schemes make use of this structure by suggesting simple formulas to extrapolate toward the exact limit from select calculations at lower accuracy.[32–35] In fact, several families of basis sets, such as Dunnings "correlation consistent" basis set,[36] are explicitly designed with the goal of extrapolation in mind. Recently, several groups have proposed machine learning-based alternatives for these simple extrapolation formulas.

Balabin and Lomakina showed that neural networks can be used to extrapolate the results of small basis set calculations toward the complete basis set (CBS) limit, i.e. along arrow (1) in Figure 2.1, even if the basis set family does not offer a dedicated extrapolation formula.[39] Schütt and VandeVondele demonstrated that the number of basis functions needed for a given accuracy can be reduced by using a machine learning-based geometry adapted atom-centered basis.[40]

The highest reduction of computational effort is achieved by methods for extrapolation to higher levels of theory, i.e. along arrow (2) in Figure 2.1. In the "molecular-orbital-based machine learning" method by Welborn et al. matrix elements of the Fock, Coulomb, and exchange matrices are used to extrapolate from a Hartree-Fock reference calculation toward MP2 and CCSD results.[41–43] An alternative approach using only the projected density matrix as input to a linear regression or neural network model was presented by Chen et al.[44] Townsend and Vogiatzis showed that their scheme of calculating coupled cluster amplitudes from MP2 results can also be used as efficient extrapolation scheme.[45]

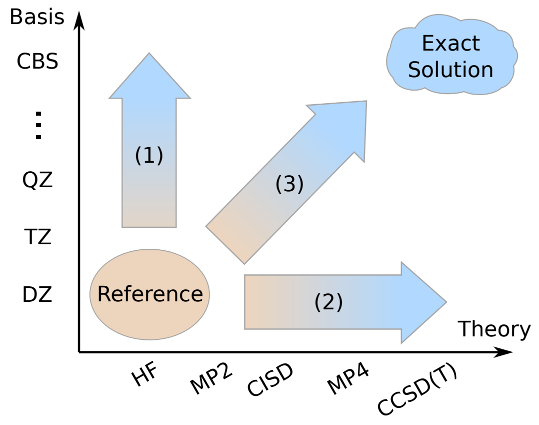


Figure 2.1.: Correction of the two approximations in the HF method. Inspired by similar graphics in Ref. 37 and Ref. 38.

Simultaneous extrapolation of both the number of basis functions and the level of theory, i.e. along arrow (3) in Figure 2.1, using a neural network was investigated in the group of Hiromi Nakai.[38,46]

Similarly, the machine learning-based extrapolation of computational results toward various experimental observables has been suggested by several groups.[47–53] These methods typically use density functional theory calculations (see Section 2.1.2) as reference since the hierarchical relationship of the wave function-based methods can not be exploited in this application.

Similar extrapolation models that do not use any of the reference calculation results as input, but instead model the deviation from higher level methods as function of the geometry such as the $\Delta$-learning approach by Ramakrishnan et al.,[54] can be constructed from the PES models presented in Section 2.2.

### 2.1.2. Density functional theory

The Hohenberg-Kohn theorem[55] states that the ground state energy of a many-electron system is uniquely determined by the electron density $n(\mathbf{r})$. Therefore, even without knowledge of the underlying wave function, the energy can be calculated as a functional of the density,

$$E[n] = T[n] + E_{\mathrm{ne}}[n] + E_{\mathrm{ee}}[n]. \tag{2.23}$$

This would result in a significant reduction in the computational effort required to calculate the PES since the two spin electron densities are functions of only three spatial degrees compared to the $4N_{\mathrm{elec}}$ degrees freedom of a wave function (three spatial, one spin degree for each electron).

Motivated by the Hartree-Fock method, the electron-electron interaction functional $E_{\mathrm{ee}}[n]$ can further be divided into the Coulomb (or Hartree) functional $J[n]$ and the exchange functional $K[n]$. The attraction between nuclei and electrons and the Coulomb part of the electron-electron interaction can be described by their classical counterparts

$$E_{\mathrm{ne}}[n] = -\sum_a \int \frac{Z_a n(\mathbf{r})}{|\mathbf{R}_a - \mathbf{r}|} \, \mathrm{d}\mathbf{r} \tag{2.24}$$

and

$$J[n] = \frac{1}{2} \int \int \frac{n(\mathbf{r}) n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \, \mathrm{d}\mathbf{r} \, \mathrm{d}\mathbf{r}'. \tag{2.25}$$

While several properties of the functionals for the kinetic energy $T[n]$ and the exchange energy $K[n]$ have been derived, the exact functional forms are unknown and remain an active field of research. Similar to post-Hartree-Fock methods an additional correlation functional $E_{\mathrm{C}}[n]$ is needed to describe the deviation of the mean-field approximation from the full electron-electron interaction.

#### Kohn-Sham density functional theory

In the Kohn-Sham approach the kinetic energy is calculated as sum of the kinetic energy of a single Slater-determinant,

$$T_S = \sum_i \langle \phi_i | -\frac{1}{2} \nabla^2 | \phi_i \rangle, \tag{2.26}$$

which is the exact solution of a non-interacting system, and a correction term $T_C$ due to the electron correlation in interacting systems. These Kohn-Sham orbitals are also used to represent the electron density,

$$n(\mathbf{r}) = \sum_i |\phi_i|^2. \tag{2.27}$$

The total energy in the Kohn-Sham approach is given by

$$E[n] = T_S[n] + E_{\mathrm{ne}}[n] + J[n] + E_{\mathrm{xc}}[n], \tag{2.28}$$

where the unknown contributions are collected in the exchange-correlation functional,

$$E_{\mathrm{xc}}[n] = (T[n] - T_S[n]) + (E_{\mathrm{ee}}[n] - J[n]). \tag{2.29}$$

The remaining task therefore is to find an expression for the exchange-correlation functional, which is typically divided into a sum of two separate terms for the exchange and correlation contributions,

$$E_{\mathrm{xc}}[n] = E_{\mathrm{x}}[n] + E_{\mathrm{c}}[n]. \tag{2.30}$$

A formal classification of approximate exchange-correlation functionals based on their complexity, commonly referred to as Jacob's ladder of density functionals, was introduced by Perdew and Schmidt.[56]

The first rung of this ladder is occupied by the so-called local density approximation (LDA), which corresponds to the limit of a slowly varying density, where the exchange functional is given by[57]

$$E_{\mathrm{x}}^{\mathrm{LDA}} = C_{\mathrm{x}} \int n(\mathbf{r})^{4/3} \, \mathrm{d}\mathbf{r}, \tag{2.31}$$

with $C_{\mathrm{x}} = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}$. Similar results have been derived for the correlation energy in both the high[58–60] and low density limit.[61] A formula for interpolation between these limits was suggested by Vosko, Wilk and Nusair (VWN).[62]

Semi-local functionals improve on the local expressions by including derivatives of the density. The Generalized-Gradient-Approximation (GGA) functionals on the second rung of Jacob's ladder depend on the first derivative of the density, typically in the form of the rescaled variable

$$s = \frac{|\nabla n(\mathbf{r})|}{n(\mathbf{r})^{4/3}}. \tag{2.32}$$

This category includes some of the most widely used functionals, and most cited scientific publications, such as the B88 exchange functional by Axel Becke,[63] the Lee-Yang-Parr (LYP) correlation functional,[64] and the Perdew-Burke-Ernzerhof (PBE) exchange-correlation functional.[65]

Third rung functionals, so-called meta-GGA functionals, depend on higher order derivatives or alternatively on the orbital kinetic energy density,

$$\tau = \frac{1}{2} \sum_{i}^{N_{\mathrm{occ}}} |\nabla \phi_i|^2, \tag{2.33}$$

where $N_{\mathrm{occ}}$ refers to the number of occupied Kohn-Sham orbitals.

The hybrid functionals, which occupy the forth rung on the ladder, additionally include some amount of "exact" exchange. Since the Kohn-Sham approach uses a single Slater determinant this corresponds to the exchange energy $K$ given by the Hartree-Fock method. Notable examples are the B3 family of functionals by Becke[66] and its most prominent member B3LYP[67] given by

$$E_{\mathrm{xc}}^{\mathrm{B3LYP}} = (1-a)E_{\mathrm{x}}^{\mathrm{LDA}} + aE_{\mathrm{x}}^{\mathrm{exact}} + b(E_{\mathrm{x}}^{\mathrm{B88}} - E_{\mathrm{x}}^{\mathrm{LDA}}) + (1-c)E_{\mathrm{c}}^{\mathrm{VWN}} + cE_{\mathrm{c}}^{\mathrm{LYP}}, \tag{2.34}$$

with $a$, $b$, and $c$ as fitting parameters.

The fifth and final rung contains functionals that additionally depend on the unoccupied Kohn-Sham orbitals and fully non-local density functionals.

While data-driven approaches for fitting of free parameters have always been a core part of the development of exchange-correlation density functionals, in recent years several groups have suggested functionals that rely heavily on modern machine learning concepts.

One of the earliest examples of machine learning-based density functionals is the neural network fit of the exchange-correlation functional by Tozer et al.[68]

Zheng et al.[69] presented a neural network approach of determining the weighting parameters for the terms in the B3LYP functional (Equation 2.34) based on system-specific properties such as the multiplicity and the dipole moment.

Building on the idea of error estimation for density functionals,[70] Wellendorff et al. suggested a Bayesian learning approach to determine the parameters of both a non-local[71] and a meta-GGA[72] exchange-correlation functional.

Similar empirical parameters are used in range-separated functionals. The electron-electron interaction is divided into a short and a long range part,

$$\frac{1}{r} = \frac{1 - \operatorname{erf}(\mu r)}{r} + \frac{\operatorname{erf}(\mu r)}{r}, \tag{2.35}$$

where $\operatorname{erf}(r)$ is the error function and $\mu$ is a parameter, and different functionals are used for the two contributions. Lui et al. presented a neural network to determine a system-specific range-separation parameter $\mu$ for a long-range-corrected exchange-correlation functional.[73]

The transferability of neural network-based approximations to the whole Hartree-exchange-correlation potential for one-dimensional systems was investigated by Nagai et al.[74]

Ji and Jung[75] presented a local environment descriptor of the electron density, which was used in combination with kernel ridge regression to fit the exchange-correlation potential in 16 small molecular systems. A similar neural network-based approach, using Maxwell-Cartesian spherical harmonics as environment descriptors, was presented by Lei and Medford.[76] Dick and Fernandez-Serra used an expansion of the density in atom-centered basis functions as input to a neural network-based correction to the PBE functional.[77,78]

The same idea of exploiting the dependence on local environments is used in convolutional neural network-based exchange-correlation functional models. Schmidt et al. trained a one-dimensional convolutional neural network to reproduce both the exact exchange–correlation energy and its derivative in a two electron model. Ryczko et al.[79] showed that deep two-dimensional convolutional neural networks can be used to predict all energy components in Kohn-Sham density functional theory, and even outright replace it by additionally mapping the external potential to the corresponding electron density. A three-dimensional convolutional neural network for the exchange-correlation functional in small two and four electron systems was investigated by Zhou et al.[80]

Nagai et al.[81] suggested neural network-based semi-local exchange-correlation functionals on the local, GGA, and meta-GGA level and compared their performance to existing density functionals on a large set of benchmark molecules.

### Orbital-free density functional theory

The re-introduction of orbitals in the Kohn-Sham formalism results in a loss of the superior scaling compared to wave function-based methods. Therefore, in order to realize the full potential of density functional theory, an approximation to the elusive exact kinetic energy functional, allowing for orbital-free calculations, has to be found.

A starting point, the local density approximation of the kinetic energy functional, known as Thomas-Fermi functional, is given by[82–84]

$$T^{\mathrm{TF}}[n] = C_{\mathrm{TF}} \int n(\mathbf{r})^{5/3} \; \mathrm{d}\mathbf{r}, \tag{2.36}$$

where $C_{\mathrm{TF}}[n] = \frac{3}{10}(3\pi^2)^{2/3}$. Von Weizsäcker suggested a gradient-based correction, which is exact for the case of a single orbital,[85]

$$T^{\mathrm{vW}} = \frac{1}{8} \int \frac{|\nabla n(\mathbf{r})|^2}{n(\mathbf{r})} \; \mathrm{d}\mathbf{r}. \tag{2.37}$$

These simple approximations are, however, not accurate enough to be useful in computational chemistry calculation. Suggested corrections including higher order derivatives[86,87] have been shown to diverge in the long-range limit.[88] Therefore, most modern day research is focused on non-linear functionals of the form

$$T^{\mathrm{NL}}[n] = C \int \int n(\mathbf{r})^{\alpha} \omega[n](\mathbf{r}, \mathbf{r}') n(\mathbf{r}')^{\beta} \; \mathrm{d}\mathbf{r} \; \mathrm{d}\mathbf{r}', \tag{2.38}$$

where $\omega$ is a dimensionless kernel, typically assumed to be a function of $|\mathbf{r} - \mathbf{r}'|$, and the exponents $\alpha$ and $\beta$ are parameters.

Snyder et al.[89] showed that the kinetic energy functional can be approximated using machine learning by training a kernel ridge regression model for a one-dimensional system of non-interacting particles. This approach has also been successfully used to describe bond breaking[90] and the universal part of the total energy density functional.[91]

Traditional approaches of finding density functionals rely heavily on the known properties of the exact functional. For example, the exact kinetic energy functional for non-interacting particles is known to transform as[92]

$$T[n_\lambda] = \lambda^2 T[n] \tag{2.39}$$

for a coordinate scaling,

$$n_\lambda(\mathbf{r}) = \lambda^3 n(\lambda \mathbf{r}), \tag{2.40}$$

that conserves the total number of particles $\int n_\lambda(\mathbf{r}) \; \mathrm{d}\mathbf{r} = \int n(\mathbf{r}) \; \mathrm{d}\mathbf{r} = N$. All of the simple approximations such as the Thomas-Fermi and the von Weizsäcker functionals fulfill this condition. Hollingsworth et al.[93] investigated if imposing this scaling law on a kernel ridge regression model can improve the machine learning kinetic energy functional.

While these models achieved chemical accuracy using only a minimal amount of training data, this mapping of density to kinetic energy represents only part of the orbital-free density functional theory problem. Following the variational principle, every density functional theory calculation involves minimizing Equation 2.23 with respect to the density

$$\frac{\delta E[n]}{\delta n} = \frac{\delta T[n]}{\delta n} + \frac{\delta E_{\mathrm{ne}}[n]}{\delta n} + \frac{\delta E_{\mathrm{ee}}[n]}{\delta n} \overset{!}{=} 0. \tag{2.41}$$

Accurate predictions of the kinetic energy functional derivative are therefore just as important as achieving chemical accuracy on the kinetic energy itself. The noisy nature and bad extrapolation behavior of machine learning models for the kinetic energy functional often result in the exploration of unphysical density regimes during the iterative search for the minimum energy density. Snyder et al.[94] suggested a projection scheme based on principal component analysis to restrict the search to the subspace of densities for which the machine learning approximation is valid.

An alternative solution to this problem was proposed by Brockherde et al.[95] Instead of minimizing the total energy functional, they used a second machine learning model to directly predict the minimum energy density for any given nuclear potential, a process they refer to as Hohenberg-Kohn mapping. This density is then used to predict the total energy using machine learning-based orbital-free density functional theory. Motivated by the $\Delta$-learning approach,[54] the same group recently showed that this two-stage approach can also be used to achieve coupled cluster level accuracy at the computational cost of orbital-free density functional theory.[96]

In Ref. 97, reprinted as Chapter 6 of this thesis, we showed that the problem of noisy functional derivatives can be reduced significantly by including reference data for the functional derivative into the training set. Since the computational effort of evaluating a kernel ridge regression model increases with the size of the data set, we additionally tested a convolutional neural network model which offers a constant evaluation time, independent of the number of training examples.

A different convolutional neural network-based approach was presented by Yao and Parkhill.[98] They used one-dimensional scans of the density and its gradient along bond directions as input to model the kinetic energy of alkane molecules.

Seino et al.[99] suggested a semi-local kinetic energy functional in the form of a neural network mapping of the density and its first three derivatives to the kinetic energy density $\tau$, where

$$T[n] = \int \tau[n] \; \mathrm{d}\mathbf{r}. \tag{2.42}$$

In later works they slightly modified their approach by fitting an enhancement factor,

$$F_{\mathrm{enh}} = \frac{\tau}{\tau^{\mathrm{TF}} + \tau^{\mathrm{vW}}}, \tag{2.43}$$

instead of the total kinetic energy density, and adding the inverse distances between $\mathbf{r}$ and the nuclei $\{1/|\mathbf{r} - \mathbf{R}_a|, \dots, 1/|\mathbf{r} - \mathbf{R}_{N_a}|\}$ to the list of neural network inputs.[100,101]

Golub and Manzhos[102] presented a similar semi-local approach. However, instead of using plain derivatives of the density they used five rescaled input variables,

$$z_1 = n(\mathbf{r})^{5/3}, \qquad z_2 = \frac{1}{8}\frac{|\nabla n(\mathbf{r})|^2}{n(\mathbf{r})} - \frac{1}{4}\Delta n(\mathbf{r}), \qquad z_3 = n(\mathbf{r})^{1/3}\left(\frac{\Delta n(\mathbf{r})}{n(\mathbf{r})}\right)^2,$$
$$z_4 = n(\mathbf{r})^{1/3}\left(\frac{\Delta n(\mathbf{r})}{n(\mathbf{r})}\right)\left(\frac{\nabla n(\mathbf{r})}{n(\mathbf{r})}\right)^2, \qquad z_5 = n(\mathbf{r})^{1/3}\left(\frac{\nabla n(\mathbf{r})}{n(\mathbf{r})}\right)^4, \tag{2.44}$$

motivated by the terms in the fourth-order gradient expansion to fit the kinetic energy density.

In a follow-up article, Manzhos and Golub[103] investigated linear and kernel-based fits of the kinetic energy density as a function of density dependent variables such as the Thomas-Fermi $\tau_{\mathrm{TF}}$, and von Weizsäcker kinetic energy density $\tau_{\mathrm{vW}}$.

## 2.2. Potential energy models

Most research is focused on directly learning the PES, i.e. mapping a given arrangement of atoms to the corresponding potential energy. These machine learning potentials represent a continuation of earlier data-driven approximations such as force fields or direct interpolation and fitting of the PES.

In order to be useful in computational chemistry calculations these approximations should possess some of the properties of the exact PES. An example is the invariance with respect to transformations such as rotation and translation of the coordinate system and the permutation of atoms of the same element. Other desirable properties for the machine learning models include the prediction of a smooth energy surface, to allow for the evaluation of forces and the Hessian matrix, and the ability to generalize to arbitrarily sized systems.

### 2.2.1. Encoding invariance

Rasmussen and Williams[104] list three approaches to incorporate known invariances into machine learning models. These methods are illustrated here by training neural networks on the simple function,

$$f(x) = x + \sin(\pi x), \tag{2.45}$$

which is point-symmetric, i.e. invariant to inversion

$$f(-x) = -f(x). \tag{2.46}$$

The presented neural networks consist of a single hidden layer containing 16 neurons with a hyperbolic tangent activation function and are trained on the mean squared error of 7 data points using the L-BFGS algorithm[105,106] as implemented in the tensorflow-probability package.[107] Note that all of the presented results are handpicked examples to highlight the strengths and weaknesses of each approach and can not demonstrate some of the intricacies such as added complexity in training.

Figure 2.2 shows the result of fitting a neural network without any attempts to account for the symmetry. As expected, the neural network accurately reproduces the function at the training examples. However, without knowledge of the point-symmetry of the underlying reference function, the generalization to other values of $x$ is extremely poor. In fact, this result would be classified as a typical example of overfitting.

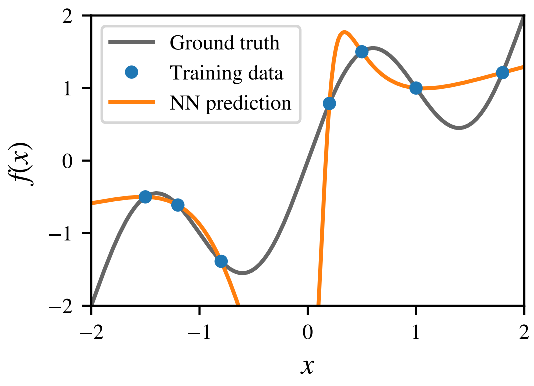


Figure 2.2.: Neural network fit of the training data without symmetry adaptions.

Given enough training data, machine learning models should be able to learn these invariances. The first approach is therefore to generate more data by applying the transformation to existing training examples. This is referred to as data augmentation and a common technique employed to include rotational invariance into image classification models. One of the earliest applications is the distortion model by Baird.[108]

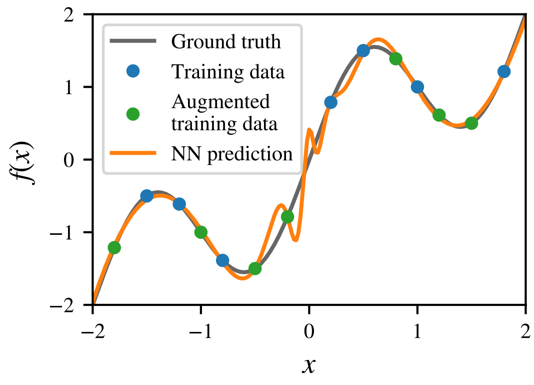


Figure 2.3.: Neural network fit of the augmented training data.

Figure 2.3 shows that this approach can significantly improve the generalization of the neural network fit for this simple example. For point-symmetry, the data augmentation approach results at most in a doubling of the number of training data, that is when applied to all examples. Note, however, that for more complex invariances such as the invariance with respect to permutation of atom of the same species this would yield a massive increase in the number of training examples that the model needs to be trained on. Even worse, there is no straight-forward strategy for extending the data augmentation approach to parametrized transformations. Translational invariance can for example be used to generate arbitrarily many additional training examples. Augmented data sets are therefore unable to fully cover this invariance.

Invariance with respect to these parametrized transformations are the main focus of the second approach, the “tangent prop” method by Simard et al.[109] Similar to other regularization techniques, the main idea is to include an additional term to the loss function in order to penalize deviations from the correct transformation behavior. For parametrized transformations this additional penalty term takes the form of a directional derivative, i.e. the tangent to the set of points that can be generated from applying the transformation to a data point. The modified loss function is given by

$$\begin{aligned}\tilde{\mathcal{L}} &= \mathcal{L} + \mu \sum_i^M \sum_j \left\| T_j(\mathbf{x}_i) - \left.\frac{\partial f_{\mathrm{ML}}(s_j(\mathbf{x},\alpha))}{\partial \alpha}\right|_{\mathbf{x}=\mathbf{x}_i,\alpha=0} \right\|^2 \\ &= \mathcal{L} + \mu \sum_i^M \sum_j \left\| T_j(\mathbf{x}_i) - \left.\frac{\partial f_{\mathrm{ML}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_i} \left.\frac{\partial s_j(\mathbf{x_i},\alpha)}{\partial \alpha}\right|_{\alpha=0} \right\|^2, \end{aligned} \tag{2.47}$$

where $\mathcal{L}$ is the original loss function, $f_{\mathrm{ML}}$ is the machine learning model, $s_j(\mathbf{x},\alpha)$ are transformations parametrized by $\alpha$, $T_j(\mathbf{x})$ are the target values for the directional derivative (for invariance $T_j(\mathbf{x}) = 0$), and $\mu$ is a scaling factor. Translational invariance for example can be enforced using three transformations corresponding to translation along the Cartesian axes,

$$s_1(\mathbf{x},\alpha) = \mathbf{x} + \alpha\hat{e}_x \qquad s_2(\mathbf{x},\alpha) = \mathbf{x} + \alpha\hat{e}_y \qquad s_3(\mathbf{x},\alpha) = \mathbf{x} + \alpha\hat{e}_z, \tag{2.48}$$

where $\hat{e}_x$, $\hat{e}_y$, and $\hat{e}_z$ denote unit vectors along the axes. For a potential energy model this yields the condition that the sum over all atomic forces must be equal to the zero vector. Note that the directional derivative in Equation 2.47 corresponds to a linearization of the transformation. For non-linear transformations this approach therefore only ensures local invariance, i.e. invariance with respect to small distortions. The “tangent prop” method can for example not be used to include invariance under (arbitrarily large) rotations.

Figure 2.5 shows the results of including an additional penalty term in the loss function. Since the point-symmetry of the example function in Equation 2.45 is a non-parametric transformation, it can not be included using the derivative approach in Equation 2.47. Instead, the penalty consists of the mean squared difference of the neural network prediction between the right hand and left hand side of Equation 2.46 evaluated on 101 evenly distributed values in the interval $[-2, 2]$. The regularization strength $\mu$ is set to 1. Note that while the penalty term can in theory be reduced arbitrarily far during the training, the resulting function is never perfectly invariant with respect to the transformations.

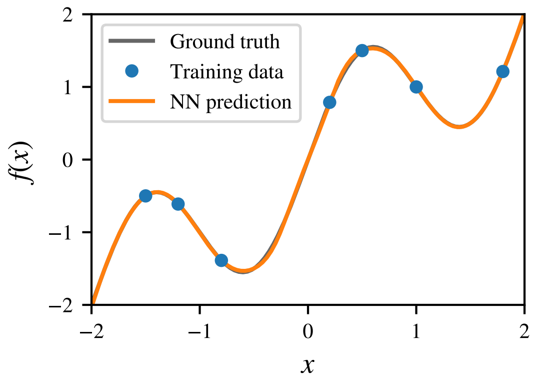


Figure 2.4.: Neural network fit with additional penalty term.

The third approach is to develop a suitable representation of the input which is invariant to the transformations. Conversion to this representation can either be a preprocessing step, the so-called feature extraction phase, or integrated into the model expression, which allows adapting its parameters during the training process, commonly referred to as deep learning. The most prominent example of this approach are convolutional neural networks,[110, 111] which by construction include translational invariance.

Figure 2.5 shows the results of fitting a neural network which includes point-symmetry. This is achieved by using the absolute value of $x$ as input to the machine learning model and multiplying the output by the sign function of $x$,

$$\tilde{f}_{\mathrm{ML}}(x) = f_{\mathrm{ML}}\left(|x|\right) \mathrm{sgn}(x). \tag{2.49}$$

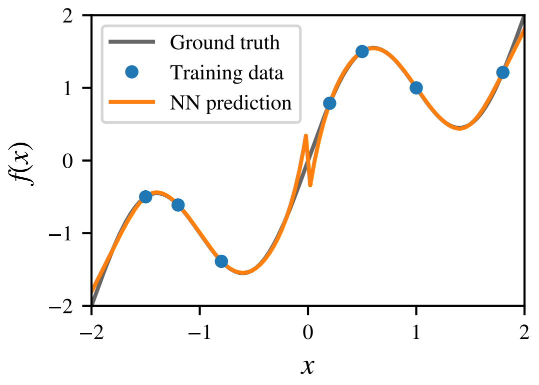


Figure 2.5.: Neural network fit using an invariant architecture.

While this results in a perfectly point-symmetric function it introduces a discontinuity at $x = 0$, highlighting the fact that additional conditions, such as smoothness of the result, might need to be considered during the construction of a transformation invariant model. This smoothness condition could for example be enforced by including an additional multiplicative correction,

$$\tilde{f}_{\mathrm{ML}}(x) = f_{\mathrm{ML}}\left(|x|\right) \mathrm{sgn}(x) \left(1 - e^{-x^2}\right). \tag{2.50}$$

For most applications in computational chemistry the invariances have to be reproduced exactly. Consider for example a molecular dynamics simulations. Evaluating the potential energy and atomic forces using a model without translational and rotational invariance would lead to variations in the energy as a molecule moves through the simulation box, thus breaking conservation of energy. Therefore, only the third approach represents a viable option.

### 2.2.2. Molecular and atomic descriptors

Several invariant representations of molecular geometries have been suggested as input for machine learning models. A simple example of such invariant features are internal coordinates, i.e. bond distances, bond angles, and dihedral angles. The Z-matrix representation built from these coordinates is, however, not uniquely defined and not invariant with respect to the ordering (indexing) of atoms.

A systematic extension is given by the construction of permutation-invariant polynomials (PIP) from these features. From a given set of internal coordinates, referred to as monomials in this context, permutation-invariant descriptors can be build as symmetric product. A detailed discussion of PIP and a review of their application to PES fitting was given by Braams and Bowman.[112] For illustrative purposes, a possible set of PIP for the $CO_2$ molecule, constructed from the three inter-atomic distances $R_{\mathrm{CC}}$, $R_{\mathrm{CO}_1}$, and $R_{\mathrm{CO}_2}$, is given by[113]

$$d_1 = R_{\mathrm{CC}}, \qquad d_2 = R_{\mathrm{CO}_1} R_{\mathrm{CO}_2}, \qquad d_3 = \left(R_{\mathrm{CO}_1} + R_{\mathrm{CO}_2}\right)/2. \tag{2.51}$$

The specific choice of polynomials is not unique; typically, PIP are generated using an automated procedure. In general the PIP approach is not transferable across different systems and limited to a few atoms as it involves no reduction of dimensionality.

The structure of covalently bonded molecules lends itself to a representation as an undirected graph, consisting of vertices (atoms) and edges (bonds). These graphs can be described by the so-called adjacency matrix $A$, where $A_{ij} = 1$ if the atoms $i$ and $j$ are bonded. Geometry information can be included by weighting the edges, for example using the inter-atomic distances,

$$A_{ij} = |\mathbf{R}_i - \mathbf{R}_j|. \tag{2.52}$$

Ralaivola showed that a molecular descriptor can be derived from (random) paths, i.e. sequences of edges, on the graph.[114]

Rupp et al.[115] suggested a vectorial descriptor built from the ordered list of eigenvalues of the Coulomb matrix,

$$M_{ij} = \begin{cases} \frac{1}{2} Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \text{for } i \neq j, \end{cases} \tag{2.53}$$

where the exponent in the diagonal term is derived from a polynomial fit of the energy of isolated atoms. Zero padding is used to ensure same length descriptor vectors for differently sized Coulomb matrices allowing for transferability across systems.

Von Lilienfeld et al. compared the performance of the Coulomb matrix descriptor to their suggestion of a Fourier series of atomic radial distribution functions.[116]

Since the representation of the Coulomb matrix by its eigenvalues might result in a loss of information, Hansen et al. investigated alternative ways of deriving a permutation-invariant descriptor from the matrix representation, such as sorting the rows and columns by their norm.[117] In a follow-up, they presented a modification of the Coulomb matrix descriptor specifically aimed at the robust description of chemical composition.[118] In this so-called Bag-of-Bonds approach the off-diagonal entries of the Coulomb matrix are used to construct a descriptor vector by sorting them into “bags” according to the atomic element of the involved atom $i$ and $j$. Equal size of the individual bags is again enforced by zero padding. The final descriptor vector is constructed by concatenation of all bags.

Huang and von Lilienfeld generalized the Bag-of-Bonds descriptor in their so-called "BA-representation", where BA stands for bonds and angles.[119] Instead of populating the bags with elements of the Coulomb matrix they suggested using the energy contributions of the universal force field (UFF) method[120] and introduced additional bags for three- and four-atom terms.

In a similar approach, Faber et al. investigated molecular descriptors generated from specially binned histograms of the bond distances, bond angles, and dihedral angles.[121]

The "many-body tensor representation" by Huo and Rupp represents a smooth, generalized version of the BA-representation.[122]

Many of the most successful machine learning PES models calculate the total energy as a sum of atomic contributions,

$$E = \sum_i^N \epsilon_i, \tag{2.54}$$

which allows for simple generalization to differently sized systems. The atomic energies $\epsilon_i$ are assumed to be functions of a descriptor vector of their local environment. This approach is often, though controversial, justified by the nearsightedness principle of quantum chemistry.[123]

As a first step in the construction of these atomic descriptors, the local environment of each atom $i$ is typically restricted by a cutoff sphere of radius $R_\text{cut}$. In order to avoid discontinuities as an atom $j$ enters and leaves the cutoff sphere, atomic descriptors are faded to zero at $R_{ij} = R_\text{cut}$ using a multiplicative cutoff function. A common choice is the cosine cutoff function,

$$f_\text{cut}(R_{ij}) = \begin{cases} \frac{1}{2}\left(\cos\left(\frac{\pi R_{ij}}{R_\text{cut}}\right) + 1\right) & \text{for } R_{ij} < R_\text{cut} \\ 0 & \text{for } R_{ij} \geq R_\text{cut}. \end{cases} \tag{2.55}$$

Note, however, that this results in a discontinuity in the second derivative at $R_{ij} = R_\text{cut}$. An infinitely differentiable alternative is given by

$$f_\text{cut}(R_{ij}) = \begin{cases} \frac{1}{1+\exp\left(\frac{R_\text{cut}\left(R_\text{cut}-2R_{ij}\right)}{R_{ij}\left(R_{ij}-R_\text{cut}\right)}\right)} & \text{for } R_{ij} < R_\text{cut} \\ 0 & \text{for } R_{ij} \geq R_\text{cut}. \end{cases} \tag{2.56}$$

The so-called atom-centered symmetry functions (ACSF), first introduced by Behler and Parrinello,[124] can be divided into two-body or radial descriptors and three-body or angular descriptors. Examples from the detailed discussion of ACSF in Ref. 125 are the radial descriptor

$$G_i^\text{rad} = \sum_j e^{-\eta(R_{ij}-R_s)^2} f_\text{cut}(R_{ij}) \tag{2.57}$$

and the angular descriptor

$$G_i^\text{ang} = 2^{1-\zeta} \sum_{j,k} (1 + \lambda \cos\theta_{ijk})^\zeta e^{-\eta\left(R_{ij}^2 + R_{ik}^2\right)} f_\text{cut}(R_{ij}) f_\text{cut}(R_{ik}). \tag{2.58}$$

Slightly modified functional forms of ACSF were suggested by Smith et al.[126] Botu and Ramprasand investigated projections of radial descriptors along select directions for the description of vectorial properties.[127]

An excellent first-principles discussion of atomic descriptors and their mathematical properties was given by Bartok et al.[128] They derived descriptors similar to ACSF by expanding the local environment,

$$\rho_i(\mathbf{R}) = \sum_j Z_j \delta\left(\mathbf{R} - \mathbf{R}_{ij}\right) f_{\text{cut}}(R_{ij}), \tag{2.59}$$

in a product of radial basis functions $g_n$ and spherical harmonics $Y_{lm}$,

$$\rho_i(\mathbf{R}) = \sum_n \sum_{l=0} \sum_{m=-l}^{l} c_{nlm} g_n(|\mathbf{R}|) Y_{lm}(\hat{\mathbf{R}}), \tag{2.60}$$

where $\hat{\mathbf{R}}$ denotes a unit vector along $\mathbf{R}$ and $c_{nlm}$ are expansion coefficients. The so-called power spectrum $p_{nl}$ and bispectrum $b_{nll_1l_2}$ can be constructed from the expansion coefficients,

$$p_{nl} = \sum_{m=-l}^{l} c^*_{nlm} c_{nlm}, \qquad b_{nll_1l_2} = \sum_{m=-l}^{l} \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} c^*_{nlm} C^{ll_1l_2}_{mm_1m_2} c_{nl_1m_1} c_{nl_2m_2}, \tag{2.61}$$

where the summation over $m$ ensures rotational invariance and $C^{ll_1l_2}_{mm_1m_2}$ are the Clebsch-Gordan coefficients. As an alternative to the radial basis functions they suggested encoding the distance information as a third angle $\theta_0 = |\mathbf{R}|/R_0$, where $R_0 > R_{\text{cut}}$ is a parameter. Using four-dimensional hyperspherical harmonics $U^j_{m'm}(\varphi, \theta, \theta_0)$ as basis for the expansion in Equation 2.60 the so-called SO(4) power spectrum and bispectrum descriptors can be constructed similar to their standard SO(3) counterparts by summation over the indices $m'$ and $m$.

In the standard approach separate radial and angular descriptors are defined for each possible pair and triple of atom types. This results in a rapid growth of the descriptor space with the number of different chemical elements and restricts the approach to data sets containing only a handful of atom types. Gastegger et al.[129] suggested modified ACSF using element-dependent weighting functions. The new radial descriptor is then given by

$$\tilde{G}_i^{\text{rad}} = \sum_j g(Z_j) e^{-\eta(R_{ij}-R_s)^2} f_{\text{cut}}(R_{ij}). \tag{2.62}$$

Similarly, the angular descriptor $G_i^{\text{ang}}$ is modified using the weighting function $h(Z_j, Z_k)$. They showed that using the simple weighting functions $g(Z_j) = Z_j$ and $h(Z_j, Z_k) = Z_j Z_k$ the generalization of machine learning energy predictions can be improved significantly.

An extension of this weighting method is obtained by allowing for vector-valued functions $\mathbf{g}(Z_j)$ and $\mathbf{h}(Z_j, Z_k)$. A weight vector $\mathbf{g}$ could for example be constructed from element-specific properties such as the ionization potential and the electron affinity. In the so-called "embedding" approach, commonly used in natural language processing, this mapping from a discrete input space to a real-valued vector is learned as part of the training procedure of a deep learning model.

The presented atomic descriptors can also be used to construct molecular descriptors for example using the "regularized entropy match" approach by De et al.[130]

### 2.2.3. Neural networks-based models

One of the earliest examples of a neural network-based PES model was presented by Sumpter and Noid in 1992.[131] Most early applications used permutation-invariant polynomials as input and are therefore limited to low dimensional systems. Extensive reviews of these efforts can be found in Ref. 3 and Ref. 4. A review of the current state of the art in neural network-based potential energy models for small systems was given by Manzhos and Carrington.[12]

In order to allow for simulations of large scale systems Hobday et al.[132] suggested to modify existing many-body potentials, such as the Tersoff potential,[133,134] using neural networks. In Chapter 3 we investigated a similar extension to a many-body potential by replacing some of the contributions in the “second moment approximation tight binding” potential with neural network expressions.

A more general approach was presented in a seminal article by Behler and Parrinello.[124] They suggested to model the total energy as a sum of atomic contributions (see Equation 2.54) described by element-specific atomic neural networks using atom-centered symmetry function descriptors as inputs.

Artrith et al.[135] extended the method to include long-range electrostatic interactions,

$$E_{el} = \frac{1}{2} \sum_{i,j} \frac{q_i q_j}{R_{ij}} \tag{2.63}$$

by modeling atomic charges $q_i$ using separate atomic neural networks. A comprehensive review of the method was given by Jörg Behler.[136]

A modification to the Behler-Parrinello approach was suggested by Liu and Kitchin who used a single neural network with multiple outputs for all chemical elements instead of separate atomic neural networks.[137] This approach, effectively identical to weight sharing between the hidden layers of the atomic neural networks, is especially useful if the number of training examples is limited or in applications where the time spent on neural network training is critical such as on the fly learning.

More recent developments typically employ deep neural network architectures where a construction of the invariant representation is part of the model and the corresponding parameters are adapted during training. Examples include the “deep tensor neural networks”,[138] and “SchNET”[139] models by the group of Tkatchenko and Müller, and the “deep potential molecular dynamics” model by Zhang et al.[140]

An alternative energy partitioning is given by the so-called many-body expansion, in which the total energy is calculated as a sum of $n$-body energy contributions,

$$E = \sum_i E^{1-\text{body}}(i) + \sum_{ij} E^{2-\text{body}}(i,j) + \sum_{ijk} E^{3-\text{body}}(i,j,k) + \dots \tag{2.64}$$

Since the contributions of higher order terms decrease rapidly, this expansion can typically be truncated at $n = 3$ or $n = 4$. An advantage of the approach is that permutation-invariant polynomials of internal coordinates can be used as input since the individual many-body terms are low-dimensional. Manzhos and Carrington presented one of the first neural network-based fits of the many-body contributions.[141] Similar to the idea of element-specific atomic neural networks, using a common $n$-body neural network for each unique combination of chemical element, as suggested by Malshe et al., can significantly reduce the number of fit parameters.[142] Lubbers et al. combined both energy partitioning approaches by modeling the atomic energy from Equation 2.54 using the many-body expansion from Equation 2.64.[143]

### 2.2.4. Kernel-based models

Several of the early examples of PES interpolation, such as the reproducing kernel Hilbert space (RKHS) method[144,145] and the modified Shepard method,[146,147] are closely related and in some cases even mathematically equivalent to kernel-based machine learning models. The use of Cartesian or internal coordinates as input does, however, limit these methods to low dimensional systems.

Analogously to the Behler-Parrinello atomic neural networks, the "Gaussian approximation potentials" (GAP) by Bartok et al.[148] partition the total energy in a sum of atomic energies. These atomic contributions are modeled using Gaussian process regression with a squared exponential kernel function in the space of SO(4) bispectrum descriptors. Latter applications of the method typically employed the so-called "Smooth overlap of atomic positions" (SOAP) kernel,[128] which can be evaluated as a dot-product kernel of SO(3) power spectrum descriptors. A detailed explanation of the method was given by Bartók and Csányi.[149]

The "spectral neighbor analysis potential" (SNAP) by Thompson et al. is closely related to GAP.[150] Instead of Gaussian process regression, a simple linear model for the atomic energy contributions in the space of SO(4) bispectrum descriptors is used, where the parameters are determined by weighted least-squares regression.

Chmiela et al. presented the "gradient-domain machine learning" (GDML) approach, a special kernel ridge regression model for atomic forces.[151] The resulting force predictions are conservative, i.e. curl free, by construction. Other machine learning potential energy models include this property by calculating forces as analytical derivatives of the total energy expression.

In a follow-up study the approach was extended to include permutational invariance resulting in the so-called symmetrized GDML (sGDML) method.[152] This is achieved by using a symmetrized kernel function,

$$k_{\mathrm{sym}}(\mathbf{x}, \mathbf{x}_i) = \sum_q^S k(\mathbf{x}, \mathbf{P}_q \mathbf{x}_i), \tag{2.65}$$

where $\mathbf{x}$ and $\mathbf{x}_i$ are molecular geometries and $\mathbf{P}_q$ is a permutation operator. While this approach is conceptually identical to the data augmentation presented in Section 2.2.1, the resulting increase in computational effort is kept low by including the additional training examples at the kernel level.

A similar approach of kernel-based learning of atomic forces was investigated in the group of De Vita. These investigations include studies of on the fly training for molecular dynamics,[153] symmetry invariant kernels,[154] and $n$-body kernels for many-body force fields.[155]

The "graph approximated energy" (GRAPE) suggested by Ferre et al. uses adjacency matrices as graph-based description of the local environments for atomic energy contributions.[156] A translation and rotation invariant kernel is derived from random walks on the local graphs.

## 2.3. Structure search

For a given method that can be used to evaluate the PES most problems in computational chemistry reduce to finding stationary points such as (local) minima and saddle points on the PES. While there are general purpose algorithms to locate stationary points on high dimensional surfaces, certain methods have been developed specifically for the application to molecular systems, significantly reducing the number of computationally expensive evaluations of the PES.

### 2.3.1. Local geometry optimization

A routine task in computational chemistry is the search for local minima corresponding to stable molecular geometries. What differentiates the optimization of molecular geometries from other problems with respect to optimization algorithms is that evaluating gradients of the PES requires a similar time as the evaluation of the energy. Typical optimization algorithms, however, assume that the evaluation of an $n$-dimensional gradient requires $n$ times the effort of a single function evaluation and are therefore trimmed to avoid frequent calculation of gradients.[157] An in-depth discussion of general optimization problems is given in Fletcher's textbook on "Practical Methods of Optimization"[158] which is the primary source for the standard optimization algorithms presented in this section.

All the presented optimization methods can be described using the following algorithm.

Starting from an initial guess $\mathbf{x}_0$ for the minimum energy structure:

1. Evaluate the PES at $\mathbf{x}_i$ to obtain the energy $E_i$, the gradient vector $\mathbf{g}_i$, and possibly the Hessian matrix $H_i$.
2. Check for convergence, typically by comparing the gradient to a threshold.
3. Fit an approximate model of the PES using the available information.
4. Calculate the displacement $\Delta\mathbf{x}$ by finding the minimum of the model PES.
5. Set $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$ and go to step 1.

The main difference of the various optimization algorithms is given by the choice of approximate model for the PES used in step 3.

#### Newton-Raphson

The standard Newton-Raphson method can be obtained by using a second order Taylor series expansion as model PES in step 3 of the optimization algorithm. Note that a first order approximation can not be used since a flat plane does not contain stationary points, rendering step 4 of the optimization algorithm impossible. Using the symbols defined in step 1 of the optimization algorithm, the second order Taylor series is given by

$$E = E_i + \Delta\mathbf{x}^\top \mathbf{g}_i + \frac{1}{2}\Delta\mathbf{x}^\top H_i \Delta\mathbf{x} + \dots \tag{2.66}$$

In this simple example the minimum of the model can be calculated analytically by setting $\frac{\partial E}{\partial \Delta \mathbf{x}} = \mathbf{0}$, which yields the so-called Newton-Raphson step,

$$\Delta \mathbf{x} = -H_i^{-1} \mathbf{g}_i. \tag{2.67}$$

Note that this algorithm is therefore only dependent on PES information from the geometry at step $i$ and immediately discards all knowledge from previous steps.

### Quasi Newton methods

Evaluation of the Hessian is computationally expensive. Therefore, in the so-called quasi Newton methods the Hessian is approximated using information from previous steps. The de-facto standard method for approximating the Hessian is the Broyded-Fletcher-Goldfarb-Shanno (BFGS) algorithm. Using the step size,

$$\Delta \mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i, \tag{2.68}$$

and the change in the gradient vector,

$$\Delta \mathbf{g}_i = \mathbf{g}_{i+1} - \mathbf{g}_i, \tag{2.69}$$

the update formula for BFGS is given by

$$H_{i+1}^{\mathrm{BFGS}} = H_i + \frac{\Delta \mathbf{g} \Delta \mathbf{g}^\top}{\Delta \mathbf{g}^\top \Delta \mathbf{x}} - \frac{H \Delta \mathbf{x} \Delta \mathbf{x}^\top H}{\Delta \mathbf{x}^\top H \Delta \mathbf{x}}. \tag{2.70}$$

In practice it is more efficient to update the inverse Hessian $G = H^{-1}$ needed in the Newton-Raphson step[i], which can be achieved using the formula

$$G_{i+1}^{\mathrm{BFGS}} = G_i + \left(1 + \frac{\Delta \mathbf{g}^\top G \Delta \mathbf{g}}{\Delta \mathbf{x}^\top \Delta \mathbf{g}}\right) \frac{\Delta \mathbf{x} \Delta \mathbf{x}^\top}{\Delta \mathbf{g}^\top \Delta \mathbf{x}} - \frac{\Delta \mathbf{g} \Delta \mathbf{x}^\top G + G \Delta \mathbf{x} \Delta \mathbf{g}^\top}{\Delta \mathbf{g}^\top \Delta \mathbf{x}}. \tag{2.71}$$

An initial guess $H_0$ or $G_0$ is needed for the first step. Typically, a unit matrix or problem-specific scaled diagonal matrices are used. For the application in molecular geometry optimizations a informed guess such as the Hessian from a force field calculation can be used to significantly accelerate convergence.[159–162]

Figure 2.6 shows the optimization of a diatomic molecule using a combination of the Newton-Raphson step and the BFGS update scheme. The PES for the molecule is given by a Morse potential,

$$E = D_\mathrm{e} \left(e^{-2\alpha(r - r_\mathrm{e})} - 2e^{-\alpha(r - r_\mathrm{e})}\right), \tag{2.72}$$

with parameters $D_\mathrm{e} = 5.0$, $\alpha = 1.0$, and $r_\mathrm{e} = 2.0$. The starting position is set to $r_0 = 3.0$ and the single entry in the Hessian is initialized to $H_0 = 2.0$. The oscillatory behavior depicted in Figure 2.6 can be eliminated in higher dimensional systems by using the Newton-Raphson formula to determine a search direction $\Delta \mathbf{x} = -\alpha H_i^{-1} \mathbf{g}_i$ and applying an iterative line search algorithm to determine a suitable step size $\alpha$.

[i]Note that this naming convention for $H$ and $G$ is reversed with respect to the notation used in the textbook by Fletcher.[158]

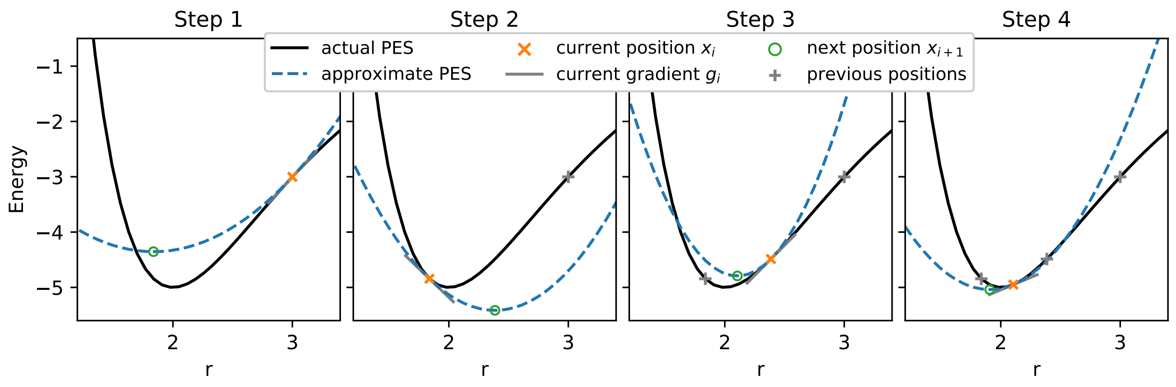


Figure 2.6.: Optimization of a dimer using the BFGS algorithm and the Newton-Raphson step.

### Machine learning-accelerated geometry optimization

Further improvement can be achieved by using all the available data to fit more complex models, such as the machine learning potentials presented in Section 2.2. For this specific application the machine learning models are not strictly required to include any invariances due to the fact that the training data has a constant atomic composition and the model is only applied to a limited region of the PES.

Figure 2.7 shows the performance of machine learning-accelerated geometry optimization for the dimer example using a 1D Gaussian process regression model combined with a standard Newton-Raphson BFGS optimization to locate the model PES minimum. A squared exponential covariance function with fixed length scale of $l = 1.69$ is used. This value is chosen such that the first step $\Delta\mathbf{x}$ is approximately the same as the first Newton-Raphson step in Figure 2.6. The main advantage illustrated by this example is that the machine learning model can make use of all available PES information and thereby avoid the oscillating behavior exhibited in Figure 2.6.

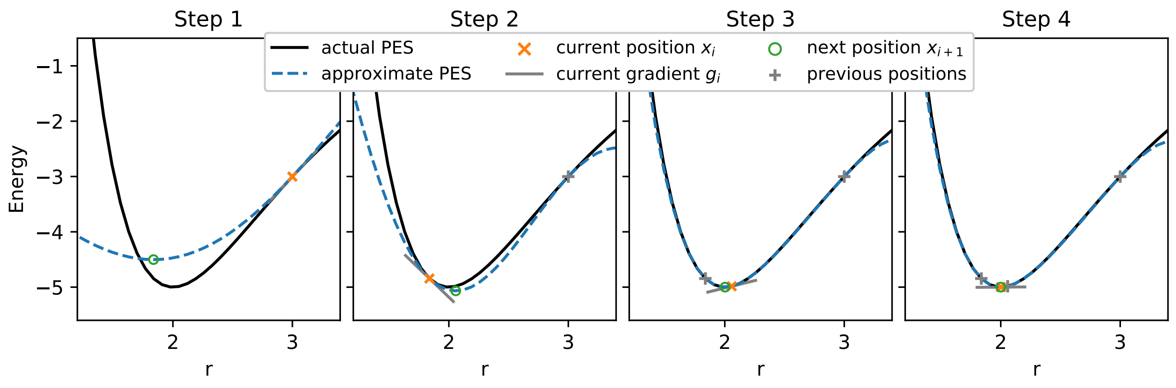


Figure 2.7.: Optimization of a dimer using Gaussian process regression to model the approximate PES.

This additional flexibility in the model PES does, however, come with several difficulties and computational overhead which has to be made up for by a significant reduction in the number of PES evaluations.

The most significant increase in computational effort is due to step 3, the fitting/training of machine learning PES approximations. Since the model has to be constructed on the fly during the optimization, the training procedure has to be executed fully automatically without manual tuning or inspection. Ideally, the model can be easily updated every iteration to incorporate the newly added data without starting from scratch. Most implementations therefore use Gaussian process regression which offers the additional advantage of a closed form solution, eliminating the dependence on randomly initialized weights typically used in neural networks.

Another source of additional computational overhead comes from the fact that there is no analytical method for finding the closest local minimum on the model PES in step 4 of the algorithm. Using an iterative search algorithm such as a quasi Newton method to locate this minimum is only feasible the evaluation of the machine learning approximation is orders of magnitude faster than evaluating the *ab initio* PES.

Denzel and Kästner[163] showed that using Gaussian process regression in Cartesian coordinates the reduction in *ab initio* PES evaluations can outweigh the additional computational overhead even for small molecules. Similar algorithms were suggested by Garijo del Río et al.,[164] and Schmitz and Christiansen.[165]

In Ref. 166, reprinted as Chapter 4 of this thesis, we showed that the GPR based optimization can be further improved by transforming the input from Cartesian coordinates to molecule-specific internal coordinate systems that are used by most state-of-the-art geometry optimization packages.

Recent advances in this field include the incorporation of higher order derivatives,[167] and the extension to anisotropic kernels.[168,169]

### 2.3.2. Transition state search

Other points of interest on the PES, especially for the investigation of reactions, are the so-called transition states. In mathematical terms these are first order saddle points, that is minima in all but one coordinate and characterized by a zero gradient and a single negative eigenvalue of the Hessian matrix. Based on this property of a negative eigenvalue, Cerjan and Miller[170] suggested an algorithm for locating transition states. After rewriting the Newton-Raphson step from Equation 2.67 in terms of eigenvalues $b_j$ and eigenvectors $\mathbf{V}_j$ of the Hessian matrix $H$,

$$\Delta\mathbf{x} = \sum_j \frac{\mathbf{V}_j \mathbf{V}_j^\top \mathbf{g}_i}{b_j}, \tag{2.73}$$

the eigenvalue spectrum corresponding to a first order saddle point can be enforced by shifting the eigenvalues,

$$\Delta\mathbf{x} = \sum_j \frac{\mathbf{V}_j \mathbf{V}_j^\top \mathbf{g}_i}{b_j - \lambda}. \tag{2.74}$$

Different methods for determining the shift parameter $\lambda$ were suggested by Cerjan and Miller[170] and Simons et al.[171]

A more general algorithm for locating stationary points was suggested by Banerjee et al.[172] In the so-called rational function optimization (RFO) method the Taylor-Series expansion of the Newton-Raphson step is replaced by a Padé approximation to order $[2, 2]$,

$$E = E_i + \frac{\Delta\mathbf{x}^\top \mathbf{g}_i + \frac{1}{2}\Delta\mathbf{x}^\top H_i \Delta\mathbf{x}}{1 + \Delta\mathbf{x}^\top S \Delta\mathbf{x}}, \tag{2.75}$$

where $S$ is a symmetric matrix of expansion coefficients, typically set to a diagonal (or unit) matrix. Note that the denominator contains no linear term in order to fulfill the conditions $\frac{\partial E}{\partial \Delta \mathbf{x}}\big|_{\Delta\mathbf{x}=\mathbf{0}} = \mathbf{g}$ and $\frac{\partial^2 E}{\partial \Delta \mathbf{x}^2}\Big|_{\Delta\mathbf{x}=\mathbf{0}} = H$. The step $\Delta\mathbf{x}$ is determined by finding the stationary points of the Padé approximation characterized by $\frac{\partial E}{\partial \Delta \mathbf{x}} = \mathbf{0}$. This yields an eigenvalue equation of dimension (n+1),

$$\begin{pmatrix} H_i & \mathbf{g}_i \\ \mathbf{g}_i^\top & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} S & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ 1 \end{pmatrix}. \tag{2.76}$$

The resulting eigenvectors correspond to steps towards different stationary points. A step in the direction of a minimum is obtained from the eigenvector for the lowest eigenvalue. Similarly, the eigenvector corresponding to the second lowest eigenvalue can be used to locate a first order saddle point. In order to increase the stability of transition state searches most optimization packages use a slight modification of the algorithm, the partitioned RFO.[173]

As shown by Denzel and Kästner[174] these single point transition state search algorithms can be accelerated using machine learning in exactly the same way as the search for minimum energy configurations. Similarly, Koistinen et al.[175] used Gaussian process regression in inverse distance coordinates to reduce the number of PES evaluations for the so-called dimer method.[176–179] A more sophisticated transition state algorithm based on the predicted variance of a Gaussian process regression model was suggested by Fernández et al.[180]

### Two-sided transition state search

A different class of transition state algorithms, often referred to as two-sided transition state search or string methods, is focused not only on locating the saddle point but also exploring the reaction path connecting two minima, commonly referred to as minimum energy path. Since these algorithms involve significantly more evaluations of the PES possible savings in computational effort due to the combination with machine learning methods are even higher.

In the nudged elastic band (NEB)[181–183] algorithm the path between two minima is discretized into a band of so-called images which are connected by fictitious springs. During the iterative relaxation procedure the images move according to two forces. The first force, corresponding to the negative PES gradient, is applied perpendicular to the band and results in a minimization of the potential energy for all images. The second force, due to the newly introduced springs, is applied tangential to the band and ensures an even distribution of images along the band. More detailed descriptions of the method and variants thereof are given in Section 5.3.1, Ref. 184, and Ref. 185.

Implementations of machine learning-accelerated nudged elastic band calculations based on atomic neural networks and Gaussian process regression in Cartesian coordinates were suggested by Peterson[186] and Koistinen et al.,[187] respectively. In Ref. 188, reprinted as Chapter 5 of this thesis, we compared the performance of these two previously suggested machine learning models and Gaussian approximation potentials for the task of accelerating NEB calculations on both a simple benchmark system and a catalytic reaction.

Recently suggested modifications for the Gaussian process regression-accelerated NEB method include different acquisition functions[189] and inverse distance coordinates.[190] Denzel et al.[191] introduced a Gaussian process regression-accelerated version of the minimum energy path method by Vaucher and Reiher.[192]

### 2.3.3. Global geometry optimization

Local minima can be used to divide the PES into so-called basins of attraction defined as the set of points from which gradient descent optimization leads to the same minimum. The number of these basins increases exponentially with the system size rendering exhaustive search for the global minimum impossible.[193] Nevertheless random search has been used successfully to locate experimentally confirmed global minima.[194] This can be attributed to chemical constraints on the search space and the structure of the PES. For example, the Bell-Evans-Polanyi[15,195–197] principle implies that low energy basins are separated from even lower lying basins by small energy barriers. This leads to a clustering of these low energy minima in regions commonly referred to as funnels.

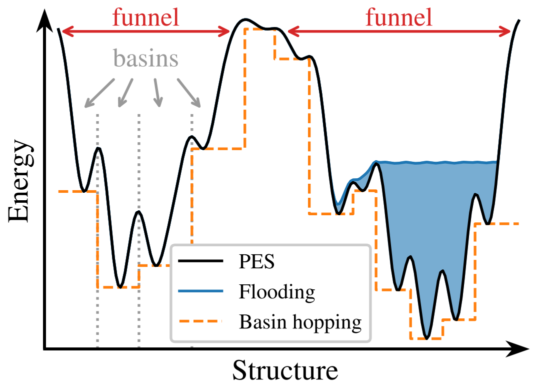


Figure 2.8.: Illustration of various global optimization methods. Inspired by similar plots in Ref. 198 and Ref. 199.

In general, global optimization algorithms consist of two parts. The first part is method to suggest a new atomic configuration, ideally corresponding to jumps from basin to basin (and funnel to funnel). Simple ideas include shaking, i.e. small random perturbation to the atomic positions, short molecular dynamics runs, and the exchange of atoms of different species.

More complex methods are typically based on distorting the PES and subsequent gradient descent or molecular dynamics steps. Stillinger et al. demonstrated that several of the local minima for 13 atom Lennard-Jones clusters can be eliminated by varying the exponent in the interaction.[200] A more general approach is to modify the PES by an additive bias potential. In metadynamics[201] and similar earlier suggestions, such as the flooding method by Grubmüller,[202] local minima are filled using a bias potential constructed as sum of Gaussians in a reduced space of coordinates typically referred to as collective variables. An opposite approach is taken by the basin hopping algorithm[203] which aims to remove all transition states between local minima by replacing the PES in a basin with the energy of its local minimum.

The second part is a method of sampling, i.e. choosing which of the suggested atomic configurations to accept, typically based on the Metropolis-Hasting method.[204] More sophisticated sampling methods include simulated annealing,[205] parallel tempering[206,207] and nested sampling.[208,209] The minima hopping algorithm[198] purposefully avoids a thermodynamic distribution by introducing a search history to discourage repeated sampling of the same basin.

A different class of global optimization algorithms, typically categorized as machine learning, are the biology inspired evolutionary algorithms. The most prominent example genetic algorithms are among the earliest applications of machine learning in computational chemistry.[210–215] Particle swarm optimization, a different variant of evolutionary algorithms, has also been suggested for global geometry optimization.[216,217]

Common to all these optimization methods is the large amount of PES evaluations needed to locate a global minimum, which restricts them to computationally inexpensive methods of calculating the potential energy. Therefore, combinations of the machine learning PES models presented in Section 2.2 with global optimization algorithms such as random structure search,[218,219] basin hopping,[220–225] and evolutionary algorithms[226–234] were suggested by several groups.

More recent examples leverage uncertainty predictions of the machine learning models for fully automated exploration of the PES based on active learning.[235–240]

Jørgensen et al.[241] introduced the "atomistic structure learning algorithm", a convolutional neural network trained using reinforcement learning to predict the global minimum of 2-dimensional structures. Suggested improvements to the method include the generalization to 3-dimensional systems and the combination with machine learning potential energy models.[242–244]

Machine learning-based dimensionality reduction techniques were also suggested as a data driven method of constructing collective variables for metadynamics[245–252] and the related variationally enhanced sampling method.[253–255]

Building on the concept of distorting the PES, Sørensen et al.[256–258] suggested an optimization step based on cluster analysis in an atomic feature space for use in basin hopping and evolutionary algorithms.

One of the most prominent challenges in the realm of global optimization is protein folding, i.e. locating the minimum energy geometry (tertiary structure) for a given sequence of amino acids (primary structure). The biannual critical assessment of structure prediction (CASP) experiment[259–261] showcases the most recent advances in this field. Since CASP12 in 2016 the competition is increasingly dominated by deep learning-based algorithms.[262–270]

This special class of global optimization algorithms typically aims to predict the contact points, that is regions where different parts of the protein chain are in close proximity. Figure 2.9 illustrates the algorithm presented in Ref. 270. First, a deep neural network is used to predict the distance matrix between specific carbon atoms of the individual residues (amino acids) using the given sequence of amino acids and features derived from comparison with a protein database as input. In a second step the final three-dimensional structure is constructed by optimization of a protein-specific potential derived from the machine learning prediction of the distance matrix.

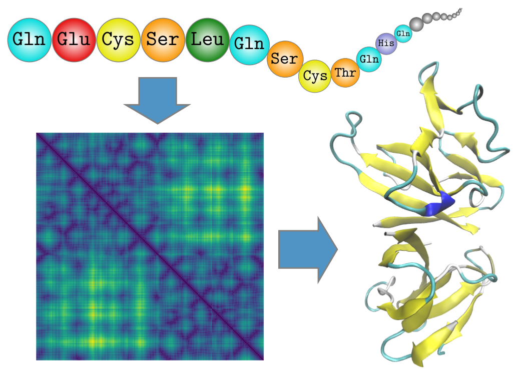


Figure 2.9.: Machine learning-based protein folding algorithm.

In CASP14 (2020) the "AlphaFold2" model, presented by the DeepMind group, achieved unprecedented accuracy with a root mean squared deviation of 1.6 Å.[271] This is comparable to the accuracy of experimentally obtained results.

# 3. Embedded atom model potentials for Pt-Ni nanoclusters improved by machine learning: a compromise between flexibility and physical meaning

This chapter corresponds to the manuscript
"Embedded atom model potentials for Pt-Ni nanoclusters improved by machine learning: a compromise between flexibility and physical meaning" by Ralf Meyer, Martin Schnedlitz, Riccardo Ferrando, and Andreas W. Hauser, currently in preparation for publication.

Contributions:

- Ralf Meyer ran the density functional theory calculations, implemented, trained, and evaluated the machine learning models, and wrote the first draft of the manuscript.
- Martin Schnedlitz generated the data set using the global optimization program.
- Riccardo Ferrando provided the global optimization program, and contributed to the final manuscript.
- Andreas W. Hauser supervised the method development, contributed to the manuscript, and provided the funding.

## 3.1. Abstract

A many-body potential for the Pt-Ni system, based on the second-moment approximation to tight binding, is systematically improved via machine learning for applications as a DFT energy predictor in the nanometer regime. In a series of modifications, the embedding function as well as the pair density functions are replaced by neural network expressions. We investigate the performance of these hybrid systems and compare them to the purely mathematically inspired Behler-Parrinello atomic neural network model, carving out the advantages and disadvantages of hard-wired physical knowledge versus maximum flexibility.

## 3.2. Introduction

Metallic nanoclusters are quantum objects of finite size, situated somewhere between few-atom systems and the bulk. For small aggregates, density functional theory (DFT) is the most suitable choice, while even smaller systems might still be accessible via more costly wave function-based methods. However, for larger systems, consisting of hundreds or thousands of atoms, practical evaluations of the highly complicated energy landscapes or potential energy surfaces (PES) are only feasible via approximations such as inter-atomic potentials or force fields.

A typical choice for mixed-metallic structures are many-body potentials derived from the embedded atom model (EAM).[272–274] As shown by Gupta,[275] the inclusion of a many-body term in addition to simple pair-wise interactions is necessary to accurately describe surface relaxation. In the EAM approach, the binding energy is calculated as a sum of $N$ atomic contributions $E = \sum_i^N E_i$. The latter, consisting of a pair interaction and a term describing the energy associated with embedding an atom in the local density of surrounding atoms, are given by

$$E_i = \frac{1}{2}\sum_{j\neq i}^{N} \phi^{\alpha\beta}(r_{ij}) + F^{\alpha}\left(\sum_{j\neq i}^{N} \rho^{\alpha\beta}(r_{ij})\right), \tag{3.1}$$

where $\phi^{\alpha\beta}(r_{ij})$ is a typically repulsive pair potential, $F^{\alpha}(\rho)$ is the embedding function, $\rho^{\alpha\beta}(r_{ij})$ is a pair-wise density function and the indices $\alpha$ and $\beta$ denote the element of the atoms $i$ and $j$, respectively. Mathematically equivalent models have been suggested by Smith and Banerjea[276] as well as Ercolessi, Tosatti, and Parrinello.[277] Prominent examples in the EAM family include the second-moment approximation to the tight-binding (SMATB),[275, 278–280] the Finnis-Sinclair,[281] and the Sutton-Chen[282] potentials. Note that this original energy expression was intended for nearly filled d-band transition metals and is only a function of pair-wise distances. The approach has also been extended to include angular dependence for the description of covalently bond materials in the so-called modified embedded atom model (MEAM).[283–285] A recent extension of the embedded atom model also takes polarization effects into account, aiming at the description of surface interactions between metal structures an polar, yet non-reactive species.[286]

In this article we revisit the idea of Hobday et al.[132] and introduce additional complexity to physically derived many-body models via machine learning methods. In a series of step-wise modifications we investigate the transition from physically motivated approaches for the Pt-Ni system, containing only a handful of parameters, towards highly flexible potentials based on neural networks featuring thousands of weight parameters. We study their performance with respect to the most challenging problem in this field: costly, hence often insufficient and inhomogeneous data sets for training.

Early applications of machine learning (ML) techniques to this problem were restricted to small, low dimensional systems and required prior knowledge for proper parametrization of the PES.[132,287–290] A breakthrough towards larger system sizes was achieved using purely mathematical models, in particular by means of neural network approaches.[4,124,135,291–293] Similar to the traditional techniques of above, these modern methods still attempt a modelling of the total energy via a summation over separable, atomic contributions, yet at much greater complexity. In recently suggested models based on deep learning the previously hand-crafted environment descriptors are incorporated into the neural network architecture and adapted during training.[138–140,294] However, this extremely high versatility comes at the cost of an overwhelming amount of fit parameters in the form of weights that need to be adjusted during network training, which translates into masses of data points required. An alternative approach is given by kernel-based models,[115,117,148,154] employing much less parameters, which can be interpreted physically,[155,295,296] and might even allow for a certain generalization across systems.

The Pt-Ni system treated in this article has been chosen for various reasons. First, it is a well-known system that has been studied theoretically by several groups,[297–300] with SMATB parameters available for pure Pt and Ni systems as well as for mixed systems.[280,301,302] Second, it is an interesting system with respect to structural features at the nanoscale: a simple comparison of surface and cohesive energies of both elements (Ni has a lower surface energy, Pt a higher cohesive energy) would suggest demixing tendencies in Pt-Ni particles, with Ni being pushed to the surface. However, the actual behavior is strongly dependent on cluster composition, size and temperature,[297] rendering this system as rather challenging to describe via many-body potentials. Third, the Pt-Ni alloy is known to be a very potent catalyst for the oxygen reduction reaction (ORR), a crucial chemical process in proton-exchange membrane fuel cells.[303–305] In fact, the Pt3Ni(111) single-crystal surface is providing one of the highest specific ORR activities known up to date.[306] A combination of this feature with the high surface-to-volume ratio of nanoclusters seems highly promising, but is handicapped by the poor stability of the currently synthesized octahedral nanoparticles.[307–309] Ternary alloying or surface doping of Pt-Ni nanoclusters has been suggested as a remedy.[303]

Our article is structured as follows. In Section 3.3 a state of the art EAM potential is used in combination with an global optimization algorithm to find the energetic minima of small Pt-Ni clusters of various compositions. In Section 3.4 the local minima identified in the optimization procedure are evaluated at DFT level. The results are then used as training and test sets throughout the remaining article. Section 3.5 describes the machine learning-based potentials and procedures used for training. In Section 3.6 the various potentials are investigated with regards to their accuracy and their ability to generalize. Our findings are summarized in Section 3.7.

## 3.3. Structure search

We start with the pre-selection of suitable Pt-Ni cluster geometries and metallic ratios for the sizes $N$ = 38, 55 and 147 atoms. In this size regime, surface energy effects are supposed to play a dominating effect, and strong deviations from bulk properties are to be expected. The basin hopping algorithm of Rossi and Ferrando[310] is used to search for global minima of Pt-Ni clusters in their various compositions. Energies and forces are evaluated using the SMATB potential, which features a negative square root embedding function without any atom-specific parameters,

$$F^{\alpha}(\rho) = -\sqrt{\rho}, \tag{3.2}$$

and combines it with the Born-Mayer[311] ion-ion repulsion, originally suggested as part of a potential for ionic crystals,

$$\phi^{\alpha\beta}(r_{ij}) = A^{\alpha\beta} e^{-p^{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}}-1\right)}. \tag{3.3}$$

The pair-wise density functions are given by

$$\rho^{\alpha\beta}(r_{ij}) = \left(\xi^{\alpha\beta} e^{-q^{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}}-1\right)}\right)^2. \tag{3.4}$$

The equilibrium distances are set to experimental values for the nearest neighbor distance in the bulk lattice, $r_0 = 2.77$ Å for Pt-Pt, $r_0 = 2.49$ Å for Ni-Ni, and the arithmetic mean $r_0 = 2.63$ Å for the mixed Pt-Ni interaction.[280, 312]

The computational demand is reduced by only evaluating the pair interactions up to a cutoff distance $b^{\alpha\beta}$. In order to avoid sudden jumps in the energy as atoms enter or leave the cutoff sphere the contributions are smoothly faded towards zero over an interval $[a^{\alpha\beta}, b^{\alpha\beta}]$ by interpolation using a fifth order polynomial. A detailed description of this scheme and all the parameters $a^{\alpha\beta}$ and $b^{\alpha\beta}$ are provided in Section B.2 of the supporting material. For the remaining parameters, we use the values given by Cheng et al.[302] as summarized in Table 3.1.

Table 3.1.: The Pt-Ni SMATB parameters from Ref. 302.

| $\alpha$ | $\beta$ | $A$ (eV) | $\xi$ (eV) | $p$ | $q$ |
|---|---|---|---|---|---|
| Pt | Pt | 0.1602 | 2.1855 | 13.00 | 3.13 |
| Pt | Ni | 0.1346 | 2.3338 | 14.838 | 3.036 |
| Ni | Ni | 0.0845 | 1.405 | 11.73 | 1.93 |

For each cluster size we investigate structures containing a single impurity of the opposite element, and mixtures of roughly 1:3 and 1:1 ratios. The optimization runs are started from a truncated octahedral structure for the $N = 38$ clusters and icosahedral geometries for $N = 55$ and $N = 147$, with random mixing of Pt and Ni atoms. Additionally, for each size a single optimization run with a mixture ratio of 1:1 is started from a so-called janus configuration, in which the clusters are clearly divided into one half consisting of Pt atoms and one half of Ni atoms.

In Figure 3.1 selected examples of the local minimum structures are presented. The $\mathrm{Pt_{20}Ni_{18}}$ cluster in panel a) shows a layered structure of Pt and Ni atoms while maintaining a truncated octahedral structure. Similar results are obtained for $\mathrm{Pt_{10}Ni_{28}}$ in panel b) where the Pt atoms only partially fill their respective layers. The $\mathrm{Pt_{28}Ni_{27}}$ structure presented in panel c) shows a continuous band of Ni atoms running along the surface layer of an icosahedral cluster. A minimal change in the composition to $\mathrm{Pt_{27}Ni_{28}}$ leads to a rearrangement towards fcc as depicted

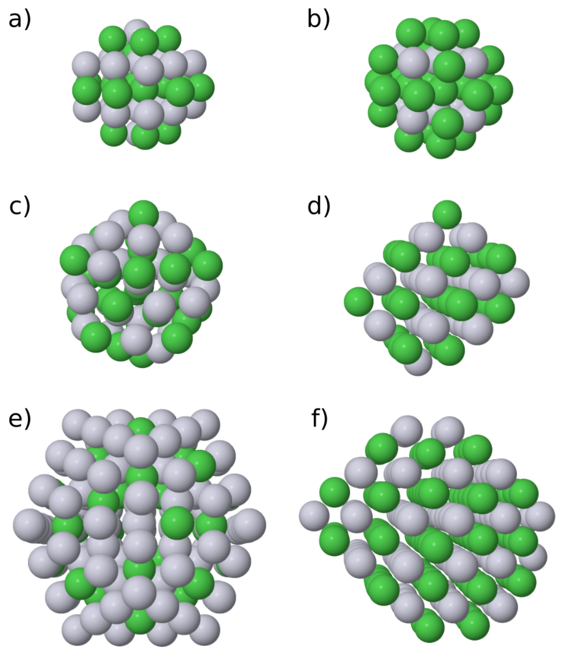


Figure 3.1.: Minimum energy configurations as predicted by SMATB for various cluster sizes and ratios of Pt (grey) to Ni (green) atoms located via a basin hopping algorithm.

in panel d). Similar results are obtained for the larger clusters containing 147 atoms. The configuration $Pt_{110}Ni_{37}$ in panel e) is favoring an icosahedral structure with the Ni atoms centered in the triangular faces of the outer most layer, and the configuration $Pt_{73}Ni_{74}$ in panel f) is favoring a layered fcc structure. Common to all global minima is a strong intermixing, i.e. the tendency to maximize the number of Pt-Ni interactions.

## 3.4. DFT data generation

In a next step, in order to validate the results of the global optimization algorithm discussed above, we use DFT to calculate the energies of the five best geometries for each composition, i.e. the lowest energy predictions of the SMATB approach. For the sake of a fair comparison, all DFT parameters are chosen to be consistent with the values used in Ref. 302 to fit the original SMATB parameters (see Table 3.1). All cluster geometries are evaluated as predicted by SMATB, i.e. without further local geometry optimizations, using the Quantum Espresso package[313,314] in a 45 x 45 x 45 $\text{Bohr}^3$ supercell. In spin unrestricted calculations we combine the PBE functional[65] and ultrasoft pseudopotentials[315,316] with a wavefunction cutoff of 40 Ryd and a density cutoff of 400 Ryd. The Brillouin zone is sampled at the Gamma-point and the smearing technique of Methfessel and Paxton[317] with a value of 0.002 Ryd is employed.

Figure 3.2 shows large discrepancies in the energy ordering predicted by DFT and SMATB. In fact, the prediction for the geometry of lowest energy is correct only for 7 of the 22 investigated systems, which suggests a systematic error in the SMATB potential. This ranking performance is only minimally better than random, given a number of just 5 geometries to choose from in each size and configuration regime. Note that the SMATB potential only predicts the

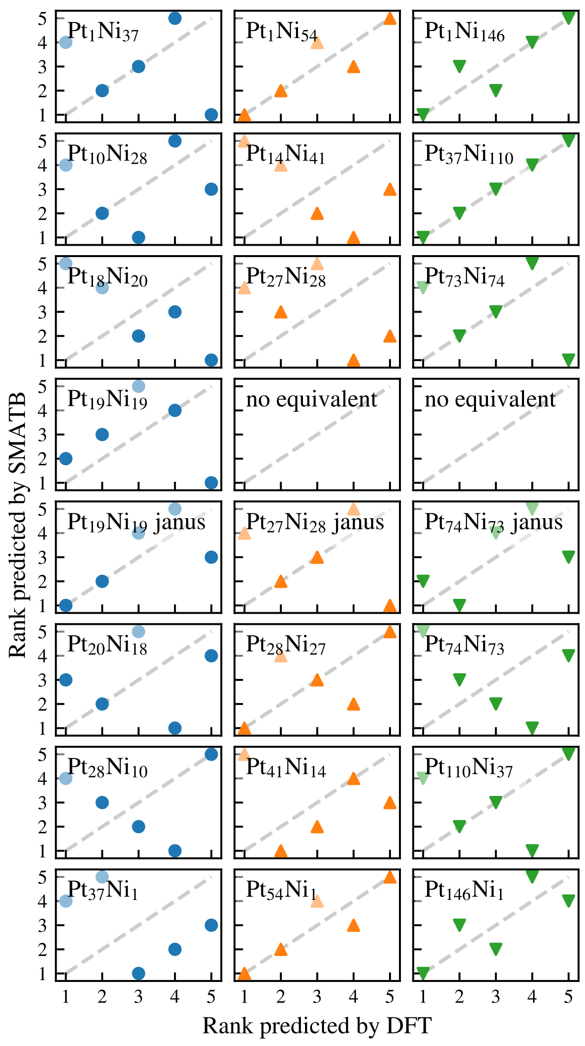


Figure 3.2.: Comparison of the energy ordering predicted by the SMATB potential and DFT for the five evaluated geometries for each composition. The plots are organized by ascending total number of atoms (columns) and ascending ratio of Pt atoms (rows).

bonding energy, whereas the DFT calculations provide an absolute electronic energy of all valence electrons as determined by the underlying atomic pseudo-potentials. Performing two additional DFT calculations on the isolated atoms, a “reference” bonding energy $E_{\mathrm{Ref}}$ can be obtained by subtracting the atomic energies from the total energy,

$$E_{\mathrm{Ref}} = E_{\mathrm{tot}}^{\mathrm{DFT}} - N_{\mathrm{Pt}} E_{\mathrm{Pt}}^{\mathrm{DFT}} - N_{\mathrm{Ni}} E_{\mathrm{Ni}}^{\mathrm{DFT}}. \tag{3.5}$$

In Figure 3.3 the energy difference between SMATB and DFT results is plotted as a function of the $N_{\mathrm{Pt}} : N$ ratio. The clear dependence of the prediction error on that ratio suggests that the SMATB potential is most effectively improved by a slight readjustment of the Pt-Ni interaction. The alternative, a possible imbalance in the relative strengths of the Pt-Pt and the Ni-Ni interaction, can not be quantified due to the lack of pure Ni or Pt cluster geometries in the data set. In addition to these 110 geometries used as test set we further evaluate all 1353

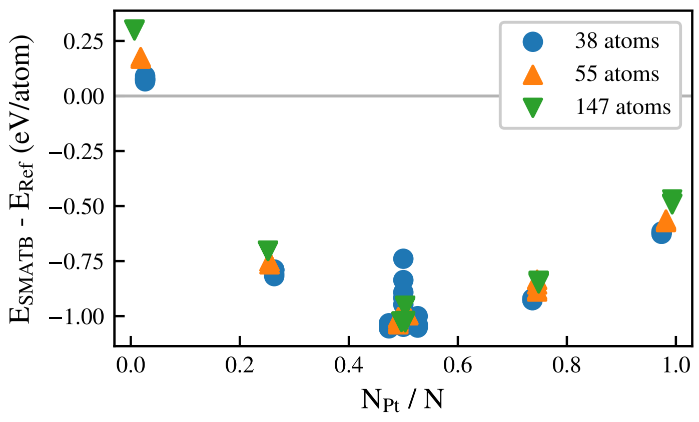


Figure 3.3.: Difference between the SMATB energy prediction and the DFT reference for the geometries of the test set as a function of the ratio of Pt atoms. The reference binding energy $E_{\mathrm{Ref}}$ is calculated from the DFT results using Equation 3.5.

intermediate best candidates which have been identified by the basin hopping algorithm. This second, extended set is split roughly 90:10 into a training set and a validation set consisting of 1217 and 136 geometries, respectively. All of the data sets include both the bonding energy calculated from Equation 3.5 and the atomic forces predicted by DFT, which substantially increases the number of individual data points that can be used to fit new potentials. A more detailed analysis of the data sets is provided in Section B.1 of the supporting material.

## 3.5. Synthesis of new potentials

In this section we discuss the stepwise improvement of the original SMATB model for the Pt-Ni system. Starting from a simple refit of the original SMATB parameters taken from Ref. 302, additional flexibility is gradually introduced by replacing some of the contributions by neural networks. Finally, a state-of-the-art neural network many-body potential as suggested by Behler and Parrinello is fitted as a lower threshold for the achievable accuracy.

In the interest of a fair comparison all models are trained in the same way. However, we note that especially those models relying only on a small number of parameters could be trained more efficiently with different methods. The model parameters are determined by minimization of the loss function

$$\mathcal{L} = \frac{1}{M}\sum_{k}^{M}\left(\frac{E_k^{\mathrm{model}} - E_k^{\mathrm{ref}}}{N_k}\right)^2 + \frac{\kappa}{M}\sum_{k}^{M}\frac{1}{3N_k}\sum_{l}^{3N_k}\left(F_{kl}^{\mathrm{model}} - F_{kl}^{\mathrm{ref}}\right)^2 + \lambda\sum_{k}^{N_w} w_k^2, \tag{3.6}$$

which consists of the mean squared error for the energy per atom as well as for all atomic force components. The weighting coefficient $\kappa$ for error contributions due to deviations in the forces is set to 1. For all of the neural network-based models a third term in the loss function, the so-called regularization term, is added in order to penalize network complexity, which can be quantified by the presence of larger weights. The regularization function is only applied to weights (not biases) in the hidden layers and controlled by the hyperparameter $\lambda$.

All of the presented models are implemented with the help of keras[318] and tensorflow[319] Python machine learning packages and evaluated in 32-bit floating point precision.

The loss function is minimized using the Adam optimizer[320] with a learning rate of $10^{-4}$ for a total of 2000 epochs with a batch size $M = 50$, resulting in 25 weight updates per epoch. During the optimization run the loss on the validation set is calculated after each epoch. The model weights achieving the lowest validation loss are used as final optimization result. Unless noted otherwise, all neural network weights $w$ are initialized randomly using the "Glorot uniform" tensorflow method[321,322] and the biases are set to zero.

All neural network-based models are trained with several architectures, i.e. number and size of hidden layers and different values for the regularization parameter $\lambda$. Only the models achieving the lowest error scores on the validation set are presented here; results for other models are presented in Section B.4 of the supporting material. However, the model performance appears to be mostly independent of our choice of hyperparameters. A small remaining variation can be attributed to the random initialization of weight parameters.

### 3.5.1. Refitting the SMATB parameters

A first improvement is obtained by simply refitting the 12 free parameters of the SMATB model to the training data, using the values in Table 3.1 as an initial guess. The standard procedure for fading long range interactions is slightly modified to allow for easier implementation in the neural network-based models. Instead of the interpolation using a fifth order polynomial, both $\phi^{\alpha\beta}$ and $\rho^{\alpha\beta}$ are multiplied by a polynomial cutoff function,

$$f_{\mathrm{cut}}^{\alpha\beta}(r) = \begin{cases} 1, & \text{for } r \leq a^{\alpha\beta} \\ 1 - 10\hat{r}^3 + 15\hat{r}^4 - 6\hat{r}^5, & \text{for } a^{\alpha\beta} < r < b^{\alpha\beta} \\ 0, & \text{for } r \geq b^{\alpha\beta} \end{cases} \tag{3.7}$$

with $\hat{r} = \frac{r - a^{\alpha\beta}}{b^{\alpha\beta} - a^{\alpha\beta}}$ as scaled distance. The range affected by the cutoff function is stretched by setting the inner cutoff $a^{\alpha\beta}$ to the second nearest neighbor distance and the outer cutoff to the fifth nearest neighbor distance, in order to avoid rapid changes in the atomic energy. Note that the exponential pair density function in Equation 3.4 is multiplied by the cutoff function before being squared to ensure a smooth second derivative. A detailed analysis of effects introduced by these modifications is given in Section B.2 of the supporting material.

The newly obtained SMATB parameters are listed in Table 3.2.

Table 3.2.: The refitted Pt-Ni SMATB parameters.

| $\alpha$ | $\beta$ | $A$ (eV) | $\xi$ (eV) | $p$ | $q$ |
|---|---|---|---|---|---|
| Pt | Pt | 0.24860 | 2.30001 | 10.11435 | 3.08657 |
| Pt | Ni | 0.22629 | 2.17392 | 10.15831 | 3.23356 |
| Ni | Ni | 0.14672 | 1.79192 | 10.22987 | 2.80600 |

### 3.5.2. Neural network-based EAM potential

In a second step we attempt to improve the SMATB model by a piece-wise introduction of neural network structures. We start with the replacement of the embedding function, then test for a replacement of the pair density functions, and combine both options eventually. All neural network functions employed in this process are one-dimensional, i.e. $\mathcal{R}^1 \to \mathcal{R}^1$. The resulting functions can be exported in tabulated form which allows for compatibility with many molecular dynamics packages. For two hidden layers such a neural network is given by

$$c(x) = w_3 \cdot (w_2 \cdot \varphi\,(w_1 \cdot x + \mathbf{b}_1) + \mathbf{b}_2) + b_3, \tag{3.8}$$

where the $w_k$ are weight matrices, the $\mathbf{b}_k$ are bias vectors and $\varphi$ is an activation function. Since the input and output to the neural network function are scalars, the first weight matrix $w_1$ and the last weight matrix $w_3$ reduce to vector quantities and the final bias vector $b_3$ reduces to a scalar. Note that the final layer is linear, i.e. no activation function is used. The hidden layers employ a hyperbolic tangent activation function. For all contributions that have not been replaced by neural network expressions, the parameters are initialized to the values of the refitted SMATB model as listed in Table 3.2.

### Neural network embedding function

A differentiation missing in the SMATB approximation is the usage of different embedding functions for the two elements Pt and Ni. Therefore, a first improvement is to distinguish with respect to index $\alpha$. In SMATB, the root dependence can be justified by the fact that the binding energies of transition metals are approximately proportional to the average width of the corresponding density of states, i.e. the square root of the second moment, with the latter being represented by sum over squares of overlap or "hopping" integrals.[280] We keep this dependence in our formulation, but provide additional flexibility by an element-dependent correction function of the pair density, $c_{\mathrm{F}}^{\alpha}(\rho)$, which appears as a multiplicative factor next to the SMATB ansatz:

$$F^{\alpha}(\rho) = -\sqrt{\rho}\; c_{\mathrm{F}}^{\alpha}(\rho). \tag{3.9}$$

This way, an asymptotically correct behavior for a vanishing density is ensured. The correction function $c_{\mathrm{F}}^{\alpha}(\rho)$ is modelled using a one-dimensional neural network. A reasonable starting guess is obtained by a specific initialization of the last (linear) layer: Its initial values for the weights are drawn from a random normal distribution with mean $\mu = 0.0$ and standard deviation $\sigma = 0.001$, and the bias is initialized to 1.0. This ensures that, initially, the modified embedding function is almost identical to the standard square root embedding function. Lowest values for the summed energy and force errors on the validation set are achieved using a regularization value of $\lambda = 10^{-5}$ and a neural network architecture consisting of two hidden layers containing 15 neurons each. Combined with the unmodified SMATB parameters, this yields a total of 584 free parameters.

### Neural network pair density functions

Modifications of the pair density functions must take the positivity constraint into account. In our expression, a neural network function $c_{\rho}^{\alpha\beta}$ is introduced, which takes the same normalized and shifted argument as apparent in the exponent of the original SMATB ansatz (see Equation 3.3). The squared product of this function with the cutoff function $f_{\mathrm{cut}}^{\alpha\beta}(r_{ij})$ as defined in Equation 3.7 yields a modified pair density function of the form

$$\rho^{\alpha\beta}(r_{ij}) = \left( c_{\rho}^{\alpha\beta} \left( \frac{r_{ij}}{r_0^{\alpha\beta}} - 1 \right) f_{\mathrm{cut}}^{\alpha\beta}(r_{ij}) \right)^2 . \tag{3.10}$$

In order to reduce the training effort, the neural network-based density functions are first trained separately to reproduce the exponential density function used in the SMATB model. The training examples are generated by evaluating the SMATB density functions at 101 equally distributed inter-atomic distance values between 0.0 and 6.2. Using the Adam algorithm, the neural network parameters are optimized until the summed mean squared error for all three interactions is below $10^{-5}$.

The lowest error score on the validation set is achieved for $\lambda = 10^{-6}$ and three hidden layers consisting of 15 neurons each, yielding already a total of 1584 adjustable parameters for this model.

#### Neural network density and embedding functions

In an obvious third modification of the original SMATB ansatz, both the embedding function as well as the pair density functions are now represented by neural networks as described above. All neural network weights are initialized using the schemes described in the corresponding sections. The lowest validation score is achieved with the hyperparameter $\lambda = 10^{-6}$ and three hidden layers containing 20 layers each. This leads to a total number of 4511 parameters in the joint NN model.

### 3.5.3. Behler-Parrinello atomic neural networks

For the sake of a more diverse evaluation of neural network capabilities when applied to the problem at hand we further employ neural networks in the style of Behler and Parrinello.[124] Contrary to the methods described above, these networks lack any physically motivated choice of structure or parameters, but are instead relying on their ability to – at least in principle – emulate any functional dependence to the desired accuracy if sufficient amounts of data can be provided. Yet, despite their purely mathematically inspired construction, Behler-Parrinello atomic neural networks are built upon the very same assumption that the total energy of a system can be represented by a sum of atomic contributions. Indeed, the energetic characterization of each atom by its local environment in a "feature space" is conceptually very close to the much older idea of describing its local "embedding" by some nonlinear function within the EAM picture. In a Behler-Parrinello network structure, for each atom $i$ of type $\alpha$ the distribution of neighboring atoms of type $\beta$ is encoded using so-called symmetry functions or descriptors. This way, any given environment can be translated into a vector in an abstract feature space and used as an input for a neural network. For the sake of a fair comparison with the SMATB ansatz, which neglects any angular dependence (at least in its standard formulation used here), only radial symmetry functions are employed in this model.

$$\mathbf{G}_i^{\alpha\beta} = \sum_{j\in\beta}^{N} e^{-\boldsymbol{\eta}^{\alpha\beta} r_{ij}^2} \cdot f_{\text{cut}}^{\alpha\beta}(r_{ij}). \tag{3.11}$$

The values for the seven entries in the $\boldsymbol{\eta}^{\alpha\beta}$ parameters are summarized in Table 3.3.

It is considered best practice to normalize the inputs for neural networks. Behler suggests restricting the variation in each descriptor entry to the interval $[-1, 1]$ by applying the transformation[136]

$$\tilde{\mathbf{G}}_i^{\alpha\beta} = \frac{2(\mathbf{G}_i^{\alpha\beta} - \mathbf{G}_{\text{min}}^{\alpha\beta})}{\mathbf{G}_{\text{max}}^{\alpha\beta} - \mathbf{G}_{\text{min}}^{\alpha\beta}} - 1, \tag{3.12}$$

Table 3.3.: Parameters for the descriptors

| $\alpha$ | $\beta$ | $a$ (Å) | $b$ (Å) | $\boldsymbol{\eta}$ (Å$^{-2}$) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Pt | Pt | 0.0 | 6.20 | 0.001 | 0.02 | 0.04 | 0.08 | 0.16 | 0.32 | 0.64 |
| Pt | Ni | 0.0 | 6.20 | 0.001 | 0.02 | 0.04 | 0.08 | 0.16 | 0.32 | 0.64 |
| Ni | Pt | 0.0 | 6.20 | 0.001 | 0.02 | 0.04 | 0.08 | 0.16 | 0.32 | 0.64 |
| Ni | Ni | 0.0 | 5.57 | 0.001 | 0.02 | 0.04 | 0.08 | 0.16 | 0.32 | 0.64 |

where $\mathbf{G}^{\alpha\beta}_{\mathrm{min}}$ and $\mathbf{G}^{\alpha\beta}_{\mathrm{max}}$ are calculated as the minimum/maximum values of all $\mathbf{G}^{\alpha\beta}_i$ in the training set for each vector entry individually. The values for these quantities are tabulated in Section B.3 of the supporting material.

The vectors encoding the Pt and the Ni environments for each atom $i$ are then concatenated $\mathbf{G}^{\alpha}_i = [\mathbf{G}^{\alpha\mathrm{Pt}}_i, \mathbf{G}^{\alpha\mathrm{Ni}}_i]$ and used as input for the atomic neural networks. This results in a mapping, $\mathcal{R}^D \to \mathcal{R}^1$, from the $D$-dimensional environment description to an atomic energy contribution. For a network architecture consisting of two hidden layers the atomic energy is then given by

$$E_i = w^{\alpha}_3 \cdot \varphi\left(w^{\alpha}_2 \cdot \varphi\left(w^{\alpha}_1 \cdot \mathbf{G}^{\alpha}_i + \mathbf{b}_1\right) + \mathbf{b}_2\right) + b_3, \tag{3.13}$$

with $w_k$ again denoting the weight matrices, the $\mathbf{b}_k$ as bias vectors and $\varphi$ as an activation function. In contrast to Equation 3.8, due to the vector input, only the final weight matrix $w_3$ and the final bias vector reduce to a vector and scalar, respectively. The bias weight of the linear last layer ($b_3$ in the example in Equation 3.13) is not allowed to vary freely during the training procedure. Instead, its value is chosen such that the atomic energy predicted for a single atom is exactly zero. Details on the necessity and the implementation of this scheme are given in Section B.3 of the supporting material.

The hyperparameter search yields the best validation score for $\lambda = 10^{-5}$ and atomic neural networks consisting of three hidden layers consisting of 20 neurons each. This neural network architecture results in a total of 2320 parameters.

## 3.6. Evaluation of the new potentials

### 3.6.1. Error scores

We start with the discussion of error scores for all methods on the test and the training/fitting sets. The root mean squared errors, summarized in Table 3.4, do confirm the intuitive expectation of reduced scores achieved with methods employing an increasing amount of parameters. However, what stands out is the actual "cost" of little additional accuracy as it is documented in the second half of the table, in comparison to the significant improvement of the original SMATB achieved by a refitting of its parameters to the new data set. This simple measure already reduces errors in the energy by an order of magnitude, a consequence of the physically meaningful foundation of the tight-binding model.

We further note that the original SMATB of Ref. 302 achieves similar force errors as the simple geometry independent models presented in Section B.1 of the supporting material. This finding is consistent with the fact that all investigated geometries are local minima predicted by these SMATB parameters and the forces are therefore close to zero.

The noticeably higher errors in the test set can be attributed to the fact that the latter contains a higher proportion of $N = 38$ clusters, for which the largest deviations are observed

Table 3.4.: Root mean squared errors for energy and forces on the training and test set achieved by the various potentials given in meV/atom and meV/Å.

| | Training set | | Test set | |
|---|---|---|---|---|
| Model | Energy | Forces | Energy | Forces |
| SMATB Ref. 302 | 844.6 | 336.6 | 830.2 | 336.2 |
| SMATB refitted | 71.3 | 202.0 | 105.7 | 208.3 |
| NN $F$ | 24.4 | 198.5 | 39.1 | 202.2 |
| NN $\rho$ | 53.7 | 177.0 | 84.0 | 174.9 |
| NN $F$ and $\rho$ | 21.1 | 169.7 | 35.6 | 167.2 |
| Behler-Parrinello | 10.2 | 132.1 | 18.1 | 122.8 |

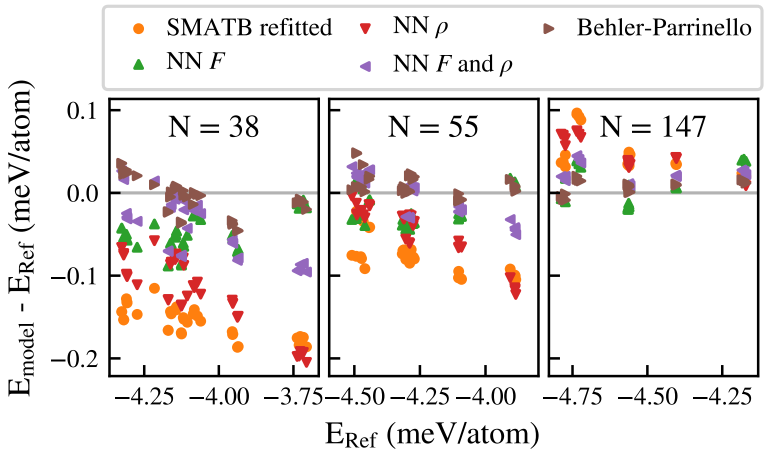


Figure 3.4.: Deviations of the energy predicted by the newly fitted models from the reference DFT energy for the geometries of the test set.

in general. This is seen best in Figure 3.4, where signed errors in the energies are plotted as a function of the DFT energies per atom. On one hand, the latter choice of $x$-axis allows for a direct judgement of the actual error size in terms of the DFT energy. On the other hand, it leads to a sorting of the various data points with respect to Pt-Ni cluster composition, since the Pt:Ni ratio enforces a rather specific average energy per atom with little variation due to cluster geometry (hence the column-like aggregations of data points). An increase in cluster size, displayed by three separate panels for $N = 38$, 55 and 147, causes a constant shift in DFT energies of about -0.3 meV per atom in each step, following the expected trend. Errors in the predictions become smaller with increasing cluster sizes. For the largest cluster size, $N = 147$, the sign of the error changes for all methods in our study, which we attribute to the models not fully capturing the dependence of the energy on the total number of particles. This may be caused by the increase in relative number of surface atoms or the stronger influence of angular dependent terms not covered by the EAM models. As a consequence, all models overestimate the binding energy of smaller clusters and underestimate the binding energy of the 147 atom cluster as this yields the lowest overall mean squared error per atom.

### 3.6.2. Simple PES scans

Energies of optimized structures are clearly the most important expectation values to be predicted, but offer little insight in actual weaknesses of a given model or method. Atomic forces, on the other hand, are very specific for a given cluster geometry and therefore hardly useful for a more general interpretation.

A more suitable approach is based on the analysis of averaged binding energies as a function of inter-atomic distances. We start with the potential energy curves for the three diatomic molecules $Pt_2$, PtNi, and $Ni_2$, and compare their shape as predicted by DFT and the newly fitted models in Figure 3.5. The admittedly challenging comparison reveals large deviations

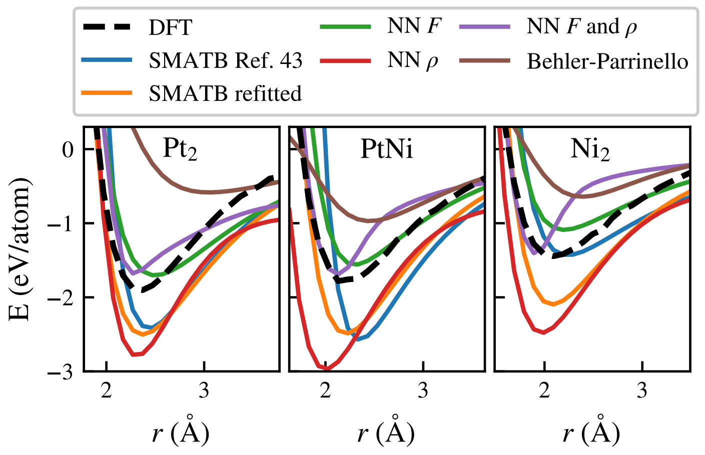


Figure 3.5.: The $Pt_2$, PtNi, and $Ni_2$ dimer potential energy curves as predicted by the various models, compared to results obtained using DFT.

from the DFT results for all model potentials. This failure follows from the lack of comparably small structures in the training set, and could have been compensated only by the addition either of data points or "hard-wired" physical knowledge on diatomics. However, even the unmodified SMATB does not reproduce the DFT curves correctly simply because it was not constructed for this purpose. Due to the lack of training data in this size regime we refrain from any deeper interpretations of this figure but move on to more representative bimetallic geometries instead.

Figure 3.6 shows the breathing mode of fully symmetrical icosahedral clusters consisting of 13 atoms. Due to a full shell of nearest neighbors for the central atom, this benchmark structure is significantly closer to the geometries found in the training set. While the binding energy and the energy minimum predicted by the models match the DFT results much more closely in this case, a significant deviation in the long range behavior is noticeable in of some of the machine learning models. However, an optically much more prominent deviation can be observed for the predictions of the Behler-Parrinello potential at short distances. Again, this can be attributed to the lack of training examples in this region, a fact which becomes most obvious for the model featuring the most opulent parametrization. A simple fix would be to include the pair repulsion term employed in all other models and use neural networks only to describe the attractive embedding energy.

Finally, in Figure 3.7 the extrapolation towards bulk properties is tested by plotting the cohesive energy as a function of the lattice constant $a$ for both pure Pt and pure Ni structures.

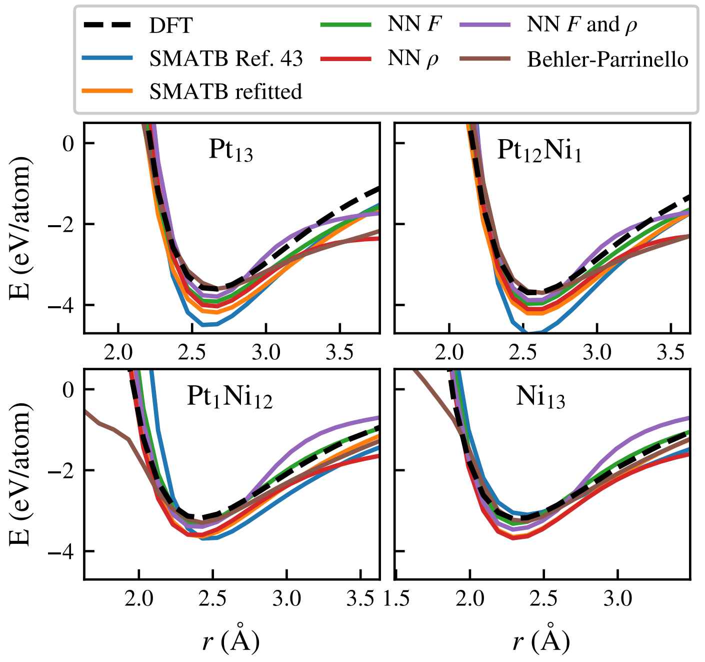


Figure 3.6.: Breathing mode of 13 atom icosahedral clusters. The top right and bottom left panels correspond to single Ni and Pt atoms surrounded by a 12 atom shell of Pt and Ni atoms, respectively.

The DFT reference is calculated using a $20 \times 20 \times 20$ Monkhorst-Pack k-point grid.[323] As can be seen in the figure, all but the original SMATB of Ref.[302] (which was fitted to exactly reproduce the experimental values) agree very well with the DFT curves obtained for bulk fcc geometries. Again, the lack of training data at shorter inter-atomic distances leads to a pronounced deviation of the Behler-Parinello potential, while all other models are also able to extrapolate correctly. Note that the DFT predictions for the cohesive energy of 5.52 eV for Pt and 4.87 eV for Ni deviate from the experimental values of 5.84 eV and 4.44 eV for Pt and Ni, respectively.[312] In other words, any of the DFT-trained models is equally suitable for bulk since relative deviations in the energy per particle are below the systematic error of the actual density functional at the chosen plane-wave energy cutoffs.

### 3.6.3. Analysis of individual contributions

Despite the underlying complexity in case of the neural network models, the pair density functions as well as the embedding functions for Pt and Ni are simple, one-dimensional functions. The former are functions of the interatomic distance between atoms $i$ and $j$, the latter are functions of the density contribution of all atoms in the environment of atom $i$. These functions can be plotted and inspected. We start with the analysis of the pair density functions $\rho^{\alpha\beta}$ in Figure 3.8.

It shows $\rho^{\mathrm{PtPt}}$, $\rho^{\mathrm{PtNi}}$, and $\rho^{\mathrm{NiNi}}$ as used by each model, plotted as a function of the inter-atomic distance, together with a histogram depicting the distribution of distances $r_{ij}$ in the data set. The histogram of NiNi interactions in the bottom panel shows a significantly lower number of short range and a higher number of long range interactions. This can be explained

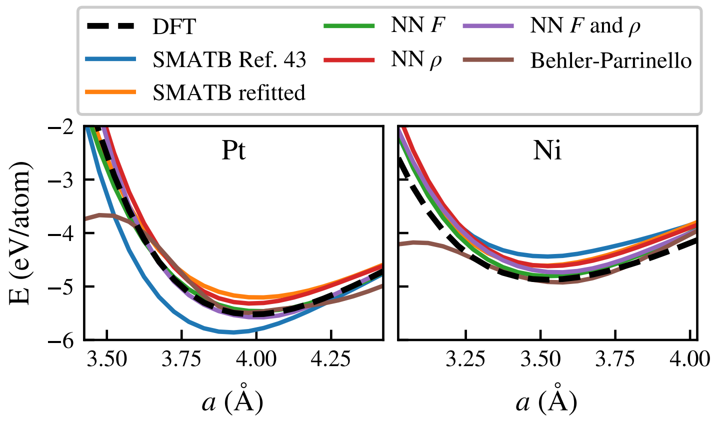


Figure 3.7.: Scan of the energy of pure Pt and Ni fcc bulk structures as a function of the lattice constant $a$.

by the lower binding energy predicted for NiNi interactions which are therefore disfavored in the global minimum search. Interestingly, both neural network-based models suggest a stronger long range contribution of the pair density functions than provided by the SMATB model. An obvious *ad hoc* correction would be switching to a modified density function $\tilde{\rho}^{\alpha\beta}$ built from two exponential functions,

$$\tilde{\rho}^{\alpha\beta}(r_{ij}) = \left(\xi_1^{\alpha\beta} e^{-q_1^{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}}-1\right)}\right)^2 + \left(\xi_2^{\alpha\beta} e^{-q_2^{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}}-1\right)}\right)^2 . \tag{3.14}$$

This measure, here conceived directly by looking at the machine learning-suggestions of Figure 3.8, even has physical meaning to it: It can be interpreted as an ansatz employing different pair density functions for $s$ and $d$ orbitals, as it had been suggested by Foiles et al. more than 30 years ago.[274]

A similar analysis can be done for the embedding functions and is presented in Figure 3.9. Here we plot the neural network correction $c_{\mathrm{F}}^{\alpha}$ to the square root embedding as a function of the summed pair density $\rho$. However, note that interpretations of the embedding functions are less straightforward due to their obvious dependence on the actual choice of pair density functions. This becomes evident already when looking at shape of both neural network-based corrections, which deviate much more than what is observed for the pair density functions. Again, the region for $\rho < 20$ should be excluded from any interpretations. Due to the lack of training data in this region, indicated by the combined histogram of summed density values, the functional behavior at short range is fully irrelevant for the fit. In the regime $20 < \rho < 40$ a damping of the embedding function for small density values is common to all corrections suggested by neural networks. This finding is consistent with Figure 3.4, indicating that the binding energy for smaller clusters is overestimated by the simple square root embedding function of SMATB. Again, an analytical model for the correction function in the region of the training points can be constructed from an extremely reduced neural network expression,

$$F^{\alpha}(\rho) = -\sqrt{\rho}\,(c_0^{\alpha} + c_1^{\alpha}\tanh(c_2^{\alpha}\rho))\,. \tag{3.15}$$

This correction function is, however, purely inspired by the neural network approach and offers no obvious physical interpretation. Note that the coefficients $c^{\alpha}$ can not simply be integrated

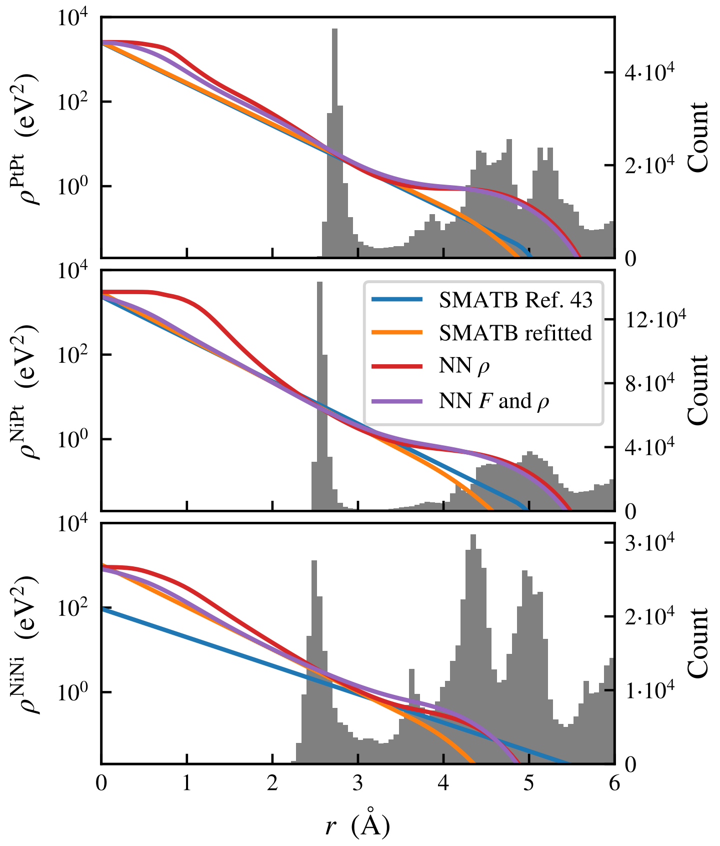


Figure 3.8.: The pair density functions as a function of the distance (left axis) and a histogram of the inter-atomic distances for all the geometries in the training data (right axis).

into a rescaling of the pair density functions $\rho^{\alpha\beta}$ (more specifically the parameter $\xi$ in the SMATB potential) due to the assumption that these functions are symmetric, i.e. $\rho^{\alpha\beta} = \rho^{\beta\alpha}$. Therefore, the rescaled square root dependence could also be interpreted as an indicator that this restriction should be lifted, and different parameters for $\rho^{\alpha\beta}$ and $\rho^{\beta\alpha}$ should be fitted instead.

## 3.6.4. Extended SMATB

Building on the insights from Section 3.6.3 we fit three extended SMATB models starting from the parameters given in Table 3.2. The goal here is to achieve a similar improvement over the standard SMATB model as observed for the neural network-based models, but using a minimal amount of additional parameters. The first model uses the embedding function in Equation 3.15, where the parameters are initialized to $c_0^\alpha = 0.5$, $c_1^\alpha = 0.5$, and $c_2^\alpha = 0.05$. The second model uses the pair density function in Equation 3.14, where $\xi_1^{\alpha\beta}$ and $q_1^{\alpha\beta}$ are initialized to the values given in Table 3.2 and the newly added parameters are initialized to $\xi_2^{\alpha\beta} = 0.5$ and $q_2^{\alpha\beta} = 0.1$. Finally, the third model again combines both the extended embedding function and the extended pair density function using the same values to initialize the new parameters.

The results of training these extended SMATB models is summarized in Table 3.5. All three investigated extended SMATB models achieve comparable error scores to their respective neural

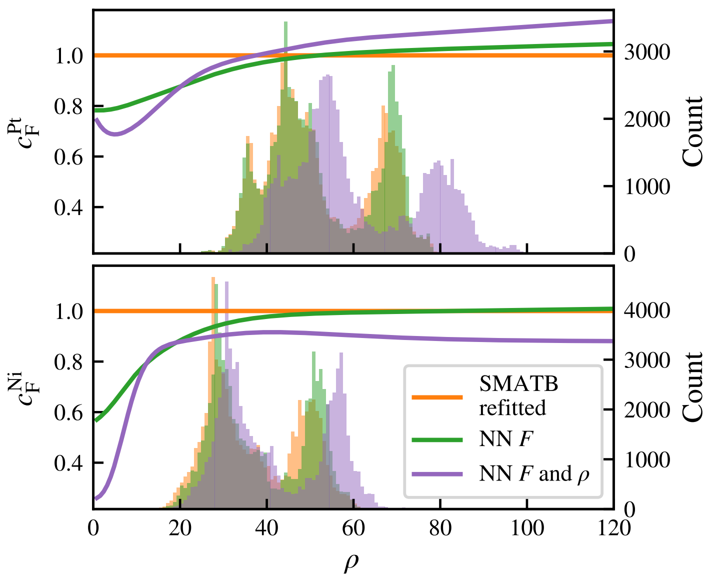


Figure 3.9.: The neural network embedding rescaling function $c_F^\alpha$ from Equation 3.9 as a function of the density (left axis) and a histogram of the summed pair density values for the training set as predicted by the investigated models (right axis).

network-based counterparts.

Table 3.5.: Root mean squared errors for energy and forces on the training and test set achieved by the extended SMATB models given in meV/atom and meV/Å.

| | Training set | | Test set | |
|---|---|---|---|---|
| Model | Energy | Forces | Energy | Forces |
| extended $F$ | 22.8 | 198.7 | 37.3 | 201.7 |
| extended $\rho$ | 56.0 | 176.7 | 84.5 | 172.9 |
| extended $F$ and $\rho$ | 17.4 | 174.0 | 27.8 | 169.5 |

A table containing the full set of optimized parameters for the three extended models is given in Section B.5 of the supporting material. It is worth noting that the optimization of both the extended pair density model and the extended embedding and pair density model yields negative $q_2^{\alpha\beta}$ parameters for all three interactions. This would lead to the unphysical prediction of increasing binding energy with increasing inter-atomic distance if multiplication with a cutoff function was skipped. We suspect that a reliable and physically meaningful fit of the second exponential term could be achieved after increasing the cutoff distance $b^{\alpha\beta}$.

## 3.7. Conclusion

An SMATB-based many-body potential for Pt-Ni systems at the nanoscale has been systematically improved by machine learning. We have shown that introducing additional complexity to established potential energy expressions via neural networks allows to derive physically meaningful extensions. Numerical correction functions produced by the networks can be "translated" and expressed by simple, appropriately parametrized analytical functions due to their slowly varying character. The obtained models, employing just a few additional parameters, yield significant improvements in the error scores. However, they do not quite reach the accuracy of purely mathematical models such as the Behler-Parrinello approach, if the overwhelming hunger of the latter for training data can somehow be satisfied.

For future work, an extended combination of machine learning and physical models for predictions of system properties such as melting temperatures, mixing behavior and phase diagrams might allow for further improvements and generalizations of atomic models. Such an endeavor would also necessitate the costly generation of suitable data sets containing geometries at higher temperatures, e.g. through molecular dynamics simulations.

Finally, symbolic regression might offer a way to automate the process of inspecting the correction functions provided by neural networks, translating them into simple analytical expressions, and fitting these new expressions to the training data.

## 3.8. Acknowledgments

Financial support by the Austrian Science Fund (FWF) under Grant P29893 is gratefully acknowledged. We further thank the IT Services (ZID) of the Graz University of Technology and the Vienna Scientific Cluster (VSC) for providing high performance computing resources and technical support.

# 4. Geometry optimization using Gaussian process regression in internal coordinate systems

This chapter corresponds to the publication
"Geometry optimization using Gaussian process regression in internal coordinate systems" by Ralf Meyer, and Andreas W. Hauser, in *The Journal of Chemical Physics*, **152** (8), 084112 (2020).

Contributions:

- Ralf Meyer implemented and tested the algorithm, ran the presented benchmark calculations, and wrote the first draft of the manuscript.
- Andreas W. Hauser supervised the method development, contributed to the manuscript, and provided the funding.

## 4.1. Abstract

Locating the minimum energy structure of molecules, typically referred to as geometry optimization, is one of the first steps of any computational chemistry calculation. Earlier research was mostly dedicated to finding convenient sets of molecule-specific coordinates for a suitable representation of the potential energy surface, where a faster convergence toward the minimum structure can be achieved. More recent approaches, on the other hand, are based on various machine learning techniques and seem to revert to Cartesian coordinates instead for practical reasons. We show that the combination of Gaussian process regression with those coordinate systems employed by state-of-the-art geometry optimizers can significantly improve the performance of this powerful machine learning technique. This is demonstrated on a benchmark set of 30 small covalently bonded molecules.

## 4.2. Introduction

The localization of extrema on the potential energy surface (PES) of a molecular system is one of the most fundamental tasks of computational chemistry. Due to the high computational effort which is necessary to obtain "chemically" accurate ($\delta$E $<$ 1 kcal/mol) estimates of electronic energy differences, the search for algorithms that use the least amount of evaluations possible to locate minima or transition states on a molecular PES has been a long standing research topic.

The even higher computational cost of evaluating Hessians limits these undertakings to gradient-based optimization algorithms in most cases. Quasi-Newton optimizers, which approximate the (inverse) Hessian matrix via gradient information taken from previous steps, are the method of choice for these tasks. Research intentions in this area include the identification of suitable coordinate systems,[324–336] the development of better initial guesses for the Hessian,[159–162] and a proper treatment of weakly coupled systems.[337–340] Even though quasi-Newton methods are forming the basis of most optimization algorithms, there exist several alternative approaches which have been designed for the specific purpose of molecular geometry optimization. Some of the most common techniques are GDIIS,[341] QUICCA,[342] FIRE,[343] and Quick-Min.[183,344]

In the more recent literature, a strong trend toward machine learning concepts can be noted. Current research in this field is dedicated to the accelerated localization of minima[163,164] and transition states[174,186–188,190,191,345] as well as to the improvement of local PES scans for better accuracy of reaction rates within instanton rate theory.[346–348] These methods employ the strategy of fitting a surrogate energy surface to those points already evaluated and provide suggestions for new geometries via a computationally much less demanding structural search on that surrogate surface. Typically, the latter is continuously updated with new information since the actual single point evaluation is performed on the real PES and then added to the dataset. So far, most of the machine learning potentials employed in these algorithms are formulated in Cartesian coordinates even though the most successful machine learning potentials and geometry optimization algorithms use alternative coordinate systems.

In this work, we aim to bridge the gap between "classic" state-of-the-art optimization algorithms employing thoughtfully designed internal coordinate systems and "novel" approaches based on machine learning. Our method of choice is Gaussian process regression (GPR), formulated in various non-Cartesian coordinate systems. We test the performance of selected algorithms for the task of geometry optimization and discuss their advantages and disadvantages in comparison to conventional approaches.

## 4.3. Methods

### 4.3.1. Geometry optimization

The main objective of this article is to investigate the influence of various coordinate systems on optimization performance. Our implementation of machine learning-accelerated geometry optimization is a combination of the ideas presented in Refs. 163 and 164. In order to provide a fair and unbiased comparison of this core feature, we restrain from any modifications or advancements suggested in the literature, which may reduce the computational overhead or accelerate the convergence of a specific approach. The remaining universal algorithm comprises the following steps:

- Step 0: Given an initial geometry $\mathbf{x}_0$, construct the coordinate transformation and initialize $\mathbf{x}_{\text{min}} = \mathbf{x}_0$ and $E_{\text{min}} = \infty$.
- Step 1: Evaluate energy $E_i$ and forces $\mathbf{F}_i$ at the current geometry $\mathbf{x}_i$. If $E_i < E_{\text{min}}$ set $\mathbf{x}_{\text{min}} = \mathbf{x}_i$ and $E_{\text{min}} = E_i$.
- Step 2: Check for convergence.
- Step 3: Add geometry $\mathbf{x}_i$, energy $E_i$ and forces $\mathbf{F}_i$ to the training set.
- Step 4: Fit the machine learning model and optimize the hyperparameters.
- Step 5: Perform a geometry optimization using the machine learning potential starting from $\mathbf{x}_{\text{min}}$. Stop if the maximum predicted force component is smaller than $0.5 f_{\text{max}}$ (where $f_{\text{max}}$ is the overall convergence threshold on force components) or the distance to $\mathbf{x}_{\text{min}}$ exceeds the trust radius $r_{\text{max}}$.
- Step 6: Use the last geometry of the machine learning optimization trajectory as a new guess for the minimum $\mathbf{x}_{i+1}$ and go to step 1. If the machine learning optimization has stopped because $r_{\text{max}}$ was exceeded, use the second to last geometry as new guess $\mathbf{x}_{i+1}$ instead.

While simple in concept, certain details of this algorithm prove crucial not only for improving the overall performance but even for achieving actual convergence in the first place. Probably, the most important detail is the necessity of small step sizes on the machine learning PES in step 5 while monitoring the distance from the starting geometry $\mathbf{x}_{\text{min}}$ in order to enforce a maximum step length $r_{\text{max}}$. This measure, which is slightly more complicated than the standard remedy of a simple down-scaling of large predicted steps, is necessary because the optimization trajectory on the machine learning PES is not a straight line in general. A similar idea was presented by Koistinen et al.[187] for their machine learning-accelerated nudged elastic band algorithm.

The starting point for the optimization on the machine learning PES is always the lowest energy geometry $\mathbf{x}_{\text{min}}$ in order to avoid unphysical "escapes" from a local minimum area on the real PES, which is not correctly represented on the yet insufficiently informed PES constructed by the machine learning model. The optimization in step 5 is carried out using the FIRE[343] method, which has been chosen due to its efficiency and robustness. It is based on a molecular dynamics engine with modified velocity calculations and adaptive time steps. In particular, in the first few steps, when the machine learning representation of the energy surface is still crude, quasi-Newton optimization algorithms tend to suggest very large displacements or might not converge at all.

The parameters of the optimizer are set to the default values of the Atomic Simulation Environment (ASE)[349] default values ($\Delta t = 0.1$ Å$\sqrt{\text{u/eV}}$ (approximately 1 fs), $\Delta t_{\text{max}} = 10\Delta t$, $\alpha_{\text{start}} = 0.1$, $f_\alpha = 0.99$, $f_{\text{inc}} = 1.1$, $f_{\text{dec}} = 0.5$ and $N_{\text{min}} = 5$). The maximum step size on the machine learning PES, however, is reduced to 0.02 Å. The global maximum step size is set to 0.2 Å. All calculations are restricted to a maximum of 150 optimization cycles. An optimization run is considered converged if a maximum gradient component of less than $f_{\text{max}} = 0.0003$ hartree/bohr and either an energy change of less than $10^{-6}$ hartree or a maximum displacement of less than 0.0003 bohr are reached. These convergence criteria were used by most previous studies on the investigated benchmark set.

We would like to emphasize that all geometry optimization steps, e.g., coordinate displacements or any measurements of step lengths, are performed in Cartesian coordinates and that it is only the machine learning model that is trained on the PES in a transformed coordinate system. This choice guarantees a fair comparison to reference calculations in Cartesian coordinates. The latter are done with the Broyden-Fletcher-Goldfarb-Shanno (BFGS)[158,350–353] method, including a line-search algorithm as implemented in the ASE program package,[349] with the same global maximum step size of 0.2 Å as used in all machine learning methods.

### 4.3.2. Gaussian process regression

This section is mostly dedicated to the necessary modifications to GPR due to the transformation of the input coordinates and our choice of certain hyperparameters of the model. We refer the interested reader to Ref. 104 for a detailed introduction to the GPR method and to Refs. 163 and 164 for the inclusion of derivative information. In summary, given a set of geometries $\{\mathbf{x}^{(1)} \dots \mathbf{x}^{(M)}\}$, corresponding energies $\{E^{(1)} \dots E^{(M)}\}$, and gradients $\{\mathbf{g}^{(1)} \dots \mathbf{g}^{(M)}\}$, the formulas needed for energy and gradient prediction of a previously unseen geometry $\mathbf{x}^\star$ are given by

$$E(\mathbf{x}^\star) = \sum_n \alpha^{(n)} k(\mathbf{x}^\star, \mathbf{x}^{(n)}) + \sum_n \sum_i \beta_i^{(n)} \frac{\partial k(\mathbf{x}^\star, \mathbf{x}^{(n)})}{\partial x_i^{(n)}} + E_{\text{mean}}(\mathbf{x}^\star), \tag{4.1}$$

and

$$\frac{\partial E(\mathbf{x}^\star)}{\partial x_k^\star} = \sum_n \alpha^{(n)} \frac{\partial k(\mathbf{x}^\star, \mathbf{x}^{(n)})}{\partial x_k^\star} + \sum_n \sum_i \beta_i^{(n)} \frac{\partial^2 k(\mathbf{x}^\star, \mathbf{x}^{(n)})}{\partial x_k^\star \partial x_i^{(n)}} + \frac{\partial E_{\text{mean}}(\mathbf{x}^\star)}{\partial x_k^\star}, \tag{4.2}$$

where the kernel $k(\mathbf{x}, \mathbf{x}')$ is a measure of similarity between different geometries. Using a constant mean model $E_{\text{mean}}(\mathbf{x}) = E_{\text{mean}}$, the parameters $\alpha^{(n)}$ and $\boldsymbol{\beta}^{(n)}$ can be determined by solving the following linear equation:

$$\begin{bmatrix} \alpha^{(1)} \\ \vdots \\ \alpha^{(M)} \\ \boldsymbol{\beta}^{(1)} \\ \vdots \\ \boldsymbol{\beta}^{(M)} \end{bmatrix} = \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \begin{bmatrix} E^{(1)} - E_{\text{mean}} \\ \vdots \\ E^{(M)} - E_{\text{mean}} \\ \mathbf{g}^{(1)} \\ \vdots \\ \mathbf{g}^{(N)} \end{bmatrix}, \tag{4.3}$$

where $\sigma_n^2$ is an assumed Gaussian noise on the training data, which increases numerical stability, and K is the extended kernel matrix given by

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}^{(m)}, \mathbf{x}^{(n)}) & \frac{\partial k(\mathbf{x}^{(m)}, \mathbf{x}^{(n)})}{\partial \mathbf{x}^{(n)}} \\ \frac{\partial k(\mathbf{x}^{(m)}, \mathbf{x}^{(n)})}{\partial \mathbf{x}^{(m)}} & \frac{\partial^2 k(\mathbf{x}^{(m)}, \mathbf{x}^{(n)})}{\partial \mathbf{x}^{(m)} \partial \mathbf{x}^{(n)}} \end{bmatrix}. \tag{4.4}$$

Due to the inclusion of derivative information, GPR cannot simply be applied to transformed input data $\mathbf{q}(\mathbf{x})$. This problem can be solved in different ways. The probably most intuitive solution is to first transform both the coordinates and the gradients into the non-Cartesian coordinate system and train the GPR using the transformed training examples. Predictions, more precisely the gradient predictions, are then transformed back into Cartesian coordinates. While straight-forward in principle, knowledge of the back transformation is needed in this case. Unfortunately, the latter can be particularly difficult to construct if the Cartesian coordinates have been transformed into a redundant set of coordinates.

Due to these difficulties, we used an alternative approach in which the GPR is performed in Cartesian coordinates. First, the kernel matrix is built in the space of the transformed coordinates, followed by a back-transformation to the Cartesian space. This is similar in concept to the approach used by descriptor-based machine learning models such as atomic neural networks[124] or Gaussian approximation potentials,[148] where no back-transformation from the descriptor space is possible.

The three basic components of the extended kernel matrix K are given by

$$k\left(\mathbf{x}, \mathbf{x}'\right) = k(\mathbf{q}\left(\mathbf{x}\right), \mathbf{q}(\mathbf{x}')) = k\left(\mathbf{q}, \mathbf{q}'\right), \tag{4.5}$$

$$\frac{\partial k(\mathbf{x}, \mathbf{x}')}{\partial x_k} = \sum_i \frac{\partial k(\mathbf{q}, \mathbf{q}')}{\partial q_i} \frac{\partial q_i}{\partial x_k} \tag{4.6}$$

and

$$\frac{\partial^2 k(\mathbf{x}, \mathbf{x}')}{\partial x_k \partial x'_l} = \sum_i \sum_j \frac{\partial q_i}{\partial x_k} \frac{\partial^2 k(\mathbf{q}, \mathbf{q}')}{\partial q_i \partial q'_j} \frac{\partial q'_j}{\partial x'_l}. \tag{4.7}$$

For all calculations presented, we use the constant mean model as suggested by Denzel and Kästner,[163]

$$E_{\text{mean}} = \max_i E_i + 10, \tag{4.8}$$

with energies measured in eV, and set the noise parameter to $\sigma_n^2 = 10^{-6}$. Unless noted otherwise, our calculations employ the squared exponential kernel,

$$k^{\text{SE}}\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_m^2 \exp\left(-\frac{1}{2} \sum_h \frac{(x_h - x'_h)^2}{l_h^2}\right). \tag{4.9}$$

Matérn kernel[354,355] (for $\nu = 5/2$) as suggested by Denzel and Kästner,[163]

$$k^{\mathrm{M}}\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_m^2 \left(1 + \sqrt{5 \sum_h \frac{(x_h - x_h')^2}{l_h^2}} + \frac{5}{3} \sum_h \frac{(x_h - x_h')^2}{l_h^2}\right) \exp\left(-\sqrt{5 \sum_h \frac{(x_h - x_h')^2}{l_h^2}}\right). \tag{4.10}$$

For torsional angles (given in radian), we use the following periodic kernel suggested by MacKay:[356]

$$k^{\mathrm{P}}\left(\boldsymbol{\phi}, \boldsymbol{\phi}'\right) = \sigma_m^2 \exp\left(-2 \sum_h \frac{\sin^2\left(\frac{\phi_h - \phi_h'}{2\lambda_h}\right)}{l_h^2}\right), \tag{4.11}$$

where setting $\lambda_h = 1.0$ yields functions periodic in $2\pi$. The hyperparameters are optimized by maximizing the log marginal likelihood using the L-BFGS algorithm,[105,106] as implemented in the SciPy open source Python library.[357] Determining a separate length scale hyperparameter for all dimensions is typically not feasible. Therefore, unless noted otherwise, we use a common value along all dimensions $l_h = l$ yielding the so-called isotropic kernels.

### 4.3.3. Internal coordinate systems

Two different sets of primitive internal coordinates are investigated in this article.

The first set employs internal coordinates which are typically used to define the Z-matrix of a molecular system. It comprises bond stretches, planar bends (or linear bending coordinates), and proper torsions. These primitives are generated from the atomic connectivity, which is determined following a scheme similar to the algorithm presented in Ref. 331. Atoms are considered connected if the square of the interatomic distance is less than 1.25 times the square of the sum of their covalent radii. This may lead to a disconnected graph consisting of several fragments. In this case, the latter are then merged iteratively by adding connections between the shortest distance atoms on different fragments.

Using this connectivity graph, a set of internal coordinates can be generated. First, bond stretch coordinates are defined for all connections. Next, the bending angles enclosed by adjacent bonds are calculated and added to the coordinate set. Bending angles larger than 175° or smaller than 5° are ignored. If the center atom along the ignored coordinate is only connected to two atoms, the bend coordinate is replaced by two linear bending coordinates: the coplanar and perpendicular bend described in Ref. 358. Torsion coordinates involving this angle are instead defined via a central bond between the two endpoints of the angle (see Ref. 359 for a more detailed description). Finally, torsion coordinates are defined for all triplets of bonds that do not form three-membered rings.

The second set of primitives, referred to as the "total connection scheme" by Billeter et al.[331] considers every atom connected to every other atom and comprises all resulting inverse internuclear distances. This set has also been used as a basis for more sophisticated molecular descriptors such as the Coulomb matrix[115] or the so-called bag of bonds vector.[118]

In neither of these two sets of internal coordinates do we take advantage of symmetry to reduce the number of primitives. Therefore, both schemes lead to highly redundant sets of coordinates. The number of inverse distances grows as $N(N-1)/2$, whereas the number of Z-matrix-derived internals grows roughly linear with the number of atoms N.

### Delocalized internal coordinates

A simple and unbiased scheme for eliminating redundancies was introduced by Baker et al. in the form of the so-called delocalized internal coordinates.[329] They represent the default coordinate system of choice for most geometry optimization algorithms. The starting point for the construction of delocalized coordinates is the general non-linear transformation from Cartesian coordinates to internal coordinates. However, at any reference geometry $\mathbf{x}_0$, this transformation can be linearized and expressed through the well-known Wilson B-matrix.[360] For small displacements from the reference geometry, the change in internal coordinates can, therefore, be written as

$$\boldsymbol{\Delta}\mathbf{q} = \mathbf{B}\boldsymbol{\Delta}\mathbf{x}, \tag{4.12}$$

where the Wilson B-matrix is given by

$$\mathbf{B} = \left.\frac{\partial \mathbf{q}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}. \tag{4.13}$$

Constructing and diagonalizing the matrix $\mathbf{G} = \mathbf{B}\mathbf{B}^\top$ yields a set of $n$ eigenvectors, with $n$ denoting the number of primitive internal coordinates. Using the matrix $\mathbf{U}$ whose columns contain the eigenvectors corresponding to non-zero eigenvalues (for numerical reasons, a threshold of $10^{-10}$ is used), a non-redundant set of coordinates can be defined as follows:

$$\mathbf{s} = \mathbf{U}^\top \mathbf{q}. \tag{4.14}$$

This reduced set of coordinates, consisting of linear combinations of primitive internal coordinates, is referred to as the "active" coordinate set throughout this article.

### Localized internal coordinates

The linear combination of primitives in delocalized internal coordinates leads to an often undesirable mixing of different types of primitives. Even though it was never used for geometry optimization, Baker et al.[329] described a scheme for localizing the delocalized internal coordinates. The individual primitives are projected onto the active subspace $\mathbf{U}$ by taking the scalar product of a unit vector $\hat{\mathbf{q}}_i$ with a unit component corresponding to the primitive internal $i$ and the columns of the matrix $\mathbf{U}$,

$$\mathbf{q}_i^{\text{proj}} = \sum_k \left(\mathbf{q}_i \cdot \mathbf{U}_k\right) \mathbf{U}_k. \tag{4.15}$$

The resulting set of projected vectors $\{\mathbf{q}_0^{\text{proj}}, \ldots \mathbf{q}_n^{\text{proj}}\}$ is then normalized and Schmidt-orthogonalized to obtain the final set of active internal coordinate vectors. The Schmidt orthogonalization ensures that the final set is non-redundant by eliminating linearly dependent coordinates. Note that this localization scheme does not yield active coordinates consisting of single primitives. However, the active coordinates are typically dominated by just a few primitives, which allows dividing them into stretch, bend, and torsion dominated subsets.

## 4.3.4. Benchmark systems

We measure the influence of the coordinate system on the optimization performance using a benchmark set of 30 molecular geometries originally introduced by Baker.[328] The 30 initial geometries are depicted in Figure 4.1. This set of covalently bonded, small molecules has been used by several groups to investigate their proposed optimization methods. Therefore, it allows us to conveniently track the continuous improvements of the state-of-the-art in geometry optimization over time.[162,334,340,342,361–366] An overview of results selected from this literature is given in Table C.1 of the supporting material.

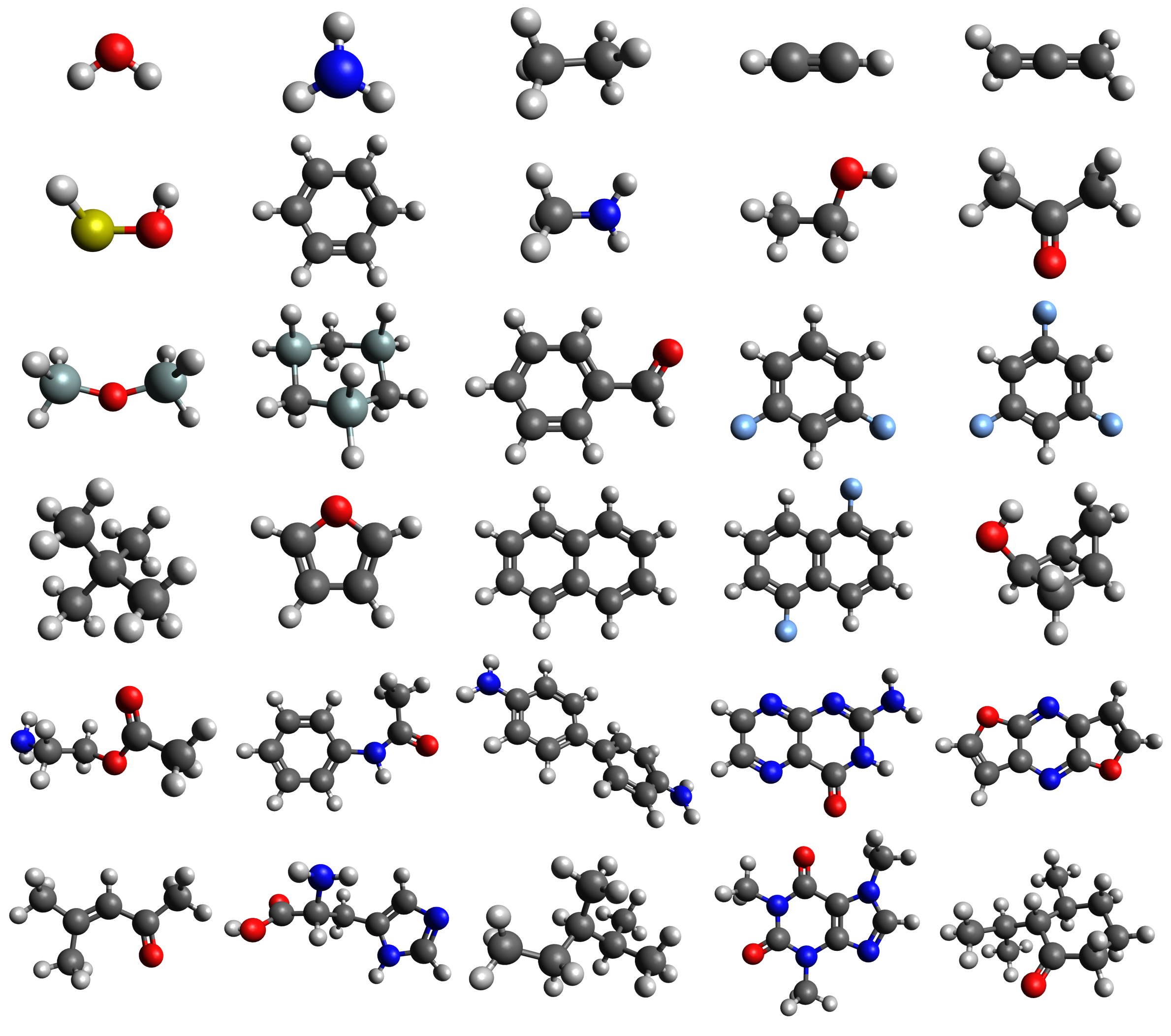


Figure 4.1.: The initial geometries of the 30 benchmark systems suggested by Baker.

The *ab initio* PES is evaluated using the Hartree-Fock method with a STO-3G basis set,[367,368] as implemented in the Q-Chem package.[369]

## 4.4. Results

The number of optimization cycles for each molecule and the sum of optimization cycles over the whole benchmark set are summarized in Table 4.1 for all of the investigated geometry optimization methods. The size of the active coordinate space generated by the various schemes is given in Table C.2 of the supporting material.

### 4.4.1. Cartesian coordinates

The optimization results in Cartesian coordinates mainly serve as reference values for the investigated internal coordinate systems. We note that the ASE implementation of BFGS-LS performs worse than the results reported by Baker[328] for Cartesian optimization using a unit Hessian initial guess. Gaussian process regression-accelerated optimization yields a reduction in *ab initio* evaluations on almost every molecular system and a sum of 545 optimization cycles. Similar to the results reported by Denzel and Kästner, we observe a reduction of up to a factor of two for the larger systems. In our study, the Matérn kernel, which yields a total sum of 543 optimization cycles, does not significantly outperform the squared exponential kernel, as reported by Denzel and Kästner. This is most likely due to differences in the implementation, such as the inclusion of a second hyperparameter $\sigma_m^2$ and the re-optimization of the hyperparameters at every step.

### 4.4.2. Redundant internal coordinates

As a next step, the potential improvements due to non-Cartesian coordinates are investigated by using the fully redundant sets of primitive internal coordinates.

Using the inverse distance coordinates in combination with an isotropic squared exponential kernel leads to a significant further reduction in the number of optimization cycles, in particular, for larger geometries. The resulting sum of 285 *ab initio* evaluations represents an improvement of almost a factor of two in comparison to GPR in Cartesian coordinates.

For the Z-matrix-derived internal coordinates, we use separate kernels for the subsets of bond stretches $\mathbf{r}$, bend angles $\boldsymbol{\theta}$, and torsion angles $\boldsymbol{\omega}$ due to the expected different length scales of these different types of coordinates,

$$C(\mathbf{q}, \mathbf{q}') = \sigma_m^2 \cdot C^{\mathrm{SE}}(\mathbf{r}, \mathbf{r}') \cdot C^{\mathrm{SE}}(\boldsymbol{\theta}, \boldsymbol{\theta}') \cdot C^{\mathrm{P}}(\boldsymbol{\omega}, \boldsymbol{\omega}'). \tag{4.16}$$

This yields a total of four hyperparameters, namely, the scaling parameter $\sigma_m^2$ and three length scale parameters $l_r$, $l_\theta$, and $l_\omega$. Linear bend coordinates are included in the subset of bend angles and therefore share the same length scale $l_\theta$. This more sophisticated model compared to redundant inverse distances yields a further reduction to a value of 232 for the sum of optimization cycles. Using a simpler model with a common length scale for all types of Z-matrix-derived primitives results in a sum of 431 optimization cycles. Details are presented in Section C.3 of the supporting material.

### 4.4.3. Non-redundant internal coordinates

The use of redundant coordinate systems increases the computational overhead of the method. Therefore, two schemes for eliminating the redundancy are investigated, namely, delocalized internal coordinates and localized internal coordinates.

While delocalization presents a completely unbiased approach, the ordering of the primitive internals influences the results of the localization algorithm due to the sequential character of the Schmidt orthogonalization procedure. We choose to sort the inverse distances $1/r_{ij}$ in ascending order of the corresponding bond lengths $r_{ij}$. While localized and delocalized inverse distances perform similar to the fully redundant set on most molecules, both fail to locate a minimum structure for the benzidine molecule. For the case of Z-matrix-derived internals, the strong mixing of different types of primitives in delocalized coordinates implies that the approach of separate kernels cannot be applied.

Table 4.1.: Number of optimization cycles until convergence for the investigated coordinates sets. The > sign denotes optimization runs that did not converge within the maximum of 150 cycles.

| | Cartesian | | | Inverse distances | | | | Z-matrix-derived internals | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Molecule | BFGS-LS | Squared exponential | Matérn | Fully redundant | Delocalized | Localized | Reduced redundancy | Fully redundant | Delocalized | Localized | Reduced redundancy |
| Water | 7 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Ammonia | 6 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 4 | 4 |
| Ethane | 7 | 6 | 6 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 |
| Acetylene | 6 | 6 | 5 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 6 |
| Allene | 6 | 6 | 4 | 5 | 6 | 8 | 5 | 6 | 6 | 7 | 6 |
| Hydroxysulfane | 17 | 14 | 15 | 10 | 10 | 11 | 10 | 7 | 13 | 7 | 7 |
| Benzene | 6 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| Methylamine | 14 | 8 | 7 | 7 | 7 | 6 | 7 | 7 | 9 | 7 | 7 |
| Ethanol | 22 | 13 | 14 | 8 | 7 | 9 | 8 | 8 | 12 | 8 | 6 |
| Acetone | 15 | 14 | 16 | 8 | 9 | 8 | 8 | 7 | 11 | 8 | 8 |
| Disilyl-ether | 17 | 13 | 15 | 8 | 10 | 9 | 8 | 8 | 10 | 9 | 8 |
| 1,3,5-Trisilacyclohexane | 35 | 21 | 22 | 14 | 12 | 11 | 13 | 8 | 23 | 13 | 8 |
| Benzaldehyde | 38 | 17 | 14 | 9 | 9 | 9 | 9 | 8 | 33 | 9 | 9 |
| 1,3-Difluorobenzene | 16 | 7 | 7 | 7 | 7 | 7 | 7 | 5 | 8 | 5 | 5 |
| 1,3,5-Trifluorobenzene | 9 | 6 | 6 | 4 | 6 | 6 | 4 | 6 | 6 | 6 | 6 |
| Neopentane | 8 | 6 | 6 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 5 |
| Furan | 15 | 9 | 9 | 7 | 7 | 7 | 7 | 6 | 9 | 6 | 6 |
| Naphthalene | 14 | 10 | 9 | 6 | 6 | 6 | 7 | 8 | 9 | 6 | 7 |
| 1,5-Difluoronaphthalene | 26 | 13 | 13 | 8 | 8 | 8 | 8 | 6 | 12 | 6 | 6 |
| 2-Hydroxybicyclopentane | 51 | 33 | 31 | 13 | 16 | 17 | 13 | 11 | 35 | 17 | 11 |
| ACHTAR10 | 53 | 35 | 34 | 17 | 18 | 18 | 18 | 12 | 21 | 11 | 10 |
| ACANIL01 | 52 | 28 | 28 | 14 | 13 | 13 | 16 | 8 | 13 | 9 | 8 |
| Benzidine | 34 | 18 | 22 | 12 | > 150 | > 150 | 13 | 10 | 15 | 9 | 10 |
| Pterin | 40 | 19 | 19 | 11 | 10 | 11 | 11 | 9 | 13 | 9 | 9 |
| Difuropyrazine | 30 | 16 | 16 | 9 | 10 | 10 | 10 | 9 | 13 | 8 | 8 |
| Mesityl-oxide | 42 | 26 | 25 | 12 | 11 | 12 | 11 | 9 | 12 | 10 | 7 |
| Histidine | 89 | 68 | 64 | 21 | 40 | 36 | 21 | 14 | 60 | 21 | 15 |
| Dimethylpentane | 44 | 22 | 22 | 12 | 13 | 13 | 12 | 10 | 13 | 9 | 7 |
| Caffeine | 60 | 30 | 30 | 12 | 13 | 13 | 13 | 8 | 16 | 11 | 9 |
| Menthone | 106 | 67 | 71 | 23 | 22 | 22 | 23 | 13 | 41 | 28 | 13 |
| Sum | 885 | 545 | 543 | 285 | > 448 | > 448 | 290 | 232 | 444 | 269 | 225 |

An additional problem is that the periodicity of torsion coordinates can no longer be included by using a periodic kernel. Rasmussen and Williams[104] showed that the periodic kernel can be derived by transforming the one-dimensional input $\omega$ into a two-dimensional space $\mathbf{u}(\omega) = (\cos(\omega), \sin(\omega))$ and applying the squared exponential kernel. We make use of this fact by transforming all torsion coordinates into $(\cos(\omega), \sin(\omega))$ before applying the delocalization or localization scheme. Using the squared exponential kernel, which introduces a total of two hyperparameters, in combination with delocalized Z-matrix-derived primitives, yields a sum of 444 optimization cycles. The localization of the primitive internals allows us to characterize them as stretch-, bend-, and torsion-dominated coordinates, where the main contribution is determined by the sum over the respective squared coefficients. Using a product of separate squared exponential kernels for each of these subsets, again resulting in a total of four hyperparameters, yields a sum of 269 optimization cycles.

### 4.4.4. Reduced redundancy internal coordinates

Since a certain degree of redundancy seems to be beneficial, we further investigate possible schemes to reduce the redundancy without aiming for a non-redundant subset. We attribute the observed performance improvements to the fact that the machine learning model is free to select a subset of coordinates which fulfills the assumption of an isotropic PES better than the active sets created by the delocalization or localization procedure.

For the inverse internuclear distance coordinates, the most successful investigated approach to reduce redundancy consists of the following two steps. First, all primitives corresponding to bond lengths below a certain threshold $r_{\text{cut}}$ are added to the active set (in the presented case $r_{\text{cut}} = 5.0$ Å, the results for other values of $r_{\text{cut}}$ are given in Table C.3 of the supporting material). This might lead to an incomplete coordinate set. Therefore, in the second step, we sort the remaining primitives from shortest to longest bond length and then add them to the active set sequentially if the corresponding B-matrix column and the B-matrix of the current active set are linearly independent. With a sum of 290 optimization cycles, this approach performs comparably to the fully redundant set of inverse distances. However, we note that the generated set is still highly redundant, as can be seen when comparing the size of the active set to the actual number of Cartesian coordinates, as listed in Table C.2 of the supporting material.

The Z-matrix-derived internals seem to perform best if they are separated into stretches, bends, and torsions since this allows using different length scales in the kernel. Therefore, the reduced redundancy approach presented here employs a separate delocalization or localization procedure on each of these three subsets. Note that torsion coordinates again have to be transformed into the $(\cos(\omega), \sin(\omega))$ space, as described in Section 4.4.3. Table 4.1 shows the results for the active set obtained by applying the localization scheme to the individual subsets yielding a sum of 225 optimization cycles. Delocalization of the coordinates within the three subsets yields a similar performance with a sum of 229 optimization cycles (see Table C.3 of the supporting material for details).

## 4.5. Discussion

A comparison of the results presented in this work to the results of previous studies in Table C.1 of the supporting materialshows that the GPR-based approaches in internal coordinates perform similar to the original algorithms suggested by Baker,[328] but cannot compare to the performance of more advanced schemes including a sophisticated guess for the initial Hessian matrix.

The main weakness of the GPR-based approach used in this article is the isotropic kernel, which, in turn, implies an isotropic Hessian given a single training geometry. Anisotropic energy surfaces can only be represented using additional data points, which increases the number of necessary optimization cycles significantly. Denzel and Kästner presented an approach to deal with single coordinates on a different length scale by introducing a separate GPR for these directions. However, our results clearly show that transforming the PES into a more suitable coordinate space allows for a more convenient treatment of anisotropy. The idea of transformation into an isotropic vector space also lies at the heart of preconditioning approaches, as they have been presented, for example, in Refs. 370 and 371. Combining machine learning-accelerated optimization with an appropriate preconditioner might, therefore, lead to competitive performance even in Cartesian coordinates.

We note that rerunning the GPR-based optimization on different machines yields slight variations in the number of optimization cycles. This is most likely due to the fact that the marginal likelihood surface used to optimize the hyperparameters is very flat in the first few steps. A possible solution for future implementations would be to use prior distributions on the hyperparameters. This would also allow us to encode prior knowledge of the Hessian matrix especially in Z-matrix-derived internals and could yield competitive performance compared to approaches with sophisticated initial Hessian guesses.

Both the GPR-based optimization and the transformation into internal coordinates introduce a computational overhead compared to the standard optimization in Cartesian coordinates. The details of the computational cost of both these methods, as well as possible schemes to reduce it, have been discussed in the respective original publications of the approaches. In our combination of GPR-based optimization with a transformed internal coordinate system, the computational overhead is dominated by the GPR-algorithm, in particular by the hyperparameter optimization via a maximization of the marginal likelihood.

Finally, we note that the methods employed in this article to reduce redundancy rely on local information, namely, the Wilson B-matrix. Therefore, a slightly worse performance can be expected for longer trajectories, possibly requiring an occasional reconstruction of the active set of coordinates.

## 4.6. Conclusion

We investigated the influence of the coordinate system on the performance of Gaussian process regression-based geometry optimization on a standard benchmark set of molecules. Significant improvements over Cartesian coordinates are observed for nearly all of the examined internal coordinate systems. This is attributed to a lower coupling of different coordinates and a higher degree of isotropy of the PES in internal coordinates.

Due to the rather modest size of the benchmark set, which consists of a small selection of covalently bonded molecules only, we refrain from a performance ranking of the tested internal coordinate systems, but note that a certain choice of coordinates will work better for certain motives and types of molecular binding. Although an obvious statement, this should be kept in mind when comparing our results to well-established optimizer packages as they are implemented in most computational chemistry program packages. The latter are still outperforming Gaussian process regression, even if formulated in internal coordinates, but take large advantage of hard-coded physical knowledge, which has been gathered through decades of research and continuous fine-tuning.

The lack of heuristics on actual force constants and couplings can only be partially compensated by the choice of a suitable set of internal coordinates, and future undertakings will have to encode this knowledge, e.g., via a pre-informed choice of hyperparameters in their machine learning models.

## 4.7. Acknowledgments

Financial support by the Austrian Science Fund (FWF) under Grant No. P29893 is gratefully acknowledged. We further thank the IT Services (ZID) of the Graz University of Technology and the Vienna Scientific Cluster (VSC) for providing high performance computing resources and technical support.

# 5. Machine learning in computational chemistry: An evaluation of method performance for nudged elastic band calculations

This chapter corresponds to the publication
"Machine Learning in Computational Chemistry: An Evaluation of Method Performance for Nudged Elastic Band Calculations" by Ralf Meyer, Klemens S. Schmuck, and Andreas W. Hauser, in *Journal of Chemical Theory and Computation*, **15** (11), 6513-6523 (2019).

Contributions:

- Ralf Meyer implemented the neural network model and Gaussian approximation potential, ran all presented benchmark results and wrote the first draft of the manuscript.
- Klemens Schmuck implemented the Gaussian process regression model and performed proof-of-concept calculations on the ethane benchmark.
- Andreas W. Hauser supervised the method development, contributed to the manuscript, and provided the funding.

## 5.1. Abstract

The localization of transition states and the calculation of reaction pathways are routine tasks of computational chemists but often very CPU-intense problems, in particular for large systems. The standard algorithm for this purpose is the nudged elastic band method, but it has become obvious that an “intelligent” selection of points to be evaluated on the potential energy surface can improve its convergence significantly. This article summarizes, compares, and extends known strategies that have been heavily inspired by the machine learning developments of recent years. It presents advantages and disadvantages and provides an unbiased comparison of neural network-based approaches, Gaussian process regression in Cartesian coordinates, and Gaussian approximation potentials. We test their performance on two example reactions, the ethane rotation and the activation of carbon dioxide on a metal catalyst, and provide a clear ranking in terms of usability for future implementations.

## 5.2. Introduction

The application of machine learning techniques such as neural networks, Gaussian process regression, and other algorithms to problems of computational chemistry has become a highly active field in recent years.[121, 136, 138, 372, 373] Equipped with a vast range of possible approaches, several closely related applications ranging from the development of atomistic potentials[6, 124, 148, 149] for minimum energy structure searches in large systems over the enhancement of structural optimizers[163, 186, 187] for the localization of extrema on a given potential energy surface (PES) and calculation of reaction rates[346, 347] to predictors for molecular spectra[374] and accelerated molecular dynamics simulations[375] have been suggested recently. Within this subset of applications, the prediction of the electronic energy as a function of the nuclear coordinates is a central concern due to the substantial cost of its evaluation, in particular for larger systems, where complicated electronic structures meet with a very high dimensionality of the corresponding electronic potential energy surface. Aiming for the localization of certain extrema on the PES is a standard problem of any computational chemist, and success is particularly hard to achieve in cases where the point of interest resembles a saddle point of first order or “transition state” (TS) in chemist’s parlance. Knowledge of the latter, together with the typically less costly localization of those PES minima that correspond to reactant and product configurations, allow, for example, the application of transition state theory within the harmonic approximation in order to estimate reaction rates and product distributions. The reaction itself can be thought as a structural rearrangement best described by the minimum energy path (MEP), a curved line connecting both minima and running over the saddle point: in a fully thermalized system, this path corresponds to the trajectory with the largest statistical weight. A commonly used approach for its evaluation is the nudged elastic band method (NEB), an iterative procedure in which the path is discretized by a chain of points or “images” on the PES.[183–185] Each of these images corresponds to a certain geometry of the system. Starting from some initial set of these images, a first guess for the actual path, the NEB technique improves the positions of these images in each iteration until convergence. Unfortunately, hundreds of energy and gradient evaluations are needed in order to converge to the MEP, with each of them taking several minutes to hours in typical problem settings. Obviously, any acceleration of this laborious procedure is highly desired. In this article, we are concerned with the methodological improvement of the NEB method by tweaking its actual choice of images in each step of the path updating process, without any loss of accuracy in the path. The latter claim excludes pragmatic trade-offs such as the growing or freezing string methods.[376–378] Obviously, the geometries of each previously evaluated path contain local knowledge of the high-dimensional PES. In standard NEB, this information is only used to make reasonable choices for placements of new images in the next step. In 2016, Peterson suggested the usage of a neural network to collect the information on the PES gained in each update,[186] which laid the foundation for a series of follow-up studies introducing also other machine learning techniques[187] based on Gaussian process regression.[104] A common strategy of all these methods is to perform any intermediate calculations of energies and gradients, which are necessary to correct the MEP estimate between two updating steps, on a tentative analytical model PES, which is easy to calculate. The latter can be considered as the best approximation to the actual *ab initio* PES given the information at hand and gets updated each time a new path has been suggested. These updates are enforced by the constraint that each new image, suggested by an algorithm working on the tentative PES, must be evaluated on the "real" *ab initio* PES. The so obtained energies and atomic forces are then used to improve the current model PES. This way it is ensured that unbiased information is added to the database only, making it possible to reach the exact MEP in principle upon convergence. Three popular machine learning techniques are applied to the task of accelerating standard-NEB: neural network potentials, Gaussian process regression in Cartesian coordinates, and Gaussian approximation potentials. All of them use energy as well as gradient information for their predictions. After presenting the details of their implementation in Section 5.3, their performance will be tested on two molecular systems in Section 5.4. The first test, the internal rotation of an ethane molecule in gas phase, is chosen for the sake of a direct comparison to the literature.[186] The second test, $CO_2$ activation on a $Pt_4^-$ atomic cluster, has been selected in order to provide a more challenging, realistic evaluation on a system of current chemical interest. A fair comparison of the various machine learning approaches is attempted by creating unbiased, authentic starting conditions without usage of extra information. In Section 5.5, we discuss the advantages and disadvantages of each method. We conclude our study with a final suggestion for an optimal choice of method and its corresponding hyperparameters.

## 5.3. Methods

We start with an overview of selected machine learning techniques that have recently been suggested for the improvement of NEB calculation performance. We summarize the main ideas, highlight similarities, and discuss differences between these approaches. Numeric values for all relevant model parameters are listed in the following subsections. For the sake of brevity, their actual effects on the corresponding algorithm are not discussed in greater detail. Instead, we refer the reader to the original publications of the individual method as provided in the corresponding sections.

### 5.3.1. Nudged elastic band

NEB is a standard algorithm of computational chemistry that determines, in an iterative fashion, the reaction path given two minima on the PES of the molecular system under investigation.[184]

First, two initial geometries, preferably those closest to a transition state of interest, are connected via a linearly interpolated path consisting of $N$ geometries or "images". Each of these images corresponds to a certain position on the potential energy surface. Neighboring images are then connected by fictitious springs to keep them equidistantly distributed along the path.[183,379] The basic algorithm attempts to minimize an objective function, which is a sum over spring contributions and the potential energies of all images. Problems such as corner-cutting and sliding-down issues are solved by a convenient projection of spring and gradient forces: While the former are considered only in their projection along the band, the latter are only allowed to act perpendicular to the band. The total force acting on each image is then given by

$$\boldsymbol{F}_i = -[\nabla_i E(\boldsymbol{x}_i) - ([\nabla_i E(\boldsymbol{x}_i)]\hat{\boldsymbol{\tau}}_i)\hat{\boldsymbol{\tau}}_i] + (\boldsymbol{F}_i^{\mathrm{s}}\hat{\boldsymbol{\tau}}_i)\hat{\boldsymbol{\tau}}_i, \tag{5.1}$$

with $\hat{\boldsymbol{\tau}}_i$ as tangent to the band, $\boldsymbol{x}_i$ as position, $\boldsymbol{F}_i^{\mathrm{s}}$ as spring force, and the subscript $i$ denoting the image. The spring force can be expressed as

$$\boldsymbol{F}_i^{\mathrm{s}} = k[(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) - (\boldsymbol{x}_i - \boldsymbol{x}_{i-1})], \tag{5.2}$$

with $k$ as the spring constant. Since the band cannot be considered smooth in general, the tangent is estimated, for example, through neighboring images,[379]

$$\hat{\boldsymbol{\tau}}_i = \frac{\boldsymbol{x}_{i+1} - \boldsymbol{x}_{i-1}}{|\boldsymbol{x}_{i+1} - \boldsymbol{x}_{i-1}|}. \tag{5.3}$$

The main procedure of the nudged elastic band comprises the following steps:

1. Initialize the band with a linear interpolation between the two minima.
2. Evaluate the energy and gradient for each image on the potential energy surface.
3. Estimate every tangent, and calculate the spring and acting force of each image.
4. Move every image on the potential energy surface with respect to the acting force.
5. Continue with step 2 until the acting force norms are below a certain threshold and convergence is achieved.

Note that the nudged elastic band algorithm converges toward the minimum energy path, but it does not deliver the transition state automatically. This feature can be added by first converging the algorithm to the (approximate) minimum energy path and then applying the climbing image method, where the spring force acting on the image with the highest energy is set to zero, and the component of the gradient force along the band is inverted. As a consequence, the image with highest energy moves up the PES in the direction of the elastic band and down perpendicular to the band, toward the actual saddle point.[185,379]

Alternatively, well-established methods such as eigenvector following techniques[173] can be applied to reasonably close geometries for an exact localization of a transition state. If the exact reaction pathway itself is not needed, much less expensive approximators of the latter can be used instead to provide good estimates for the TS structure as well.[376–378] The initial linear interpolation of the band is sometimes problematic because atoms can get too close to each other or may even collide upon rearrangement. These issues of unphysical proximities can be solved with the so-called image-dependent pair potential technique[380] (IDPP), where the actual PES is replaced with a well-behaved analytical surface, which is significantly cheaper to evaluate. For that, the algorithm first calculates the pairwise distance between all atoms at each minimum geometry. These distances are then used to linearly interpolate between pairwise distances for each intermediate image, which gives the optimal distance between the atoms. For each atom, the deviation of the actual pairwise distances to the optimal is calculated. The image-dependent pair potential is given as a weighted sum over all these deviations. The nudged elastic band method as described above can then be applied to this potential in order to obtain an initial pathway on which atomic collisions are avoided. Our computational study is based on the NEB algorithm as implemented in the ASE suite of programs.[349] In all calculations, the spring constant between images is set to the ASE default value of $k = 0.1$ eV/Å. Two convergence thresholds are used. Once the first threshold of a maximum force on all atoms $T_{\text{CIon}} = 0.5$ eV/Å is reached, the highest energy image is switched to climbing mode. This second stage of the algorithm is run until the maximum force on all atoms is below $T_{\text{MEP}} = 0.05$ eV/Å. As suggested by the ASE documentation, the Fast Inertial Relaxation Engine (FIRE)[343] algorithm is used to optimize the NEB. We use the ASE default values for the parameters of the optimizer, which follow the suggestions of the original publication of the algorithm. The time step is set to $\Delta t = 0.1$ Å$\sqrt{\text{u/eV}}$ (ASE time units), which corresponds to approximately 1 fs. A maximum time step of $\Delta t_{\text{max}} = 10\Delta t$ is used. The remaining parameters are set to $\alpha_{\text{start}} = 0.1$, $f_\alpha = 0.99$, $f_{\text{inc}} = 1.1$, $f_{\text{dec}} = 0.5$, and $N_{\text{min}} = 5$. The ASE implementation uses a maximum step size, which was set to 0.2 Å.

### 5.3.2. Machine learning-accelerated nudged elastic band

Our implementation of the machine learning-accelerated nudged elastic band (ML-NEB) follows a simple algorithm outlined by Peterson, referred to as the "all-images-evaluated" (AIE) algorithm by Koistinen et al.:[187]

1. Generate initial guess for the MEP
2. Calculate *ab initio* energies and forces for all images
3. Check for convergence $F_{\text{max}} < T_{\text{MEP}}$
4. Add images to the training set
5. Train machine learning potential
6. Run a complete NEB on the updated machine learning potential, starting from the same initial guess obtained in step 1
7. Go to step 2

The NEB calculation in step 6 switches to climbing image NEB when the maximum force drops below a threshold of 0.5 eV/Å. The same value is used for classic as well as ML-enhanced NEB. In order to avoid oscillation near the true MEP, the convergence threshold for the machine learning prediction of the MEP is reduced to $T^{\text{ML}}_{\text{MEP}} = 0.025$ eV/Å. In addition to restricting the number of iterations in both phases of the NEB algorithm to $N^{\text{ML}}_{\text{max}} = 250$ and $N^{\text{ML}}_{\text{max,CI}} = 150$, we also implemented the suggestion of Koistinen et al.[187] to abort the NEB optimization on the machine learning PES as soon as the distance from any image to the nearest training example exceeds the trust radius, $r_{\text{max}}$.

### 5.3.3. Machine learning potentials

Standard techniques of approximating potential energy surfaces used to employ physically motivated models with a relatively small number of meaningful parameters, which are then fitted to high level *ab initio* data. More recently, this traditional concept has been challenged by a purely mathematically inspired class of atomistic potentials, where physically motivated terms are sacrificed for higher flexibility. These machine learning potentials can achieve high accuracy but typically use significantly more parameters and therefore need more reference data. The following sections outline the main ideas of the three investigated machine learning potentials including the prediction formulas for the total energy. For more details on the inclusion of energy and gradient information in all learning procedures under discussion we refer to the original publications. We note that the usage of atomic forces as additional input data is essential.

#### Neural network potentials

A highly prominent machine learning ansatz has been introduced by Behler and Parrinello.[124,136] The main idea of their approach is to assume that the total energy can be written as a sum of atomic energy contributions

$$E = \sum_{i}^{N} \epsilon_i(\mathbf{d}_i), \tag{5.4}$$

which are functions of a set of descriptors $\mathbf{d}_i$ of the local environment of the atom. These atomic energy functions are modeled using neural networks (NNs). For each atom type, one neural network, referred to as atomic neural network, is employed.

The local environment is encoded by "descriptors", sometimes also denoted as "symmetry functions" or "features", which ensure a constant number of inputs for the individual atomic neural networks. They also offer the possibility to enforce physical invariances and symmetries of the PES, for example, with respect to translation or rotation. Invariance with respect to permutation of same-type atoms is automatically ensured by usage of the same atomic neural network. The local environment is limited by a cutoff sphere of radius $R_\mathrm{c}$. A cutoff function is used to ensure that all descriptors fade to zero at the cutoff radius and to avoid discontinuities in the PES as atoms enter or leave the cutoff sphere:

$$f_\mathrm{c}(R_{ij}) = \begin{cases} \frac{1}{2}\left[\cos\left(\frac{\pi R_{ij}}{R_\mathrm{c}}\right) + 1\right] & \text{for } R_{ij} \leq R_\mathrm{c} \\ 0 & \text{for } R_{ij} > R_\mathrm{c}, \end{cases} \tag{5.5}$$

where $R_{ij}$ denotes the distance between atoms $i$ and $j$. For the descriptors, we use functional forms originally suggested by Behler,[125] combined with the parameters given by Artrith and Kolpak.[381] This set employs a combination of two- and three-body descriptors. The former, referred to as $G^2$ by Behler, are sums of Gaussians multiplied by the cutoff function,

$$G_i^2 = \sum_{j\neq i} e^{-\eta R_{ij}^2} \cdot f_\mathrm{c}(R_{ij}), \tag{5.6}$$

where $\eta$ is a parameter of the descriptor. The three-body descriptor, termed $G^4$ by Behler, combines cosine polynomials with radial Gaussians multiplied by the cutoff function,

$$G_i^4 = 2^{1-\zeta} \sum_{j\neq i} \sum_{k\neq i,j} (1 + \lambda \cos\theta_{ijk})^\zeta \, e^{-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)} f_\mathrm{c}(R_{ij}) f_\mathrm{c}(R_{ik}) f_\mathrm{c}(R_{jk}), \tag{5.7}$$

with $\zeta$, $\lambda$, and $\eta$ as parameters and $\theta_{ijk}$ as the angle between the vectors $\mathbf{R}_{ij}$ and $\mathbf{R}_{ik}$. In typical neural network applications, the descriptors are scaled to a fixed range, for example, -1 to 1, as suggested by Behler, in order to avoid biasing. However, we use the raw descriptor values without scaling for practical reasons in the given application: A rescaling each time a new data point is added to the training set would prohibit the usage of weights from previous fits.

With the help of these descriptors a certain input geometry of a molecule is translated into a vector, $\mathbf{d}_i$, which serves as the input to a feed-forward type neural network with two hidden layers. The hyperbolic tangent is used as activation function, which leads to an atomic energy function of the form

$$\epsilon_i(\mathbf{d}) = b^3 + \sum_l w_l^{23} \tanh\left(b_l^2 + \sum_k w_{kl}^{12} \tanh\left(b_k^1 + \sum_j w_{jk}^{01} d_j\right)\right). \tag{5.8}$$

The parameters $b$ and $w$ of this function, referred to as biases and weights, are shared by atomic energy functions for all atoms of the same type.

The actual learning process, given a set of molecular geometries, total energies, and forces, corresponds to a minimization of the loss function

$$\mathcal{L} = \frac{1}{M}\sum_i^M \left(E_i^{\mathrm{NN}} - E_i^{\mathrm{Ref}}\right)^2 + \frac{\beta}{3NM}\sum_i^M\sum_j^{3N}\left(F_{i,j}^{\mathrm{NN}} - F_{i,j}^{\mathrm{Ref}}\right)^2 + \lambda\sum_k^{N_w} w_k^2, \tag{5.9}$$

which consists of the mean squared error (MSE) of the total energies, the MSE of the forces, and a regularization term that penalizes more complicated models. $M$ denotes the number of training examples, $N$ the number of atoms, and $N_w$ the number of weights, and $\beta$ and $\lambda$ are hyperparameters.

Our implementation of neural network potentials is based on the TensorFlow library,[319] and the loss function is minimized with respect to the weights using the L-BFGS algorithm[105,106] as implemented in the SciPy open source Python library.[357]

Reliable black box training of neural networks is a difficult task. We follow the approach of Peterson and run the optimization until the root-mean-square energy deviation is less than 0.001 eV/atom and the root-mean-square deviation of the forces is less than 0.05 eV/Å. Additionally, we restrict the number of L-BFGS iterations to a maximum of $10^4$. A minimum of 200 iterations is enforced in order to ensure new training data is adequately represented by the neural network potential. At each L-BFGS step, a maximum of 200 previous iterations are used for the Hessian approximation.[105] For the first step, the weight matrices $w$ are initialized randomly, and the bias vectors $b$ are set to zero, with the exception of the linear output layer, where the bias is set to the energy value of a single atom of the respective element. Subsequent optimizations start from the result of the previous step. This not only accelerates the training phase but also stabilizes the whole ML-NEB algorithm, which would otherwise be prone to oscillations about the MEP.

### Cartesian Gaussian process regression

The simplest possible approach to fitting an energy surface is to model the energy as a function of the Cartesian coordinates of the atoms. In principle, any machine learning regression algorithm could be used to model this function. Koistinen et al. suggested Gaussian process regression (GPR) in Cartesian coordinates (abbreviated as CC-GPR in this article) because of the closed form solution for the parameters and a black-box method for optimizing the hyperparameters (maximizing the marginal likelihood).

The total energy is written as a linear model in a high dimensional space, typically referred to as feature space:

$$E(\mathbf{x}) = \sum_h w_h \phi_h(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \tag{5.10}$$

where $\boldsymbol{\phi}(\cdot)$ denotes the transformation from the Cartesian input space into the higher dimensional feature space and $w_h$ are the coefficients or weights of the linear fit. The energy prediction for a previously unseen geometry $\mathbf{x}^\star$ is given by

$$E(\mathbf{x}^\star) = \mathbf{E}^{\text{ref}} \left(\Phi^\top \Phi + \sigma^2 \mathbf{I}\right)^{-1} \Phi^\top \boldsymbol{\phi}(\mathbf{x}^\star), \tag{5.11}$$

where $\mathbf{E}^{\text{ref}}$ is the vector of all reference energies, $\Phi$ is a matrix with the transformed reference geometries $\{\boldsymbol{\phi}(\mathbf{x}_i), \ldots, \boldsymbol{\phi}(\mathbf{x}_M)\}$ along the columns, and $\sigma^2$ is the assumed variance of the training data, which improves numerical stability of the matrix inversion. See Ref. 104 for a detailed description of the derivation of this result. For the prediction, only the inner product in feature space is needed, which is typically replaced by a so-called kernel (or covariance function in the context of GPR), $C(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$. In practice, knowledge of the actual transformation into feature space replaced by the kernel is not needed.

Therefore, a central component of the GPR approach is the covariance function, which acts as a measure of similarity between different points on the PES. We use the infinitely differentiable squared exponential covariance function[187]

$$C\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_c^2 + \sigma_m^2 \exp\left(-\frac{1}{2}\sum_{h=1}^{3N} \frac{(x_h - x_h')^2}{l_h^2}\right), \tag{5.12}$$

where $\sigma_c^2$, $\sigma_m^2$, and $\mathbf{l} = \{l_1, \ldots, l_{3N}\}$ are the hyperparameters of the covariance function. Note that $\sigma_c^2$ introduces a constant energy offset, which compensates for the absence of a mean model. The case of a common length scale along all dimensions, $l_h = l$ is referred to as isotropic squared exponential covariance function, whereas the general case is anisotropic. In addition to fixing the constant term $\sigma_c^2 = 100$ eV$^2$ as suggested by Koistinen et al.,[187] we also fix the scaling factor $\sigma_m^2 = 1$ eV$^2$, leaving only the length scale $\mathbf{l}$ as free hyperparameter. Denzel and Kästner[163] reported better performance in optimization tasks using the Matérn kernel[354,355](for $\nu = 5/2$),

$$C\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_c^2 + \sigma_m^2 \left(1 + \sqrt{5\sum_{h=1}^{3N} \frac{(x_h - x_h')^2}{l_h^2}} + \frac{5}{3}\sum_{h=1}^{3N} \frac{(x_h - x_h')^2}{l_h^2}\right) \exp\left(-\sqrt{5\sum_{h=1}^{3N} \frac{(x_h - x_h')^2}{l_h^2}}\right). \tag{5.13}$$

Again, we fix the hyperparameters $\sigma_c^2$ and $\sigma_m^2$ to the aforementioned values and only optimize the length scale hyperparameter $\mathbf{l}$.

### Gaussian approximation potentials

Gaussian approximation potentials (GAP)[148,149] introduced by Bartok et al. feature a similar approach as the neural network potentials of Behler and Parrinello. The total energy is written as a sum of atomic contributions, which in turn are modeled as functions of descriptors of the local environment.

$$E = \sum_i^N \epsilon_i(\mathbf{d}_i) = \sum_i^N \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{d}_i). \tag{5.14}$$

These atomic energy functions, $\epsilon_i(\mathbf{d}_i)$, are fitted to the total energy using Gaussian process regression, as hinted at by the linear expansion in basis functions $\boldsymbol{\phi}$. The covariance of two total energies $E_a$ and $E_b$ is then given by a sum of covariances of atomic contributions (see Ref. 149 for a detailed derivation):

$$\begin{aligned}
\langle E_a E_b \rangle &= \left\langle \sum_{i\in a} \epsilon_i(\mathbf{d}_i) \sum_{j\in b} \epsilon_j(\mathbf{d}_j) \right\rangle \\
&= \left\langle \sum_{i\in a} \sum_{j\in b} \sum_{hh'} w_h w_{h'} \phi_h(\mathbf{d}_i)\phi_{h'}(\mathbf{d}_j) \right\rangle \\
&= \sum_{i\in a} \sum_{j\in b} \sum_{hh'} \langle w_h w_{h'} \rangle \, \phi_h(\mathbf{d}_i)\phi_{h'}(\mathbf{d}_j) \\
&= \sigma_w^2 \sum_{i\in a} \sum_{j\in b} \sum_{h} \phi_h(\mathbf{d}_i)\phi_h(\mathbf{d}_j) \\
&= \sigma_w^2 \sum_{i\in a} \sum_{j\in b} C(\mathbf{d}_i, \mathbf{d}_j).
\end{aligned} \tag{5.15}$$

Typically, this approach is combined with the so-called bispectrum descriptors or the smooth overlap of atomic positions kernel (SOAP),[128] which are similar to the Behler symmetry function descriptors but use different radial and angular expansions. For the sake of a fair comparison, we will be using the same descriptor set for both descriptor-based approaches in our study. The SOAP approach was recently expanded to allow coupling of different atomic species,[130] similar to the embedding concept used in SchNet.[139] In this work, we choose not to couple descriptions of different atomic species as this is closest to the neural network potential ansatz.

In addition to the squared exponential covariance function, we also test the normalized dot product covariance function as has been suggested by Bartok and Csányi:[149]

$$C\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_c^2 + \sigma_m^2 \frac{\left(\mathbf{x}\cdot\mathbf{x}' + \sigma_0^2\right)^\xi}{\left(\mathbf{x}\cdot\mathbf{x} + \sigma_0^2\right)^{\xi/2} \left(\mathbf{x}'\cdot\mathbf{x}' + \sigma_0^2\right)^{\xi/2}}, \tag{5.16}$$

where the additive constant $\sigma_0^2$ is a hyperparameter that can be optimized and the exponent $\xi$ is a fixed hyperparameter. The parameters $\sigma_c^2$ and $\sigma_m^2$ are set to the same value as for the squared exponential kernel.

## 5.4. Results

Two molecular systems have been chosen for our study. The first part is concerned with the internal rotation of ethane, a standard problem that has been used before in the same context of machine learning applications. Although not particularly interesting, it serves as a benchmark and allows for evaluations against other strategies in the literature. However, as will be made clear in later sections of this article, the advantage of its well-behaved, rather simple, and symmetric PES introduces a 2-fold intrinsic bias: First, it is virtually impossible for any meaningful NEB implementation not to converge. Instabilities in a chosen algorithm or sensitivities to unfortunate initialization (e.g., the random weights in NN approaches) are barely detectable. Second, the high symmetry provides bigger advantages for certain methods than for others. More concrete, it tips the scale toward descriptor-based machine learning potentials.

Therefore, our choice for the second part is a more challenging, less symmetric reaction in a metallic system that is of current interest to the community.[382–384] We are concerned with the activation of $CO_2$ over a $Pt_4^-$ atomic cluster ion, where a C=O bond is ruptured and an O-Pt bond is formed. This problem is not only challenging from the NEB point of view but also from the point of the computational evaluation of the PES. We note that convergence of *ab initio* methods in metallic systems is particularly problematic, leading to much more time-intense single-point evaluations, discontinuous results for the PES due to wrong or incomplete convergence, and a much higher risk for unphysical choices of geometries during the NEB algorithm. Special care has to be taken here.

In both systems to be investigated, the *ab initio* PES is calculated using the Q-Chem program package.[369] Details of the method chosen for each system are provided in the corresponding sections.

### 5.4.1. Test 1: Ethane rotation

The rotation about the C-C bond in ethane is one of the examples used in Peterson's original description of machine learning-accelerated NEB[186] and has been used by Jónsson et al.[380] to demonstrate their IDPP initial guess. In order to reproduce the original results, we use the B3LYP functional[62,64,66,67] and the 6-31+G* basis set[385–388] as well as 9 interior images between the two minima. We do, however, use a slightly different initial guess in order to avoid the long and complex MEP described by Peterson, which would create an unfair disadvantage for methods based on Cartesian coordinates (including standard-NEB method without machine learning). This is achieved by setting the `remove_rotation_and_translation` flag in the ASE NEB implementation for the IDPP optimization. The complete initial path used for all ethane NEB runs is given in Section D.1 of the supporting material. As suggested by Koistinen et al., the trust radius for the ML-NEB algorithm is set to half the length of the initial path, that is, $r_{\max} \approx 1.3$ Å.

The plain NEB method without machine learning acceleration needs 21 evaluations of the whole band until reaching convergence. Note the fundamentally different outcome in comparison to the 48 steps reported by Peterson, which simply follows from a more natural choice of coordinate representation. All methods to be tested in this work start from the same geometries, and their performance will be tested against the 21 evaluations necessary with standard NEB.

#### Neural network potentials

For our tests of the neural network potential, we use the same network architecture as suggested by Peterson, which consists of two hidden layers containing five neurons each. Still, reproducing results of previous studies proves difficult due to the random weight initialization used when training a neural network. The performance is also influenced by the choice of the hyperparameters $\beta$ and $\lambda$ in the loss function (Equation 5.9), which are not mentioned in the study of Peterson.

A small hyperparameter study for the regularization parameter shows that $\lambda = 10^{-4}$ (see Section D.2 of the supporting material) yields the best results for $\beta = 1$, converging after four evaluations of the band (i.e., evaluation of the initial guess and three MEP predictions of the ML method) in the best case and six evaluations in the worst case. A typical progression of the PES and MEP as predicted by the neural network potential is presented in Figure 5.1, which shows NEB projections onto a two-dimensional cut through the multidimensional PES along the two most important internal coordinates, the C-C bond length and the dihedral angle. It depicts three update steps of the band, starting with the initial path (gray) and a first prediction (black) in the leftmost picture. The picture in the middle shows the second step, where the prediction of the first step has been replaced by actual data evaluated on the unknown *ab initio* surface. Note how this additional data affects the landscape and with it the next prediction. In the rightmost picture, this procedure has been repeated. More data is now available, and the onset of convergence can be observed as the prediction and the last path evaluation are almost identical. Note that even the converged path does not fully agree with the MEP one would obtain on that particular two-dimensional energy surface due to its projective character; the converged path is the true MEP with respect to all degrees of freedom, that is, also to those not shown.

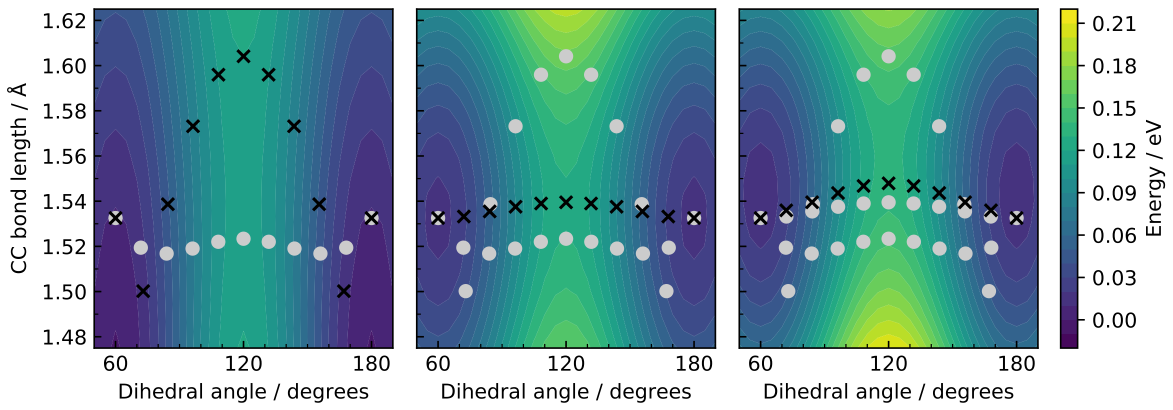


Figure 5.1.: Three reaction pathway updates for the internal rotation of ethane, obtained with the NN-based NEB method, and projected onto the energy landscape created by two prominent internal coordinates, the C-C bond length and the dihedral angle. The NEB prediction of each step (black) is replaced by corresponding data points (grey) evaluated on the *ab initio* PES in the next step.

### Cartesian Gaussian process regression

Our evaluation of this ML method starts with a test of the isotropic squared exponential covariance function as a suitable measure of similarities between images. The length scale is optimized each time the model is fitted by maximizing the marginal likelihood using the L-BFGS algorithm.[105,106] For the first iteration, the starting value of the optimization is set to $l = 1$ Å. Subsequent iterations use the optimization result of the previous step as starting value. Aiming at a perfect reproduction of the *ab initio* PES, the noise parameter is chosen to be as small as $\sigma^2 = 10^{-8}$ eV$^2$; the same value as used in Ref. 187. With these parameters, the MEP is found after only 7 evaluations of the band. Using the Matérn covariance function and the same parameters, convergence is reached after 8 iterations.

For the sake of comparison, we also investigate the performance of anisotropic covariance functions, optimizing all 24 length scale hyperparameters during the fit, again starting from a value of 1 Å. With this modification, the number of band evaluations until convergence is increased to 10 for the squared exponential covariance function and reduced slightly to 7 evaluations for the Matérn covariance function. Anisotropic covariance functions are not recommended for general application, as the high dimensionality of the hyperparameter space increases the computational effort for the fit significantly. Also, it leads to multiple local maxima on the marginal likelihood surface, which makes the hyperparameter optimization less reliable.

### Gaussian approximation potentials

In the course of this study, it became obvious that a combination of advantageous features of both methods studied so far, the NN and the Cartesian GPR ansatz, would be highly desirable. Such a possibility is provided by Gaussian approximation potentials, where the translation-, rotation-, and permutation-invariant descriptor space of a neural network potential is combined with atomic energy summations within the closed-form expression of Gaussian process regression.

We compare the performance of the isotropic squared exponential and Matérn covariance function as well as the normalized dot product covariance function (for exponents $\xi = 2$ and $\xi = 4$). Anisotropic covariance functions are omitted from this study as the problems already shown for Cartesian GPR would be even worse for GAP due to the high dimensionality of the descriptor vectors. The noise term is set to $\sigma^2 = 10^{-6}$ eV$^2$. This increase over the value for the Cartesian GPR was necessary to avoid numerical issues during the Cholesky decomposition. This can probably be attributed to the fact that there is no guarantee that the total energy can be written as a sum of atomic contributions or that these atomic contributions are functions of a limited set of descriptors. For the Matérn covariance function, the algorithm converged after 6 iterations of the whole band. For the other three covariance functions, 4 iterations were sufficient.

### Summary on ethane rotation

The aim of this first investigation was to provide the crucial, yet missing comparison of two state-of-the-art ML-NEB methods and to introduce the new concept of combining their advantages in the form of the GAP-based approach, given a benchmark system that has been used for the same purpose in the past. Our results with respect to NEB improvement are summarized in Table 5.1. This preliminary outcome already demonstrates the significant acceleration of the NEB convergence by any machine learning ansatz. However, the final numbers of band iterations are very closely grouped, so a more detailed discussion of performance differences seems inappropriate. Clearly, the simplicity of this benchmark system limits its suitability for an advanced study. Therefore, comments and further discussion will be postponed to Section 5.5.

Table 5.1.: Summary of the savings in *ab initio* evaluations for the investigated machine learning potentials for the ethane benchmark.

| Machine learning method | $N_{\text{steps}}$ |
|---|---|
| Reference value (no ML) | 21 |
| NN, 5-5 architecture | $5 \pm 1$ |
| CC-GPR, isotropic squared exp. | 7 |
| CC-GPR, anisotropic squared exp. | 10 |
| CC-GPR, isotropic Matérn | 8 |
| CC-GPR, anisotropic Matérn | 7 |
| GAP, isotropic squared exp. | 4 |
| GAP, isotropic Matérn | 6 |
| GAP, dot product $\xi = 2$ | 4 |
| GAP, dot product $\xi = 4$ | 4 |

## 5.4.2. Test 2: Bond breaking in heterogeneous catalysis

For the second part, we have chosen the $CO_2$ activation on a $Pt4_4^-$ cluster, a reaction that has recently been studied both experimentally and computationally by Green et al. in Ref. 384. The aim of their study was to illustrate effects of metal cluster size on the amount of $CO_2$ activation. After adsorption from the gas phase, the molecule remains strongly bound to the cluster, but a larger barrier prevents it from dissociation. The task at hand is to locate the rate determining transition state of the fully unconstrained system between the two neighboring minima and identify the MEP by application of NEB comprising 7 intermediate images. The complete initial path as constructed via the IDPP approximation is given in Section D.3 of the supporting material. The *ab initio* PES is calculated employing unrestricted DFT with the B3P86 functional[66,389] and the D95 basis set[390,391] in combination with the MWB60 Stuttgart/Dresden effective core potential.[392] An energy profile of the MEP is plotted in Figure 5.2. For the ML-NEB, we use a significantly reduced trust radius of only a tenth of the

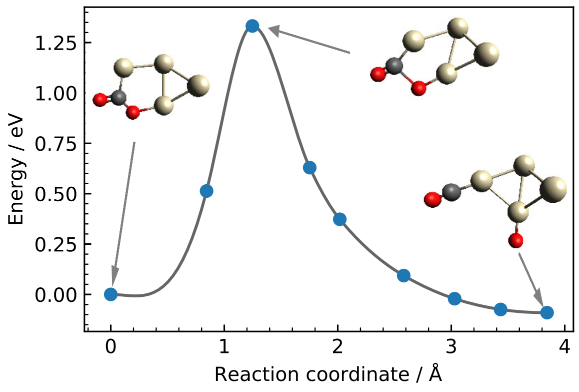


Figure 5.2.: Reaction pathway for the dissociation of $CO_2$ over a $Pt_4^-$ atomic cluster, including the geometries of both minima and the rate determining TS.

initial path ($r_{\text{max}} \approx 0.284$ Å) in order to avoid unphysical geometries, which are difficult to evaluate *ab initio*. The standard NEB algorithm, which is used as baseline for the comparison of the ML-NEB calculations, takes 86 iterations until convergence for this benchmark system. All results of this study are summarized in Figure 5.3 in which the maximum force per atom is plotted as a function of the number of *ab initio* band evaluations showing the convergence behavior for the various machine learning potentials.

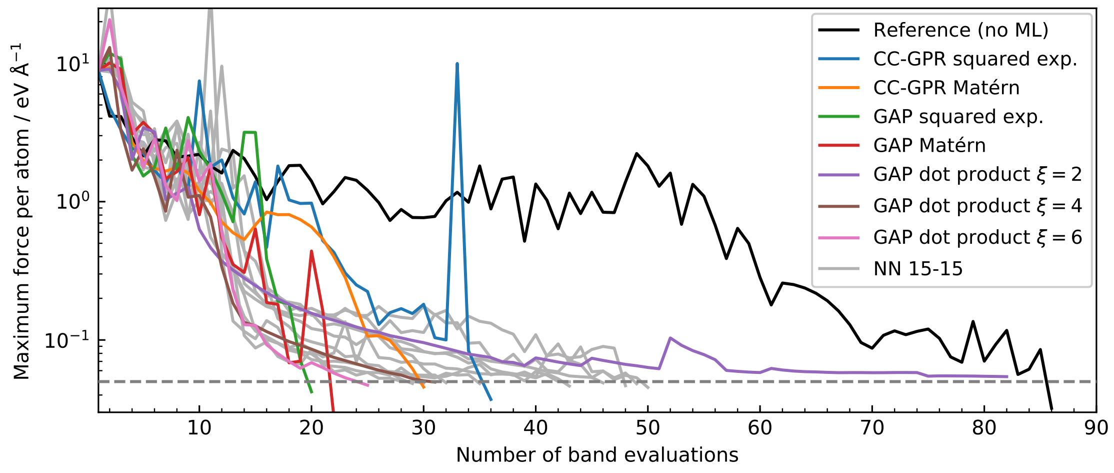


Figure 5.3.: The maximum force per atom, recorded for all methods, and plotted as a function of the number of band evaluations. Due to its statistical character (random initialization of the weights), several curves are presented for the NN-based approach. Note the sudden spike of the Cartesian GPR approach (squared exponential, blue curve) at step 33, which is related to the exploration of a badly represented part of the PES.

### Neural network potentials

Finding a suitable set of hyperparameters and neural network architecture for this more challenging benchmark proves difficult. A small regularization parameter of $\lambda = 10^{-6}$ is chosen in order to avoid restricting the neural network complexity, and the force weighting parameter is set to $\beta = 1$. Three network architectures, all featuring two hidden layers with 5, 15, or 30 neurons per layer, are investigated with respect to their potential savings in *ab initio* evaluations and their consistency in achieving those results. A detailed analysis of all the conducted calculations can be found in Section D.4 of the supporting material. While all three architectures yield similar results for their respective best performing runs, reducing the number of band evaluations by roughly a factor of 3, the results for the medium sized 15–15 network architecture feature the smallest variance. The average number of *ab initio* band evaluations for this setup is 40.5 with a maximum of 50 and a minimum of 29 iterations.

### Cartesian Gaussian process regression

Cartesian GPR parameters do not need to be adapted to the varying molecular systems. Therefore, the noise parameter remains set to $\sigma^2 = 10^{-8}$ eV$^2$. For this more complicated PES only the performance of isotropic covariance functions is investigated, as the optimization of all 21 length scale parameters for the anisotropic case would become computationally more and more demanding after the first few iterations. This rapidly increasing computational effort is also the reason why a slightly different hyperparameter optimization scheme is used. Starting from $l = 1$ Å for the first step and the previous optimization result for subsequent steps, the hyperparameter is only optimized for the first ten steps and kept fixed for the remaining algorithm. A motivation for this choice and an analysis of its benefits is given in Section 5.5 and Section D.5 of the supporting material. Using this scheme, the Cartesian GPR yields a reduction to 36 and 30 evaluations of the band for the squared exponential and the Matérn covariance function, respectively.

### Gaussian approximation potentials

As for the ethane system, the noise term is set to $\sigma^2 = 10^{-6}$ eV$^2$ to avoid numerical issues. The hyperparameters are optimized using the same scheme as described above for Cartesian GPR, that is, their values are frozen after 10 iterations. The combination of GAP with a squared exponential covariance function achieved the best overall performance of all investigated machine learning potentials, reducing the number of steps from 86 without machine learning to just 20 evaluations of the band. Using the Matérn kernel, a similar performance of 22 iterations is achieved. Given a sufficiently high complexity to accurately represent the PES, the GAP implementation featuring a dot product kernel yields competitive performance, as it reduces the number of iterations needed to 31 and 25 for $\xi = 4$ and $\xi = 6$, respectively. The convergence curve for $\xi = 2$ (see Figure 5.3) clearly shows the problems that arise if the PES cannot be approximated accurately enough by the machine learning potential. In such a case, the predicted MEP starts to oscillate around the true MEP without reaching convergence.

## 5.5. Discussion

Before starting with the actual discussion of the ML potential performances, we would like to emphasize certain aspects that make the NEB-driven PES exploration differ significantly from typical applications of machine learning methods in computational chemistry. Standard ML potentials should reproduce certain properties of the PES, such as invariance to certain transformations and (infinite) differentiability. Transferability to other problems, for example, to molecular systems of varying size, is another prerequisite for most applications. However, these strong demands are not relevant for the task at hand. This is probably most obvious from the fact that even a Cartesian coordinate representation, where the degrees of freedom describing molecular motion are coupled in an unnecessarily complicated way and even basic invariance to translation and rotation is not given, still yields a reduction in computational effort. Leaving aside the question of whether other coordinate representations are better suited for the NEB algorithm itself,[393] this is a strong indication that, in principle, any representation and its corresponding machine learning potential seem suitable for this task. Therefore, of the three criteria mentioned above, only a repeated differentiability remains an important claim as forces need to be used as input and output data besides the energy.

We start with the discussion of the neural network approach, which proved difficult due to various reasons. Training of the neural network, a difficult optimization task by itself, tends to get stuck in local minima of the loss function, and an automatization of the process is not straightforward. Another complication is the determination of the optimal loss function hyperparameters $\beta$ and $\lambda$, a necessary prerequisite before running the actual ML-NEB algorithm. In our treatments of both test systems, we fixed the value for $\beta = 1$ and chose small regularization parameters of $\lambda = 10^{-4}$ and $\lambda = 10^{-6}$, for the ethane and the $Pt_4^-CO_2$ systems, respectively. An adjustment to the complexity of the model is achieved by a tuning of the neural network architecture.

While the latter had been kept fixed in the first task for the sake of a direct comparison to the work of Peterson,[186] the influence of the architecture on the algorithm performance has been investigated for the second problem, that is, the $CO_2$ activation, in the course of a small separate study (please see Section D.4 of the supporting material for details).

In direct comparison, the Cartesian GPR approach stands out by its simple and easy to implement formalism. Another advantage is the avoidance of any additional assumptions on the mathematical form of the PES, for example, it being a sum of atomic energies. Computational savings could probably be increased even further by tuning various details of the algorithm. Approaches presented by Denzel and Kästner[163] are overshooting in separate dimensions, decreasing the length scale according to a fixed shrinking scheme and nesting of multiple GPRs in a multilevel ansatz. While Cartesian coordinates are the simplest possible representation of molecular geometries, extrema on the PES are located in delocalized internal coordinates[329] by most *ab initio* packages. Constructing the GPR model in these coordinates could improve the performance of the algorithm as the invariance with respect to translation and rotation is encoded in the coordinate transformation.

Regarding the GAP approach, we investigated the performance of the dot product kernel because the corresponding feature space is finite and easy to construct. This opens the possibility of performing the costly matrix inversion (Cholesky decomposition) in a possibly smaller feature space. Details of this idea can be found in Section D.6 of the supporting material. Given the relatively small dimension $H = 50$ of our descriptor set, in the case of $\xi = 2$, it would be computationally less expensive to determine the weights directly in feature space after 12 steps of the ML-NEB algorithm for the ethane system. The descriptor space grows rapidly with the number of different elements. For the $Pt_4^-CO_2$ benchmark system with its $H = 93$ descriptors, performing the matrix inversion in feature space only becomes favorable after 87 ML-NEB iterations. The dimension of the feature space increases similarly fast for higher values of $\xi$, which rules out this possibility for $\xi = 4$ or $\xi = 6$.

Common to both the Cartesian GPR and the GAP is that hyperparameter optimization at each step is the bottleneck of the fitting procedure. The optimization is computationally demanding and can lead to major instabilities if a new minimum is found for the set of hyperparameters since large parts of the PES become ill represented. Since the hyperparameters do not change significantly after the first few iterations, as can be seen in Figure D.2 (top panel) of the supporting material, we suggest fixing them after a certain number of steps (10 steps in our case). This reduces the computational effort of the fitting without drastically increasing the number of *ab initio* evaluations needed. Figure D.2 (lower panel) of the supporting material illustrates the convergence behavior for Cartesian GPR and GAP (similar to Figure 5.3) when the hyperparameters are optimized after each step. Fixing the hyperparameter bears the additional advantage that only the part of the kernel matrix corresponding to new training data needs to be calculated after each iteration. Remarkably, even the Cholesky decomposition, used here to solve the system of linear equations, can be accelerated by this measure, since a decomposition into upper and lower triangular matrices is not perturbed by the extra rows and columns added per optimization step. This makes the algorithm effectively scale as $\mathcal{O}(M^2)$ instead of $\mathcal{O}(M^3)$, with $M$ denoting the number of training examples. A graphical illustration of this finding is provided in Section D.7 of the supporting material.

Finally, we would like to mention another variant of machine learning NEB, the one-image-evaluated algorithm (OIE) as suggested by Koistinen et al.,[187] in which only one image (typically the one with the highest prediction uncertainty) is evaluated and added to the training set before refitting the machine learning potential. All methods discussed here offer approaches to estimate the uncertainty of the model prediction at each point and could therefore be used in combination with the OIE algorithm.

## 5.6. Conclusion

We compared three machine learning-based potential energy approximations, neural network potentials, Gaussian process regression in Cartesian coordinates, and Gaussian approximation potentials, for their ability to accelerate the nudged elastic band method, a standard algorithm for reaction path evaluations in computational chemistry. Their performance has been tested on two molecular systems, one chosen for the sake of a direct comparison to the literature (ethane internal rotation) and the other one in order to provide a more realistic evaluation on a system of current chemical interest ($CO_2$ activation on $Pt_4^-$).

All methods provide a significant speedup (factors between 2 and 4) of band convergence at a computational effort that is mostly negligible in comparison to the costs for a typical single-point evaluation on the *ab initio* surface of any molecular system of chemical interest, so there is not a single reason to remain with the unmodified standard algorithm.

The simplicity and relatively high symmetry of the PES for ethane rotation disqualifies this system for an unbiased comparison among machine learning enhanced NEB calculations, since descriptor-based methods have a certain advantage in such a case. Also, the number of evaluations needed is very low in all cases (ranging from 4 to 10 in comparison to 21 in standard-NEB) and therefore not representative. However, on the less symmetric and more challenging $Pt_4^-/CO_2$ system, it became clear that neural network-based ML-NEB falls behind the two other approaches in several ways. Besides providing the lowest speedup (a factor of 2 on average), its performance is strongly fluctuating due to the random initialization of its weights. Another negative aspect to be mentioned is its complicated implementation in comparison to the other methods, and its sensitivity to the many parameters that have to be selected and tested by hand. Cartesian Gaussian process regression and Gaussian approximation potentials on the other side, with the latter being a combination of a descriptor-based approach with a kernel method, are comparably easy and straightforward in their implementation. In particular, direct approximation of the PES using Gaussian process regression (providing a factor of 3) stands out in this regard due to its unbeatable simplicity but also because this method is even working in an implementation employing Cartesian coordinates. The latter are typically avoided in any kind of geometry optimization due to their inherent strong coupling with respect to molecular motion and the resulting computational difficulties. This finding suggests an implementation, for example, in delocalized internal coordinates, a standard of many optimizers in quantum chemistry packages, as the most reasonable next step to take in order to make machine learning enhanced NEB the new standard. Another positive feature is the avoidance of any additional assumptions, such as the somewhat unphysical summation over atomic contributions to obtain the total energy.

This being said, we would like to mention that our newly introduced ansatz featuring Gaussian approximation potentials showed the best performance in our study (acceleration by a factor of 4) but comes at the cost of having to choose a suitable set of descriptors and might be more difficult to generalize in order to be applicable to arbitrary systems. Despite the potential for further speedups, we therefore recommend Gaussian process regression as the most convenient machine learning extension of the NEB algorithm for a widespread use in computational chemistry packages.

## 5.7. Acknowledgments

Financial support by the Austrian Science Fund (FWF) under Grant P29893 is gratefully acknowledged. We further thank the Central Informatics Service (ZID) of the Graz University of Technology and the Vienna Scientific Cluster (VSC) for providing high performance computing resources and technical support.

# 6. Machine learning approaches toward orbital-free density functional theory: Simultaneous training on the kinetic energy density functional and its functional derivative

This chapter corresponds to the publication
"Machine Learning Approaches toward Orbital-free Density Functional Theory: Simultaneous Training on the Kinetic Energy Density Functional and Its Functional Derivative" by Ralf Meyer, Manuel Weichselbaum, and Andreas W. Hauser, in *Journal of Chemical Theory and Computation*, **16** (9), 5685-5694 (2020).

Contributions:

- Ralf Meyer implemented and trained the kernel ridge regression model, evaluated the performance for all presented models, and wrote the first draft of the manuscript.
- Manuel Weichselbaum generated the data sets, implemented and trained the neural network models, and contributed to the final manuscript.
- Andreas W. Hauser supervised the method development, contributed to the manuscript, and provided the funding.

## 6.1. Abstract

Orbital-free approaches might offer a way to boost the applicability of density functional theory by orders of magnitude in system size. An important ingredient for this endeavor is the kinetic energy density functional. Snyder et al. [*Phys. Rev. Lett.* **2012**, *108*, 253002] presented a machine learning approximation for this functional achieving chemical accuracy on a one-dimensional model system. However, a poor performance with respect to the functional derivative, a crucial element in iterative energy minimization procedures, enforced the application of a computationally expensive projection method. In this work we circumvent this issue by including the functional derivative into the training of various machine learning models. Besides kernel ridge regression, the original method of choice, we also test the performance of convolutional neural network techniques borrowed from the field of image recognition.

## 6.2. Introduction

Over past decades density functional theory (DFT) has evolved into a powerful standard tool of computational chemistry.[394,395] Although originally intended as an orbital-free ansatz, where all contributions to the electronic energy of a system are represented by functionals of the electron density, the reintroduction of orbitals within the Kohn-Sham framework is a *de facto* standard of most modern programs.[55,396] The crucial term which triggered this development is the expression of the kinetic energy for a system of interacting fermions, which is much better covered within the picture of occupied molecular orbitals, i.e. eigenfunctions of an effective one-electron operator in a mean-field approximation. In modern functionals, the small deviations from the true kinetic energy are compensated by approximative functional expressions for the exchange and correlation interactions of an $N$-electron system. Although the local density approximation (LDA) of Kohn and Sham[396] is uniquely defined by the properties of the uniform gas, the strategy for further refinements is not clear at all.

Among the most successful current approaches for kinetic energy density functionals (KEDFs) is the class of nonlocal functionals, which are built from three parts,

$$T[n] = T^{\mathrm{TF}} + T^{\mathrm{vW}} + T^{\mathrm{NL}}, \tag{6.1}$$

with $T^{\mathrm{TF}} = C_{\mathrm{TF}} \int n^{5/3}\ \mathrm{d}\mathbf{r}$ as the Thomas-Fermi functional,[82–84] $T^{\mathrm{vW}} = \frac{1}{8} \int |\nabla n|^2/n\ \mathrm{d}\mathbf{r}$ as the semilocal von Weizsäcker functional[85] and $T^{\mathrm{NL}}$ as an additional nonlocal term. A widely used ansatz for the nonlocal part is of the form

$$T^{\mathrm{NL}} = C \int \int n(\mathbf{r})^{\alpha} \omega[n](\mathbf{r}, \mathbf{r}') n(\mathbf{r}')^{\beta}\ \mathrm{d}\mathbf{r}\ \mathrm{d}\mathbf{r}', \tag{6.2}$$

with $\omega$ denoting a dimensionless kernel, typically assumed to be a function of $|\mathbf{r} - \mathbf{r}'|$, and the exponents $\alpha$ and $\beta$ as parameters. This form encompasses state-of-the-art non-local functionals such as the Wang-Teter,[397] Smargiassi-Madden,[398] Perrot,[399] Wang-Govind-Carter,[400,401] Huang-Carter[402] and Mi-Genova-Pavanello[403] functionals. With varying degrees of success, these functionals have been applied to metallic and semiconducting bulk systems containing up to 1 million atoms,[404–407] to metallic clusters[408–410] and to molecular systems.[411,412]

Following Ref. 89, the idea of using machine learning (ML) methods to approximate density functionals has been investigated by several groups recently. The original ML model for the

KEDF has been shown to successfully describe bond breaking,[90] and was extended to include basis set independence[413] as well as scale-invariance conditions.[93] The same ML model has also been employed for direct fits of $F[n]$, the universal part of the total energy density functional.[91] A very interesting ML model was investigated by Yao and Parkhill, who used a 1D convolutional neural network to fit the kinetic energy as a function of the density projected onto bond directions.[98] Machine learning approximations, in particular neural networks, have also been suggested for semilocal KEDFs.[99–101] However, all of these ML-based KEDFs were deemed to be inadequate for an application in iterative calculations of the minimum energy density, mostly due to large errors on the predicted functional derivative. As a consequence, the focus has shifted toward a direct prediction of the minimum energy density from the nuclear potential, thereby bypassing the need for iterative calculations.[95,414–418]

One of the earliest machine learning models for density functionals was presented by Tozer et al. for the exchange-correlation (XC) functional.[68] In addition to explorations of the XC functional,[74,80,81,419] machine learning approximations have also been applied to other technicalities of DFT.[79,420–422]

In this article, we follow up on the first tests by Snyder et al.[89] and investigate if the original idea of learning the kinetic energy functional for a usage in iterative calculations can be "salvaged" by a simultaneous training of the machine learning model on both the kinetic energy functional and its functional derivative. In addition to the application of kernel ridge regression we evaluate the performance of convolutional neural networks, one of the most successful and widely used ML architectures to date. Our approach is motivated by the fact that the underlying mathematical expression is very similar to the nonlocal contribution given by Equation 6.2 (if the kernel $\omega$ is assumed to be a function of $|\mathbf{r}-\mathbf{r}'|$) and shows translational invariance, which might enable a better generalization, especially for large systems.

## 6.3. Methods

### 6.3.1. Data generation

A one-dimensional model system of noninteracting spinless Fermions is used to train and test the ML KEDFs. It consists of $N$ particles in a hard wall box within the interval $0 \leq x \leq 1$ and an external potential built from a linear combination of three Gaussians,[89]

$$V(x) = -\sum_{i=1}^{3} a_i \exp\left(-\frac{(x-b_i)^2}{2{c_i}^2}\right), \tag{6.3}$$

with parameters $a$, $b$, and $c$ randomly sampled from uniform distributions in the intervals $[1, 10]$, $[0.4, 0.6]$ and $[0.03, 0.1]$, respectively. The 1D Schrödinger equation for these potentials is solved on a grid of $G = 500$ points using Numerov's method,[423] yielding a set of eigenfunctions $\psi_j^k(x)$ and corresponding eigenvalues $E_j^k$ for each potential $V_j(x)$, ordered from lowest to highest energy with increasing index $k$. These solutions are then used to calculate all components of the training data for an $N$-particle system, namely the density,

$$n_j(x) = \sum_{k}^{N} \left(\psi_j^k(x)\right)^2, \tag{6.4}$$

the kinetic energy density, here defined as

$$\tau_j(x) = \frac{1}{2}\sum_{k=1}^{N}\left(\nabla\psi_j^k(x)\right)^2, \tag{6.5}$$

the kinetic energy

$$T_j = \int_0^1 \tau_j(x)\ \mathrm{d}x, \tag{6.6}$$

and the kinetic energy functional derivative

$$\frac{\delta T[n_j]}{\delta n_j(x)} = \mu_j - V_j(x), \tag{6.7}$$

with $\mu_j = \sum_k^N E_j^k/N$ denoting the total energy per particle. The discretized version of these functions, written as vectors for clarity $n_j(x) \to \mathbf{n}_j$, $\tau_j(x) \to \boldsymbol{\tau}_j$ and $\delta T[n_j]/\delta n_j(x) \to \nabla_{\mathbf{n}_j} T_j/\Delta x$, are used to train the ML models. The error in the eigenenergies due to discretization is estimated to be below $10^{-3}$ kcal/mol by comparing the solutions to calculations on a 10 times finer grid. However, in the context of the ML models all of the computed quantities are considered exact. The $M = 100$ parameter triplets for $a$, $b$ and $c$ as listed in the Supporting Material of Ref. 89 are used as training data in order to recreate the original study as closely as possible. On the basis of Equation 6.3, we generate 1000 additional random potentials as a test set.

### 6.3.2. Kernel ridge regression

We start with a brief review of the kernel ridge regression (KRR) approach as introduced in Ref. 89. This ansatz is then extended by the inclusion of the functional derivative into the training in order to improve its capabilities. Details on the derivation of the equations used in the following can be found in Section E.2 of the supporting material. An elaborate discussion of KRR model training can be found in Ref. 117.

In KRR the simple regularized linear fit of ridge regression is extended toward nonlinear data through the introduction of a kernel function:

$$T^{\mathrm{ML}}(\mathbf{n}) = \sum_j^M \alpha_j k(\mathbf{n}_j, \mathbf{n}), \tag{6.8}$$

with $\alpha_j$ as the fit coefficients, the kernel function $k(\mathbf{n}_i, \mathbf{n}_j)$, which can be interpreted as a measure of similarity between two densities, and with $\{\mathbf{n}_1, \ldots, \mathbf{n}_M\}$ as the $M$ training examples. The coefficients $\alpha_j$ are determined by minimizing the cost function

$$\mathcal{L} = \sum_j^M \left\|T^{\mathrm{ML}}(\mathbf{n}_j) - T_j\right\|^2 + \lambda\sum_{i,j}^M \alpha_i k(\mathbf{n}_i, \mathbf{n}_j)\alpha_j, \tag{6.9}$$

where the second term is a regularization function scaled by the parameter $\lambda$ and $T_j$ are the kinetic energies corresponding to the training densities $\mathbf{n}_j$. Setting the derivative with respect to the $\alpha_j$ equal to zero yields a matrix equation for the fit coefficients,

$$\begin{pmatrix}\alpha_1\\ \vdots\\ \alpha_M\end{pmatrix} = (K + \lambda\mathbf{I}_M)^{-1}\begin{pmatrix}T_1\\ \vdots\\ T_M\end{pmatrix}. \tag{6.10}$$

The matrix $K$ contains the values of the kernel function for the $M$ training examples, so $K_{ij} = k(\mathbf{n}_i, \mathbf{n}_j)$ and $\mathbf{I}_M$ is a unit matrix of size $M$.

Snyder et al.[89] have shown already that the discretized functional derivative can easily be calculated from Equation 6.8, yielding

$$\frac{\nabla_{\mathbf{n}} T^{\mathrm{ML}}(\mathbf{n})}{\Delta x} = \sum_j^M \frac{\alpha_j}{\Delta x} \nabla_{\mathbf{n}} k(\mathbf{n}_j, \mathbf{n}). \tag{6.11}$$

In our study, we expand on this idea by including the functional derivatives of the training examples into the model using additional fit coefficients $\boldsymbol{\beta}_j$. The kinetic energy of the extended model is then given by

$$T^{\mathrm{ML}}(\mathbf{n}) = \sum_j^M \alpha_j k(\mathbf{n}_j, \mathbf{n}) + \sum_j^M \frac{\boldsymbol{\beta}_j^\top \cdot \nabla_{\mathbf{n}_j} k(\mathbf{n}_j, \mathbf{n})}{\Delta x}. \tag{6.12}$$

Derivation with respect to the input density gives the new formula for the functional derivative:

$$\frac{\nabla_{\mathbf{n}}^\top T^{\mathrm{ML}}(\mathbf{n})}{\Delta x} = \sum_j^M \frac{\alpha_j}{\Delta x} \nabla_{\mathbf{n}}^\top k(\mathbf{n}_j, \mathbf{n}) + \sum_j^M \frac{\boldsymbol{\beta}_j^\top \cdot \left(\nabla_{\mathbf{n}_j} \cdot \nabla_{\mathbf{n}}^\top k(\mathbf{n}_j, \mathbf{n})\right)}{(\Delta x)^2}. \tag{6.13}$$

Note that each of the newly introduced coefficients $\boldsymbol{\beta}_j$ is a vector of size $G$. Therefore, the number of parameters grows from $M$ to $M(1+G)$. The cost function is extended by the squared error of the functional derivative and an additional regularization term for the new weights $\boldsymbol{\beta}_j$,

$$\begin{aligned}\mathcal{L} =& \sum_j^M \left\| T^{\mathrm{ML}}(\mathbf{n}_j) - T_j \right\|^2 + \frac{\kappa}{G} \sum_j^M \left\| \frac{\nabla_{\mathbf{n}_j} T^{\mathrm{ML}}(\mathbf{n}_j)}{\Delta x} - \frac{\nabla_{\mathbf{n}_j} T_j}{\Delta x} \right\|^2 \\ &+ \lambda \sum_{i,j}^M \left( \alpha_i K_{ij} \alpha_j + \frac{\boldsymbol{\beta}_i^\top \cdot \left(\nabla_{\mathbf{n}_i} \cdot \nabla_{\mathbf{n}_j}^\top k(\mathbf{n}_i, \mathbf{n}_j)\right) \cdot \boldsymbol{\beta}_j}{(\Delta x)^2} \right),\end{aligned} \tag{6.14}$$

with $\nabla_{\mathbf{n}_j} T_j / \Delta x$ denoting the reference value for the discretized functional derivative corresponding to the training density $\mathbf{n}_j$. Minimizing this extended cost function with respect to the coefficients yields

$$\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_M \\ \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_M \end{pmatrix} = (K_{\mathrm{ext}} + \Lambda)^{-1} \begin{pmatrix} T_1 \\ \vdots \\ T_M \\ \frac{\nabla_{\mathbf{n}_1} T_1}{\Delta x} \\ \vdots \\ \frac{\nabla_{\mathbf{n}_M} T_M}{\Delta x} \end{pmatrix}, \tag{6.15}$$

with an extended regularization matrix

$$\Lambda = \begin{pmatrix} \lambda \mathbf{I}_M & 0 \\ 0 & \frac{\lambda G}{\kappa} \mathbf{I}_{MG} \end{pmatrix}, \tag{6.16}$$

where $\mathbf{I}_M$ and $\mathbf{I}_{MG}$ are unit matrices of size $M$ and $MG$, respectively, and an extended kernel matrix

$$K_{\mathrm{ext}} = \begin{pmatrix} K & J' \\ J & H \end{pmatrix}, \tag{6.17}$$

where $K$ is a $(M \times M)$ matrix with elements $K_{ij} = k(\mathbf{n}_i, \mathbf{n}_j)$, $J$ and $J'$ are matrices of size $(MG \times M)$ and $(M \times MG)$, respectively, and contain the gradient vectors of $k(\mathbf{n}_i, \mathbf{n}_j)$ with respect to the input densities $J_{ij} = \frac{1}{\Delta x}\nabla_{\mathbf{n}_i} k(\mathbf{n}_i, \mathbf{n}_j)$ and $J'_{ij} = \frac{1}{\Delta x}\nabla^\top_{\mathbf{n}_j} k(\mathbf{n}_i, \mathbf{n}_j)$. Finally, $H$ is a $(MG \times MG)$ blocked matrix consisting of the Hessian matrices of $k(\mathbf{n}_i, \mathbf{n}_j)$ given by $H_{ij} = \frac{1}{(\Delta x)^2}\nabla_{\mathbf{n}_i} \cdot \nabla^\top_{\mathbf{n}_j} k(\mathbf{n}_i, \mathbf{n}_j)$.

Following Ref. 89, we use the squared exponential kernel for all of the presented KRR models,

$$k(\mathbf{n}_i, \mathbf{n}_j) = \exp\left(-\frac{1}{2}\frac{\|\mathbf{n}_i - \mathbf{n}_j\|^2}{\sigma^2}\right), \tag{6.18}$$

where the hyperparameter $\sigma$ denotes the length scale on which the training densities vary.

### 6.3.3. Convolutional neural networks

Convolutional neural networks[110,111] (CNNs) are on the forefront of the ongoing deep learning revolution[424,425] and achieve unprecedented accuracy in their main field of application, image recognition.[426] CNNs represent a subclass of standard feed forward neural networks, designed for the specific purpose of an efficient inclusion of spatial information in pixel-based image processing. Despite their origin in visual pattern recognition, CNNs have been applied successfully to numerous other tasks, including also the approximation of density functionals.[80,98,420]

A single convolutional layer typically consists of several filters or steps of input processing. A pass through a single convolutional filter in one dimension is given by

$$\mathbf{z}_{(g)} = f\left(b + \sum_{q=1}^{\sigma_w} \mathbf{n}_{(q)} \mathbf{w}_{(g-q)}\right), \tag{6.19}$$

where the index "$(g)$" refers to the $g$th element of a vector (parentheses are used to distinguish grid point indices from training example indices), $f$ is an activation function, $b$ is a bias parameter, $\mathbf{w}$ is a vector of weight parameters for the filter (commonly referred to as "kernel"), and $\sigma_w$ is the filter width. Equation 6.19 is only valid for indices $(g)$ where the input and the convolutional kernel $\mathbf{w}$ fully overlap (referred to as "valid padding"). The resulting output vector $\mathbf{z}$ is therefore smaller than the input. Alternatively, the input vector can be padded with zeros to ensure that the output is of the same size as the input, a technique referred to as "same padding".

In the course of this article we investigate the performance of both a standard CNN and a residual neural network (ResNet),[427] with the latter referring to a network featuring a more sophisticated architecture: In addition to conventional convolutional layers, ResNets use so-called skip connections through which the feed-forward signal can bypass several layers and is directly added to the output of a later layer. Connections of this type are known to improve the training process, in particular if training data is limited, as they are forming a less complicated, "coarse" network within the actual network structure.

Both investigated models use 32 filters per convolutional layer with a filter width of $\sigma_w = 100$ and employing the softplus activation function.[428,429] The standard CNN consists of five convolutional layers using valid padding. This results in an flattened output vector containing 160 entries, which is then reduced to a single scalar, the kinetic energy prediction, using a weighted sum, referred to as “linear dense layer” in community parlance. The more complicated ResNet model consists of three blocks of two convolutional layers. Each of these blocks is bypassed by a skip connection. The three blocks are followed by a final convolutional layer with a single filter. In order to allow for skip connections all convolutions employ same padding. This architecture results in an output vector of the same size as the input density, which is interpreted as kinetic energy density. Finally, the kinetic energy (see Equation 6.6) is calculated by integrating over the output using the trapezoidal rule. The batch normalization layers[430] typically employed in ResNets worsen the training performance in regression tasks and are therefore not used. Schematics of both models are presented in Figure 6.1. We use the keras[318] and tensorflow[319] Python packages to implement and train both types of neural networks.

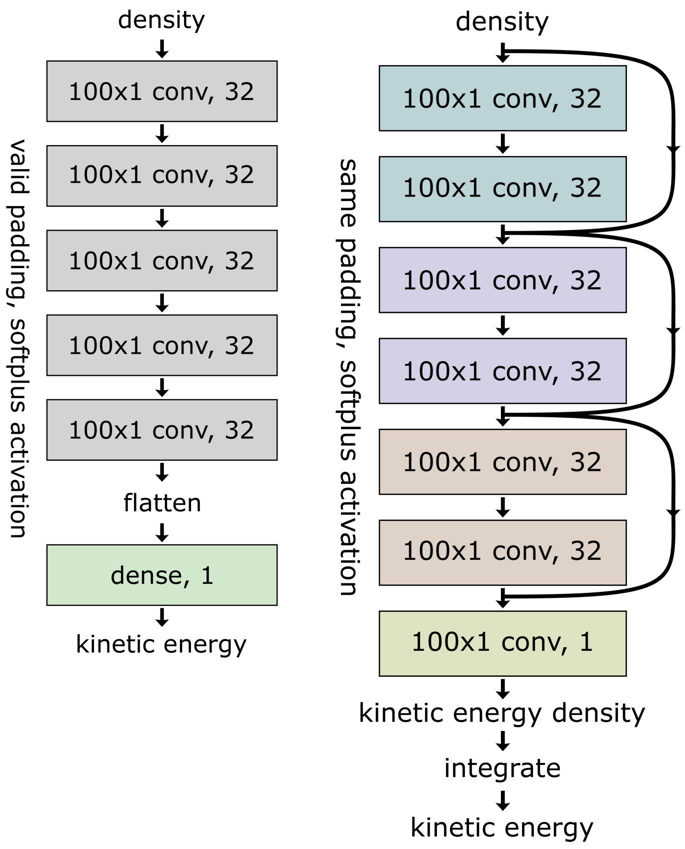


Figure 6.1.: Schematic depiction of the NN architectures used for the standard CNN (left) and the ResNet model (right). Note the appearance of skip connections for the latter.

The bias and weight parameters are determined by minimizing a cost function similar to Equation 6.14. Since the ResNet model offers predictions of the kinetic energy densities $\boldsymbol{\tau}_j$, an additional error term can be added to the cost function,

$$\begin{aligned}\mathcal{L} = \quad & \frac{\iota_T}{M}\sum_j^M ||T^{\mathrm{ML}}(\mathbf{n}_j) - T_j||^2 + \frac{\iota_\tau}{MG}\sum_j^M ||\boldsymbol{\tau}^{\mathrm{ML}}(\mathbf{n}_j) - \boldsymbol{\tau}_j||^2 \\ & + \frac{\kappa}{MG}\sum_j^M \left\| \frac{\nabla_{\mathbf{n}_j} T^{\mathrm{ML}}(\mathbf{n}_j)}{\Delta x} - \frac{\nabla_{\mathbf{n}_j} T_j}{\Delta x} \right\|^2 + \lambda||\mathbf{w}||^2, \end{aligned} \tag{6.20}$$

where the weighting coefficients are set to $\iota_T = 0.2$, $\iota_\tau = 0$, $\kappa = 1$, and $\lambda = 2.5 \cdot 10^{-4}$ for the standard CNN (since it does not predict the kinetic energy density) and to $\iota_T = 0$, $\iota_\tau = 1$, $\kappa = 1$, and $\lambda = 2.5 \cdot 10^{-4}$ for the ResNet. The $L_2$-regularization term is applied to weight parameters exclusively and not to bias parameters.

The network parameters are initialized randomly according to the "Glorot uniform" tensorflow method[321,322] and trained using the Adam optimizer[320] for 100 000 epochs. We use a two-stage learning rate schedule, where the learning rate stays constant at $10^{-4}$ for the first 21 800 epochs and is lowered by 10% every 1000 epochs for the remaining training procedure, resulting in an exponential decay. This greatly improves the overall convergence, as the inclusion of derivative information leads to large variations of the cost function during the training. A more detailed discussion of the training procedure as well as its convergence behavior is given in Section E.4 of the supporting material.

## 6.4. Results and discussion

Each of the following investigations can be split into two different parts with respect to their objectives. In the first part, the model performance is tested by using the exact densities of the test set as input to the ML models and evaluating the error of both the kinetic energy and the functional derivative. In the second part, the derivative prediction of the ML models is used to iteratively find the minimum energy density for the potentials of the test set. For these densities, the error of the kinetic energy is reported together with the deviation from the exact minimum energy density. This way, the impact and the magnitude of both types of error contributions, one stemming from the model itself, and the other caused by wrong minimum energy density predictions, should become clear and traceable for the reader.

### 6.4.1. Training on the functional derivative

As a first test we investigate if the inclusion of derivative information can improve the fit quality of the machine learning models on the data sets for $N = 1$. Table 6.1 summarizes the mean value, the standard deviation and the maximum value of both the absolute error of the kinetic energy $|\Delta T|$ and the integral over the absolute error of the functional derivative $|\Delta \frac{\delta T}{\delta n}|$ for all of the investigated models.

As a reference, we reproduce the KRR results from Ref. 89 by using the reported hyperparameters ($\sigma = 43$ and $\lambda = 12 \cdot 10^{-14}$). Slight deviations between our results and the previous work in Table 6.1 can be attributed to the fact that we use a different randomly generated test set. The hyperparameters for the extended KRR model including derivative information (referred to as "ext. KRR" in Table 6.1) are determined using a rough grid search and 5-fold cross validation. The minimum of the sum of the mean absolute validation errors for kinetic energy and functional derivative is obtained for $\sigma = 30.58$ and $\lambda = 10^{-12}$. The influence

Table 6.1.: Absolute error values on the $N = 1$ test set for all of the machine learning models given in kcal/mol.

| | $\lvert\Delta T\rvert$ | | | $\lvert\Delta \frac{\delta T}{\delta n}\rvert$ | | |
|---|---|---|---|---|---|---|
| model | mean | std | max | mean | std | max |
| KRR Ref. 89 | 0.15 | 0.24 | 3.2 | – | – | – |
| KRR this work | 0.163 | 0.29 | 4.6 | 29313.2 | 345.5 | 30610.9 |
| ext. KRR | 0.004 | 0.02 | 0.6 | 3.4 | 4.3 | 50.7 |
| CNN | 0.044 | 0.10 | 2.3 | 31.5 | 25.0 | 370.1 |
| ResNet | 0.015 | 0.02 | 0.3 | 10.1 | 7.0 | 110.7 |

of the weighting parameter $\kappa$, which is set to 1, as well as a more detailed description of the hyperparameter search is given in Section E.3 of the supporting material. Comparison of the

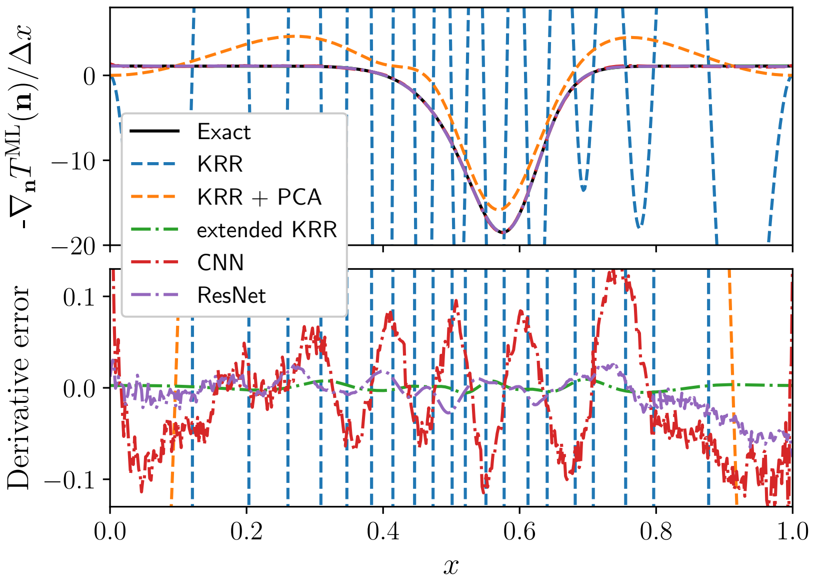


Figure 6.2.: Comparison of exact functional derivative (solid lines) and predictions by standard KRR with and without the PCA projection detailed in Section 6.4.2 (dashed lines) as well as prediction from the ML models trained on derivative information (dashed-dotted lines). The parameters for the shown potential are $a = \{4.43, 7.18, 9.03\}$, $b = \{0.0532, 0.587, 0.568\}$ and $c = \{0.0754, 0.0406, 0.0554\}$.

two KRR models shows that the inclusion of derivative information into the KRR approach not only drastically reduces the error on the functional derivative, as illustrated for a sample potential in Figure 6.2, but also improves the accuracy of the kinetic energy prediction by reducing the standard deviation and the maximum of the absolute error. The slight increase in the mean kinetic energy error originates from a trade-off in accuracy between kinetic energy and its derivative. Section E.3 of the supporting material shows that the cross validation error of the kinetic energy is actually lowest for the hyperparameter values $\sigma = 11.50$ and $\lambda = 10^{-14}$, whereas the lowest error on the functional derivative is obtained for $\sigma = 30.58$ and $\lambda = 10^{-12}$.

Both the simpler CNN as well as the more sophisticated ResNet achieve lower mean absolute errors for both the kinetic energy and its derivative than the standard KRR model. We attribute the better performance of the extended KRR method to a lack of smoothness exhibited by the neural network models as shown in the bottom panel of Figure 6.2.

### 6.4.2. Finding minimum energy densities using principal component analysis

In the next step we address the question of applicability with respect to a direct minimization of kinetic energy density. As will be shown in the following, an unconstrained search still remains impossible despite the drastic improvements in the prediction accuracy of the functional derivative.

The reason for this failure lies in the notoriously noisy nature of machine learning approximations. Already emphasized in Ref. 89, this was discussed in greater detail in a follow-up investigation on nonlinear gradient denoising.[94] For reasons of comparability, we use the same local principal component analysis (PCA) approach as was introduced in the original publication as an *ad hoc* remedy and investigate if the improved accuracy of the models trained on the functional derivative translates to lower errors on the iteratively found densities. Additionally, we test if the PCA search space can be increased for these models.

Starting from the average density of all training examples, $\mathbf{n}^{(0)} = \frac{1}{M}\sum_j \mathbf{n}_j$, the minimum energy density is found by simple gradient descent,

$$\mathbf{n}^{(j+1)} = \mathbf{n}^{(j)} - \eta \mathbf{P}_{m,l}(\mathbf{n}^{(j)}) \left[ \mathbf{V} + \frac{\nabla_{\mathbf{n}} T^{\mathrm{ML}}(\mathbf{n}^{(j)})}{\Delta x} \right], \tag{6.21}$$

where the projection matrix $P_{m,l}(\mathbf{n})$ is acting on the functional derivative and constraining the search space, $\eta$ is the step size and $\mathbf{V}$ is the discretized potential. We note that more sophisticated optimization methods such as conjugate-gradient are known to significantly accelerate the convergence,[431] but this is not relevant for the intended comparison. For a given density $\mathbf{n}$ the local PCA algorithm starts by calculating the difference matrix $X^\top = (\mathbf{n}_{j_1} - \mathbf{n}, \ldots, \mathbf{n}_{j_m} - \mathbf{n})$ for the $m$ closest training densities and diagonalizing the covariance matrix $C = X^\top X/m$. The projection matrix is then constructed from the eigenvectors $\mathbf{w}_k$ corresponding to the $l$ largest magnitude eigenvalues $P_{m,l}(\mathbf{n}) = \sum_{k=1}^{l} \mathbf{w}_k \cdot \mathbf{w}_k^\top$. Figure 6.2 shows the effect of this projection on the functional derivative prediction made by the standard KRR model with parameters $m = 30$ and $l = 5$. For all calculations presented in this article we keep $m = 30$, but vary the size of the search space via the parameter $l$. Section S7 of the Supporting Information shows the effect of the projection on the functional derivative prediction for different values of $l$. The iterative minimization algorithm is considered converged once the integral over the absolute projected functional derivative is smaller than $10^{-6}$ hartree/particle. We use a step size of $\eta = 10^{-3}$ and restrict the maximum number of iterations to 4000 cycles.

The results for all of the 1000 random potentials in the test set are summarized in Table 6.2. Again, the inclusion of derivative information reduces both errors significantly when compared to those obtained with the simple KRR. The final error can be attributed to two different sources. The first contribution stems from the model error due to the ML approximation as has already been discussed in Section 6.4.1. A second contribution arises due to the difference in the corresponding minimum energy densities $\Delta n$, which is in turn caused by the model error and the restriction of the search space in the PCA. This limited flexibility in the search for $l = 5$ is likely the cause of the similar error values achieved by all of the ML models using derivative

Table 6.2.: Absolute kinetic energy errors $\Delta T$ for the iteratively found densities in kcal/mol as well as the integrated absolute error of the densities $\Delta n$ for the $N = 1$ test set.

| | | $\|\Delta T\|$ | | | $\|\Delta n\| \cdot 10^4$ | | |
|---|---|---|---|---|---|---|---|
| model | $l$ | mean | std | max | mean | std | max |
| KRR Ref. 89 | 5 | 3.0 | 5.3 | 46 | – | – | – |
| KRR this work | 5 | 2.85 | 7.00 | 87.34 | 45.0 | 54.0 | 503.9 |
| ext. KRR | 5 | 0.46 | 1.05 | 15.35 | 14.9 | 11.5 | 85.0 |
| ext. KRR | 10 | 0.04 | 0.22 | 5.95 | 0.8 | 0.7 | 10.7 |
| ext. KRR | 15 | 0.04 | 0.22 | 5.97 | 0.3 | 0.5 | 10.8 |
| CNN | 5 | 0.57 | 1.40 | 21.69 | 15.2 | 12.0 | 95.5 |
| CNN | 10 | 0.29 | 0.77 | 13.26 | 5.5 | 8.5 | 171.0 |
| ResNet | 5 | 0.51 | 1.25 | 19.59 | 14.9 | 11.5 | 85.5 |
| ResNet | 10 | 0.09 | 0.21 | 5.72 | 1.0 | 0.9 | 14.7 |
| ResNet | 15 | 0.09 | 0.22 | 5.86 | 2.0 | 2.4 | 19.6 |

information. We therefore also investigated larger $l$ values while keeping $m = 30$. Runs using standard KRR fail for every value $l > 5$, not reaching convergence and predicting sharply peaked densities instead. Similarly, the performance of the simple CNN deteriorates quickly for $l > 10$. For the ResNet and extended KRR models the errors reduce up to an $l$ value of about 15, at which point the iterative algorithm leaves the valid region of the ML models. Note, however, that for $l$ values in the range between 10 and 15 the ML approximations including derivative information achieve errors an order of magnitude lower than standard KRR.

Alternatively, KRR can also be used in iterative calculations without the PCA by introducing a constant offset,

$$\tilde{T}^{\mathrm{ML}}(\mathbf{n}) = b + \sum_{j}^{M} \alpha_j k(\mathbf{n}_j, \mathbf{n}), \tag{6.22}$$

and using a small length scale hyperparameter $\sigma$ in the kernel function. This can be used to penalize densities far from the training data and effectively acts similarly to PCA, while using all of the training densities ($m = M$). This approach is inspired by the use of Gaussian process regression for molecular geometry optimization,[163, 164, 166] where a similar idea ensures that the iterative search does not stray too far from the training data. A more detailed explanation and results for the densities provided by this method can be found in Section E.6 the supporting material. However, both the need for local PCA and the alternative approach of small length scales and a constant offset are indications that the ML density functionals do not properly generalize and are only valid in close vicinity of the training examples.

### 6.4.3. Toward a real world use case

While Section 6.4.2 shows that models trained on the functional derivative allow for significantly larger search spaces, the iterations will, given enough flexibility, inevitably leave the region where the machine learning approximations are valid. This typically leads to sharply peaked or rapidly oscillating densities. A straightforward solution for this problem is to use physically motivated penalty terms for these unphysical densities such as the von Weizsäcker kinetic energy functional,[85] and to train the machine learning model on the difference between the exact kinetic energy and the von Weizsäcker model,

$$T^{\mathrm{ML}} = T - T^{\mathrm{vW}} = T - \int \frac{n'(x)^2}{8n(x)} \, \mathrm{d}x. \tag{6.23}$$

The prediction of a previously unseen density is then given by the sum of the machine learning and the von Weizsäcker model functionals, $T^{\mathrm{ML}}+T^{\mathrm{vW}}$, where the derivative term $n'(x) = dn/dx$ in the latter contribution is introducing an energy penalty for rapid changes in the density, which effectively restricts the search space to physically reasonable densities. Since the von Weizsäcker model already yields the exact solution for the case of a single spatial orbital (discussed in Section 6.4.1), we test this approach for two-particle densities instead. In our toy model, this already corresponds to the occupation of two spatial orbitals since there is no spin degree of freedom taken into consideration.

At first, we again investigate the model performance for fixed densities. The hyperparameters for both the standard and the extended KRR models are readjusted using the same grid search and 5-fold cross-validation as in Section 6.4.1. This yields $\sigma = 35.16$ and $\lambda = 10^{-12}$ for the standard KRR and $\sigma = 26.59$, $\lambda = 10^{-12}$, and $\kappa = 1.0$ for the extended KRR. More details on the hyperparameter search are provided in Section E.3 of the supporting material.

As can be seen in Table 6.3, all of the three investigated models achieve significantly better performance on the two-particle densities than on the single-particle densities. The analysis of the training sets in Section E.1 of the supporting material suggests that even though kinetic energies in the $N = 2$ data set are showing a larger variance, most of it can be captured by a simple linear model. We attribute this to the fact that the second particle is less influenced by the relatively shallow potentials and the more difficult to learn semilocal contribution is already covered by the von Weizsäcker functional. Even though chemical accuracy is achieved by all models on this less challenging data set, extended KRR yields a mean absolute error for the kinetic energy that is 2 orders of magnitude lower than that obtained with ResNet or standard KRR.

Table 6.3.: Error values of the machine learning approximations on the $N = 2$ test set in kcal/mol.

| | $\lvert\Delta T\rvert$ | | | $\lvert\Delta \frac{\delta T}{\delta n}\rvert$ | | |
|---|---|---|---|---|---|---|
| model | mean | std | max | mean | std | max |
| KRR | 0.0355 | 0.0588 | 0.8752 | 2957.85 | 16.00 | 2990.36 |
| ext. KRR | 0.0002 | 0.0008 | 0.0233 | 0.12 | 0.15 | 2.18 |
| ResNet | 0.0483 | 0.2837 | 6.8116 | 6.78 | 11.36 | 223.35 |

Regarding efficiency and feasibility, the local PCA introduces a significant computational overhead and would most likely prohibit a large scale application of ML density functionals to realistic problems. We therefore opt for a more traditional approach of using a basis of sine functions instead. This introduces the necessity of ensuring both the positivity and the proper normalization of the density throughout the iterative algorithm. While the correct norm can simply be enforced by a Lagrange multiplier, the positivity constraint is typically included by iterating on the variable $\boldsymbol{\varphi} = \sqrt{\mathbf{n}}$ instead of the density. The steepest descent update rule for this new variable is given by

$$\boldsymbol{\varphi}^{(j+1)} = \boldsymbol{\varphi}^{(j)} - \eta P_K \left[ 2\boldsymbol{\varphi}^{(j)} \left( \mathbf{V} + \frac{\nabla_{\mathbf{n}} T(\mathbf{n}^{(j)})}{\Delta x} - \mu \right) \right], \tag{6.24}$$

where $P_K$ is a projection matrix and $\mu$ is the Lagrange multiplier used to ensure the conservation of the number of particles $N$. A detailed derivation of this result, a possible way of determining $\mu$ as well as an explanation why this procedure is not necessary for the local PCA is provided in Section E.5 of the supporting material. The matrix used to project the functional derivative onto the basis of sine functions is constructed via

$$P_K = \frac{1}{\Delta x} \sum_{k=1}^{K} \mathbf{w}_k \cdot \mathbf{w}_k^\top, \tag{6.25}$$

with $\mathbf{w}_k = \sqrt{2}\sin(k\pi\mathbf{x})$. Note that this matrix does not need to be reconstructed in every iteration as it is no longer dependent on the density at step $j$. The size of the search space is now determined by the maximum wavenumber $K$. The starting value for the variable $\boldsymbol{\varphi}$ is the square root of the initial density $\mathbf{n}^{(0)}$ projected onto the basis of sine functions $\boldsymbol{\varphi}^{(0)} = P_K\left[\sqrt{\mathbf{n}^{(0)}}\right]$, where the starting density is again given by the average over the training data $\mathbf{n}^{(0)} = \frac{1}{M}\sum_j \mathbf{n}_j$. The maximum number of iterations is restricted to 4000 and calculations are considered converged once the integral over the absolute projected functional derivative drops below a threshold of $10^{-6}\ \sqrt{\text{hartree/particle}}$. Note that this differs from the convergence criterion in Section 6.4.2 because convergence is monitored using the functional derivative with respect to the variable $\boldsymbol{\varphi}$ instead of the density. The step size is reduced to $\eta = 10^{-4}$ in order to avoid oscillations in the convergence behavior.

Table 6.4.: Absolute kinetic energy error $\Delta T$ for the iteratively found densities in kcal/mol as well as the integrated absolute error of the densities $\Delta n$ on the $N = 2$ test set, compared between the KRR variants and ResNet for increased search spaces.

| | | $\lvert\Delta T\rvert$ | | | $\lvert\Delta n\rvert \cdot 10^4$ | | |
|---|---|---|---|---|---|---|---|
| model | $K$ | mean | std | max | mean | std | max |
| KRR | 10 | 8.431 | 1.138 | 16.686 | 109.0 | 23.0 | 222.8 |
| KRR | 20 | 21.365 | 0.826 | 22.456 | 133.9 | 18.4 | 194.9 |
| KRR | 40 | 23.882 | 0.846 | 25.003 | 139.8 | 18.4 | 200.5 |
| ext. KRR | 10 | 0.523 | 0.827 | 7.353 | 24.8 | 16.0 | 102.1 |
| ext. KRR | 20 | 0.074 | 0.069 | 0.789 | 0.5 | 0.6 | 2.9 |
| ext. KRR | 40 | 0.076 | 0.069 | 0.789 | 0.1 | 0.1 | 1.1 |
| ResNet | 10 | 1.239 | 6.537 | 142.649 | 25.8 | 18.6 | 248.9 |
| ResNet | 20 | 0.877 | 6.634 | 146.324 | 2.9 | 11.8 | 253.3 |
| ResNet | 40 | 0.877 | 6.635 | 146.327 | 2.7 | 11.8 | 253.3 |

Our results are summarized in Table 6.4. The large errors on the functional derivative of the standard KRR model lead to poor results for iteratively found densities. This is further emphasized by the steadily increasing error when the search space grows from $K = 10$ to $K = 20$ and $K = 40$. The iterative search is, however, stable even for the larger $K$ values due to the von Weizsäcker penalty term. Extended KRR clearly yields the best predictions for the minimum energy densities, while the ResNet approach barely manages to achieve a mean absolute error for the kinetic energy within chemical accuracy.

### 6.4.4. Larger training set

The previous sections are somewhat biased due to the small number of training examples - a regime where KRR excels. In a final test we therefore investigate the performance of the neural network-based density functional for an increasing number of training examples. A similar investigation for the extended KRR model is not feasible since the training effort for the KRR model scales with $\mathcal{O}(M^3)$ and the memory requirements grow with $\mathcal{O}(M^2)$. We note, however, that an alternative approach to reduce the computational effort would be to use sparse kernel-based machine learning algorithms such as support vector regression[432–434] or sparsified Gaussian process regression.[435–437] These methods could combine the high accuracy of kernel ridge regression even for small training sets with the potential of increasing the region where the model is valid by incorporating a significantly wider range of training examples.

The hyperparameters of the ResNet model and the training procedure have to be adjusted due to the increased number of examples. Instead of evaluating the cost function involving all of the training data (batch learning), a random subset or "batch" of 100 examples is used to calculate the gradient descent step and to update the NN parameters (i.e. minibatch learning). The models are trained for a total of 300 000 such iterations with a learning rate of $10^{-4}$ during the first 40 000 steps after which the learning rate is reduced by 10% every 2000 steps. In addition, the regularization factor $\lambda$ is lowered as well (see Section E.4 of the supporting material for details).

Table 6.5.: Absolute error values for the kinetic energy $\Delta T$ and its functional derivative (in kcal/mol) on the $N = 2$ test set, achieved by the ResNet model trained on sets of varying size.

| | $\lvert\Delta T\rvert$ | | | $\lvert\Delta \frac{\delta T}{\delta n}\rvert$ | | |
|---|---|---|---|---|---|---|
| M | mean | std | max | mean | std | max |
| 100 | 0.049 | 0.284 | 6.814 | 6.78 | 11.36 | 223.36 |
| 1000 | 0.012 | 0.063 | 1.922 | 3.12 | 2.81 | 66.31 |
| 10 000 | 0.007 | 0.018 | 0.528 | 2.79 | 1.92 | 45.19 |
| 100 000 | 0.007 | 0.009 | 0.138 | 2.39 | 1.36 | 19.40 |

Table 6.5 shows that the larger amount of training examples leads to a steady reduction in every error score. The most significant improvement is observed for the standard deviation and the maximum error, the metrics most closely related to the generalization properties of the model.

We use a basis of $K = 40$ sine functions for the iterative calculation of minimum energy densities. Instead of enforcing a convergence threshold, the calculations stop after a fixed number of 10 000 iterations for the sake of simplified parallelization on GPUs. Typically, the final error values are reached within the first 10% of the iterations. The large overhead in terms of iterations is used to investigate the numerical stability of the iterations on noisy ML predictions of the derivative.

The error scores on the iteratively found densities, summarized in Table 6.6, are clearly

Table 6.6.: Absolute kinetic energy error $\Delta T$ in kcal/mol as well as the integrated absolute error $\Delta n$ of the iteratively found densities on the $N = 2$ test set, employing the ResNet on training sets of increasing size.

| | $\|\Delta T\|$ | | | $\|\Delta n\| \cdot 10^4$ | | |
|---|---|---|---|---|---|---|
| M | mean | std | max | mean | std | max |
| 100 | 0.856 | 6.591 | 145.58 | 2.7 | 11.8 | 253.0 |
| 1000 | 0.151 | 1.196 | 32.65 | 0.7 | 1.9 | 50.5 |
| 10 000 | 0.062 | 0.228 | 6.48 | 0.6 | 0.6 | 13.7 |
| 100 000 | 0.047 | 0.070 | 0.89 | 0.6 | 0.5 | 5.5 |

improving with an increasing number of training examples. The ResNet trained on 100 000 densities achieves a performance similar to the extended KRR model trained on just 100 examples. While this may suggest that KRR should be the obvious choice, one has to keep in mind that the evaluation times for the ResNet model are independent of the number of training examples, and that training examples are typically available in abundance: In fact, every single step in a self-consistent Kohn-Sham DFT calculation could serve as training input.

## 6.5. Conclusion

The predictive capabilities of kernel ridge regression and convolutional neural networks, two well-established machine learning techniques, have been tested on a one-dimensional model system of noninteracting spinless fermions with respect to the kinetic energy and its functional derivative. Extending the work of Snyder et al.,[89] we have investigated if the original idea of learning the kinetic energy functional for usage in iterative calculations of minimum energy densities can be “salvaged” by a simultaneous training of machine learning models on both the kinetic energy functional and its functional derivative. Besides kernel ridge regression, the method of choice in the original paper, we have evaluated the performance of convolutional neural networks, one of the most successful and widely used machine learning architectures to date.

In general, the inclusion of the functional derivative not only improves the prediction accuracy for the functional derivative, but also leads to better generalization toward out-of-training data. This is underlined by the fact that iterative calculations of the minimum energy density are significantly more stable and lead to lower deviations in both the final kinetic energy and the converged density. However, the usage of derivative information in the kernel ridge regression technique increases the computational effort significantly and prohibits its application to larger data sets. Neural networks, on the other hand, do not show these limitations. Of the two flavors tested in this study, conventional convolutional networks and the more advanced ResNets, the latter variant achieves competitive results already on small training sets and improves its performance steadily with increasing data at minimal additional computational cost.

Very recently, it has already been shown for the exchange-correlation functional that convolutional neural network-based density functionals can easily be extended toward three-dimensional systems.[80] Using similar techniques for the kinetic energy functional might bring us closer to the ambitious objective of a truly orbital-free density functional theory.

## 6.6. Acknowledgements

Financial support by the Austrian Science Fund (FWF) under Grant P29893 is gratefully acknowledged. We thank the Central Informatics Service (ZID) of the Graz University of Technology for providing high performance computing resources and technical support. Furthermore, we are grateful for resources and support offered by Colaboratory, a free service of Google Research.

# 7. Conclusion and outlook

Over the last few years several extremely promising proofs of concept and first large scale applications of machine learning in computational chemistry have been presented. In this thesis methodological improvements to some of these previously suggested algorithms have been investigated. However, the presented methods only mark the very beginning of a possible paradigm shift in the field of computational chemistry and material science.

The next few years will be a crucial phase for the future of machine learning-based computational chemistry, determining whether these algorithms can make the jump from "niche" applications to wide-spread use throughout the community. In discussing these future developments it is often helpful to look at past examples. The latest example of a large shift in the field was the rise of density functional theory, which has become the workhorse of computational chemistry. Due to the approximative and often empirical nature, the probably most important ingredient in its highly successful thirty-year journey from the publication of the Hohenberg-Kohn theorem and Kohn-Sham method to its wide-spread application was building trust and confidence in the calculated results.

Modern machine learning methods find themselves in a similar situation. While several examples of algorithms reaching chemical accuracy have been presented, there is still wide-spread skepticism regarding the reliability and robustness of these results. Therefore, in the short term future, the first machine learning-based methods to be applied routinely are probably those classes of algorithms which do not influence the final calculation accuracy, such as automated selection of basis sets and active spaces, algorithms assisting structural search, improved initial guesses for electron densities or expansion coefficients in self-consistent calculations, and other convergence accelerating methods.

Another lesson to be learned from the example of density functional theory is that the availability in main stream computational chemistry packages is a necessary prerequisite for building trust. The main reason for B3LYPs popularity is most likely not its superior accuracy but rather the fact that it was the first hybrid functional to be implemented in the Gaussian program package. While most of the presented machine learning methods are published as open-source project, the burden of adaptation to a specific application or system typically lies with the user.

Even though machine learning methods will undoubtedly find their way into many computational chemistry algorithms, the exact extent and role of these methods in the long term future is unclear. Some of the more enthusiastic researchers have even speculated that machine learning models might outright replace most of the approximative methods used today such as force fields, semi-empirical methods, and even density functional theory. In this vision of the future, only the highest accuracy methods remain relevant and will mostly be used to generate training data for machine learning models.

From a physicist's point of view, the probably most interesting current development concerns the increased drive towards an actual understanding of the reasons behind a certain machine learning performance as opposed to black box fitting of large data sets. Methods, such as symbolic regression, can offer more interpretable results and might thereby help to gain insights into the underlying physics.

# A. Machine learning overview

This chapter provides a quick overview of the two machine learning-based regression models used throughout the thesis and refer the interested reader to more detailed explanations and literature.

## A.1. Feed forward neural networks

Feed forward neural networks are a special category of neural networks organized in sequential layers. The basic functional form of a single layer for a $D$-dimensional input is given by

$$z_j(\mathbf{x}) = \varphi\left(b_j + \sum_i^D w_{ij} x_i\right), \tag{A.1}$$

where $z_j$ are the elements of the output vector, $w_{ij}$ is a weight matrix, $b_j$ are bias variables, and $\varphi$ denotes an activation function. Feed forward neural networks are constructed by sequentially passing the input through a series of these layers,

$$y^{\mathrm{NN}}(\mathbf{x}) = z^L\left(z^{L-1}\left(\dots z^2\left(z^1(\mathbf{x})\right)\right)\right), \tag{A.2}$$

where the layers $z^1, \dots, z^{L-1}$ are referred to as hidden layers, and $z^L$ is the output layer.

The universal approximation theorem for neural networks states that any region of a function can be approximated arbitrarily well by single hidden layer feed forward neural networks.[438] From a mathematical point of view the hidden layer neurons can be interpreted as linearly independent basis functions and the output layer as linear combination thereof. Figure A.1 illustrates this by plotting the output of the 16 hidden layer neurons for the example shown in Figure 2.2.

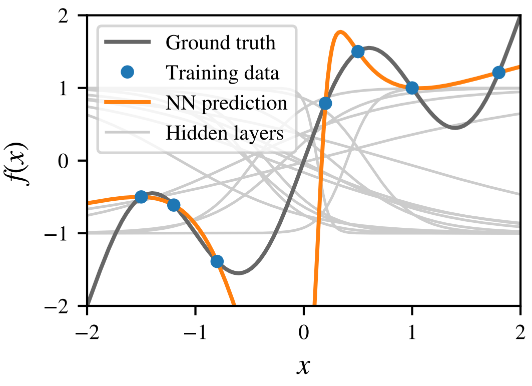


Figure A.1.: Illustration of the intermediate representation in the hidden layer.

Therefore, a necessary requirement for the activation functions is some amount of non-linearity, otherwise Equation A.2 would be reduced to a simple linear fit. Common choices include the hyperbolic tangent function $\varphi(x) = \tanh(x)$ or the "softplus" function $\varphi(x) = \ln(1 + e^x)$. The activation function in the output layer is specific to the learning task. For regression a linear activation function is used to account for the arbitrary choice in scale of measuring the output. In classification tasks the so-called "softmax" function is used, which allows to interpret the output as probability for each class.

A second requirement for the linear independence of the hidden layer functions is a suitable initialization of the weights. Typically, the weight matrices are initialized randomly and bias parameters are set to zero.

During training, the neural network parameters, i.e. the weights and biases, are determined by minimizing a loss function. For the task of regression this loss function is typically composed of the summed or mean squared error on a set of $M$ training examples and a regularization term used to penalize unnecessary complexity,

$$\mathcal{L} = \sum_i^M \left( y^{\mathrm{NN}}(\mathbf{x}_i) - y_i^{\mathrm{ref}} \right)^2 + \lambda \mathcal{R}(w, b), \tag{A.3}$$

where the weighting parameter $\lambda$ is referred to as regularization strength. A common choice is the L2-regularization $\mathcal{R}(w,b) = \sum_l^{L-1} \sum_{ij} |w_{ij}^l|^2$ applied to the weights of the hidden layers.

The computational minimization of the loss function with respect to the parameters is achieved using gradient-based optimization methods. While these training algorithms are typically based on the simple gradient descent method, additional modifications such as momentum and parameter-specific step sizes are used to accelerate convergence and to escape local minima. A further modification, aimed to reduce the computational effort and memory requirements, is the so-called mini-batch learning. Instead of evaluating the gradient of Equation A.3 by calculating the full sum over the training set, only a small subset of examples, referred to as batch, is used. The resulting noise, i.e. deviation from the true gradient, can be helpful to avoid local minima.

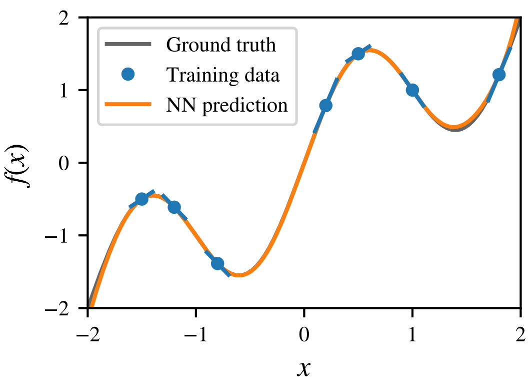


Figure A.2.: Neural network fit including derivative information.

Since most physics problems involve solving of a differential equation, training data for the derivative of the target function $y$ can be obtained with minimal additional computational cost. Reference information on gradients $\mathbf{g}^{\mathrm{ref}}$ can be included by extending the loss function,

$$\tilde{\mathcal{L}} = \mathcal{L} + \kappa \sum_i^M \left( \nabla y^{\mathrm{NN}}(\mathbf{x}_i) - \mathbf{g}_i^{\mathrm{ref}} \right)^2, \tag{A.4}$$

where $\kappa$ is a weighting parameter for the relative importance of the error on the gradients. Figure A.2 shows the significant improvements in the generalization behavior that can be achieved by including gradient information on the simple example from Section 2.2.1.

In summary, neural networks provide a capable and versatile framework for function fitting, especially suited for large training sets due to the fact that the number of parameters and evaluation time is independent of the number of training examples. However, since the loss function typically exhibits numerous local minima, careful tuning of the neural network architecture and training parameters is required to achieve accurate results.

Detailed explanations of the both the basics of neural networks and the more sophisticated deep learning architectures can be found in Ref. 356 and Ref. 424.

## A.2. Kernel ridge regression / Gaussian process regression

Kernel ridge regression and Gaussian process regression are closely related extensions of ridge regression and Bayesian regression toward non-linear data. While both methods are derived differently, the final prediction formulas are mathematically identical. The probabilistic nature of Gaussian process regression provides additional expressions for the uncertainty of the predictions. Since Section E.2 contains a detailed derivation of kernel ridge regression, this section is focused on summarizing the final results and applying them to the simple curve fitting example from Section 2.2.1.

The prediction of an previously unseen example is given by

$$y(\mathbf{x}) = \sum_{i}^{M} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{A.5}$$

where $k$ is a similarity measure referred to as kernel or covariance function in the context of Gaussian process regression and the weights $\alpha_i$ are given by

$$\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_M \end{pmatrix} = \left[ \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_M) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_M, \mathbf{x}_1) & \dots & k(\mathbf{x}_M, \mathbf{x}_M) \end{pmatrix} + \lambda \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix} \right]^{-1} \begin{pmatrix} y_1^{\text{ref}} \\ \vdots \\ y_M^{\text{ref}} \end{pmatrix}, \tag{A.6}$$

with the regularization parameter $\lambda$. Similar to the example of a single hidden layer neural network, the kernel functions can be interpreted as linearly independent basis functions, which in this case are anchored on the training examples.

A common choice for the kernel function is the so-called squared exponential kernel, sometimes referred to as Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left( \frac{1}{2} \sum_{d}^{D} \frac{(x_d - x'_d)^2}{l_d^2} \right), \tag{A.7}$$

where the length scales $l_d$ are additional parameters. Figure A.3 illustrates the importance of a suitable scale parameter $l$ by fitting the example from Section 2.2.1 using a squared exponential kernel with different length scale parameter values.

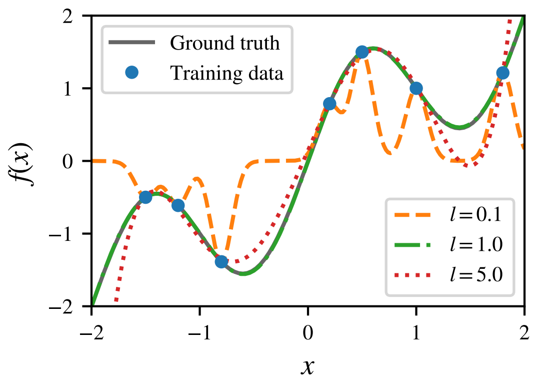


Figure A.3.: Kernel ridge regression fit using different length scale parameters.

Additional model parameters beyond the fit weights $\alpha_i$, such as the kernel parameters and the regularization parameter $\lambda$, are referred to as hyperparameters, commonly denoted as $\boldsymbol{\theta}$. In kernel ridge regression hyperparameter optimization is typically achieved by cross-validation, i.e. training on a subset of the training set and evaluating the generalization error on the excluded examples. Gaussian process regression additionally offers a probabilistic approach to hyperparameter tuning by maximizing the marginal likelihood,

$$p(y|X, \boldsymbol{\theta}) = \int p(y|X, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\alpha}, \tag{A.8}$$

where $X = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ is a matrix consisting of the input vectors for the training examples.

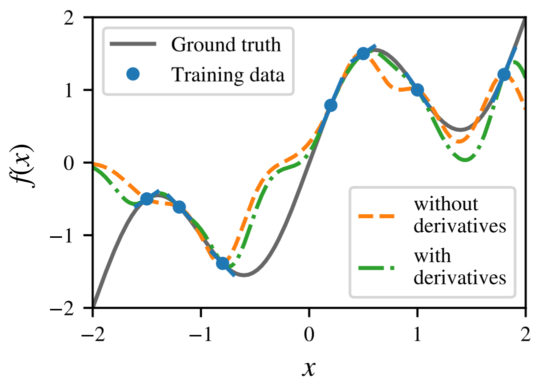


Figure A.4.: Kernel ridge regression fit including derivative information.

Derivative information can be included by using additional weights for the kernel derivatives,

$$y(\mathbf{x}) = \sum_{i}^{M} \left( \alpha_i k(\mathbf{x}_i, \mathbf{x}) + \boldsymbol{\beta}_i^\top \nabla k(\mathbf{x}_i, \mathbf{x}) \right). \tag{A.9}$$

A detailed derivation of this result and the formula for the newly introduced weight vectors $\boldsymbol{\beta}_i$ is given in Section E.2.

Figure A.4 shows the improvements achieved by including reference values for the derivative using a squared exponential kernel with a length scale of $l = 0.2$.

In summary, kernel-based methods offer a highly accurate approach for function fitting, given a suitable set of hyperparameters. The method is, however, restricted to smaller data sets as the computational effort for training and prediction grows quickly with the number of training examples.

An in-depth discussion of Gaussian process regression and the relation to other machine learning methods can be found in the book "Gaussian Processes for Machine Learning" by Rasmussen and Williams.[104]

# B. Supporting material: Embedded atom model potentials for Pt-Ni nanoclusters improved by machine learning: a compromise between flexibility and physical meaning

In Section B.1 the DFT data set is inspected in detail and analyzed with the simplest of models. Section B.2 provides an explanation of the cutoff scheme used to reduce the computational effort. Descriptor parameters and modifications of the Behler-Parrinello model are discussed in Section B.3. Results of the hyperparameter search conducted for all neural network-based models are presented in Section B.4. Finally, Section B.5 lists the parameters of the newly suggested extended SMATB models.

## B.1. Data set

The data set comprises a total of 1463 cluster geometries evaluated at the DFT level. 167 of these structures consist of 38 atoms, 352 of 55 atoms, and 944 contain 147 atoms. This distribution emerges from the fact that the global optimization routine applied to clusters of the three sizes identified more local minima for the larger clusters. The test set consists of the 5 lowest energy geometries (as determined by SMATB) for each of the 22 optimization runs, yielding a total of 110 geometries. The remaining 1353 geometries are split into roughly 10 % validation set (136 geometries) and 90 % training set (1217 geometries). Including both the bonding energy and the components of the force vectors the training, validation, and test set contain a total of 421 475, 47 530, and 25 880 data points, respectively. In Figure B.1 the distribution of binding energies per atom and their dependence on the fraction of Pt atoms is plotted. The right side panel shows three distinct binding energy curves for the three investigated cluster sizes.

The complexity of the data set can be quantified by the error scores achieved by simple models. We refrained from presenting these models in the main article due to their trivial nature, but present them here in order to underline that accurate relations between cluster energy, size and composition are found easily – but useless for the task of geometry optimization. A simple, constant (i.e. geometry independent) model is given by

$$E = \sum_{i}^{N} E^{\alpha} = N^{\mathrm{Pt}} E^{\mathrm{Pt}} + N^{\mathrm{Ni}} E^{\mathrm{Ni}}. \tag{B.1}$$

Fitting the two parameters on the training set yields $E^{\mathrm{Pt}} = -4.7154$ eV and $E^{\mathrm{Ni}} = -4.1317$ eV. A slightly more refined model is obtained by fitting a pair binding energy for each bond

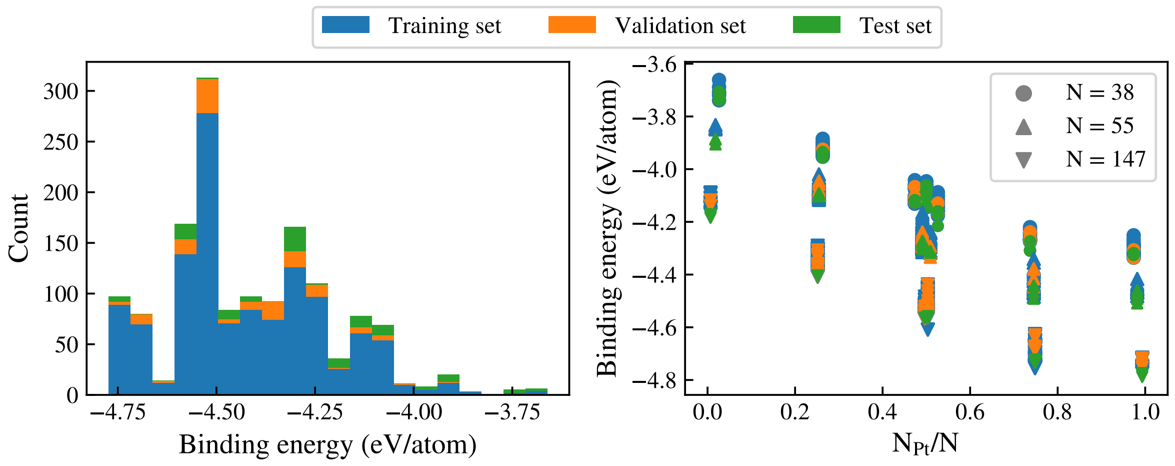


Figure B.1.: Histogram of the DFT binding energies (left panel) and a plot of the binding energy as a function of the Pt concentration (right panel) for the various geometries in the three data sets.

type,

$$E = \sum_{ij \in bonds} E^{\alpha\beta} = N^{\mathrm{PtPt}} E^{\mathrm{PtPt}} + N^{\mathrm{NiPt}} E^{\mathrm{NiPt}} + N^{\mathrm{NiNi}} E^{\mathrm{NiNi}}. \tag{B.2}$$

Here we use a distance criterion $r_{ij} < r_{\mathrm{cut}}$ to determine which pairs of atoms are participating in bonding. A rough scan over the cutoff radius $r_{\mathrm{cut}}$ yields the best results for a value of $r_{\mathrm{cut}} = 3.1$ Å with bond energies of $E^{\mathrm{PtPt}} = -0.5358$ eV, $E^{\mathrm{NiPt}} = -0.5110$ eV, and $E^{\mathrm{NiNi}} = -0.4494$ eV. Further improvement might be achievable via element-specific cutoff radii $r_{\mathrm{cut}}^{\alpha\beta}$. The errors of these basic models for energies and forces are given in Table B.1. While energy predictions are very good on the given data set, as can already be anticipated from the linear relationships seen in the right panel of Figure B.1 (binding energy per atom as a function of Pt content), both models are obviously predicting zero forces for all structures since Equations 1 and 2 are lacking any dependence on geometry. A first geometry-dependent model of similarly low complexity can be constructed by replacing the atomic neural networks in the Behler-Parrinello model by a linear fit,

$$E_i^{\alpha} = \mathbf{w}^{\alpha\mathrm{Ni}} \cdot \mathbf{G}_i^{\alpha\mathrm{Ni}} + \mathbf{w}^{\alpha\mathrm{Pt}} \cdot \mathbf{G}_i^{\alpha\mathrm{Pt}} + c^{\alpha}, \tag{B.3}$$

where the constant offset term $c^{\alpha}$ can either be treated as an additional fit parameter or be determined by enforcing zero atomic energy for a single, isolated atom. We elaborate on the reasoning behind the second choice in Section B.3.

Table B.1 summarizes the error scores achieved by these models on the three data sets. Only the model with fitted offset parameters achieves lower error scores than the simple pair binding energy model. Interestingly, the fixed offset Behler-Parrinello model shows significantly higher deviations in the force predictions, even compared to the simple geometry independent models. This can again be attributed to the fact that all three data sets only include geometries near local minima for which the forces are close to zero. A closer inspection of the Behler-Parrinello models given in Section B.3 reveals that including the offset in the fit parameters leads to severe over-fitting and unphysical predictions outside of the region covered by the data sets.

Table B.1.: Root mean squared errors for energy and forces achieved by the simple models given in meV/atom and meV/Å.

| | Training set | | Validation set | | Test set | |
|---|---|---|---|---|---|---|
| model | Energy | Forces | Energy | Forces | Energy | Forces |
| Atomic energy | 161.8 | 321.1 | 152.2 | 318.4 | 230.1 | 326.2 |
| Pair binding energy | 126.6 | 321.1 | 119.9 | 318.4 | 157.8 | 326.2 |
| Linear Behler-Parrinello, fitted offset | 112.9 | 238.2 | 105.8 | 235.9 | 162.1 | 228.8 |
| Linear Behler-Parrinello, fixed offset | 125.0 | 531.7 | 114.5 | 532.3 | 146.7 | 469.2 |

## B.2. Long range cutoff

In order to reduce the number of interactions to evaluate, the pair-wise interactions $\phi^{\alpha\beta}$ and $\rho^{\alpha\beta}$ in the SMATB expressions are typically forced to fade out in a region between $r = a^{\alpha\beta}$ and $r = b^{\alpha\beta}$ using an interpolation function $f_{\text{int}}^{\alpha\beta}$. For example, the modified pair potential $\tilde{\phi}$ is then given by

$$\tilde{\phi}^{\alpha\beta}(r) = \begin{cases} \phi^{\alpha\beta}(r), & \text{for } r \leq a^{\alpha\beta} \\ f_{\text{int}}^{\alpha\beta}(r), & \text{for } a^{\alpha\beta} < r < b^{\alpha\beta} \\ 0, & \text{for } r \geq b^{\alpha\beta}. \end{cases} \tag{B.4}$$

For the sake of simplicity the atom type superscripts are dropped for the remainder of this discussion. A common choice for the interpolation function is a fifth-order polynomial,

$$f(r) = c_0 + c_1 r + c_2 r^2 + c_3 r^3 + c_4 r^4 + c_5 r^5, \tag{B.5}$$

where the parameters $c_i$ are determined using the following 6 continuity conditions,

$$\begin{aligned} &f(a) = \phi(b) \qquad && f'(a) = \phi'(a) \qquad && f''(a) = \phi''(a) \\ &f(b) = 0 && f'(b) = 0 && f''(b) = 0. \end{aligned} \tag{B.6}$$

The interpolation approach requires evaluating the values $\phi(b)$, $\phi'(a)$, and $\phi''(a)$ at each step of the parameter fitting, which introduces additional complexity especially in the case of neural network-based pair interactions. Therefore, in this article, fading contributions are realized by a multiplication with a cutoff function,

$$\tilde{\phi}(r) = \phi(r) \cdot f_{\text{cut}}(r), \tag{B.7}$$

which is inspired by a similar use of cutoff functions in the Behler-Parrinello model. Following the standard SMATB fading approach mentioned before, we employ a fifth order polynomial now for the cutoff function,

$$f_{\text{cut}}(r) = \begin{cases} 1, & \text{for } r \leq a \\ 1 - 10\left(\frac{r-a}{b-a}\right)^3 + 15\left(\frac{r-a}{b-a}\right)^4 - 6\left(\frac{r-a}{b-a}\right)^5, & \text{for } a < r < b \\ 0, & \text{for } r \geq b. \end{cases} \tag{B.8}$$

The SMATB potential by Cheng et al.[302] uses fifth and sixth nearest neighbor distances as inner and outer cutoff for the Ni–Ni interaction ($a^{\text{NiNi}} = 5.57$ Å and $b^{\text{NiNi}} = 6.10$ Å) and the values of the third and fourth neighbor distance in bulk Pt for the cutoffs of both the Pt–Ni and the Pt–Pt interaction ($a^{\mathrm{PtNi}} = a^{\mathrm{PtPt}} = 4.80$ Å and $b^{\mathrm{PtNi}} = b^{\mathrm{PtPt}} = 5.54$ Å).

Preliminary investigations showed that for the investigated clusters similar accuracy can be achieved using significantly smaller cutoff values. In order to minimize the computational effort of the basin hopping algorithm we use second and third nearest neighbor distances for the cutoffs $a$ and $b$ during the global optimization. The cutoffs for the mixed interaction are defined as $a^{\alpha\beta} = \max\left(a^{\alpha\alpha}, a^{\beta\beta}\right)$ and $b^{\alpha\beta} = \min\left(b^{\alpha\alpha}, b^{\beta\beta}\right)$. All of the presented evaluations of the SMATB potential by Cheng et al.[302] after the data generation phase and the subsequent splitting into training, validation, and test sets use the original larger cutoff values in order to correctly investigate its accuracy.

Small fading regions $a^{\alpha\beta} < r < b^{\alpha\beta}$, where the binding energy changes rapidly, result in high forces on the atoms in this region, which in turn yield high fitting errors for the forces. Therefore, we define the mixed interaction cutoff as $a^{\alpha\beta} = \min\left(a^{\alpha\alpha}, a^{\beta\beta}\right)$ and $b^{\alpha\beta} = \max\left(b^{\alpha\alpha}, b^{\beta\beta}\right)$ in order to expand the fading region and train models for various cutoff distances $b$ to investigate the influence on the fit accuracy.

Table B.2 summarizes the results for a linear Behler-Parrinello and SMATB model. The cutoff value $a$ is fixed at second nearest neighbor distance $a^{\mathrm{PtPt}} = 3.92$ Å, and $a^{\mathrm{NiNi}} = 3.52$ Å and the cutoff value $b$ is varied from third to fifth and seventh nearest neighbor distances.

Table B.2.: Root mean squared errors for energy and forces in meV/atom and meV/Å achieved by using various cutoff values $b$ in Å.

| | | | Training set | | Validation set | | Test set | |
|---|---|---|---|---|---|---|---|---|
| model | $b^{\mathrm{PtPt}}$ | $b^{\mathrm{NiNi}}$ | Energy | Forces | Energy | Forces | Energy | Forces |
| linear Behler-Parrinello | 4.81 | 4.32 | 172.6 | 577.3 | 161.9 | 584.4 | 237.2 | 504.6 |
| | 6.20 | 5.57 | 120.9 | 532.8 | 110.0 | 533.2 | 142.0 | 470.0 |
| | 7.34 | 6.59 | 143.8 | 523.6 | 131.0 | 521.5 | 176.0 | 470.1 |
| SMATB | 4.81 | 4.32 | 70.8 | 207.2 | 66.9 | 199.8 | 107.6 | 215.4 |
| | 6.20 | 5.57 | 70.7 | 201.9 | 65.9 | 195.9 | 107.8 | 207.5 |
| | 7.34 | 6.59 | 69.6 | 200.7 | 64.7 | 195.1 | 106.0 | 205.7 |

As expected, the accuracy of the fit increases with increasing cutoff distance. For the geometries of the training set, these three cutoff distances result in an average of 20.6, 37.1, and 56.0 neighbors, respectively. All models presented in the main article use the second nearest neighbor distances for $a$ and the fifth nearest neighbor distances for $b$ ($b^{\mathrm{PtPt}} = 6.2$ Å and $b^{\mathrm{NiNi}} = 5.57$ Å). This choice represents a trade off between accuracy and computational efficiency.

## B.3. Behler-Parrinello model

### B.3.1. Descriptors

Table B.3 gives the maximum values for the descriptor vectors $\mathbf{G}_{\max}^{\alpha\beta}$ needed for the transformation in Equation 3.12 of the main article.

Table B.3.: Maximum values for the descriptor vectors used for normalization.

| $\alpha$ | $\beta$ | $\mathbf{G}_{\max}^{\alpha\beta}$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Pt | Pt | 11.341068 | 9.163056 | 7.438487 | 5.101074 | 2.655911 | 0.836476 | 0.090367 |
| Pt | Ni | 13.407259 | 11.024802 | 9.073774 | 6.321983 | 3.391382 | 1.120701 | 0.138323 |
| Ni | Pt | 10.947930 | 8.841274 | 7.159221 | 4.919796 | 2.735640 | 0.911357 | 0.106661 |
| Ni | Ni | 11.204494 | 9.439022 | 7.967277 | 5.840071 | 3.395881 | 1.318860 | 0.220886 |

The minimum value for all descriptors $\mathbf{G}_{\min}^{\alpha\beta}$ is the zero vector $\mathbf{0}$ obtained for the geometries with the highest Pt and Ni concentrations, respectively. Figure B.2 shows the distribution of the normalized descriptor vector entries for the training set in the form of "violin plots", i.e. a series of histograms describing the relative occurrence of vector entries for the 7 components of $\mathbf{G}_{\max}^{\alpha\beta}$.

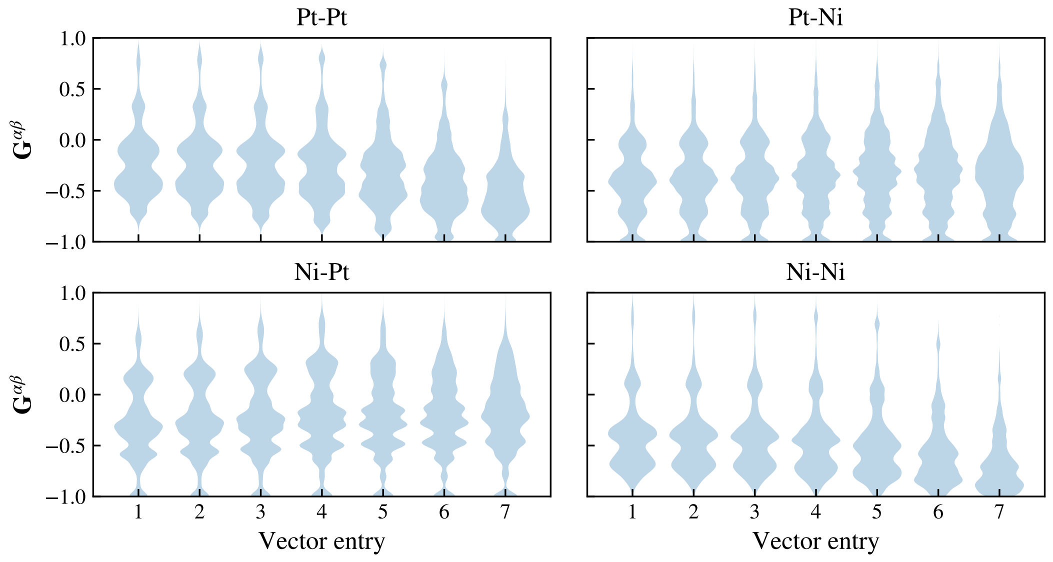


Figure B.2.: Violin plots of the entries for the descriptor vectors evaluated on the training set.

### B.3.2. Offset / Single atom energy

Typically, Behler-Parrinello style atomic neural networks employ a final linear layer. This is motivated by the fact that both the unit of measurement and the zero-point of the energy scale can be chosen arbitrarily. The zero-point can, however, not easily be determined from the presented data sets since they only contain geometries close to equilibrium, which in turn leads to wrong predictions of the dissociation energies. In order to enforce the correct dissociation behavior we exclude the offset, i.e. the bias variable of the final linear layer, from the set fitting parameters and instead determine its value from the remaining neural network parameters. The offset parameter is then given by the neural network expression for the input of a single, isolated atom. For the chosen input normalization this corresponds to the vector $-\mathbf{1}$.

Figure B.3 illustrates the influence of this offset parameter choice for both a linear and a two hidden layers, 15 hidden neurons each, Behler-Parrinello model by plotting the potential energy curve of a icosahedral 13 atom cluster breathing mode. Especially for the linear Behler-

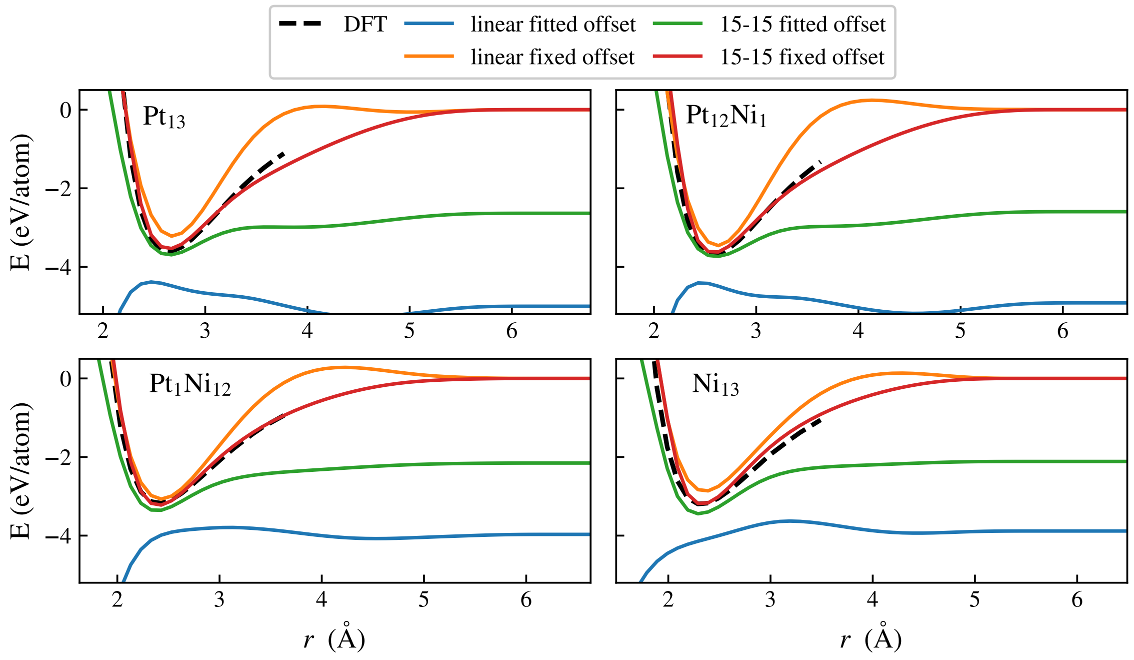


Figure B.3.: Potential energy curve for the breathing mode of a icosahedral 13 atom cluster as predicted by Behler-Parrinello models using differently determined offset parameters.

Parrinello model fixing the offset parameter not only ensures the correct atomic energy for isolated atoms but also greatly improves the fit quality throughout the PES scan. Note that the obviously incorrect behaviour exhibited by the models which include the offset in the fit parameters does not become apparent from the error scores achieved on the data sets presented in Table B.4. This can again be attributed to the extremely limited region of the PES spanned by the geometries in the data set.

Table B.4.: Root mean squared errors for energy and forces (in meV/atom and meV/Å) achieved by Behler-Parrinello models using differently determined offset parameters.

| | Training set | | Validation set | | Test set | |
|---|---|---|---|---|---|---|
| model | Energy | Forces | Energy | Forces | Energy | Forces |
| Linear, fitted offset | 112.9 | 238.2 | 105.8 | 235.9 | 162.1 | 228.8 |
| Linear, fixed offset | 125.0 | 531.7 | 114.5 | 532.3 | 146.7 | 469.2 |
| 15-15, fitted offset | 14.1 | 134.5 | 12.9 | 134.1 | 24.9 | 128.0 |
| 15-15, fixed offset | 14.1 | 139.8 | 13.3 | 140.1 | 22.4 | 132.4 |

## B.4. Hyperparameters / Model selection

Tables B.5-B.8 show the training and validation scores achieved by the neural network-based models using different hyperparameters. The final model hyperparameters are selected based on the lowest sum of energy and force root mean squared errors on the validation set.

Table B.5.: Root mean squared errors for energy and forces for the neural network embedding function model, given in meV/atom and meV/Å, respectively.

| | | Number of parameters | Training set | | Validation set | |
|---|---|---|---|---|---|---|
| $\lambda$ | Hidden layers | | Energy | Forces | Energy | Forces |
| $10^{-5}$ | **15-15** | 584 | **24.4** | 198.5 | **22.6** | 192.4 |
| | 15-15-15 | 1064 | 25.1 | 198.5 | 23.0 | 192.4 |
| | 20-20 | 974 | 24.7 | 198.5 | 22.7 | 192.4 |
| | 20-20-20 | 1814 | 24.5 | 198.5 | 22.6 | 192.4 |
| $10^{-6}$ | 15-15 | 584 | 24.6 | 198.6 | 22.7 | 192.4 |
| | 15-15-15 | 1064 | 24.7 | 198.5 | 22.8 | 192.4 |
| | 20-20 | 974 | 24.7 | 198.3 | 22.9 | 192.2 |
| | 20-20-20 | 1814 | 24.6 | **197.9** | 23.5 | **191.8** |

Table B.6.: Root mean squared errors for energy and forces for the neural network pair density function model, given in meV/atom and meV/Å, respectively.

| | | Number of parameters | Training set | | Validation set | |
|---|---|---|---|---|---|---|
| $\lambda$ | Hidden layers | | Energy | Forces | Energy | Forces |
| $10^{-5}$ | 15-15 | 864 | 55.7 | 177.1 | 49.8 | 175.6 |
| | 15-15-15 | 1584 | **53.4** | 177.9 | **46.7** | 175.9 |
| | 20-20 | 1449 | 55.1 | 177.6 | 49.2 | 175.9 |
| | 20-20-20 | 2709 | 55.5 | 178.9 | 48.6 | 177.1 |
| $10^{-6}$ | 15-15 | 864 | 56.4 | 177.1 | 49.9 | 175.6 |
| | **15-15-15** | 1584 | 53.7 | **177.0** | 47.2 | **175.4** |
| | 20-20 | 1449 | 53.6 | 177.2 | 47.8 | 175.9 |
| | 20-20-20 | 2709 | 56.4 | 179.6 | 49.1 | 177.6 |

Table B.7.: Root mean squared errors for energy and forces for the neural network embedding and pair density function model, given in meV/atom and meV/Å, respectively.

| $\lambda$ | Hidden layers | Number of parameters | Training set | | Validation set | |
|---|---|---|---|---|---|---|
| | | | Energy | Forces | Energy | Forces |
| $10^{-5}$ | 15-15 | 1436 | 21.4 | 172.4 | 16.5 | 171.2 |
| | 15-15-15 | 2636 | 20.8 | 172.6 | 15.9 | 171.2 |
| | 20-20 | 2411 | 19.3 | 170.4 | 15.1 | 169.1 |
| | 20-20-20 | 4511 | **19.1** | 171.3 | **14.0** | 169.9 |
| $10^{-6}$ | 15-15 | 1436 | 19.8 | 171.9 | 15.1 | 170.5 |
| | 15-15-15 | 2636 | 19.8 | 171.6 | 14.2 | 169.8 |
| | 20-20 | 2411 | 22.1 | 171.1 | 16.9 | 170.3 |
| | **20-20-20** | 4511 | 21.1 | **169.7** | 15.0 | **168.6** |

Table B.8.: Root mean squared errors for energy and forces for the Behler-Parrinello model, given in meV/atom and meV/Å, respectively.

| $\lambda$ | Hidden layers | Number of parameters | Training set | | Validation set | |
|---|---|---|---|---|---|---|
| | | | Energy | Forces | Energy | Forces |
| $10^{-5}$ | 15-15 | 960 | 14.1 | 139.8 | 13.3 | 140.1 |
| | 15-15-15 | 1440 | 10.7 | 134.1 | 10.4 | 134.6 |
| | 20-20 | 1480 | 13.0 | 138.1 | 12.0 | 138.0 |
| | **20-20-20** | 2320 | **10.2** | **132.1** | **8.0** | **132.9** |
| $10^{-6}$ | 15-15 | 960 | 11.2 | 137.9 | 9.5 | 138.0 |
| | 15-15-15 | 1440 | 13.3 | 135.4 | 12.2 | 135.4 |
| | 20-20 | 1480 | 13.0 | 138.1 | 12.3 | 137.9 |
| | 20-20-20 | 2320 | 11.8 | 132.5 | 10.5 | 133.2 |

## B.5. Extended SMATB models

Tables B.9-B.11 list all parameters of the extended SMATB models.

Table B.9.: The parameters for the extended embedding model.

| $\alpha$ | $\beta$ | $A$ (eV) | $\xi$ (eV) | $p$ | $q$ | $c_0$ | $c_1$ | $c_2$ |
|---|---|---|---|---|---|---|---|---|
| Pt | Pt | 0.20458 | 2.30194 | 10.78673 | 2.43044 | 0.54345 | 0.40588 | 0.03907 |
| Pt | Ni | 0.19395 | 2.18161 | 10.79773 | 2.64396 | – | – | – |
| Ni | Ni | 0.15870 | 1.87030 | 9.71559 | 2.15242 | 0.54559 | 0.44691 | 0.04554 |

Table B.10.: The parameters for the extended pair density model.

| $\alpha$ | $\beta$ | $A$ (eV) | $\xi_1$ (eV) | $\xi_2$ (eV) | $p$ | $q_1$ | $q_2$ |
|---|---|---|---|---|---|---|---|
| Pt | Pt | 0.30929 | 2.19693 | 0.65551 | 10.54191 | 4.49866 | $-0.64894$ |
| Pt | Ni | 0.26590 | 2.09543 | 0.50503 | 9.53831 | 3.82265 | $-0.73671$ |
| Ni | Ni | 0.22111 | 1.93536 | 0.28418 | 9.03036 | 3.35666 | $-0.99338$ |

Table B.11.: The parameters for the model combining the extended embedding function and the extended pair density function.

| $\alpha$ | $\beta$ | $A$ (eV) | $\xi_1$ (eV) | $\xi_2$ (eV) | $p$ |
|---|---|---|---|---|---|
| Pt | Pt | 0.31425 | 2.26808 | 0.76412 | 10.28350 |
| Pt | Ni | 0.25494 | 2.16532 | 0.51045 | 10.01660 |
| Ni | Ni | 0.20860 | 1.95425 | 0.24465 | 8.81490 |

| $q_1$ | $q_2$ | $c_0$ | $c_1$ | $c_2$ |
|---|---|---|---|---|
| 3.67648 | $-0.36707$ | 0.70972 | 0.29959 | 0.02336 |
| 3.29996 | $-0.56913$ | – | – | – |
| 2.61048 | $-1.23366$ | 0.68304 | 0.30589 | 0.03824 |

# C. Supporting material: Geometry optimization using Gaussian process regression in internal coordinate systems

## C.1. Previous studies on the benchmark set

Baker's optimization benchmark set of small molecules[328] has been used in several previous studies in order to evaluate method performance. The respective best performance reported by these investigations is summarized in Table C.1. Note that some of the studies used different convergence criteria and/or different *ab initio* methods to evaluate energies and gradients.

- Ref. 328: Baker used natural internal coordinates and the initial Hessian as predicted by the CVFF force field in the original publication of the benchmark set.
- Ref. 162: Lindh et al. used 'local normal modes', i.e. the eigenvectors of the approximate Hessian, in combination with a constantly updated model Hessian inspired by typical force field expressions
- Ref. 361: Peng et al. used a redundant internal coordinate system and a diagonal initial Hessian in the redundant internal coordinate space.
- Ref. 362: Eckert et al. used natural internal coordinates in combination with the model Hessian of Ref. 162.
- Ref. 365: Farkas et al. used a redundant internal coordinate system in combination with the rational function optimization approach[172] and a Hessian initial guess suggested by Schlegel.[159]
- Ref. 364: Lindh et al introduced the so-called force-constant weighted redundant coordinates, i.e. re-weighted delocalized internal coordinates, and the model Hessian from Ref. 162.
- Ref. 366: Bakken and Helgaker introduced the extra-redundant coordinate system in which the redundant coordinate system is extended by auxiliary bonds on pairs of atoms closer than 2.5 times the sum of their covalent radii. The initial Hessian was calculated using the model Hessian from Ref. 162.
- Ref. 342: Németh and Challacombe used a curvefit in redundant internal coordinates starting from a diagonal initial Hessian.

[a]Different convergence criteria.
[b]Different *ab initio* method.

Table C.1.: Summary of the number of optimization cycles reported by previous studies on the benchmark set.

| Molecule | Baker Ref. 328 | Lindh Ref. 162 | Peng[a] Ref. 361 | Eckert Ref. 362 | Farkas[a] Ref. 365 | Lindh[a] Ref. 364 | Bakken Ref. 366 | Németh[a] Ref. 342 | Swart[ab] Ref. 340 |
|---|---|---|---|---|---|---|---|---|---|
| Water | 6 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| Ammonia | 6 | 5 | 4 | 6 | 5 | 4 | 5 | 4 | 4 |
| Ethane | 5 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 4 |
| Acetylene | 6 | 5 | 4 | 6 | 5 | 4 | 4 | 4 | 4 |
| Allene | 5 | 5 | 4 | 4 | 3 | 4 | 4 | 3 | 4 |
| Hydroxysulfane | 8 | 8 | 7 | 7 | 7 | 7 | 7 | 8 | 6 |
| Benzene | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 3 |
| Methylamine | 6 | 5 | 4 | 5 | 3 | 4 | 4 | 4 | 4 |
| Ethanol | 6 | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 5 |
| Acetone | 6 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 5 |
| Disilyl-ether | 8 | 11 | 7 | 9 | 7 | 9 | 8 | 9 | 8 |
| 1,3,5-Trisilacyclohexane | 8 | 8 | 11 | 6 | 9 | 7 | 9 | 5 | 6 |
| Benzaldehyde | 6 | 5 | 4 | 5 | 4 | 4 | 4 | 5 | 5 |
| 1,3-Difluorobenzene | 5 | 5 | 4 | 5 | 3 | 4 | 4 | 4 | 4 |
| 1,3,5-Trifluorobenzene | 5 | 4 | 4 | 4 | 3 | 3 | 4 | 3 | 4 |
| Neopentane | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 5 | 4 |
| Furan | 8 | 7 | 5 | 6 | 6 | 6 | 5 | 6 | 5 |
| Naphthalene | 5 | 6 | 4 | 6 | 4 | 5 | 5 | 6 | 5 |
| 1,5-Difluoronaphthalene | 6 | 6 | 4 | 6 | 4 | 6 | 5 | 6 | 5 |
| 2-Hydroxybicyclopentane | 15 | 10 | 11 | 9 | 9 | 12 | 9 | 7 | 10 |
| ACHTAR10 | 12 | 8 | 9 | 9 | 8 | 10 | 8 | 10 | 8 |
| ACANIL01 | 8 | 8 | 6 | 8 | 6 | 7 | 7 | 11 | 5 |
| Benzidine | 9 | 10 | 7 | 7 | 7 | 8 | 9 | 8 | 6 |
| Pterin | 10 | 9 | 8 | 9 | 8 | 8 | 8 | 12 | 8 |
| Difuropyrazine | 9 | 7 | 6 | 7 | 6 | 6 | 6 | 6 | 8 |
| Mesityl-oxide | 7 | 6 | 5 | 6 | 5 | 5 | 5 | 5 | 5 |
| Histidine | 19 | 20 | 14 | 14 | 13 | 19 | 16 | 11 | 14 |
| Dimethylpentane | 12 | 10 | 9 | 10 | 9 | 12 | 9 | 6 | 5 |
| Caffeine | 12 | 7 | 6 | 7 | 6 | 6 | 6 | 7 | 7 |
| Menthone | 13 | 14 | 11 | 10 | 11 | 13 | 12 | 13 | 8 |
| Sum | 240 | 215 | 183 | 196 | 176 | 196 | 185 | 187 | 173 |

- Ref. 340: Swart and Bickelhaupt used a modified version of the force-constant weighted redundant coordinates from Ref. 364 and the GDIIS[341] update method in combination with the initial model Hessian from Ref. 162.

## C.2. Size of the active space

Table C.2 lists the size of the active coordinate space for all of the investigated coordinate systems, together with the number of vibrational degrees of freedom, the symmetry group, and the reduced number of variables needed to describe the geometry after taking into account the molecular symmetry. Note that the number of delocalized and localized inverse distance coordinates can be lower than the number of vibrational degrees of freedom for planar molecules. This is due to out-of-plane bending, which can not be described by a change of internuclear distances within the first-order approximation of the Wilson B-matrix formalism.

Table C.2.: Number of coordinates (degrees of freedom) generated by the various schemes.

| | | | | | Inverse distances | | | | Z-matrix-derived internals | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Molecule | Vibrational degrees of freedom | Symmetry | Symmetry reduced coordinates | Cartesian coordinates | Fully redundant | Delocalized | Localized | Reduced redundancy | Fully redundant | Delocalized | Localized | Reduced redundancy |
| Water | 3 | $C_{2v}$ | 2 | 9 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Ammonia | 6 | $C_{3v}$ | 2 | 12 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| Ethane | 18 | $D_{3d}$ | 3 | 24 | 28 | 18 | 18 | 28 | 28 | 18 | 18 | 22 |
| Acetylene | 7 | $D_{\infty h}$ | 2 | 12 | 6 | 3 | 3 | 6 | 7 | 7 | 7 | 7 |
| Allene | 15 | $D_{2d}$ | 3 | 21 | 21 | 15 | 15 | 21 | 18 | 15 | 15 | 15 |
| Hydroxysulfane | 6 | $C_1$ | 6 | 12 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| Benzene | 30 | $D_{6h}$ | 2 | 36 | 66 | 21 | 21 | 66 | 54 | 30 | 30 | 32 |
| Methylamine | 15 | $C_s$ | 10 | 21 | 21 | 15 | 15 | 21 | 21 | 15 | 15 | 17 |
| Ethanol | 21 | $C_s$ | 13 | 27 | 36 | 21 | 21 | 36 | 33 | 21 | 21 | 27 |
| Acetone | 24 | $C_{2v}$ | 8 | 30 | 45 | 24 | 24 | 45 | 36 | 24 | 24 | 28 |
| Disilyl-ether | 21 | $C_{2v}$ | 7 | 27 | 36 | 21 | 21 | 36 | 27 | 21 | 21 | 25 |
| 1,3,5-Trisilacyclohexane | 48 | $C_{3v}$ | 11 | 54 | 153 | 48 | 48 | 144 | 108 | 48 | 48 | 78 |
| Benzaldehyde | 36 | $C_s$ | 25 | 42 | 91 | 25 | 25 | 85 | 63 | 36 | 36 | 38 |
| 1,3-Difluorobenzene | 30 | $C_{2v}$ | 11 | 36 | 66 | 21 | 21 | 64 | 54 | 30 | 30 | 32 |
| 1,3,5-Trifluorobenzene | 30 | $D_{3h}$ | 4 | 36 | 66 | 21 | 21 | 63 | 54 | 30 | 30 | 32 |
| Neopentane | 45 | $T_d$ | 3 | 51 | 136 | 45 | 45 | 136 | 82 | 45 | 45 | 58 |
| Furan | 21 | $C_{2v}$ | 8 | 27 | 36 | 15 | 15 | 36 | 38 | 21 | 21 | 23 |
| Naphthalene | 48 | $D_{2h}$ | 9 | 54 | 153 | 33 | 33 | 129 | 93 | 48 | 48 | 52 |
| 1,5-Difluoronaphthalene | 48 | $C_{2h}$ | 17 | 54 | 153 | 33 | 33 | 125 | 93 | 48 | 48 | 52 |
| 2-Hydroxybicyclopentane | 36 | $C_1$ | 36 | 42 | 91 | 36 | 36 | 91 | 100 | 36 | 36 | 65 |
| ACHTAR10 | 42 | $C_1$ | 42 | 48 | 120 | 42 | 42 | 100 | 66 | 42 | 42 | 55 |
| ACANIL01 | 51 | $C_s$ | 34 | 57 | 171 | 51 | 51 | 139 | 87 | 51 | 51 | 55 |
| Benzidine | 72 | $D_2$ | 18 | 78 | 325 | 72 | 72 | 211 | 129 | 72 | 72 | 100 |
| Pterin | 45 | $C_s$ | 31 | 51 | 136 | 31 | 31 | 105 | 81 | 45 | 45 | 49 |
| Difuropyrazine | 42 | $C_{2h}$ | 15 | 48 | 120 | 29 | 29 | 89 | 86 | 42 | 42 | 48 |
| Mesityl-oxide | 45 | $C_s$ | 28 | 51 | 136 | 45 | 45 | 106 | 69 | 45 | 45 | 51 |
| Histidine | 54 | $C_1$ | 54 | 60 | 190 | 54 | 54 | 155 | 97 | 54 | 54 | 81 |
| Dimethylpentane | 63 | $C_1$ | 63 | 69 | 253 | 63 | 63 | 222 | 118 | 63 | 63 | 85 |
| Caffeine | 66 | $C_s$ | 42 | 72 | 276 | 66 | 66 | 183 | 122 | 66 | 66 | 76 |
| Menthone | 81 | $C_1$ | 81 | 87 | 406 | 81 | 81 | 321 | 170 | 81 | 81 | 122 |

## C.3. Additional results

In the course of this study, several other choices of coordinate systems have been considered during preliminary investigations. Some of these intermediate results, which were crucial in motivating the approaches presented in the main article, are summarized in Table C.3.

The calculations using the fully redundant set of Z-matrix-derived internals and an isotropic squared exponential kernel employ the approach of transforming torsion angles $\omega$ to the $(\cos(\omega), \sin(\omega))$ space in order to properly take into account the periodicity of these coordinates and to avoid jumps in the coordinates at $\omega = \pm\pi$.

Reducing the redundancy of inverse distance primitives is achieved using the procedure detailed in the main article. The results presented here use different values for the threshold $r_{\text{cut}}$, which enables a certain tuning of the degree of redundancy.

Table C.3.: Size of the active coordinate space and optimization cycles for additional investigated coordinates systems.

| | fully redundant Z-matrix internals single length scale | | Reduced redundancy inverse distances $r_{\text{cut}} = 3.0$ Å | | Reduced redundancy inverse distances $r_{\text{cut}} = 4.0$ Å | | Reduced redundancy Z-matrix internals delocalized subsets | |
|---|---|---|---|---|---|---|---|---|
| Molecule | size of active space | opt. cycles | size of active space | opt. cycles | size of active space | opt. cycles | size of active space | opt. cycles |
| Water | 3 | 5 | 3 | 5 | 3 | 5 | 3 | 5 |
| Ammonia | 6 | 5 | 6 | 4 | 6 | 4 | 6 | 4 |
| Ethane | 37 | 6 | 25 | 5 | 28 | 5 | 22 | 6 |
| Acetylene | 7 | 6 | 5 | 5 | 6 | 5 | 7 | 6 |
| Allene | 22 | 6 | 18 | 5 | 21 | 5 | 15 | 6 |
| Hydroxysulfane | 7 | 13 | 6 | 10 | 6 | 10 | 6 | 7 |
| Benzene | 78 | 5 | 39 | 4 | 57 | 4 | 32 | 4 |
| Methylamine | 27 | 9 | 20 | 7 | 21 | 7 | 17 | 7 |
| Ethanol | 45 | 11 | 29 | 8 | 35 | 8 | 27 | 7 |
| Acetone | 48 | 11 | 33 | 10 | 44 | 8 | 28 | 8 |
| Disilyl-ether | 33 | 10 | 23 | 9 | 27 | 8 | 25 | 8 |
| 1,3,5-Trisilacyclohexane | 162 | 24 | 66 | 26 | 108 | 11 | 78 | 10 |
| Benzaldehyde | 91 | 11 | 48 | 9 | 69 | 9 | 38 | 9 |
| 1,3-Difluorobenzene | 78 | 9 | 39 | 8 | 55 | 8 | 32 | 6 |
| 1,3,5-Trifluorobenzene | 78 | 6 | 39 | 7 | 54 | 4 | 32 | 5 |
| Neopentane | 118 | 6 | 82 | 5 | 130 | 5 | 58 | 6 |
| Furan | 54 | 10 | 25 | 7 | 33 | 9 | 23 | 6 |
| Naphthalene | 137 | 9 | 67 | 7 | 97 | 7 | 52 | 6 |
| 1,5-Difluoronaphthalene | 137 | 10 | 67 | 8 | 95 | 8 | 52 | 6 |
| 2-Hydroxybicyclopentane | 154 | 34 | 65 | 15 | 84 | 15 | 65 | 12 |
| ACHTAR10 | 92 | 21 | 61 | 12 | 76 | 13 | 55 | 11 |
| ACANIL01 | 125 | 13 | 79 | 31 | 113 | 17 | 55 | 8 |
| Benzidine | 189 | 16 | 103 | 41 | 149 | 24 | 100 | 9 |
| Pterin | 117 | 14 | 58 | 12 | 80 | 11 | 49 | 9 |
| Difuropyrazine | 126 | 14 | 51 | 10 | 73 | 10 | 48 | 8 |
| Mesityl-oxide | 95 | 14 | 63 | 25 | 89 | 11 | 51 | 8 |
| Histidine | 142 | 56 | 78 | 150 | 120 | 22 | 81 | 14 |
| Dimethylpentane | 172 | 18 | 114 | 12 | 177 | 12 | 85 | 8 |
| Caffeine | 176 | 16 | 101 | 22 | 136 | 17 | 76 | 9 |
| Menthone | 254 | 43 | 152 | 18 | 230 | 21 | 122 | 11 |
| Sum | | 431 | | 497 | | 303 | | 229 |

# D. Supporting material: Machine learning in computational chemistry: An evaluation of method performance for nudged elastic band calculations

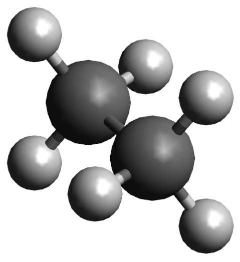
In Section D.1 of this Supporting Material we list the images of the initial path for the ethane rotation in Cartesian coordinates. In Section D.2 we present a small study of the influence of the regularization hyperparameter $\lambda$ on the performance of a neural network potential on the ethane benchmark. Section D.3 contains all images of the initial path for the carbon dioxide activation on $Pt_4^-$ in Cartesian coordinates. The influence of the architecture of a neural network on the performance in the latter reaction is discussed in Section D.4. Section D.5 shows the value of the optimized hyperparameters as a function of the number of iterations. In Section D.6 the concept of determining the weights of a Gaussian approximation potential in feature space is explained in detail. Finally, the computational advantages of fixing the kernel hyperparameters are discussed and exemplified in Section D.7.

## D.1. Ethane initial path images

While the ethane rotation is a simple example and the initial guess can easily be generated by following the ASE documentation, the complete initial path is given here for enhanced reproducibility.

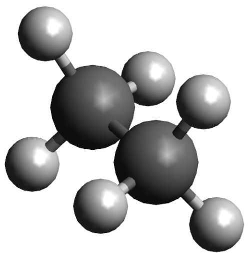
Minimum A:

```
H       1.02170625       0.00000000       1.16513318
H      -0.51085312       0.88482357       1.16513318
H      -0.51085312      -0.88482357       1.16513318
C       0.00000000       0.00000000       0.76627284
C       0.00000000       0.00000000      -0.76627284
H       0.51085312      -0.88482357      -1.16513318
H      -1.02170625       0.00000000      -1.16513318
H       0.51085312       0.88482357      -1.16513318
```



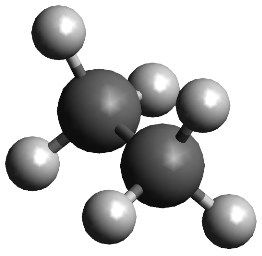
Intermediate image 1:

```
H       1.00893029       0.10477194       1.17840874
H      -0.59520030       0.82137329       1.17840874
H      -0.41372999      -0.92614523       1.17840874
C      -0.00000000       0.00000000       0.75971474
C       0.00000000       0.00000000      -0.75971474
H       0.41372999      -0.92614523      -1.17840874
H      -1.00893029       0.10477194      -1.17840874
H       0.59520030       0.82137329      -1.17840874
```



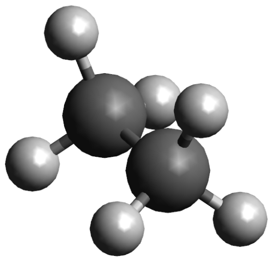
Intermediate image 2:

```
H       0.98725415       0.20944538       1.18901010
H      -0.67501210       0.75026448       1.18901010
H      -0.31224205      -0.95970986       1.18901010
C      -0.00000000       0.00000000       0.75840771
C       0.00000000      -0.00000000      -0.75840771
H       0.31224205      -0.95970986      -1.18901010
H      -0.98725415       0.20944538      -1.18901010
H       0.67501210       0.75026448      -1.18901010
```



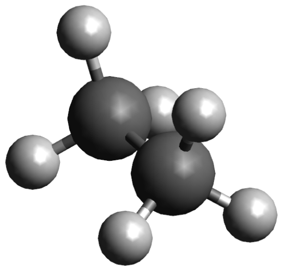
Intermediate image 3:

```
H       0.95651139       0.31188421       1.19683628
H      -0.74835534       0.67242105       1.19683628
H      -0.20815604      -0.98430526       1.19683628
C      -0.00000000       0.00000000       0.75951438
C       0.00000000      -0.00000000      -0.75951438
H       0.20815604      -0.98430526      -1.19683628
H      -0.95651139       0.31188421      -1.19683628
H       0.74835534       0.67242105      -1.19683628
```



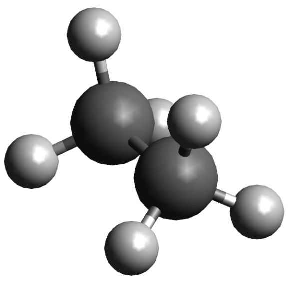
Intermediate image 4:

```
H       0.91703959       0.40991241       1.20163881
H      -0.81351436       0.58922338       1.20163881
H      -0.10352523      -0.99913579       1.20163881
C      -0.00000000       0.00000000       0.76101522
C       0.00000000      -0.00000000      -0.76101522
H       0.10352523      -0.99913579      -1.20163881
H      -0.91703959       0.40991241      -1.20163881
H       0.81351436       0.58922338      -1.20163881
```



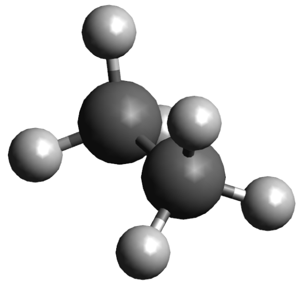
Intermediate image 5:

```
H       0.86953477       0.50202614       1.20324334
H      -0.86953477       0.50202614       1.20324334
H      -0.00000000      -1.00405227       1.20324334
C      -0.00000000       0.00000000       0.76169026
C       0.00000000      -0.00000000      -0.76169026
H       0.00000000      -1.00405227      -1.20324334
H      -0.86953477       0.50202614      -1.20324334
H       0.86953477       0.50202614      -1.20324334
```



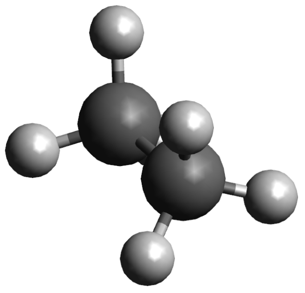
Intermediate image 6:

```
H        0.81351436        0.58922338        1.20163881
H       -0.91703959        0.40991241        1.20163881
H        0.10352523       -0.99913579        1.20163881
C       -0.00000000        0.00000000        0.76101522
C        0.00000000       -0.00000000       -0.76101522
H       -0.10352523       -0.99913579       -1.20163881
H       -0.81351436        0.58922338       -1.20163881
H        0.91703959        0.40991241       -1.20163881
```



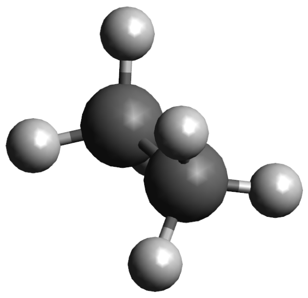
Intermediate image 7:

```
H        0.74835534        0.67242105        1.19683628
H       -0.95651139        0.31188421        1.19683628
H        0.20815604       -0.98430526        1.19683628
C       -0.00000000        0.00000000        0.75951438
C        0.00000000       -0.00000000       -0.75951438
H       -0.20815604       -0.98430526       -1.19683628
H       -0.74835534        0.67242105       -1.19683628
H        0.95651139        0.31188421       -1.19683628
```



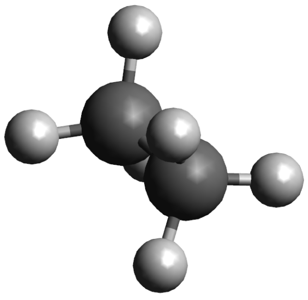
Intermediate image 8:

```
H        0.67501210        0.75026448        1.18901010
H       -0.98725415        0.20944538        1.18901010
H        0.31224205       -0.95970986        1.18901010
C       -0.00000000        0.00000000        0.75840771
C        0.00000000       -0.00000000       -0.75840771
H       -0.31224205       -0.95970986       -1.18901010
H       -0.67501210        0.75026448       -1.18901010
H        0.98725415        0.20944538       -1.18901010
```



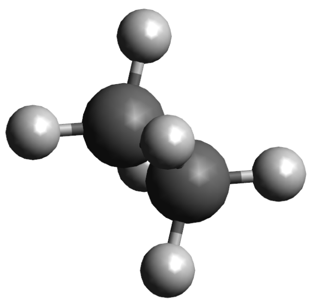
Intermediate image 9:

```
H        0.59520030        0.82137329        1.17840874
H       -1.00893029        0.10477194        1.17840874
H        0.41372999       -0.92614523        1.17840874
C       -0.00000000        0.00000000        0.75971474
C        0.00000000       -0.00000000       -0.75971474
H       -0.41372999       -0.92614523       -1.17840874
H       -0.59520030        0.82137329       -1.17840874
H        1.00893029        0.10477194       -1.17840874
```



Minimum B:

```
H        0.51085312        0.88482357        1.16513318
H       -1.02170625        0.00000000        1.16513318
H        0.51085312       -0.88482357        1.16513318
C        0.00000000       -0.00000000        0.76627284
C        0.00000000        0.00000000       -0.76627284
H       -0.51085312       -0.88482357       -1.16513318
H       -0.51085312        0.88482357       -1.16513318
H        1.02170625       -0.00000000       -1.16513318
```



## D.2. Hyperparameters for the neural network potential of ethane

The regularization parameter $\lambda$ in the loss function effectively controls the complexity of the neural network function and is typically used to avoid overfitting. In this context it can be used to favor flat (that is slowly varying) potential energy surface predictions. This may be advantageous to avoid unphysical regions of the PES, which are difficult to evaluate *ab initio*. In order to determine a suitable value for the regularization parameter we calculate 10 individual runs with varying values for $\lambda$. Table D.1 summarizes the results of this small hyperparameter study for a fixed value of $\beta = 1$. The best performance and lowest variance is achieved by setting the regularization to $\lambda = 10^{-4}$. Test runs for a larger value of $10^{-2}$ did not converge at all due to the restricted flexibility of the neural network function.

Table D.1.: Summary of the number of *ab intio* band evaluations till convergence for the neural network potentials model using varying regularization $\lambda$.

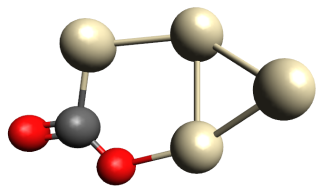
| | *ab initio* evaluations | | |
|---|---|---|---|
| Run # | $\lambda = 10^{-4}$ | $\lambda = 10^{-6}$ | $\lambda = 10^{-8}$ |
| 0 | 4 | 7 | 5 |
| 1 | 4 | 6 | 8 |
| 2 | 6 | 5 | not conv. |
| 3 | 5 | 8 | 7 |
| 4 | 5 | 8 | 7 |
| 5 | 6 | not conv. | 6 |
| 6 | 5 | 7 | not conv. |
| 7 | 4 | 5 | 10 |
| 8 | 5 | 6 | 11 |
| 9 | 6 | 5 | not conv. |
| min | 4 | 5 | 5 |
| max | 6 | $> 20$ | $> 20$ |
| median | 5.0 | 6.5 | 9.0 |

## D.3. $Pt_4^-$+$CO_2$ initial path images

Initial path for all NEB calculations of the $Pt_4^-$+$CO_2$ system. The minima are optimized starting from the geometries given by Green et al.[384]

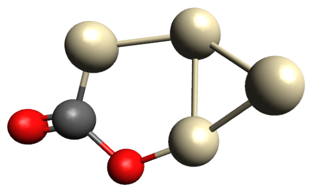
Minimum A:

| | | | |
|---|---|---|---|
| Pt | 0.00000000 | 0.00000000 | 0.00000000 |
| Pt | -2.51013000 | 2.34854000 | -0.00615000 |
| Pt | 1.83325000 | 1.19925000 | 1.17686000 |
| Pt | -0.00000000 | 2.59290000 | 0.00000000 |
| C | -2.89549000 | 0.46347000 | -0.32385000 |
| O | -1.97587000 | -0.48460000 | -0.34359000 |
| O | -4.14068000 | 0.31161000 | -0.45994000 |



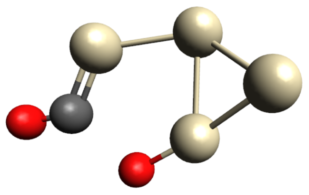
Intermediate image 1:

| | | | |
|---|---|---|---|
| Pt | -0.00760961 | 0.01734639 | -0.02196494 |
| Pt | -2.46412947 | 2.29118900 | -0.00856104 |
| Pt | 1.78208238 | 1.21084159 | 1.21700735 |
| Pt | 0.02997968 | 2.61325072 | -0.03077237 |
| C | -3.02069611 | 0.55650939 | -0.29829468 |
| O | -1.75930879 | -0.63245319 | -0.38691441 |
| O | -4.24923809 | 0.37448610 | -0.42716991 |



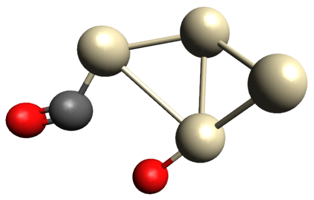
Intermediate image 2:

| | | | |
|---|---|---|---|
| Pt | -0.01201091 | 0.03836466 | -0.04267510 |
| Pt | -2.41621920 | 2.23458816 | -0.01079449 |
| Pt | 1.73079688 | 1.22229150 | 1.25713986 |
| Pt | 0.06003039 | 2.63348710 | -0.06158979 |
| C | -3.14769026 | 0.64847794 | -0.27283520 |
| O | -1.54608925 | -0.78377638 | -0.43171296 |
| O | -4.35773766 | 0.43773702 | -0.39420232 |



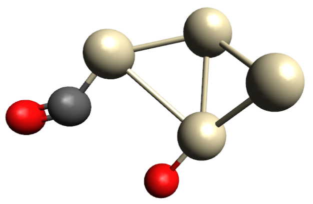
Intermediate image 3:

| | | | |
|---|---|---|---|
| Pt | -0.01644321 | 0.06353525 | -0.06205571 |
| Pt | -2.36722635 | 2.17740369 | -0.01315643 |
| Pt | 1.67997609 | 1.23331197 | 1.29675330 |
| Pt | 0.08992652 | 2.65330692 | -0.09212495 |
| C | -3.27518358 | 0.74085072 | -0.24734303 |
| O | -1.33497002 | -0.93812371 | -0.47740641 |
| O | -4.46499945 | 0.50088516 | -0.36133677 |



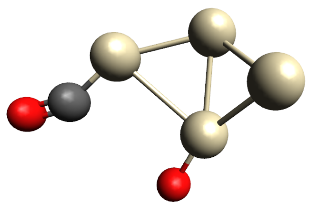
Intermediate image 4:

| | | | |
|---|---|---|---|
| Pt | -0.02717895 | 0.08879492 | -0.08168104 |
| Pt | -2.31940788 | 2.11770889 | -0.01603298 |
| Pt | 1.62997835 | 1.24378136 | 1.33554745 |
| Pt | 0.11950356 | 2.67253512 | -0.12216938 |
| C | -3.40043032 | 0.83540027 | -0.22159004 |
| O | -1.12121167 | -1.09061598 | -0.52195280 |
| O | -4.57017309 | 0.56356542 | -0.32879120 |



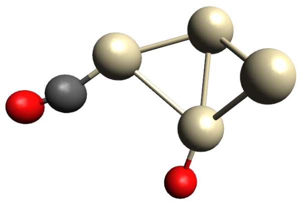
Intermediate image 5:

| | | | |
|---|---|---|---|
| Pt | -0.04722575 | 0.10628214 | -0.10419185 |
| Pt | -2.27515166 | 2.05427207 | -0.01959165 |
| Pt | 1.58035651 | 1.25401164 | 1.37389017 |
| Pt | 0.14888399 | 2.69141713 | -0.15193410 |
| C | -3.52175658 | 0.93344588 | -0.19533258 |
| O | -0.90007648 | -1.23431209 | -0.56310501 |
| O | -4.67395003 | 0.62605322 | -0.29640498 |



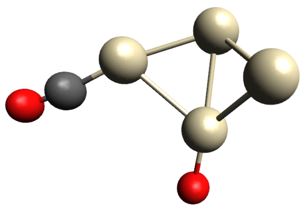
Intermediate image 6:

| | | | |
|---|---|---|---|
| Pt | -0.07137771 | 0.11279586 | -0.13036257 |
| Pt | -2.23485726 | 1.98772757 | -0.02361717 |
| Pt | 1.53008225 | 1.26457298 | 1.41262321 |
| Pt | 0.17842946 | 2.71049943 | -0.18194937 |
| C | -3.64034931 | 1.03465502 | -0.16851098 |
| O | -0.67259201 | -1.36820939 | -0.60113000 |
| O | -4.77825541 | 0.68912853 | -0.26372313 |



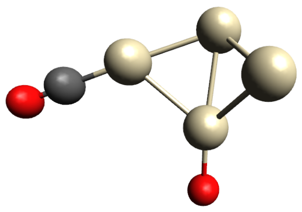
Intermediate image 7:

| | | | |
|---|---|---|---|
| Pt | -0.09226870 | 0.11445409 | -0.15789912 |
| Pt | -2.19670233 | 1.92046370 | -0.02768412 |
| Pt | 1.47877187 | 1.27571863 | 1.45207570 |
| Pt | 0.20825604 | 2.73002976 | -0.21241903 |
| C | -3.75860395 | 1.13669599 | -0.14146792 |
| O | -0.44446410 | -1.49919905 | -0.63866339 |
| O | -4.88390884 | 0.75300687 | -0.23061213 |



Minimum B:

| | | | |
|---|---|---|---|
| Pt | -0.10809266 | 0.11925179 | -0.18419984 |
| Pt | -2.15822425 | 1.85469435 | -0.03149471 |
| Pt | 1.42691527 | 1.28726131 | 1.49188210 |
| Pt | 0.23816035 | 2.74981794 | -0.24309706 |
| C | -3.87820278 | 1.23713046 | -0.11466073 |
| O | -0.21935266 | -1.63419886 | -0.67775294 |
| O | -4.99012327 | 0.81721300 | -0.19734681 |



## D.4. Neural network architecture for the $Pt_4^-$+$CO_2$ system

In order to determine the influence of the neural network architecture on the ML-NEB performance three architectures, consisting of two hidden layers containing 5, 15, and 30 neurons each, are compared. In order to account for the statistical nature of the neural network training procedure, the algorithm is run 10 times for each architecture. Figure D.1 shows the

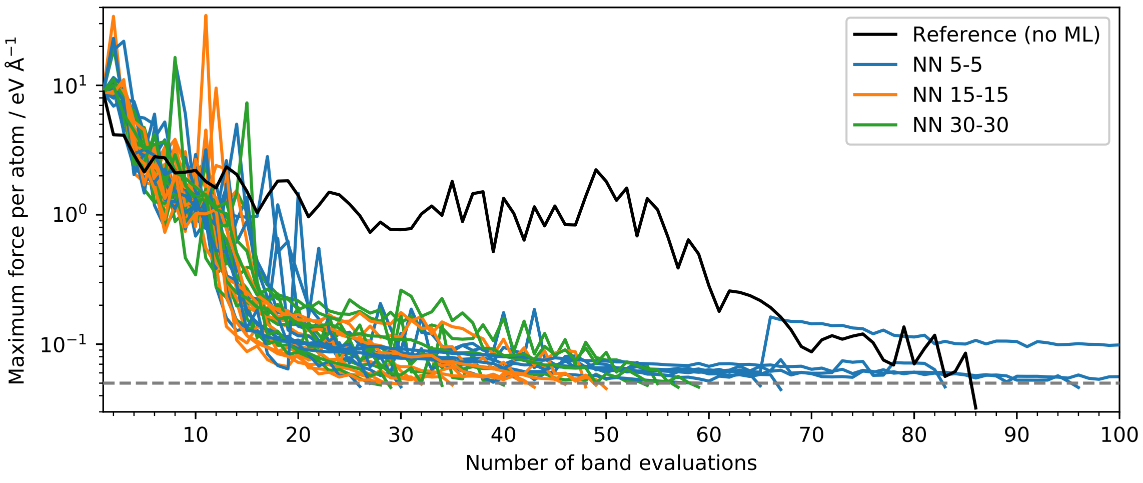


Figure D.1.: Comparison of the convergence behavior of differently sized neural network potentials.

convergence behavior of the various neural network sizes compared to the reference calculation without machine learning. For easier analysis the results of the individual runs are given in Table D.2.

Table D.2.: Detailed listing of the performance of various architectures for neural network potentials

| | *ab initio* evaluations | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Run # | | | | | | | | | | Statistics | | | |
| Layers | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | min | max | median | mean |
| 5-5 | - | 30 | 67 | 65 | 59 | 26 | 83 | - | 27 | 96 | 26 | $> 100$ | 66.0 | – |
| 15-15 | 32 | 43 | 35 | 48 | 29 | 43 | 49 | 46 | 50 | 30 | 29 | 50 | 43.0 | 40.5 |
| 30-30 | 40 | 30 | 55 | 54 | 28 | 30 | 34 | 29 | 57 | 59 | 28 | 59 | 37.0 | 41.6 |

Remarkably, the best individual run was achieved by the simplest 5-5 architecture. Figure D.1, however, also clearly shows that the lower complexity of this setup leads to difficulties in the final convergence for several runs. The two larger network architectures yield similar performance, with the medium sized 15-15 atomic neural networks achieving the smallest variance in the number of *ab initio* evaluations. This is reflected by the slightly smaller average number of iterations of 40.5 compared to 41.6 for the largest architecture. Judging by the median the 30-30 network with a value of 37 should be preferred over the 15-15 architecture with a value of 43. Our choice of presenting the mid-sized network in the main article is motivated by the fact that this leads to a reduction in the number of parameters of the model simplifying the training procedure significantly.

## D.5. Hyperparameters for the Gaussian process regression-based methods

Figure D.2 shows the optimization results for the single hyperparameter used for the Cartesian GPR and the GAP as a function of the number of steps in the ML-NEB algorithm. Comparison

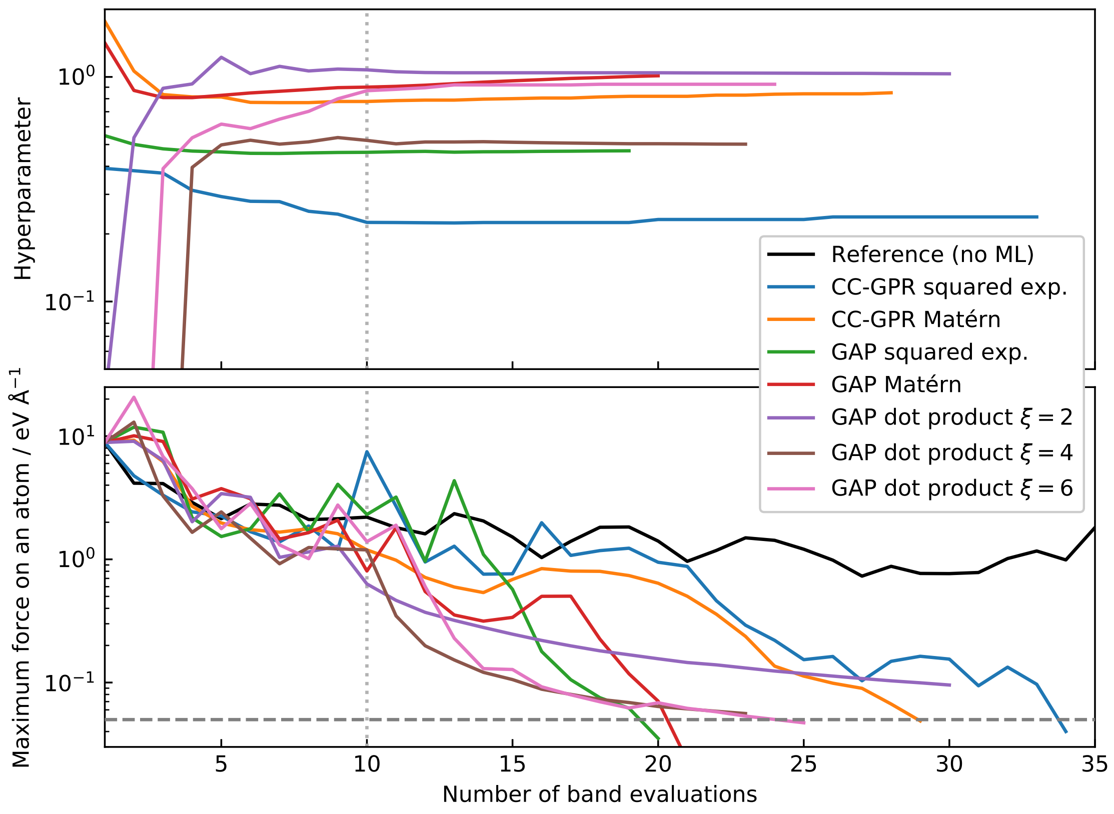


Figure D.2.: Values for the single hyperparameter (top panel) (in Å for Cartesian GPR, unitless for GAP) and maximum force per atom (lower panel) plotted as a function of the number of band evaluations for the various Gaussian process regression-based machine learning potentials.

with Figure 5.3 from the main article shows, that optimizing the hyperparameters at every step does slightly reduce the number of *ab initio* evaluations needed to find the MEP, at the cost of increased computational effort for the fitting process. This high time requirement is the reason why the runs for the GAP using the dot product kernel with $\xi = 2$ and $\xi = 4$ had to be stopped before reaching convergence. The top panel shows that the hyperparameters approach a constant value after the first few steps, which motivates our approach of freezing the hyperparameters after 10 band evaluations (marked by the dotted vertical line). It also clearly shows that determining the optimal value for the $\sigma_0^2$ hyperparameter of the dot product kernel proved to be very hard without sufficient training data.

## D.6. Kernel trick versus linear fit in feature space

The derivation of the energy prediction formula for the Gaussian process-based method (Equation 5.11 of the main article) involves the so-called ”kernel-trick”, replacing the inner product of vectors in feature space by a kernel. A prerequisite for this step is applying the following matrix identity, often referred to as push-through identity,[439]

$$(\lambda \mathbf{I} + PQ)^{-1} P = P\,(\lambda \mathbf{I} + QP)^{-1}\,, \tag{D.1}$$

to the prediction formula for GPR by setting $P = \Phi^\top$ and $Q = \Phi$. In general the matrices $\Phi$ and $\Phi^\top$ are not square: $\Phi \in \mathbb{R}^{D_1 \times D_2}, \Phi^\top \in \mathbb{R}^{D_2 \times D_1}$ with $D_1 \neq D_2$. As a result the matrices that have to be inverted are of different sizes. Due to the high computational effort of matrix inversion, choosing to inverting the smaller one can result in significant speed up. For the application in GPR this gives the choice of doing the matrix inversion either in feature space or the space spanned by the training data. Typically the feature space is very high dimensional or even of infinite dimension as for example in the case of the squared exponential kernel. However, the dimension of the GAP feature space for the dot product kernel is finite and given by:

$$D_1 = \binom{H+\xi}{\xi} \cdot N_{\text{atomtypes}}, \tag{D.2}$$

where $H$ is the dimension of the input space (descriptor space), $\xi$ is the exponent of the dot product kernel and $N_{\text{atomtypes}}$ is the number of elements in the molecule. This has to be compared to the dimension of the kernel matrix, which for the ML-NEB is a simple function of a few parameters:

$$D_2 = (2 + N_{\text{images}} N_{\text{steps}}) \cdot (1 + 3N_{\text{atoms}}), \tag{D.3}$$

where the first term is the number of training examples consisting of the two minima and the number of intermediate images $N_{\text{images}}$ times the number of *ab initio* evaluations of the band $N_{\text{steps}}$ and the second term represents the number of pieces of information per data point consisting of the value for the total energy and the $3N_{\text{atoms}}$ values of the atomic forces.

This allows to determine the number of iterations after which inverting the matrix in feature space becomes computationally cheaper, namely when $D_1 < D_2$. For the ethane system, consisting of two elements $N_{\text{atomtypes}} = 2$, the size of our descriptor set is $H = 50$. Using a value of $\xi = 2$ and plugging in the remaining values of $N_{\text{atoms}} = 8$ and $N_{\text{images}} = 9$ gives a value of 12 iterations after which $D_1 < D_2$. Due to the larger descriptor set of $H = 93$ and the increased number of elements $N_{\text{atomtypes}} = 3$ the trade-off is only reached after 87 *ab intio* evaluations for the $Pt_4^-CO_2$ benchmark ($N_{\text{atoms}} = 7$, $N_{\text{images}} = 7$).

While determining the weights in feature space does not prove useful for ML-NEB on the presented systems this concept might offer a way to reduce computational cost for tasks involving few atom types or easy access to large amounts of training data.

## D.7. Computational savings due to fixed hyperparameters

Using the simple example of fitting the one dimensional cardinal sine function using training data for the function value and the first derivative, the possible computational savings can be illustrated clearly. The left panel of Figure D.3 shows the starting situation. Given two training

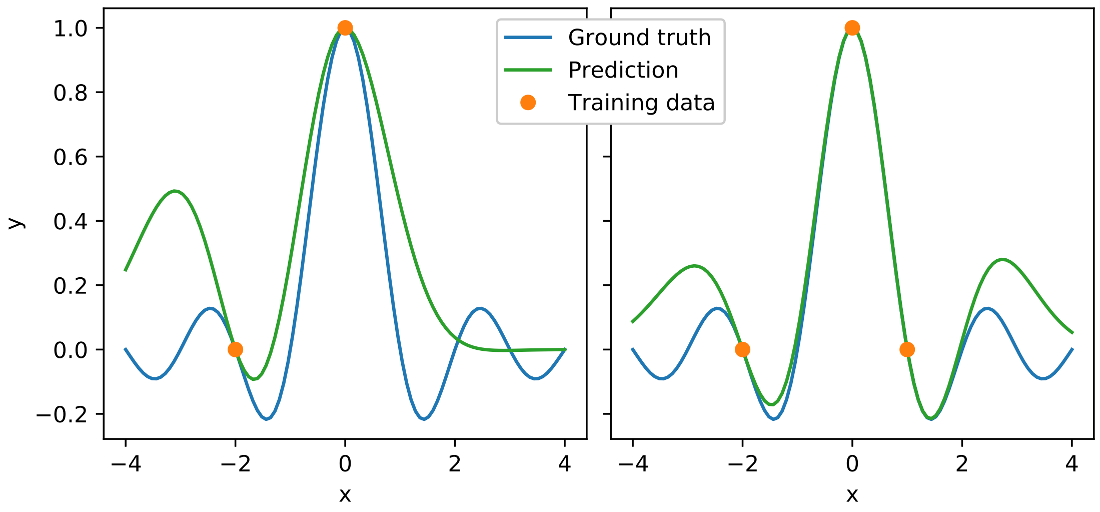


Figure D.3.: Gaussian process regression fit of the cardinal sine function using 2 and 3 training points for the left and right panel, respectively

points (orange) the cardinal sine function is approximated using a Gaussian process regression with a squared exponential covariance function ($l = 1$). The kernel (covariance) matrix $C$ in Equation D.4 is organized as follows: The first row (and column) corresponds to the function value at the first training point ($x = 0$), the second row to the first derivative at the same point. The covariance between the function value and the first derivative is zero (function value and derivative can be adjusted independently). Third and forth row are organized in a similar fashion, the former corresponding to the function value at the second training point ($x = -2$), the latter to the derivative at the training point. A crucial part of the Gaussian process regression is solving a set of linear equations which is done by calculating the Cholesky decomposition of the kernel matrix. The lower triangular matrix $L$ corresponding to $C$ is also given in Equation D.4.

$$C = \begin{pmatrix} 1. & 0. & 0.135 & 0.271 \\ 0. & 1. & -0.271 & -0.406 \\ 0.135 & -0.271 & 1. & 0. \\ 0.271 & -0.406 & 0. & 1. \end{pmatrix} \quad L = \begin{pmatrix} 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. \\ .135 & -0.271 & 0.953 & 0. \\ 0.271 & -0.406 & -0.154 & 0.859 \end{pmatrix} \tag{D.4}$$

In the next step the training set is expanded and a third data point (at $x = 1$) is added. Given a fixed kernel hyperparameter only the covariance of the new data point and the previous training data has to evaluated. The computational effort of this scales linearly in the number of training examples, in contrast to the quadratic scaling of reevaluating the whole covariance matrix. In Equation D.5 the updated parts of the matrix are highlighted in light blue.

$$C' = \begin{pmatrix} 1. & 0. & 0.135 & 0.271 & 0.607 & -0.607 \\ 0. & 1. & -0.271 & -0.406 & 0.607 & 0. \\ 0.135 & -0.271 & 1. & 0. & 0.011 & -0.033 \\ 0.271 & -0.406 & 0. & 1. & 0.033 & -0.089 \\ 0.607 & 0.607 & 0.011 & 0.033 & 1. & 0. \\ -0.607 & 0. & -0.033 & -0.089 & 0. & 1. \end{pmatrix} \tag{D.5}$$

The determining factor in the overall $\mathcal{O}(M^3)$ scaling of the GPR algorithm is the Cholesky decomposition, where $M$ denotes the number of training examples. Just updating the result of a previous decomposition, however, can significantly reduce the computational effort. Equation D.6 highlights the elements of the lower triangular matrix that have to be updated, which can be achieved in $\mathcal{O}(M^2)$ operations.

$$L' = \begin{pmatrix} 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. & 0. & 0. \\ 0.135 & -0.271 & 0.953 & 0. & 0. & 0. \\ 0.271 & -0.406 & -0.154 & 0.859 & 0. & 0. \\ 0.607 & 0.607 & 0.098 & 0.152 & 0.481 & 0. \\ -0.607 & 0. & 0.051 & 0.097 & 0.723 & 0.311 \end{pmatrix} \tag{D.6}$$

# E. Supporting material: Machine learning approaches toward orbital-free density functional theory: Simultaneous training on the kinetic energy density functional and its functional derivative

In the first section of this supporting material the data sets used in the main article are inspected and the accuracy achieved by simple linear models is investigated. Section E.2 gives a detailed derivation of the kernel ridge regression algorithm and the method used to include derivative information. Section E.3 and Section E.4 discuss hyperparameter choices and training procedures for kernel ridge regression and neural networks, respectively. In Section E.5 the algorithm used to find minimum energy densities is explained. Section E.6 shows the investigations of kernel ridge regression models using a constant offset term, an alternative approach to the principal component analysis method employed in the main article. Section E.7 details the influence of the principal component analysis on the functional derivative predictions. Section E.8 contains a plot of the learning curve. Section E.9 shows computational timings for the various machine learning models.

## E.1. Data sets

Training and test data are supposed to be created as closely as possible to the data used by Snyder et al.[89] in order to clearly demonstrate the improved accuracy achieved by including the functional derivative into the training algorithm. In Section E.1.1 and Section E.1.2 the data sets for $N = 1$ and $N = 2$, respectively, are inspected in greater detail. Additionally, the performance achieved by simple models is investigated as reference for the more complex models explored in the main article. The parameter triplets $a$, $b$, and $c$, used to generate the potentials of the training and test set, are available online in CSV format.

### E.1.1. Data for $N = 1$

The training set consists of a total of $M = 100$ densities. Figure E.1 shows a typical example of an input density as well as a histogram of the corresponding kinetic energies. In order to evaluate the complexity of the data set we fit simple regression models and test their performance on the test set. The results for all of these models are summarized in Table E.1.

The simplest possible model is a constant model:

$$T^{\text{const}} = b \tag{E.1}$$

where the least squares solution for the parameter $b$ is given by the mean kinetic energies of the training examples $b = \sum_j^M T_j/M$. The mean absolute error achieved by this model is known

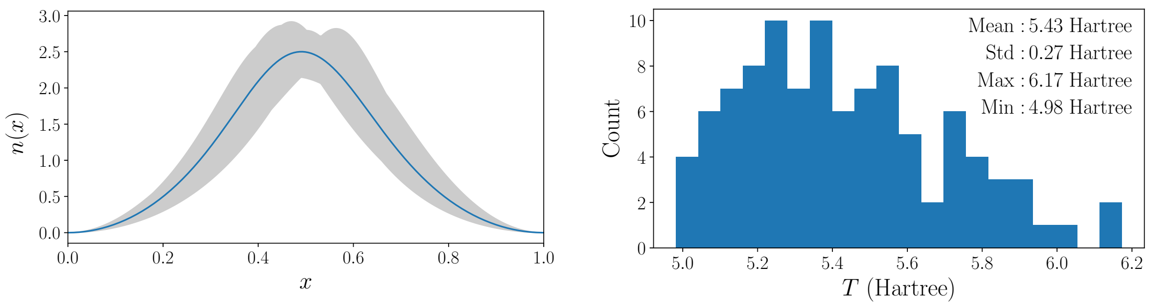


Figure E.1.: Left panel: The shaded region shows the variation of densities in the training set for $N = 1$. The density corresponding to the first potential in the training set is shown as solid line. Right panel: A Histogram and the statistical parameters of the distribution of kinetic energies in the training set.

Table E.1.: Absolute error values for the simple models on the $N = 1$ data set given in kcal/mol.

| | $\lvert\Delta T\rvert$ | | |
|---|---|---|---|
| model | mean | std | max |
| Constant model | 138.3 | 107.3 | 847.4 |
| Linear model | 2429.8 | 9924.9 | 162087.9 |
| Ridge regression | 54.3 | 52.1 | 417.1 |

as mean absolute deviation (MAD) in statistics. A slightly more complex model is given by a standard least squares linear fit, relating the $G = 500$ dimensional input densities to the kinetic energy:

$$T^{\text{linear}}(\mathbf{n}) = \sum_{g}^{G} \mathbf{w}_{(g)}\mathbf{n}_{(g)} = \mathbf{w}^{\top} \cdot \mathbf{n}, \tag{E.2}$$

where the index $(g)$ denotes the $g$-th entry of a vector and the brackets are used to distinguish grid point indices from training example indices. Note that the absolute error for this model is significantly higher than for the constant model. This is most likely due to the fact that 100 training examples are insufficient to fit 500 weight parameters. Ridge regression addresses this problem by introducing an additional regularization term that is used to penalize large weights in the linear fit. Since ridge regression is a linear variant of kernel ridge regression used in the main article, the achieved errors represent an upper bound to the expected performance of KRR. The results are obtained with a regularization parameter of $\lambda = 10^{-6}$ and the ridge regression algorithm as implemented in the scikit-learn Python package.[440]

### E.1.2. Data for $N = 2$

The data set for $N = 2$ used in Section 6.4.3 and Section 6.4.4 of the main article is based on the same potentials as the data presented the previous section. While the addition of a second particle leads to higher values for the total kinetic energy, the standard deviation does not increase significantly. We attribute this to the fact that the second particle is less influenced by the potential and the corresponding wave function resembles that of an particle moving freely in a hard wall box. A summary of this training set is given in Figure E.2.

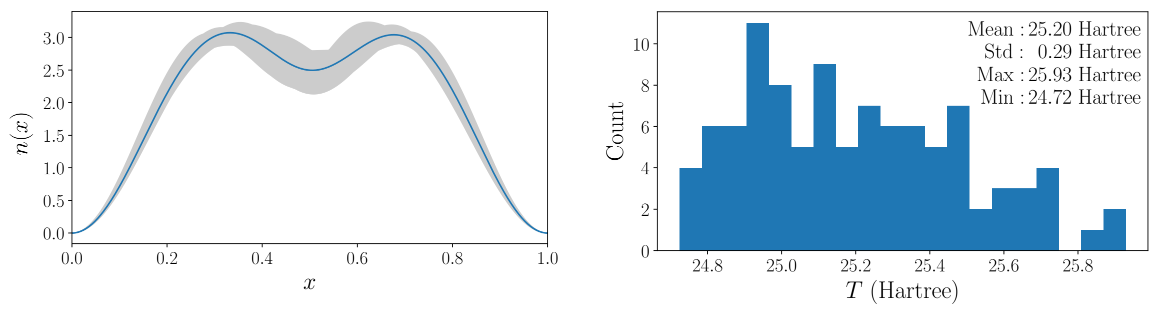


Figure E.2.: Left panel: The shaded region shows the variation of densities in the training set for $N = 2$. The density corresponding to the first potential in the training set is shown as solid line. Right panel: A Histogram and the statistical parameters of the distribution of kinetic energies.

Table E.2 shows the error values achieved by the simple models presented in the previous section. The most notable difference to the results for $N = 1$ is the significant improvement in the accuracy of the linear model.

Table E.2.: Absolute error values (in kcal/mol) achieved by the simple models on the $N = 2$ test set.

| | $\lvert\Delta T\rvert$ | | |
|---|---|---|---|
| model | mean | std | max |
| Constant model | 145.1 | 111.7 | 730.5 |
| Linear model | 121.2 | 409.9 | 6987.7 |
| Ridge regression | 34.6 | 30.8 | 322.3 |

In Section 6.4.3 and Section 6.4.4 of the main article the kinetic energy is given by a machine learning correction to the von Weizsäcker functional:

$$T = T^{\mathrm{ML}} + T^{\mathrm{vW}}. \tag{E.3}$$

The models are therefore not trained on the data set presented in Figure E.2 but rather on the difference between these exact values and the prediction obtained with the von Weizsäcker functional.

In Figure E.3 the kinetic energy predictions of the von Weizsäcker functional and the Thomas-Fermi functional (for the sake of completeness) are plotted versus the exact solutions in order to show that the von Weizsäcker model can not describe systems consisting of two spatial orbitals. While the Thomas-Fermi model roughly captures the correct functional correlation, the predictions by the von Weizsäcker functional exhibit an opposing trend. The accuracy of the Thomas-Fermi model can be improved further by adding a constant term. Fitting this constant offset using the training set yields reduced values of 110.9 kcal/mol, 83.1 kcal/mol,

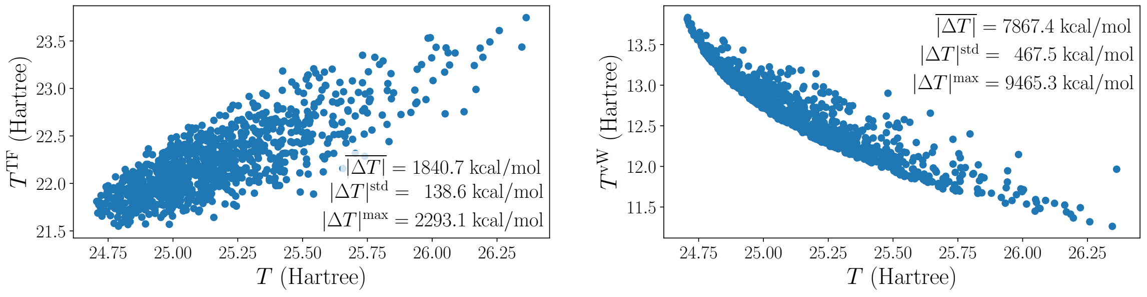


Figure E.3.: Comparison of the exact kinetic energy values of the $N = 2$ test set on the x-axis and the corresponding predictions by the Thomas-Fermi functional (left panel) and the von Weizsäcker functional (right panel) on the y-axis.

and 453.5 kcal/mol for the mean, the standard deviation and the maximum of the absolute error, respectively.

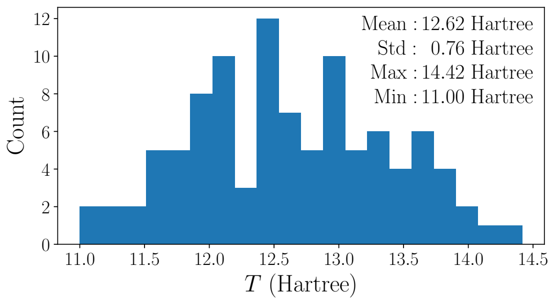


Figure E.4.: Histogram of the kinetic energies in the training set for $N = 2$ after subtracting the von Weizsäcker kinetic energy.

Due to the opposing functional behavior of the von Weizsäcker model, subtracting the von Weizsäcker kinetic energy from the exact values leads to an increased variance in the training data for the machine learning model as depicted in Figure E.4.

Table E.3 shows that this fact is also reflected in the reduced accuracy achieved by the constant model. Note, however, that both linear models achieve significantly lower mean absolute errors. We conclude that the von Weizsäcker term captures some of the non-linear contributions to the kinetic energy in this data set.

Table E.3.: Absolute error values $\Delta T$ (in kcal/mol), achieved by the simple models trained on the $N = 2$ data set after subtracting the von Weizsäcker kinetic energy.

| | $\lvert\Delta T\rvert$ | | |
|---|---|---|---|
| model | mean | std | max |
| Constant model | 378.8 | 279.4 | 1543.5 |
| Linear model | 87.6 | 413.5 | 8067.3 |
| Ridge regression | 12.9 | 13.3 | 113.6 |

## E.2. Kernel ridge regression

In this section we provide a detailed derivation of the Kernel Ridge Regression (KRR) algorithm. In Section E.2.1 the KRR method is reviewed and a standard notation valid in both the supporting material and the main manuscript is introduced. Section E.2.2 shows how this concept can be extended to include training data for the functional derivative. Note that in both sections the formulas are derived in the so-called weight space view and transformed only in the last step to the kernel space expressions used in the main article.

### E.2.1. Standard KRR

The main idea in KRR is that even non-linear data can be described by a linear model after the transformation to a higher dimensional vector space,

$$T^{\mathrm{ML}}\left(\mathbf{n}\right) = \sum_{d}^{D} w_d \phi_d\left(\mathbf{n}\right) = \mathbf{w}^\top \boldsymbol{\phi}\left(\mathbf{n}\right), \tag{E.4}$$

where $w_d$ are the fit coefficients and $\phi$ denotes the transformation from the input space $\mathbb{R}^G$ to a higher dimensional feature space $\mathbb{R}^D$. The weights are determined by minimization of a cost function consisting of the squared error and a regularization term

$$\mathcal{L} = \sum_{j}^{M} \left(T^{\mathrm{ML}}(\mathbf{n}_j) - T_j\right)^2 + \lambda \sum_{d}^{D} w_d^2, \tag{E.5}$$

where the parameter $\lambda$ controls the regularization strength. The cost function is minimized by setting the derivative with respect to $w_k$ equal to zero:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_k} &= 2 \sum_{j}^{M} \left( \sum_{d}^{D} w_d \phi_d\left(\mathbf{n}_j\right) - T_j \right) \phi_k\left(\mathbf{n}_j\right) + 2\lambda w_k \overset{!}{=} 0 \\ \sum_{j}^{M} \sum_{d}^{D} w_d \phi_d\left(\mathbf{n}_j\right) \phi_k\left(\mathbf{n}_j\right) + \lambda w_k &= \sum_{j}^{M} T_j \phi_k\left(\mathbf{n}_j\right). \end{aligned} \tag{E.6}$$

Derivation with respect to all weights gives a total of $D$ such equations which can be rewritten as a matrix equation:

$$\left(\Phi\Phi^\top + \lambda \mathbf{I}_D\right) \mathbf{w} = \Phi \mathbf{T}, \tag{E.7}$$

where $\Phi$ is a $(D \times M)$ matrix whose columns contain the transformed input densities and $\mathbf{I}_D$ is a unit matrix of size $(D \times D)$. The weights can be calculated by inversion of the matrix on the left hand side:

$$\mathbf{w} = \left(\Phi\Phi^\top + \lambda \mathbf{I}_D\right)^{-1} \Phi \mathbf{T}. \tag{E.8}$$

This expression for $\mathbf{w}$ can be rearranged further by the following matrix identity, often referred to as push-through identity:[439, 441, 442]

$$\left(A + P^\top R^{-1} Q\right)^{-1} P^\top R^{-1} = A^{-1} P^\top \left(R + Q A^{-1} P^\top\right)^{-1}. \tag{E.9}$$

Setting $A = \lambda \mathbf{I}_D$, $P = Q = \Phi^\top$ and $R^{-1} = \mathbf{I}_M$ yields:

$$\mathbf{w} = \Phi \left( \Phi^\top \Phi + \lambda \mathbf{I}_M \right)^{-1} \mathbf{T}. \tag{E.10}$$

One of the advantages of this rearrangement is that the costly matrix inversion can be performed either on the $(D \times D)$ matrix in Equation E.8 or the $(M \times M)$ matrix in Equation E.10. As an analog to the $D$-dimensional weight vector $\mathbf{w}$ the $M$-dimensional vector $\boldsymbol{\alpha}$ is defined as:

$$\mathbf{w} = \sum_j \alpha_j \phi\left(\mathbf{n}_j\right) \quad \text{with} \quad \boldsymbol{\alpha} = \left( \Phi^\top \Phi + \lambda \mathbf{I}_M \right)^{-1} \mathbf{T}. \tag{E.11}$$

Plugging the weight vector back into the linear model in Equation E.4 yields:

$$T^{\mathrm{ML}}\left(\mathbf{n}\right) = \mathbf{T} \left( \Phi^\top \Phi + \lambda \mathbf{I}_M \right)^{-1} \Phi^\top \boldsymbol{\phi}\left(\mathbf{n}\right) = \boldsymbol{\alpha}^\top \Phi^\top \boldsymbol{\phi}\left(\mathbf{n}\right). \tag{E.12}$$

Another advantage of applying the push-through identity is that it allows for the so-called kernel trick, where the scalar product in feature space is replaced with a kernel function $\phi(\mathbf{n}_i)^\top \phi(\mathbf{n}_j) = k\left(\mathbf{n}_i, \mathbf{n}_j\right)$. After defining the kernel matrix $K = \Phi^\top \Phi$ with elements $K_{ij} = k\left(\mathbf{n}_i, \mathbf{n}_j\right)$ and a vector $\mathbf{k}(\mathbf{n}) = \Phi^\top \boldsymbol{\phi}\left(\mathbf{n}\right)$ with elements $k_j(\mathbf{n}) = k\left(\mathbf{n}_j, \mathbf{n}\right)$ the final result can be written as:

$$T^{\mathrm{ML}}\left(\mathbf{n}\right) = \mathbf{T}\left(K + \lambda \mathbf{I}\right)^{-1} \mathbf{k}\left(\mathbf{n}\right) = \boldsymbol{\alpha}^\top \mathbf{k}\left(\mathbf{n}\right) = \sum_j^M \alpha_j k\left(\mathbf{n}_j, \mathbf{n}\right). \tag{E.13}$$

As shown by Snyder et al.[89] the corresponding prediction for the functional derivative is given by:

$$\frac{\nabla_{\mathbf{n}} T^{\mathrm{ML}}(\mathbf{n})}{\Delta x} = \sum_j^M \frac{\alpha_j}{\Delta x} \nabla_{\mathbf{n}} k\left(\mathbf{n}_j, \mathbf{n}\right). \tag{E.14}$$

Note that the division by $\Delta x$ in the discretized functional derivative is necessary to eliminate the dependence on the number of grid points $G$.

## E.2.2. Including derivative information

In a similar fashion, taking the derivative of Equation E.4 with respect to the input densities, yields the prediction of the functional derivative in weight space:

$$\frac{\nabla_{\mathbf{n}} T^{\mathrm{ML}}(\mathbf{n})}{\Delta x} = \sum_d^D \frac{w_d}{\Delta x} \nabla_{\mathbf{n}} \phi_d\left(\mathbf{n}\right). \tag{E.15}$$

The $g$-th component of the gradient is denoted as:

$$\left( \frac{\nabla_{\mathbf{n}} T^{\mathrm{ML}}(\mathbf{n})}{\Delta x} \right)_{(g)} = \sum_d^D \frac{w_d}{\Delta x} \Big( \nabla_{\mathbf{n}} \phi_d\left(\mathbf{n}\right) \Big)_{(g)} = \sum_d^D \frac{w_d}{\Delta x} \frac{\partial \phi_d\left(\mathbf{n}\right)}{\partial \mathbf{n}_{(g)}}, \tag{E.16}$$

where the brackets are used to distinguish grid point indices from training example indices. Using this expression, the cost function can be extended to include the squared error of the functional derivative weighted by an additional hyperparameter $\kappa$:

$$\mathcal{L} = \sum_j^M \left( T^{\mathrm{ML}}(\mathbf{n}_j) - T_j \right)^2 + \frac{\kappa}{G} \sum_j^M \sum_g^G \left( \left( \frac{\nabla_{\mathbf{n}_j} T^{\mathrm{ML}}(\mathbf{n}_j)}{\Delta x} \right)_{(g)} - \left( \frac{\nabla_{\mathbf{n}_j} T_j}{\Delta x} \right)_{(g)} \right)^2 + \lambda ||\mathbf{w}||^2, \tag{E.17}$$

where $\frac{\nabla_{\mathbf{n}_j} T_j}{\Delta x}$ are the reference vectors of the discretized functional derivative. The weights are determined by setting the derivative with respect to $w_k$ equal to zero:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_k} =& 2\sum_j^M \left(\sum_d^D w_d \phi_d\left(\mathbf{n}_j\right) - T_j\right)\phi_k\left(\mathbf{n}_j\right)\\ &+ 2\frac{\kappa}{G}\sum_j^M\sum_g^G \left(\sum_d^D \frac{w_d}{\Delta x}\frac{\partial \phi_d\left(\mathbf{n}_j\right)}{\partial \mathbf{n}_{j,(g)}} - \left(\frac{\nabla_{\mathbf{n}_j} T_j}{\Delta x}\right)_{(g)}\right)\frac{1}{\Delta x}\frac{\partial \phi_k\left(\mathbf{n}_j\right)}{\partial \mathbf{n}_{j,(g)}} + 2\lambda w_k \overset{!}{=} 0.\end{aligned} \tag{E.18}$$

Collecting all terms proportional to the weights on the left hand side yields

$$\begin{aligned}&\sum_j^M\sum_d^D \left(w_d \phi_d\left(\mathbf{n}_j\right)\phi_k\left(\mathbf{n}_j\right) + \frac{\kappa}{G}\sum_g^G \frac{w_d}{(\Delta x)^2}\frac{\partial \phi_d\left(\mathbf{n}_j\right)}{\partial \mathbf{n}_{j,(g)}}\frac{\partial \phi_k\left(\mathbf{n}_j\right)}{\partial \mathbf{n}_{j,(g)}}\right) + \lambda w_k\\ &= \sum_j^M \left(T_j \phi_k\left(\mathbf{n}_j\right) + \frac{\kappa}{G}\sum_g^G \left(\frac{\nabla_{\mathbf{n}_j} T_j}{\Delta x}\right)_{(g)}\frac{1}{\Delta x}\frac{\partial \phi_k\left(\mathbf{n}_j\right)}{\partial \mathbf{n}_{j,(g)}}\right).\end{aligned} \tag{E.19}$$

By extending the matrices defined in Equation E.7 this can again be rewritten as a matrix equation

$$\left(\Phi_{\text{ext}}\ S\ \Phi_{\text{ext}}^\top + \lambda \mathbf{I}_D\right)\mathbf{w} = \Phi_{\text{ext}}\ S\ \mathbf{T}_{\text{ext}}, \tag{E.20}$$

with the extended transformation matrix of shape $(D \times M(1+G))$:

$$\Phi_{\text{ext}} = \begin{pmatrix} \phi_1(\mathbf{n}_1) & \dots & \phi_1(\mathbf{n}_M) & \frac{\nabla_{\mathbf{n}_1}^\top \phi_1(\mathbf{n}_1)}{\Delta x} & \dots & \frac{\nabla_{\mathbf{n}_M}^\top \phi_1(\mathbf{n}_M)}{\Delta x}\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ \phi_D(\mathbf{n}_1) & \dots & \phi_D(\mathbf{n}_M) & \frac{\nabla_{\mathbf{n}_1}^\top \phi_D(\mathbf{n}_1)}{\Delta x} & \dots & \frac{\nabla_{\mathbf{n}_M}^\top \phi_D(\mathbf{n}_M)}{\Delta x}\end{pmatrix}, \tag{E.21}$$

the extended target vector:

$$\mathbf{T}_{\text{ext}} = \begin{pmatrix} T_1 & \dots & T_M & \frac{\nabla_{\mathbf{n}_1}^\top T_1}{\Delta x} & \dots & \frac{\nabla_{\mathbf{n}_M}^\top T_M}{\Delta x}\end{pmatrix}^\top, \tag{E.22}$$

and a $(M(1+G) \times M(1+G))$ diagonal matrix containing the scaling factor for the relative importance of derivative information:

$$S = \begin{pmatrix} \mathbf{I}_M & 0\\ 0 & \frac{\kappa}{G}\mathbf{I}_{MG}\end{pmatrix}. \tag{E.23}$$

Solving Equation E.20 for the weight vector $\mathbf{w}$ and applying the push-through identity of Equation E.9 by setting $A = \lambda \mathbf{I}_D$, $P = Q = \Phi_{\text{ext}}^\top$ and $R^{-1} = S$ yields

$$\mathbf{w} = \left(\Phi_{\text{ext}}\ S\ \Phi_{\text{ext}}^\top + \lambda \mathbf{I}_D\right)^{-1}\Phi_{\text{ext}}\ S\ \mathbf{T}_{\text{ext}} = \Phi_{\text{ext}}\left(\Phi_{\text{ext}}^\top \Phi_{\text{ext}} + \Lambda\right)^{-1}\mathbf{T}_{\text{ext}}, \tag{E.24}$$

with the regularization matrix

$$\Lambda = \lambda S^{-1} = \begin{pmatrix} \lambda \mathbf{I}_M & 0\\ 0 & \frac{\lambda G}{\kappa}\mathbf{I}_{MG}\end{pmatrix}. \tag{E.25}$$

The weights are again rewritten as

$$\mathbf{w}^\top = \sum_j \alpha_j \phi(\mathbf{n}_j)^\top + \frac{\boldsymbol{\beta}_j^\top \cdot \left(\nabla_{\mathbf{n}_j} \phi(\mathbf{n}_j)^\top\right)}{\Delta x}, \tag{E.26}$$

where the coefficients $\alpha$ and $\beta$ are determined by solving the matrix equation

$$\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_M \\ \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_M \end{pmatrix} = \left(\Phi_{\text{ext}}^\top \Phi_{\text{ext}} + \Lambda\right)^{-1} \begin{pmatrix} T_1 \\ \vdots \\ T_M \\ \frac{\nabla_{\mathbf{n}_1} T_1}{\Delta x} \\ \vdots \\ \frac{\nabla_{\mathbf{n}_M} T_M}{\Delta x} \end{pmatrix} = \left(K_{\text{ext}} + \Lambda\right)^{-1} \mathbf{T}_{\text{ext}}. \tag{E.27}$$

Note that the individual $\beta_j$ are vectors of length $G$. The extended kernel matrix $K_{\text{ext}}$ is calculated by applying the kernel trick to the extended transformation matrix $\Phi_{\text{ext}}$:

$$K_{\text{ext}} = \begin{pmatrix} K & J' \\ J & H \end{pmatrix} =$$

$$= \begin{pmatrix} k(\mathbf{n}_1, \mathbf{n}_1) & \dots & k(\mathbf{n}_1, \mathbf{n}_M) & \frac{\nabla_{\mathbf{n}_1}^\top k(\mathbf{n}_1, \mathbf{n}_1)}{\Delta x} & \dots & \frac{\nabla_{\mathbf{n}_M}^\top k(\mathbf{n}_1, \mathbf{n}_M)}{\Delta x} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{n}_M, \mathbf{n}_1) & \dots & k(\mathbf{n}_M, \mathbf{n}_M) & \frac{\nabla_{\mathbf{n}_1}^\top k(\mathbf{n}_M, \mathbf{n}_1)}{\Delta x} & \dots & \frac{\nabla_{\mathbf{n}_M}^\top k(\mathbf{n}_M, \mathbf{n}_M)}{\Delta x} \\ \frac{\nabla_{\mathbf{n}_1} k(\mathbf{n}_1, \mathbf{n}_1)}{\Delta x} & \dots & \frac{\nabla_{\mathbf{n}_1} k(\mathbf{n}_1, \mathbf{n}_M)}{\Delta x} & \frac{\nabla_{\mathbf{n}_1} \cdot \nabla_{\mathbf{n}_1}^\top k(\mathbf{n}_1, \mathbf{n}_1)}{(\Delta x)^2} & \dots & \frac{\nabla_{\mathbf{n}_1} \cdot \nabla_{\mathbf{n}_M}^\top k(\mathbf{n}_1, \mathbf{n}_M)}{(\Delta x)^2} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\nabla_{\mathbf{n}_M} k(\mathbf{n}_M, \mathbf{n}_1)}{\Delta x} & \dots & \frac{\nabla_{\mathbf{n}_M} k(\mathbf{n}_M, \mathbf{n}_M)}{\Delta x} & \frac{\nabla_{\mathbf{n}_M} \cdot \nabla_{\mathbf{n}_1}^\top k(\mathbf{n}_M, \mathbf{n}_1)}{(\Delta x)^2} & \dots & \frac{\nabla_{\mathbf{n}_M} \cdot \nabla_{\mathbf{n}_M}^\top k(\mathbf{n}_M, \mathbf{n}_M)}{(\Delta x)^2} \end{pmatrix} \tag{E.28}$$

Using the new weights from Equation E.26 for the prediction of the kinetic energy of a previously unseen density $\mathbf{n}$ in Equation E.4 yields:

$$T^{\text{ML}}(\mathbf{n}) = \sum_j^M \alpha_j k(\mathbf{n}_j, \mathbf{n}) + \sum_j^M \sum_g^G \frac{\beta_{j,(g)}}{\Delta x} \frac{\partial k(\mathbf{n}_j, \mathbf{n})}{\partial \mathbf{n}_{j,(g)}} = \sum_j^M \alpha_j k(\mathbf{n}_j, \mathbf{n}) + \sum_j^M \frac{\boldsymbol{\beta}_j^\top \cdot \nabla_{\mathbf{n}_j} k(\mathbf{n}_j, \mathbf{n})}{\Delta x}. \tag{E.29}$$

The corresponding prediction of the derivative is given by:

$$\left(\frac{\nabla_{\mathbf{n}} T^{\text{ML}}(\mathbf{n})}{\Delta x}\right)_{(g)} = \sum_j^M \frac{\alpha_j}{\Delta x} \frac{\partial k(\mathbf{n}_j, \mathbf{n})}{\partial \mathbf{n}_{(g)}} + \sum_j^M \sum_{g'}^G \frac{\boldsymbol{\beta}_{j,(g')}}{(\Delta x)^2} \frac{\partial^2 k(\mathbf{n}_j, \mathbf{n})}{\partial \mathbf{n}_{j,(g')} \partial \mathbf{n}_{(g)}}, \tag{E.30}$$

or similarly in the vector notation used in the main article by:

$$\frac{\nabla_{\mathbf{n}}^\top T^{\text{ML}}(\mathbf{n})}{\Delta x} = \sum_j^M \frac{\alpha_j}{\Delta x} \nabla_{\mathbf{n}}^\top k(\mathbf{n}_j, \mathbf{n}) + \sum_j^M \frac{\boldsymbol{\beta}_j^\top \cdot \left(\nabla_{\mathbf{n}_j} \cdot \nabla_{\mathbf{n}}^\top k(\mathbf{n}_j, \mathbf{n})\right)}{(\Delta x)^2}. \tag{E.31}$$

## E.3. Kernel ridge regression hyperparameter search

The hyperparameters for the KRR models are determined using 5-fold cross validation on a rectangular search grid. A total of 29 points, log-uniformly distributed between 10 and 500, are used for the length scale parameter $\sigma$, and 5 points, log-uniformly distributed between $10^{-10}$ and $10^{-14}$, for the regularization parameter $\lambda$. Both the mean absolute error for the kinetic energy and the functional derivative are evaluated using cross validation. The hyperparameters are chosen such that the sum of these two errors is minimized. Note that this choice is biased toward the derivative score as the mean absolute error on the derivative is typically significantly larger.

As a first test we investigate if the extended KRR model yields similar tendencies for the hyperparameters when the number of training examples is increased as reported for standard KRR in Ref. 89. Figure E.5 shows that the rough grid search in fact displays a similar trend toward smaller values for both $\sigma$ and $\lambda$ for an increasing training set size $M$. As expected

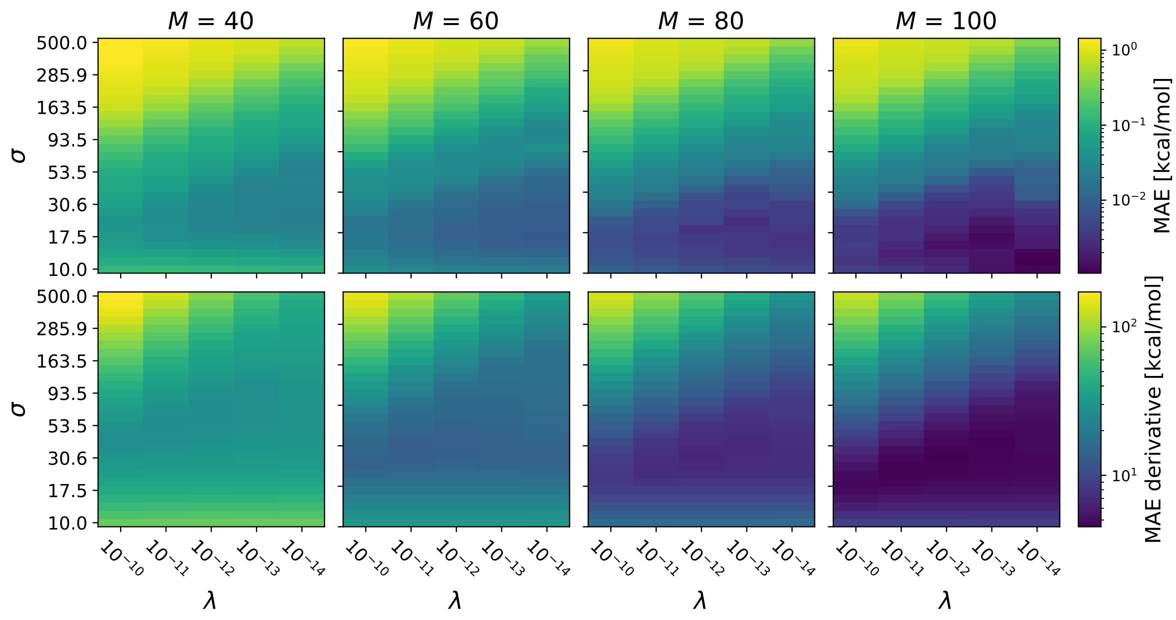


Figure E.5.: Mean absolute error of the kinetic energy (top row) and the functional derivative (bottom row) as a function of the hyperparameters $\sigma$ and $\lambda$ for an increasing number of training examples $M$ and $\kappa = 1$.

for a machine learning model, both errors decrease with the number of training examples. Chemical accuracy, defined as a mean absolute error for the kinetic energy below 1 kcal/mol, is already reached using $M = 40$ training examples for a broad range of hyperparameters. This improvement over the $M = 80$ necessary training examples reported by Snyder et al.[89] is attributed to the inclusion of derivative information. The lowest summed error on the $M = 100$ set is achieved for $\sigma = 30.58$ and $\lambda = 10^{-12}$.

The influence of the weighting parameter $\kappa$ is investigated by repeating the grid search on $M = 100$ training examples for various values of $\kappa$. Figure E.6 shows the cross validation score for both the kinetic energy and the functional derivative for $\kappa \in \{0.1, 1, 10, 100, 1000\}$. Note that the blank regions in Figure E.6 denote areas where the matrix inversion in Equation E.24,

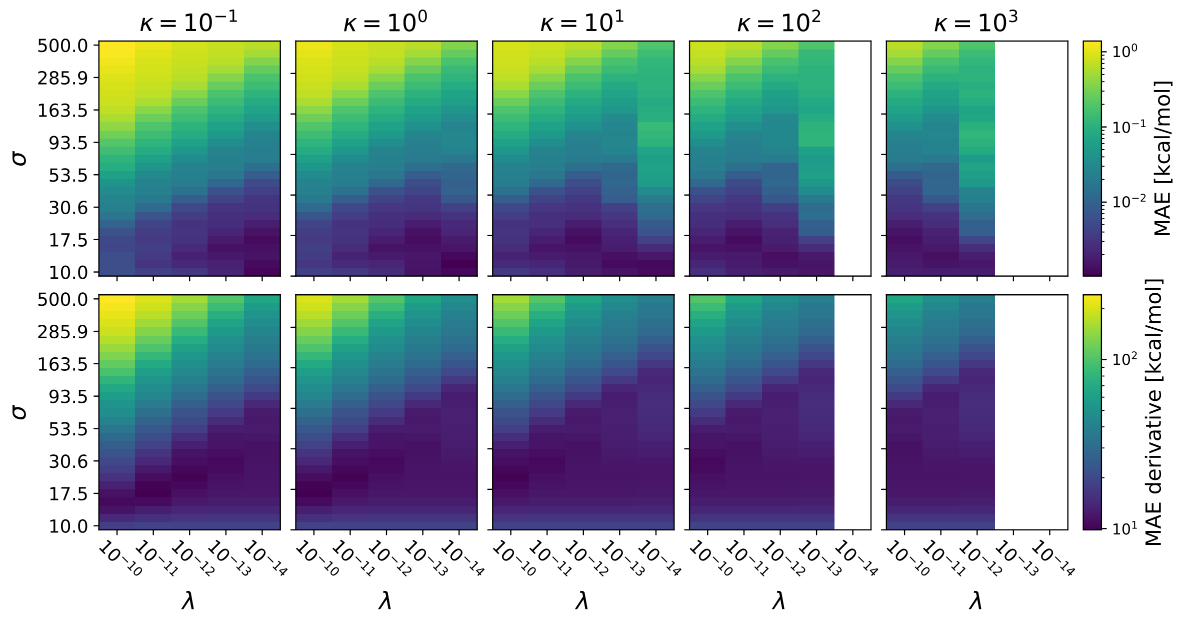


Figure E.6.: Mean absolute error of the kinetic energy (top row) and the functional derivative (bottom row) for the different values of the hyperparameters $\kappa$, $\sigma$ and $\lambda$ on the $M = 100$, $N = 1$ training set.

more precisely the Cholesky factorization used to solve Equation E.27, failed due to numerical noise. As expected, the larger values of $\kappa$ lead to an increased parameter range with low errors on the functional derivative and conversely a decreased parameter range with the lowest errors for the kinetic energy. Note, however, that the parameter region in which chemical accuracy is reached increases for large $\kappa$ values. The overall lowest summed error is reached for $\kappa = 0.1$, $\sigma = 17.49$ and $\lambda = 10^{-11}$. Nevertheless, the weighting parameter is set to $\kappa = 1$ for all of the hyperparameter investigations, since it is difficult to judge the relative importance of the errors for the presented application.

Figure E.7 shows the results of the hyperparameter search for the $N = 2$ data set. The best performance is achieved for values of $\sigma = 35.16$, $\lambda = 10^{-12}$ and $\sigma = 26.59$, $\lambda = 10^{-12}$ for the standard and extended KRR models, respectively. The hyperparameters for standard KRR are chosen based solely on the MAE of the kinetic energy and a value of $\kappa = 1.0$ is used for extended KRR. Again, the inclusion of derivative information significantly improves the error on the kinetic energy for a wide range of hyperparameters, yielding chemical accuracy on all points of the hyperparameter grid.

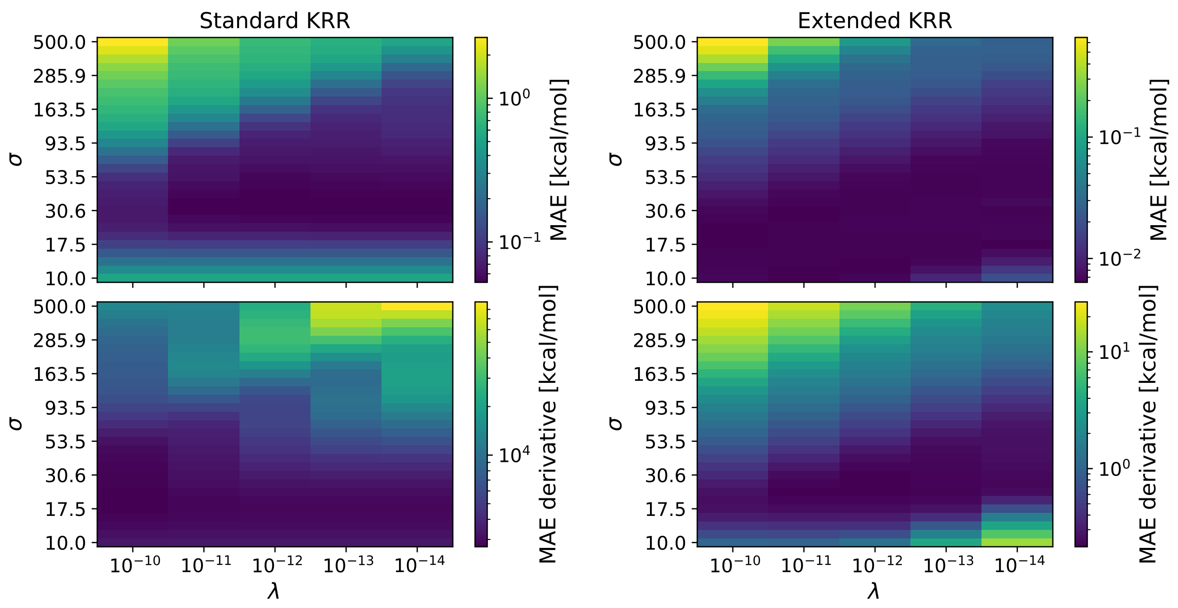


Figure E.7.: Cross validation scores of the grid search for both the standard KRR model (left) and the extended KRR model (right) on the $N = 2$ training set.

## E.4. Neural network hyperparameters and training behavior

This section summarizes all the hyperparameters used during the training of the convolutional neural network models for the sake of reproducibility, but without discussing their influence on the actual training behavior or the final model performance.

The neural network models are trained by minimizing the following cost function:

$$\begin{aligned}\mathcal{L} =& \frac{\iota_T}{M}\sum_j^M ||T^{\mathrm{ML}}(\mathbf{n}_j) - T_j||^2 + \frac{\iota_\tau}{MG}\sum_j^M ||\boldsymbol{\tau}^{\mathrm{ML}}(\mathbf{n}_j) - \boldsymbol{\tau}_j||^2 \\ &+ \frac{\kappa}{MG}\sum_j^M \left\| \frac{\nabla_{\mathbf{n}_j} T^{\mathrm{ML}}(\mathbf{n}_j)}{\Delta x} - \frac{\nabla_{\mathbf{n}_j} T_j}{\Delta x} \right\|^2 + \lambda ||\mathbf{w}||^2.\end{aligned} \tag{E.32}$$

Most choices for the hyperparameters are shared by all of the models. The remaining hyperparameters are listed in Table E.4. Since the relative weighting of contributions in the cost function is over determined by four scaling factors, the weighting parameter $\kappa$ is set to a fixed value of $\kappa = 1$. The network parameters are determined with the Adam optimizer,[320] a variation of the steepest descent algorithm, with the tensorflow[319] default hyperparameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. The gradient of the cost function with respect to the weights and biases is calculated using a batch size of 100 examples. This corresponds to the whole training set for the investigations in Section 6.4.1 through Section 6.4.3 of the main article. If the norm of the gradient surpasses a threshold value of 100 it is rescaled to a length of 100. This process is referred to as *clip by norm* in tensorflow.

Starting from an initial learning rate of $10^{-4}$ the learning rate stays constant for the first $N_{\mathrm{constant}}$ training steps and is then reduced by a factor 0.9 every $N_{\mathrm{decay}}$ steps. Figure E.8 shows the learning rate schedule used for training the convolutional neural networks. The number of

Table E.4.: Summary of the hyperparameters used for training of the neural networks.

| model | N | data set[a] | vW[b] | M | epochs | $\iota_T$ | $\iota_\tau$ | $\lambda$ | $N_{\text{constant}}$ | $N_{\text{decay}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| CNN | 1 | recreated | - | 100 | 100 000 | 0.2 | - | $2.5 \cdot 10^{-4}$ | 21 800 | 1000 |
| ResNet | 1 | recreated | - | 100 | 100 000 | 0 | 1 | $2.5 \cdot 10^{-4}$ | 21 800 | 1000 |
| ResNet | 2 | recreated | Yes | 100 | 100 000 | 0 | 1 | $2.5 \cdot 10^{-4}$ | 21 800 | 1000 |
| ResNet | 2 | generated | Yes | 1000 | 30 000 | 0 | 1 | $2.5 \cdot 10^{-5}$ | 40 000 | 2000 |
| ResNet | 2 | generated | Yes | 10 000 | 3000 | 0 | 1 | $2.5 \cdot 10^{-5}$ | 40 000 | 2000 |
| ResNet | 2 | generated | Yes | 100 000 | 300 | 0 | 1 | $1.0 \cdot 10^{-7}$ | 40 000 | 2000 |

[a] *recreated* refers to the fact that the parameters for the potentials are taken from Ref. 89 whereas *generated* refers to new randomly generated parameters.
[b] Denotes that the von Weizsäcker predictions have been subtracted before training.

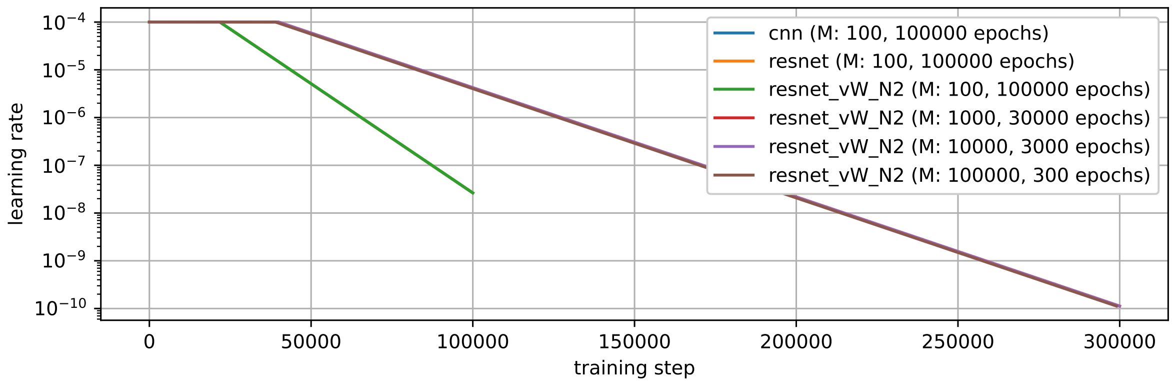


Figure E.8.: Plot of the learning rate schedule.

training steps on the x-axis refers to the number of weight updates and is given by the number of epochs times $M$ divided by the batch size.

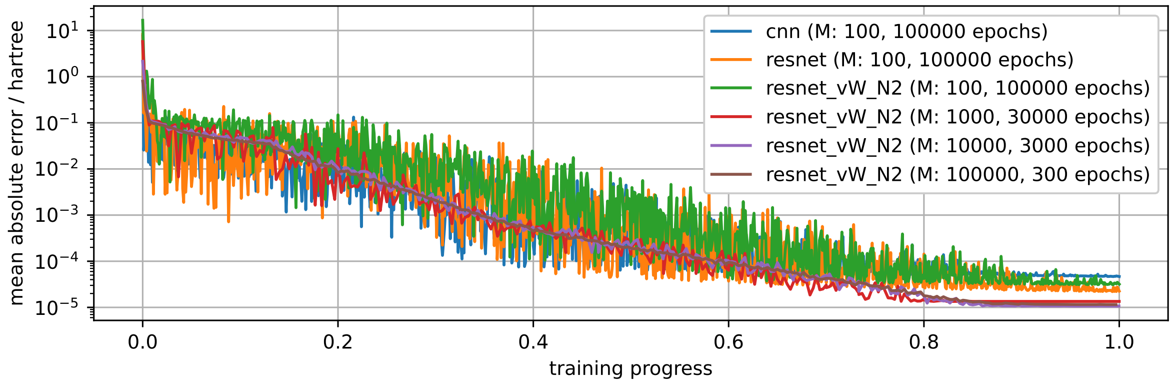


Figure E.9.: Plot of the mean absolute error of the kinetic energy during the training of the neural network models.

In Figure E.9 the mean absolute errors for the kinetic energy on the training set are plotted as a function of the training progress. It shows that in addition to the lower final error reached by

ResNets trained on larger data sets, the variation in the error curve during the training is also reduced. For all of the presented models the final training error in Figure E.9 is significantly lower than the reported test errors. The generalization properties can be improved by using additional training data as shown in Section 6.4.4of the main article.

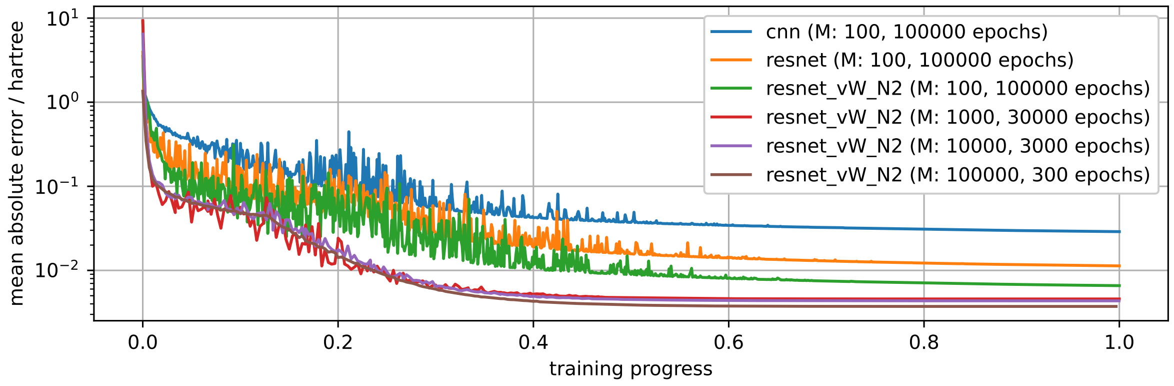


Figure E.10.: Plot of the mean absolute error of the kinetic energy derivative during the training of the neural network models.

Figure E.10 shows the mean absolute error for the functional derivative as a function of the training process. Comparing Figure E.9 and Figure E.10 shows that the early stages of the training are dominated by a reduction in the derivative error. In part due to the decreasing learning rate, the derivative error stops improving roughly at the half-way point of the training process while the kinetic energy error keeps decreasing.

## E.5. Finding minimum energy densities

### E.5.1. Iteration on the density and local principal component analysis

The direct minimization algorithm used to find the minimum energy density is a standard steepest descent optimization using a Lagrange multiplier $\mu$ to ensure the correct number of particles $N$. The corresponding Lagrange function is given by:

$$\mathcal{L}[n] = E[n] - \mu \left( \int n \ \mathrm{d}x - N \right), \tag{E.33}$$

where the total energy functional $E[n]$ for non-interacting particles is simply the sum of kinetic energy $T[n]$ and potential energy:

$$E[n] = T[n] + \int nV \ \mathrm{d}x. \tag{E.34}$$

The steepest descent update rule with step size $\eta$ is given by:

$$n_{i+1} = n_i - \eta \frac{\delta \mathcal{L}}{\delta n} = n_i - \eta \left( \frac{\delta T[n]}{\delta n} + V - \mu \right), \tag{E.35}$$

where the Lagrange multiplier $\mu$ is chosen such that the norm $\int n_{i+1} \ \mathrm{d}x = N$ is ensured. Assuming that $n_i$ is properly normalized, this is equivalent to

$$\int \left( \frac{\delta T[n]}{\delta n} + V - \mu \right) \ \mathrm{d}x \overset{!}{=} 0 \tag{E.36}$$

$$\mu = \frac{\int \left( \frac{\delta T}{\delta n} + V \right) \ \mathrm{d}x}{\int 1 \ \mathrm{d}x}. \tag{E.37}$$

The introduction of a projection operator $P$ for the functional derivative as suggested by Snyder et al.:[89]

$$n_{i+1} = n_i - \eta P \left[ \frac{\delta T[n]}{\delta n} + V - \mu \right], \tag{E.38}$$

yields a similar result for the Lagrange multiplier $\mu$:

$$\mu = \frac{\int P \left[ \frac{\delta T[n]}{\delta n} + V \right] \ \mathrm{d}x}{\int P[1] \ \mathrm{d}x}. \tag{E.39}$$

Note that the local principal component analysis (PCA) of Ref. 89 uses the difference of the density at the current step $n_i$ and a subset of the training densities as basis for the projection operator. Since the integral over the difference of two valid densities is zero, the same will be true for all functions projected onto this subspace. The correct norm for the density $n_{i+1}$ is therefore ensured without the need for a Lagrange multiplier.

### E.5.2. Iteration on the variable $\varphi$

In Section 6.4.3 and Section 6.4.4 of the main article a basis of sine functions is used to restrict the search space instead of the local PCA. This necessitates additionally enforcing a non-negativity constraint, which is typically achieved by iterating on the variable $\varphi = \sqrt{n}$ instead of the density $n$. The steepest descent update rule for $\varphi$ is given by:

$$\varphi_{i+1} = \varphi_i - \eta \frac{\delta \mathcal{L}}{\delta \varphi_i} = \varphi_i - \eta \frac{\delta \mathcal{L}}{\delta n_i} \frac{\delta n_i}{\delta \varphi_i} = \varphi_i - 2\eta\varphi_i \left( \frac{\delta T}{\delta n} + V - \mu \right), \tag{E.40}$$

where the Lagrange multiplier $\mu$ is again chosen such that the norm $\int n_{i+1}\ \mathrm{d}x = \int \varphi_{i+1}^2\ \mathrm{d}x = N$ is ensured:

$$\begin{aligned}\int \varphi_{i+1}^2\ \mathrm{d}x &= \int \left(\varphi_i - 2\eta\varphi_i\left(\frac{\delta T}{\delta n} + V - \mu\right)\right)^2\ \mathrm{d}x \\ &= \int \varphi_i^2\ \mathrm{d}x - 4\eta\int\varphi_i^2\left(\frac{\delta T}{\delta n} + V - \mu\right)\ \mathrm{d}x + 4\eta^2\int\varphi_i^2\left(\frac{\delta T}{\delta n} + V - \mu\right)^2\ \mathrm{d}x.\end{aligned} \tag{E.41}$$

Assuming that $\varphi_i$ is properly normalized $\int \varphi_i^2\ \mathrm{d}x = \int \varphi_{i+1}^2\ \mathrm{d}x = N$ this simplifies to:

$$\begin{aligned}-4\eta\int\varphi_i^2\left(\frac{\delta T}{\delta n} + V - \mu\right)\ \mathrm{d}x + 4\eta^2\int\varphi_i^2\left(\frac{\delta T}{\delta n} + V - \mu\right)^2\ \mathrm{d}x &\overset{!}{=} 0 \\ \int\varphi_i^2\left(\frac{\delta T}{\delta n} + V - \mu\right)\ \mathrm{d}x &= \eta\int\varphi_i^2\left(\frac{\delta T}{\delta n} + V - \mu\right)^2\ \mathrm{d}x.\end{aligned} \tag{E.42}$$

Since the step size $\eta$ is an arbitrary positive number the integrals on both sides have to be equal to zero, which yields

$$\mu = \frac{\int \varphi_i^2\left(\frac{\delta T}{\delta n} + V\right)\ \mathrm{d}x}{\int\varphi_i^2\ \mathrm{d}x}. \tag{E.43}$$

Inspired by the local PCA method, the basis of sine functions is introduced using a projection operator acting on the functional derivative:

$$\varphi_{i+1} = \varphi_i - \eta P\left[\frac{\delta\mathcal{L}}{\delta\varphi_i}\right] = \varphi_i - \eta P\left[2\varphi_i\left(\frac{\delta T}{\delta n} + V - \mu\right)\right]. \tag{E.44}$$

The norm of $n_{i+1}$ is then given by

$$\begin{aligned}\int\varphi_{i+1}^2\ \mathrm{d}x = &\int\varphi_i^2\ \mathrm{d}x - 2\int\eta\varphi_i P\left[2\varphi_i\left(\frac{\delta T}{\delta n} + V - \mu\right)\right]\ \mathrm{d}x \\ &+ \int\eta^2\left(P\left[2\varphi_i\left(\frac{\delta T}{\delta n} + V - \mu\right)\right]\right)^2\ \mathrm{d}x.\end{aligned} \tag{E.45}$$

Again assuming that $\int \varphi_i^2\ \mathrm{d}x = \int \varphi_{i+1}^2\ \mathrm{d}x = N$ yields:

$$\int\varphi_i P\left[\varphi_i\left(\frac{\delta T}{\delta n} + V - \mu\right)\right]\ \mathrm{d}x = \eta\int\left(P\left[\varphi_i\left(\frac{\delta T}{\delta n} + V - \mu\right)\right]\right)^2\ \mathrm{d}x. \tag{E.46}$$

Using the same arguments as before and the fact that $P$ is a linear operator the multiplier $\mu$ can be calculated via

$$\mu = \frac{\int\varphi_i P\left[\varphi_i\left(\frac{\delta T}{\delta n} + V\right)\right]\ \mathrm{d}x}{\int\varphi_i P\left[\varphi_i\right]\ \mathrm{d}x}. \tag{E.47}$$

## E.6. Kernel ridge regression with offset

The main problem of the iterative calculation of minimum energy densities is that the search algorithm is likely to leave the valid region, i.e. the region where the machine learning model offers correct predictions. Snyder et al.[89] solved this by restricting the search space using a projection onto the subspace of training densities. Denzel and Kästner[163] suggested a different solution for a similar problem in machine learning-accelerated molecular geometry optimization. The extrapolation behavior of kernel-based machine learning models can be tuned by introducing a constant offset:

$$T^{\mathrm{ML}}(\mathbf{n}) = b + \sum_{j}^{M} \alpha_j k(\mathbf{n}_j, \mathbf{n}). \tag{E.48}$$

In Figure E.11 the effect of this offset term is demonstrated for a one-dimensional example. The models predict an output value of $b$ for examples far away from the training data. Using larger values, therefore, offers the possibility of introducing an energy penalty for these out-of-training examples and thereby ensuring that the iterative search is restricted to the region spanned by the training examples. Note that this one-dimensional model trained on $M = 4$ examples does not display the problems encountered in the $G = 500$ dimensional model trained on just $M = 100$ examples discussed in the main article.

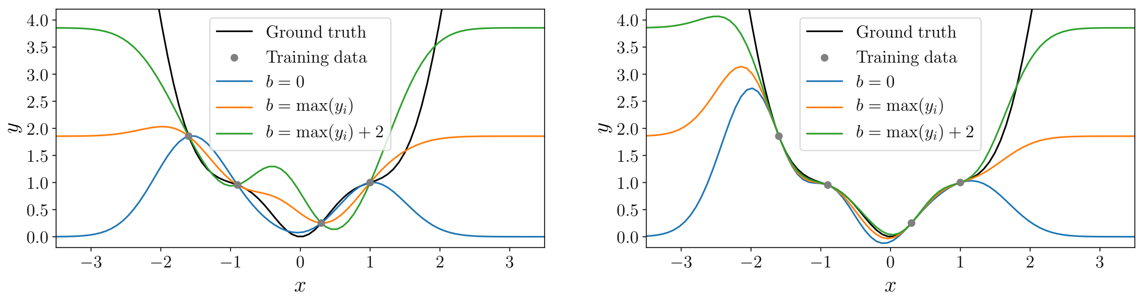


Figure E.11.: Kernel ridge regression of the function $y(x) = (x + \frac{1}{4}\sin(\pi x))^2$ using different offset values $b$. The model shown in the left panel is trained only on function values $y_i$, whereas the training data for the model in the right panel also includes derivative information $\mathrm{d}y_i/\ \mathrm{d}x$. Both models use a length scale of $\sigma = 0.5$ and parameters $\kappa = 1$ and $\lambda = 10^{-8}$.

As a first test the model errors on the $N = 1$ data set are evaluated. The results of a small grid search for the parameters $b \in \{\max(T_j), \max(T_j)+5, \max(T_j)+10\}$ and $\sigma \in \{2.5, 5.0, 10.0\}$ are summarized in Table E.5. All presented models use the extended KRR formalism and $\kappa = 1$, $\lambda = 10^{-12}$ for the remaining hyperparameters. Table E.5 shows that the largest value for the length scale parameter of $\sigma = 10.0$ yields the best accuracy for both the kinetic energy and the functional derivative. The value of the offset parameter $b$ has a smaller influence on the model performance than the length scale parameter.

The results for the iteratively found densities are summarized in Table E.6. All three models with length scale $\sigma = 10.0$ show a large number of minimization runs that leave the region spanned by the training examples, leading to extremely large mean absolute error values. While the results for $\sigma = 2.5$ suggest a better behavior during the minimization (at least

Table E.5.: Absolute error values on the test set for extended KRR models with offset term given in kcal/mol.

| | | $\|\Delta T\|$ | | | $\|\Delta \frac{\delta T}{\delta n}\|$ | | |
|---|---|---|---|---|---|---|---|
| $b$ | $\sigma$ | mean | std | max | mean | std | max |
| $\max(T_j)$ | 2.5 | 0.256 | 2.253 | 61.617 | 55.932 | 95.658 | 1746.875 |
| $\max(T_j) + 5$ | 2.5 | 1.005 | 7.484 | 158.321 | 84.234 | 225.494 | 3929.710 |
| $\max(T_j) + 10$ | 2.5 | 2.260 | 16.813 | 338.897 | 183.468 | 477.609 | 8025.145 |
| $\max(T_j)$ | 5.0 | 0.014 | 0.098 | 2.502 | 20.638 | 34.946 | 391.505 |
| $\max(T_j) + 5$ | 5.0 | 0.006 | 0.033 | 0.732 | 17.325 | 28.796 | 339.674 |
| $\max(T_j) + 10$ | 5.0 | 0.012 | 0.102 | 2.162 | 14.902 | 25.382 | 306.731 |
| $\max(T_j)$ | 10.0 | 0.002 | 0.010 | 0.165 | 6.650 | 9.340 | 105.226 |
| $\max(T_j) + 5$ | 10.0 | 0.002 | 0.008 | 0.130 | 6.520 | 9.148 | 103.361 |
| $\max(T_j) + 10$ | 10.0 | 0.001 | 0.007 | 0.111 | 6.394 | 8.972 | 101.919 |

for $b = \max(T_j) + 5$ and $b = \max(T_j) + 10$), the final errors achieved by these models are clearly limited by the poor model performance shown in Table E.5. A length scale of $\sigma = 5.0$, therefore, represents a balanced trade-off between model error and the ability to restrict the search space. In fact, the performance for $b = \max(T_j) + 10$ and $\sigma = 5.0$ is comparable to the results achieved using the principal component analysis presented in the main article. However, in general this method is not practical for large scale applications as it is difficult to determine a set of hyperparameters during training which will lead to models suitable for a usage in iterative minimizations. This problem does not arise in the original application of this approach in geometry optimization tasks as the training set is constantly extended by additional *ab-initio* calculations during minimization.

Table E.6.: Absolute kinetic energy error values for the iteratively found densities in kcal/mol as well as the integrated absolute error of the densities.

| | | $\|\Delta T\|$ | | | $\|\Delta n\| \cdot 10^4$ | | |
|---|---|---|---|---|---|---|---|
| $b$ | $\sigma$ | mean | std | max | mean | std | max |
| $\max(T_j)$ | 2.5 | 15.143 | 52.469 | 381.315 | 17861.5 | 55933.6 | 259048.3 |
| $\max(T_j) + 5$ | 2.5 | 2.884 | 12.282 | 209.400 | 13.5 | 38.8 | 534.0 |
| $\max(T_j) + 10$ | 2.5 | 4.364 | 16.793 | 265.146 | 20.2 | 49.9 | 607.7 |
| $\max(T_j)$ | 5.0 | 258.283 | 105.156 | 514.125 | 6199.3 | 7862.7 | 25692.3 |
| $\max(T_j) + 5$ | 5.0 | 0.128 | 0.838 | 17.815 | 2.4 | 5.5 | 80.5 |
| $\max(T_j) + 10$ | 5.0 | 0.075 | 0.642 | 15.196 | 1.2 | 3.5 | 71.0 |
| $\max(T_j)$ | 10.0 | 5468.034 | 257.717 | 5958.720 | 5674.8 | 392.5 | 7273.1 |
| $\max(T_j) + 5$ | 10.0 | 3787.131 | 243.600 | 4265.972 | 5226.8 | 405.1 | 6817.0 |
| $\max(T_j) + 10$ | 10.0 | 2365.444 | 215.082 | 2814.292 | 4748.9 | 417.8 | 6292.2 |

## E.7. Influence of the local PCA on the functional derivative

Figure E.12 shows the effect of the local PCA projection on the functional derivative prediction of the various models on the sample potential of Figure 6.2 of the main article. The

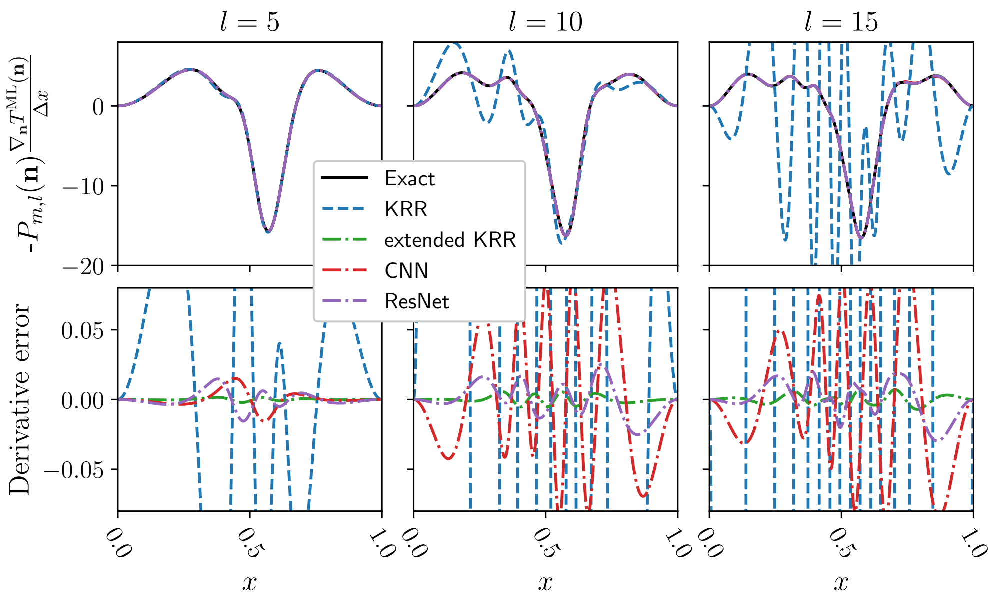


Figure E.12.: Projected functional derivative (top row) predicted by the machine learning models as well as the difference to the exact solution (bottom row) for the projection parameters $m = 30$ and various values of $l$ (columns).

corresponding integrated absolute error values for the projected functional derivatives are summarized in Table E.7. Note that both the exact and the machine learning predicted functional derivatives are projected.

Table E.7.: Absolute error values for the projected functional derivative on the $N = 1$ test set given in kcal/mol.

| | $\|\Delta P_{m=30,l=5}(n)\frac{\delta T}{\delta n}\|$ | | | $\|\Delta P_{m=30,l=10}(n)\frac{\delta T}{\delta n}\|$ | | | $\|\Delta P_{m=30,l=15}(n)\frac{\delta T}{\delta n}\|$ | | |
|---|---|---|---|---|---|---|---|---|---|
| model | mean | std | max | mean | std | max | mean | std | max |
| KRR | 147.5 | 177.8 | 1326.2 | 2008.1 | 879.6 | 4951.6 | 7874.8 | 3132.0 | 17518.6 |
| ext. KRR | 0.6 | 1.1 | 15.8 | 1.7 | 2.2 | 26.0 | 2.3 | 2.8 | 30.9 |
| CNN | 6.5 | 10.3 | 119.1 | 21.6 | 20.1 | 317.4 | 25.2 | 22.0 | 359.5 |
| ResNet | 1.9 | 2.5 | 54.1 | 4.7 | 4.4 | 81.4 | 7.6 | 5.8 | 85.0 |

## E.8. Learning curve

Figure E.13 shows the learning curve, that is the accuracy of the models as a function of the number of the training examples, for the presented machine learning models. The errors are evaluated on the $N = 1$ test set while the first $M = 40$, 60, 80, and 100 training examples are used as training examples.

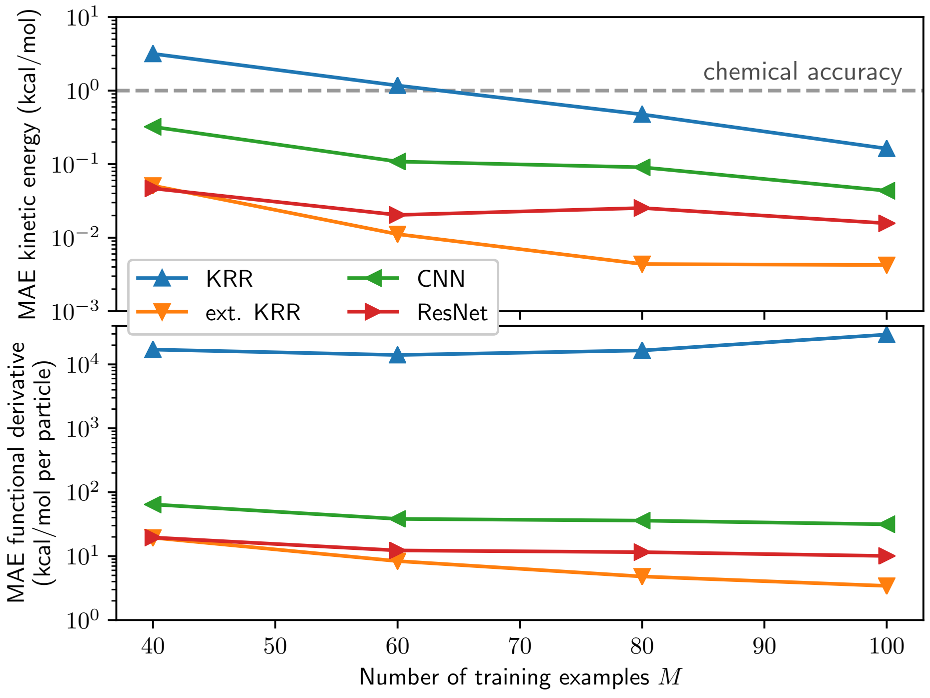


Figure E.13.: Errors on the $N = 1$ test set achieved by the various models as a function of the number of training examples $M$.

We use the hyperparameters given by Snyder et al. for the standard KRR method. A description of the method for optimizing the extended KRR hyperparameters as well as a detailed analysis of the results are given in Section E.3. The hyperparameters for both KRR models are summarized in Table E.8. For all of the neural network training runs the same hyperparameters as presented in Section E.4 is used and the batch size is reduced to the number of training examples $M$.

Table E.8.: Hyperparameters used in the training of the KRR models for different training set sizes $M$.

| | KRR | | ext. KRR | |
|---|---|---|---|---|
| $M$ | $\sigma$ | $\lambda \times 10^{14}$ | $\sigma$ | $\lambda \times 10^{12}$ |
| 40 | 238 | 57 600 | 61.49 | 10 |
| 60 | 95 | 10 000 | 35.16 | 10 |
| 80 | 48 | 4489 | 30.58 | 1 |
| 100 | 43 | 12 | 30.58 | 1 |

Figure E.13 shows a general trend of decreasing error with increasing number of training examples. Once again the advantage of including derivative information becomes apparent as all three models trained on derivatives achieve chemical accuracy for the smallest training set size $M = 40$.

Note the slight increase in the kinetic energy error for the ResNet when the number of training examples $M$ is increased from 60 to 80. This variation in the performance is to be expected due to the random initialization of the weights at the beginning of the training process. A detailed comparison of neural network learning curves could therefore only be done after averaging the results over several training runs.

## E.9. Computational timing

Figure E.14 shows the computational time required by the various models for evaluating a single density of the test set. All times are measured by averaging over 100 runs on a workstation equipped with an Intel i7-920 and without GPU acceleration for tensorflow. Note that the individual evaluation times could be improved significantly by further optimization in the computational implementation and more specialized hardware. Nevertheless, this graph clearly shows that the computational effort for the kernel-based methods increases with the number of training examples while the neural network models maintain a constant evaluation time.

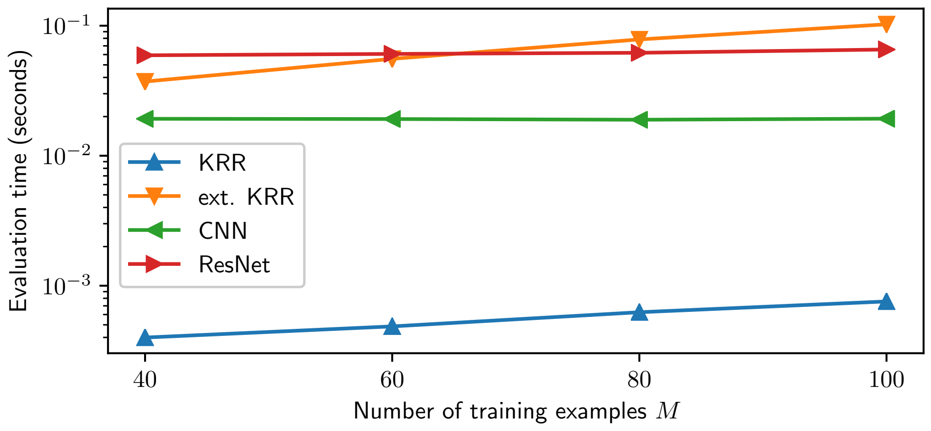


Figure E.14.: Evaluation time of the various models averaged over 100 runs as a function of the number of training examples.

# Bibliography

[1] P. A. M. Dirac and R. H. Fowler, "Quantum mechanics of many-electron systems," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 123, pp. 714–733, 1929.

[2] T. M. Mitchell, *Machine Learning.* McGraw-Hill Education, 1997.

[3] C. M. Handley and P. L. A. Popelier, "Potential Energy Surfaces Fitted by Artificial Neural Networks," *The Journal of Physical Chemistry A*, vol. 114, pp. 3371–3383, 2010.

[4] J. Behler, "Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations," *Physical Chemistry Chemical Physics*, vol. 13, pp. 17930–17955, 2011.

[5] J. Behler, "Representing potential energy surfaces by high-dimensional neural network potentials," *Journal of Physics: Condensed Matter*, vol. 26, p. 183001, 2014.

[6] J. Behler, "Perspective: Machine learning potentials for atomistic simulations," *The Journal of Chemical Physics*, vol. 145, p. 170901, 2016.

[7] J. Behler, "First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems," *Angewandte Chemie International Edition*, vol. 56, pp. 12828–12840, 2017.

[8] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, "Recent advances and applications of machine learning in solid-state materials science," *npj Computational Materials*, vol. 5, pp. 1–36, 2019.

[9] P. O. Dral, "Quantum Chemistry in the Age of Machine Learning," *The Journal of Physical Chemistry Letters*, vol. 11, pp. 2336–2347, 2020.

[10] T. Mueller, A. Hernandez, and C. Wang, "Machine learning for interatomic potential models," *The Journal of Chemical Physics*, vol. 152, p. 050902, 2020.

[11] S. Manzhos, "Machine learning for the solution of the Schrödinger equation," *Machine Learning: Science and Technology*, vol. 1, p. 013002, 2020.

[12] S. Manzhos and T. Carrington, "Neural Network Potential Energy Surfaces for Small Molecules and Reactions," *Chemical Reviews*, 2020.

[13] J. Westermayr and P. Marquetand, "Machine Learning for Electronically Excited States of Molecules," *Chemical Reviews*, 2020.

[14] M. Born and R. Oppenheimer, "Zur Quantentheorie der Molekeln," *Annalen der Physik*, vol. 389, pp. 457–484, 1927.

[15] F. Jensen, *Introduction to computational chemistry.* Chichester, England Hoboken, NJ: John Wiley & Sons, 2007.

[16] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural network methods in quantum mechanics," *Computer Physics Communications*, vol. 104, pp. 1–14, 1997.

[17] H. Nakanishi and M. Sugawara, "Numerical solution of the Schrödinger equation by a microgenetic algorithm," *Chemical Physics Letters*, vol. 327, pp. 429–438, 2000.

[18] M. Sugawara, "Numerical solution of the Schrödinger equation by neural network and genetic algorithm," *Computer Physics Communications*, vol. 140, pp. 366–380, 2001.

[19] S. M. Manzhos and T. C. Tucker Carrington, "An improved neural network method for solving the Schrödinger equation," *Canadian Journal of Chemistry*, 2009.

[20] P. Teng, "Machine-learning quantum mechanics: Solving quantum mechanics problems using radial basis function networks," *Physical Review E*, vol. 98, p. 033305, 2018.

[21] J. Han, L. Zhang, and W. E, "Solving many-electron Schrödinger equation using deep neural networks," *Journal of Computational Physics*, vol. 399, p. 108929, 2019.

[22] Y. Zeiri, E. Fattal, and R. Kosloff, "Application of genetic algorithm to the calculation of bound states and local density approximations," *The Journal of Chemical Physics*, vol. 102, pp. 1859–1862, 1995.

[23] D. E. Makarov and H. Metiu, "Using Genetic Programming To Solve the Schrödinger Equation," *The Journal of Physical Chemistry A*, vol. 104, pp. 8540–8545, 2000.

[24] R. Saha, P. Chaudhury, and S. P. Bhattacharyya, "Direct solution of Schrödinger equation by genetic algorithm: test cases," *Physics Letters A*, vol. 291, pp. 397–406, 2001.

[25] R. Hoffmann, "An Extended Hückel Theory. I. Hydrocarbons," *The Journal of Chemical Physics*, vol. 39, pp. 1397–1412, 1963.

[26] J. H. V. Lenthe, R. Zwaans, H. J. J. V. Dam, and M. F. Guest, "Starting SCF calculations by superposition of atomic densities," *Journal of Computational Chemistry*, vol. 27, pp. 926–932, 2006.

[27] J. J. Cartus, "A neural network based approach to SCF initial guesses," Master's thesis, 2019.

[28] J. P. Coe, "Machine Learning Configuration Interaction," *Journal of Chemical Theory and Computation*, vol. 14, pp. 5739–5749, 2018.

[29] J. P. Coe, "Machine Learning Configuration Interaction for ab Initio Potential Energy Curves," *Journal of Chemical Theory and Computation*, vol. 15, pp. 6179–6189, 2019.

[30] W. Jeong, S. J. Stoneburner, D. King, R. Li, A. Walker, R. Lindh, and L. Gagliardi, "Automation of Active Space Selection for Multireference Methods via Machine Learning on Chemical Bond Dissociation," *Journal of Chemical Theory and Computation*, vol. 16, pp. 2389–2399, 2020.

[31] J. Townsend and K. D. Vogiatzis, "Data-Driven Acceleration of the Coupled-Cluster Singles and Doubles Iterative Solver," *The Journal of Physical Chemistry Letters*, vol. 10, pp. 4129–4135, 2019.

[32] A. K. Wilson and J. Dunning, Thom H., "Benchmark calculations with correlated molecular wave functions. X. Comparison with "exact" MP2 calculations on Ne, HF, H2O, and N2," *The Journal of Chemical Physics*, vol. 106, pp. 8718–8726, 1997.

[33] A. Karton and J. M. L. Martin, "Comment on: "Estimating the Hartree–Fock limit from finite basis set calculations" [Jensen F (2005) Theor Chem Acc 113:267]," *Theoretical Chemistry Accounts*, vol. 115, pp. 330–333, 2006.

[34] L. A. Curtiss, K. Raghavachari, G. W. Trucks, and J. A. Pople, "Gaussian-2 theory for molecular energies of first- and second-row compounds," *The Journal of Chemical Physics*, vol. 94, pp. 7221–7230, 1991.

[35] L. A. Curtiss, K. Raghavachari, P. C. Redfern, V. Rassolov, and J. A. Pople, "Gaussian-3 (G3) theory for molecules containing first and second-row atoms," *The Journal of Chemical Physics*, vol. 109, pp. 7764–7776, 1998.

[36] T. H. Dunning, "Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen," *The Journal of Chemical Physics*, vol. 90, pp. 1007–1023, 1989.

[37] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory.* Dover Publications, Inc., first ed., 1996.

[38] T. Nudejima, Y. Ikabata, J. Seino, T. Yoshikawa, and H. Nakai, "Machine-learned electron correlation model based on correlation energy density at complete basis set limit," *The Journal of Chemical Physics*, vol. 151, p. 024104, 2019.

[39] R. M. Balabin and E. I. Lomakina, "Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies," *The Journal of Chemical Physics*, vol. 131, p. 074104, 2009.

[40] O. Schütt and J. VandeVondele, "Machine Learning Adaptive Basis Sets for Efficient Large Scale Density Functional Theory Simulation," *Journal of Chemical Theory and Computation*, vol. 14, pp. 4168–4175, 2018.

[41] M. Welborn, L. Cheng, and T. F. Miller, "Transferability in Machine Learning for Electronic Structure via the Molecular Orbital Basis," *Journal of Chemical Theory and Computation*, vol. 14, pp. 4772–4779, 2018.

[42] L. Cheng, M. Welborn, A. S. Christensen, and T. F. Miller, "A universal density matrix functional from molecular orbital-based machine learning: Transferability across organic molecules," *The Journal of Chemical Physics*, vol. 150, p. 131103, 2019.

[43] L. Cheng, N. B. Kovachki, M. Welborn, and T. F. Miller, "Regression Clustering for Improved Accuracy and Training Costs with Molecular-Orbital-Based Machine Learning," *Journal of Chemical Theory and Computation*, vol. 15, pp. 6668–6677, 2019.

[44] Y. Chen, L. Zhang, H. Wang, and W. E, "Ground State Energy Functional with Hartree–Fock Efficiency and Chemical Accuracy," *The Journal of Physical Chemistry A*, vol. 124, pp. 7155–7165, 2020.

[45] J. Townsend and K. D. Vogiatzis, "Transferable MP2-Based Machine Learning for Accurate Coupled-Cluster Energies," *Journal of Chemical Theory and Computation*, vol. 16, pp. 7453–7461, 2020.

[46] Y. Ikabata, R. Fujisawa, J. Seino, T. Yoshikawa, and H. Nakai, "Machine-learned electron correlation model based on frozen core approximation," *The Journal of Chemical Physics*, vol. 153, p. 184108, 2020.

[47] L. Hu, X. Wang, L. Wong, and G. Chen, "Combined first-principles calculation and neural-network correction approach for heat of formation," *The Journal of Chemical Physics*, vol. 119, pp. 11501–11507, 2003.

[48] X. Wang, L. Hu, L. Wong, and G. Chen, "A Combined First-principles Calculation and Neural Networks Correction Approach for Evaluating Gibbs Energy of Formation," *Molecular Simulation*, vol. 30, pp. 9–15, 2004.

[49] X. Wang, L. Wong, L. Hu, C. Chan, Z. Su, and G. Chen, "Improving the Accuracy of Density-Functional Theory Calculation: The Statistical Correction Approach," *The Journal of Physical Chemistry A*, vol. 108, pp. 8514–8525, 2004.

[50] X.-M. Duan, Z.-H. Li, G.-L. Song, W.-N. Wang, G.-H. Chen, and K.-N. Fan, "Neural network correction for heats of formation with a larger experimental training set and new descriptors," *Chemical Physics Letters*, vol. 410, pp. 125–130, 2005.

[51] J. Wu and X. Xu, "The X1 method for accurate and efficient prediction of heats of formation," *The Journal of Chemical Physics*, vol. 127, p. 214105, 2007.

[52] H. Li, L. Shi, M. Zhang, Z. Su, X. Wang, L. Hu, and G. Chen, "Improving the accuracy of density-functional theory calculation: The genetic algorithm and neural network approach," *The Journal of Chemical Physics*, vol. 126, p. 144101, 2007.

[53] J. Wu and X. Xu, "Improving the B3LYP bond energies by using the X1 method," *The Journal of Chemical Physics*, vol. 129, p. 164103, 2008.

[54] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Big Data Meets Quantum Chemistry Approximations: The Δ-Machine Learning Approach," *Journal of Chemical Theory and Computation*, vol. 11, pp. 2087–2096, 2015.

[55] P. Hohenberg and W. Kohn, "Inhomogeneous Electron Gas," *Physical Review*, vol. 136, pp. B864–B871, 1964.

[56] J. P. Perdew and K. Schmidt, "Jacob's ladder of density functional approximations for the exchange-correlation energy," *AIP Conference Proceedings*, vol. 577, pp. 1–20, 2001.

[57] P. A. M. Dirac, "Note on Exchange Phenomena in the Thomas Atom," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 26, pp. 376–385, 1930.

[58] M. Gell-Mann and K. A. Brueckner, “Correlation Energy of an Electron Gas at High Density,” *Physical Review*, vol. 106, pp. 364–368, 1957.

[59] K. Sawada, “Correlation Energy of an Electron Gas at High Density,” *Physical Review*, vol. 106, pp. 372–383, 1957.

[60] W. J. Carr and A. A. Maradudin, “Ground-State Energy of a High-Density Electron Gas,” *Physical Review*, vol. 133, pp. A371–A374, 1964.

[61] W. J. Carr, “Energy, Specific Heat, and Magnetic Properties of the Low-Density Electron Gas,” *Physical Review*, vol. 122, pp. 1437–1446, 1961.

[62] S. H. Vosko, L. Wilk, and M. Nusair, “Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis,” *Canadian Journal of Physics*, vol. 58, pp. 1200–1211, 1980.

[63] A. D. Becke, “Density-functional exchange-energy approximation with correct asymptotic behavior,” *Physical Review A*, vol. 38, pp. 3098–3100, 1988.

[64] C. Lee, W. Yang, and R. G. Parr, “Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density,” *Physical Review B*, vol. 37, pp. 785–789, 1988.

[65] J. P. Perdew, K. Burke, and M. Ernzerhof, “Generalized Gradient Approximation Made Simple,” *Physical Review Letters*, vol. 77, pp. 3865–3868, 1996.

[66] A. D. Becke, “Density-functional thermochemistry. III. The role of exact exchange,” *The Journal of Chemical Physics*, vol. 98, pp. 5648–5652, 1993.

[67] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch, “Ab Initio Calculation of Vibrational Absorption and Circular Dichroism Spectra Using Density Functional Force Fields,” *Journal of Physical Chemistry*, vol. 98, pp. 11623–11627, 1994.

[68] D. J. Tozer, V. E. Ingamells, and N. C. Handy, “Exchange-correlation potentials,” *The Journal of Chemical Physics*, vol. 105, pp. 9200–9213, 1996.

[69] X. Zheng, L. Hu, X. Wang, and G. Chen, “A generalized exchange-correlation functional: the Neural-Networks approach,” *Chemical Physics Letters*, vol. 390, pp. 186–192, 2004.

[70] J. J. Mortensen, K. Kaasbjerg, S. L. Frederiksen, J. K. Nørskov, J. P. Sethna, and K. W. Jacobsen, “Bayesian Error Estimation in Density-Functional Theory,” *Physical Review Letters*, vol. 95, p. 216401, 2005.

[71] J. Wellendorff, K. T. Lundgaard, A. Møgelhøj, V. Petzold, D. D. Landis, J. K. Nørskov, T. Bligaard, and K. W. Jacobsen, “Density functionals for surface science: Exchange-correlation model development with Bayesian error estimation,” *Physical Review B*, vol. 85, p. 235149, 2012.

[72] J. Wellendorff, K. T. Lundgaard, K. W. Jacobsen, and T. Bligaard, “mBEEF: An accurate semi-local Bayesian error estimation density functional,” *The Journal of Chemical Physics*, vol. 140, p. 144107, 2014.

[73] Q. Liu, J. Wang, P. Du, L. Hu, X. Zheng, and G. Chen, “Improving the Performance of Long-Range-Corrected Exchange-Correlation Functional with an Embedded Neural Network,” *The Journal of Physical Chemistry A*, vol. 121, pp. 7273–7281, 2017.

[74] R. Nagai, R. Akashi, S. Sasaki, and S. Tsuneyuki, “Neural-network Kohn-Sham exchange-correlation potential and its out-of-training transferability,” *The Journal of Chemical Physics*, vol. 148, p. 241737, 2018.

[75] H. Ji and Y. Jung, “A local environment descriptor for machine-learned density functional theory at the generalized gradient approximation level,” *The Journal of Chemical Physics*, vol. 148, p. 241742, 2018.

[76] X. Lei and A. J. Medford, “Design and analysis of machine learning exchange-correlation functionals via rotationally invariant convolutional descriptors,” *Physical Review Materials*, vol. 3, p. 063801, 2019.

[77] S. Dick and M. Fernandez-Serra, “Learning from the density to correct total energy and forces in first principle simulations,” *The Journal of Chemical Physics*, vol. 151, p. 144102, 2019.

[78] S. Dick and M. Fernandez-Serra, “Machine learning accurate exchange and correlation functionals of the electronic density,” *Nature Communications*, vol. 11, p. 3509, 2020.

[79] K. Ryczko, D. A. Strubbe, and I. Tamblyn, “Deep learning and density-functional theory,” *Physical Review A*, vol. 100, p. 022512, 2019.

[80] Y. Zhou, J. Wu, S. Chen, and G. Chen, “Toward the Exact Exchange–Correlation Potential: A Three-Dimensional Convolutional Neural Network Construct,” *Journal of Physical Chemistry Letters*, vol. 10, pp. 7264–7269, 2019.

[81] R. Nagai, R. Akashi, and O. Sugino, “Completing density functional theory by machine learning hidden messages from molecules,” *npj Computational Materials*, vol. 6, p. 43, 2020.

[82] L. H. Thomas, “The calculation of atomic fields,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 23, pp. 542–548, 1927.

[83] E. Fermi, “Statistical method to determine some properties of atoms,” *Rendiconti Lincei. Scienze Fisiche e Naturali*, vol. 6, p. 5, 1927.

[84] E. Fermi, “Eine statistische Methode zur Bestimmung einiger Eigenschaften des Atoms und ihre Anwendung auf die Theorie des periodischen Systems der Elemente,” *Zeitschrift für Physik*, vol. 48, pp. 73–79, 1928.

[85] C. F. v. Weizsäcker, “Zur Theorie der Kernmassen,” *Zeitschrift für Physik*, vol. 96, pp. 431–458, 1935.

[86] C. H. Hodges, “Quantum Corrections to the Thomas–Fermi Approximation—The Kirzhnits Method,” *Canadian Journal of Physics*, vol. 51, pp. 1428–1437, 1973.

[87] D. R. Murphy, “Sixth-order term of the gradient expansion of the kinetic-energy density functional,” *Physical Review A*, vol. 24, pp. 1682–1688, 1981.

[88] Y. Tal and R. F. W. Bader, “Studies of the energy density functional approach. I. Kinetic energy,” *International Journal of Quantum Chemistry*, vol. 14, pp. 153–168, 1978.

[89] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, “Finding Density Functionals with Machine Learning,” *Physical Review Letters*, vol. 108, p. 253002, 2012.

[90] J. C. Snyder, M. Rupp, K. Hansen, L. Blooston, K.-R. Müller, and K. Burke, “Orbital-free bond breaking via machine learning,” *The Journal of Chemical Physics*, vol. 139, p. 224104, 2013.

[91] L. Li, T. E. Baker, S. R. White, and K. Burke, “Pure density functional for strong correlation and the thermodynamic limit from machine learning,” *Physical Review B*, vol. 94, p. 245129, 2016.

[92] M. Levy and J. P. Perdew, “Hellmann-Feynman, virial, and scaling requisites for the exact universal density functionals. Shape of the correlation potential and diamagnetic susceptibility for atoms,” *Physical Review A*, vol. 32, pp. 2010–2021, 1985.

[93] J. Hollingsworth, L. Li, T. E. Baker, and K. Burke, “Can exact conditions improve machine-learned density functionals?,” *The Journal of Chemical Physics*, vol. 148, p. 241743, 2018.

[94] J. C. Snyder, M. Rupp, K.-R. Müller, and K. Burke, “Nonlinear gradient denoising: Finding accurate extrema from inaccurate functional derivatives,” *International Journal of Quantum Chemistry*, vol. 115, pp. 1102–1114, 2015.

[95] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, “Bypassing the Kohn-Sham equations with machine learning,” *Nature Communications*, vol. 8, p. 872, 2017.

[96] M. Bogojeski, L. Vogt-Maranto, M. E. Tuckerman, K.-R. Müller, and K. Burke, “Quantum chemical accuracy from density functional approximations via machine learning,” *Nature Communications*, vol. 11, p. 5223, 2020.

[97] R. Meyer, M. Weichselbaum, and A. W. Hauser, “Machine Learning Approaches toward Orbital-free Density Functional Theory: Simultaneous Training on the Kinetic Energy Density Functional and Its Functional Derivative,” *Journal of Chemical Theory and Computation*, vol. 16, pp. 5685–5694, 2020.

[98] K. Yao and J. Parkhill, “Kinetic Energy of Hydrocarbons as a Function of Electron Density and Convolutional Neural Networks,” *Journal of Chemical Theory and Computation*, vol. 12, pp. 1139–1147, 2016.

[99] J. Seino, R. Kageyama, M. Fujinami, Y. Ikabata, and H. Nakai, “Semi-local machine-learned kinetic energy density functional with third-order gradients of electron density,” *The Journal of Chemical Physics*, vol. 148, p. 241705, 2018.

[100] J. Seino, R. Kageyama, M. Fujinami, Y. Ikabata, and H. Nakai, “Semi-local machine-learned kinetic energy density functional demonstrating smooth potential energy curves,” *Chemical Physics Letters*, vol. 734, p. 136732, 2019.

[101] M. Fujinami, R. Kageyama, J. Seino, Y. Ikabata, and H. Nakai, "Orbital-free density functional theory calculation applying semi-local machine-learned kinetic energy density functional and kinetic potential," *Chemical Physics Letters*, vol. 748, p. 137358, 2020.

[102] P. Golub and S. Manzhos, "Kinetic energy densities based on the fourth order gradient expansion: performance in different classes of materials and improvement via machine learning," *Physical Chemistry Chemical Physics*, vol. 21, pp. 378–395, 2018.

[103] S. Manzhos and P. Golub, "Data-driven kinetic energy density fitting for orbital-free DFT: Linear vs Gaussian process regression," *The Journal of Chemical Physics*, vol. 153, p. 074104, 2020.

[104] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2005.

[105] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, pp. 1190–1208, 1995.

[106] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-scale Bound-constrained Optimization," *ACM Transactions on Mathematical Software*, vol. 23, pp. 550–560, 1997.

[107] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "TensorFlow Distributions," 2017.

[108] H. S. Baird, *Document Image Defect Models*, pp. 546–556. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992.

[109] P. Simard, B. Victorri, Y. LeCun, and J. Denker, "Tangent Prop - A formalism for specifying selected invariances in an adaptive network," in *Advances in Neural Information Processing Systems 4* (J. E. Moody, S. J. Hanson, and R. P. Lippmann, eds.), pp. 895–903, Morgan-Kaufmann, 1992.

[110] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[111] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[112] B. J. Braams and J. M. Bowman, "Permutationally invariant potential energy surfaces in high dimensionality," *International Reviews in Physical Chemistry*, vol. 28, pp. 577–606, 2009.

[113] B. Jiang and H. Guo, "Permutation invariant polynomial neural network approach to fitting potential energy surfaces," *The Journal of Chemical Physics*, vol. 139, p. 054112, 2013.

[114] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi, "Graph kernels for chemical informatics," *Neural Networks*, vol. 18, pp. 1093–1110, 2005.

[115] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, "Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning," *Physical Review Letters*, vol. 108, p. 058301, 2012.

[116] O. A. von Lilienfeld, R. Ramakrishnan, M. Rupp, and A. Knoll, "Fourier series of atomic radial distribution functions: A molecular fingerprint for machine learning models of quantum chemical properties," *International Journal of Quantum Chemistry*, vol. 115, pp. 1084–1093, 2015.

[117] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, "Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies," *Journal of Chemical Theory and Computation*, vol. 9, pp. 3404–3419, 2013.

[118] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, "Machine Learning Predictions of Molecular Properties: Accurate Many-Body Potentials and Nonlocality in Chemical Space," *The Journal of Physical Chemistry Letters*, vol. 6, pp. 2326–2331, 2015.

[119] B. Huang and O. A. von Lilienfeld, "Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity," *The Journal of Chemical Physics*, vol. 145, p. 161102, 2016.

[120] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff, "UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations," *Journal of the American Chemical Society*, vol. 114, pp. 10024–10035, 1992.

[121] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld, "Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error," *Journal of Chemical Theory and Computation*, vol. 13, pp. 5255–5264, 2017.

[122] H. Huo and M. Rupp, "Unified Representation of Molecules and Crystals for Machine Learning," *arXiv preprint arXiv:1704.06439*, 2018.

[123] W. Kohn, "Density Functional and Density Matrix Method Scaling Linearly with the Number of Atoms," *Physical Review Letters*, vol. 76, pp. 3168–3171, 1996.

[124] J. Behler and M. Parrinello, "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces," *Physical Review Letters*, vol. 98, p. 146401, 2007.

[125] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," *The Journal of Chemical Physics*, vol. 134, p. 074106, 2011.

[126] J. S. Smith, O. Isayev, and A. E. Roitberg, "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost," *Chemical Science*, vol. 8, pp. 3192–3203, 2017.

[127] V. Botu and R. Ramprasad, "Learning scheme to predict atomic forces and accelerate materials simulations," *Physical Review B*, vol. 92, p. 094306, 2015.

[128] A. P. Bartók, R. Kondor, and G. Csányi, “On representing chemical environments,” *Physical Review B*, vol. 87, p. 184115, 2013.

[129] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, and P. Marquetand, “wACSF—Weighted atom-centered symmetry functions as descriptors in machine learning potentials,” *The Journal of Chemical Physics*, vol. 148, p. 241709, 2018.

[130] S. De, A. P. Bartók, G. Csányi, and M. Ceriotti, “Comparing molecules and solids across structural and alchemical space,” *Physical Chemistry Chemical Physics*, vol. 18, pp. 13754–13769, 2016.

[131] B. G. Sumpter and D. W. Noid, “Potential energy surfaces for macromolecules. A neural network technique,” *Chemical Physics Letters*, vol. 192, pp. 455–462, 1992.

[132] S. Hobday, R. Smith, and J. Belbruno, “Applications of neural networks to fitting interatomic potential functions,” *Modelling and Simulation in Materials Science and Engineering*, vol. 7, pp. 397–412, 1999.

[133] J. Tersoff, “New empirical approach for the structure and energy of covalent systems,” *Physical Review B*, vol. 37, pp. 6991–7000, 1988.

[134] J. Tersoff, “Empirical interatomic potential for silicon with improved elastic properties,” *Physical Review B*, vol. 38, pp. 9902–9905, 1988.

[135] N. Artrith, T. Morawietz, and J. Behler, “High-dimensional neural-network potentials for multicomponent systems: Applications to zinc oxide,” *Physical Review B*, vol. 83, p. 153101, 2011.

[136] J. Behler, “Constructing high-dimensional neural network potentials: A tutorial review,” *International Journal of Quantum Chemistry*, vol. 115, pp. 1032–1050, 2015.

[137] M. Liu and J. R. Kitchin, “SingleNN: Modified Behler–Parrinello Neural Network with Shared Weights for Atomistic Simulations with Transferability,” *The Journal of Physical Chemistry C*, vol. 124, pp. 17811–17818, 2020.

[138] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, “Quantum-chemical insights from deep tensor neural networks,” *Nature Communications*, vol. 8, p. 13890, 2017.

[139] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, “SchNet – A deep learning architecture for molecules and materials,” *The Journal of Chemical Physics*, vol. 148, p. 241722, 2018.

[140] L. Zhang, J. Han, H. Wang, R. Car, and W. E, “Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics,” *Physical Review Letters*, vol. 120, p. 143001, 2018.

[141] S. Manzhos and T. Carrington, “A random-sampling high dimensional model representation neural network for building potential energy surfaces,” *The Journal of Chemical Physics*, vol. 125, p. 084109, 2006.

[142] M. Malshe, R. Narulkar, L. M. Raff, M. Hagan, S. Bukkapatnam, P. M. Agrawal, and R. Komanduri, “Development of generalized potential-energy surfaces using many-body expansions, neural networks, and moiety energy approximations,” *The Journal of Chemical Physics*, vol. 130, p. 184102, 2009.

[143] N. Lubbers, J. S. Smith, and K. Barros, “Hierarchical modeling of molecular energies using a deep neural network,” *The Journal of Chemical Physics*, vol. 148, p. 241715, 2018.

[144] T. Hollebeek, T.-S. Ho, and H. Rabitz, “Constructing multidimensional molecular potential energy surfaces from ab initio data,” *Annual Review of Physical Chemistry*, vol. 50, pp. 537–570, 1999.

[145] T.-S. Ho and H. Rabitz, “Reproducing kernel Hilbert space interpolation methods as a paradigm of high dimensional model representations: Application to multidimensional potential energy surface construction,” *The Journal of Chemical Physics*, vol. 119, pp. 6433–6442, 2003.

[146] J. Ischtwan and M. A. Collins, “Molecular potential energy surfaces by interpolation,” *The Journal of Chemical Physics*, vol. 100, pp. 8080–8088, 1994.

[147] G. G. Maisuradze, D. L. Thompson, A. F. Wagner, and M. Minkoff, “Interpolating moving least-squares methods for fitting potential energy surfaces: Detailed analysis of one-dimensional applications,” *The Journal of Chemical Physics*, vol. 119, pp. 10002–10014, 2003.

[148] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, “Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons,” *Physical Review Letters*, vol. 104, p. 136403, 2010.

[149] A. P. Bartók and G. Csányi, “Gaussian approximation potentials: A brief tutorial introduction,” *International Journal of Quantum Chemistry*, vol. 115, pp. 1051–1057, 2015.

[150] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker, “Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials,” *Journal of Computational Physics*, vol. 285, pp. 316–330, 2015.

[151] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, “Machine learning of accurate energy-conserving molecular force fields,” *Science Advances*, vol. 3, p. e1603015, 2017.

[152] S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko, “Towards exact molecular dynamics simulations with machine-learned force fields,” *Nature Communications*, vol. 9, p. 3887, 2018.

[153] Z. Li, J. R. Kermode, and A. De Vita, “Molecular Dynamics with On-the-Fly Machine Learning of Quantum-Mechanical Forces,” *Physical Review Letters*, vol. 114, p. 096405, 2015.

[154] A. Glielmo, P. Sollich, and A. De Vita, “Accurate interatomic force fields via machine learning with covariant kernels,” *Physical Review B*, vol. 95, p. 214302, 2017.

[155] A. Glielmo, C. Zeni, and A. De Vita, “Efficient nonparametric $n$-body force fields from machine learning,” *Physical Review B*, vol. 97, p. 184307, 2018.

[156] G. Ferré, T. Haut, and K. Barros, “Learning molecular energies using localized graph kernels,” *The Journal of Chemical Physics*, vol. 146, p. 114107, 2017.

[157] H. B. Schlegel, “Optimization of equilibrium geometries and transition structures,” *Journal of Computational Chemistry*, vol. 3, pp. 214–218, 1982.

[158] R. Fletcher, *Practical Methods of Optimization; (2nd Ed.)*. New York, NY, USA: Wiley-Interscience, 1987.

[159] B. H. Schlegel, “Estimating the hessian for gradient-type geometry optimizations,” *Theoretica Chimica Acta*, vol. 66, pp. 333–340, 1984.

[160] J. D. Head and M. C. Zerner, “An approximate Hessian for molecular geometry optimization,” *Chemical Physics Letters*, vol. 131, pp. 359 – 366, 1986.

[161] T. H. Fischer and J. Almlof, “General methods for geometry and wave function optimization,” *Journal of Physical Chemistry*, vol. 96, pp. 9768–9774, 1992.

[162] R. Lindh, A. Bernhardsson, G. Karlström, and P.-Å. Malmqvist, “On the use of a Hessian model function in molecular geometry optimizations,” *Chemical Physics Letters*, vol. 241, pp. 423–428, 1995.

[163] A. Denzel and J. Kästner, “Gaussian process regression for geometry optimization,” *The Journal of Chemical Physics*, vol. 148, p. 094114, 2018.

[164] E. Garijo del Río, J. J. Mortensen, and K. W. Jacobsen, “Local Bayesian optimizer for atomic structures,” *Physical Review B*, vol. 100, p. 104103, 2019.

[165] G. Schmitz and O. Christiansen, “Gaussian process regression to accelerate geometry optimizations relying on numerical differentiation,” *The Journal of Chemical Physics*, vol. 148, p. 241704, 2018.

[166] R. Meyer and A. W. Hauser, “Geometry optimization using Gaussian process regression in internal coordinate systems,” *The Journal of Chemical Physics*, vol. 152, p. 084112, 2020.

[167] A. Denzel and J. Kästner, “Hessian Matrix Update Scheme for Transition State Search Based on Gaussian Process Regression,” *Journal of Chemical Theory and Computation*, vol. 16, pp. 5083–5089, 2020.

[168] G. Raggi, I. F. Galván, C. L. Ritterhoff, M. Vacher, and R. Lindh, “Restricted-Variance Molecular Geometry Optimization Based on Gradient-Enhanced Kriging,” *Journal of Chemical Theory and Computation*, vol. 16, pp. 3989–4001, 2020.

[169] E. Garijo del Río, S. Kaappa, J. A. Garrido Torres, T. Bligaard, and K. W. Jacobsen, “Machine learning with bond information for local structure optimizations in surface science,” *The Journal of Chemical Physics*, vol. 153, p. 234116, 2020.

[170] C. J. Cerjan and W. H. Miller, “On finding transition states,” *The Journal of Chemical Physics*, vol. 75, pp. 2800–2806, 1981.

[171] J. Simons, P. Joergensen, H. Taylor, and J. Ozment, “Walking on potential energy surfaces,” *The Journal of Physical Chemistry*, vol. 87, pp. 2745–2753, 1983.

[172] A. Banerjee, N. Adams, J. Simons, and R. Shepard, “Search for stationary points on surfaces,” *The Journal of Physical Chemistry*, vol. 89, pp. 52–57, 1985.

[173] J. Baker, “An algorithm for the location of transition states,” *Journal of Computational Chemistry*, vol. 7, pp. 385–395, 1986.

[174] A. Denzel and J. Kästner, “Gaussian Process Regression for Transition State Search,” *Journal of Chemical Theory and Computation*, vol. 14, pp. 5777–5786, 2018.

[175] O.-P. Koistinen, V. Ásgeirsson, A. Vehtari, and H. Jónsson, “Minimum Mode Saddle Point Searches Using Gaussian Process Regression with Inverse-Distance Covariance Function,” *Journal of Chemical Theory and Computation*, vol. 16, pp. 499–509, 2020.

[176] G. Henkelman and H. Jónsson, “A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives,” *The Journal of Chemical Physics*, vol. 111, pp. 7010–7022, 1999.

[177] R. A. Olsen, G. J. Kroes, G. Henkelman, A. Arnaldsson, and H. Jónsson, “Comparison of methods for finding saddle points without knowledge of the final states,” *The Journal of Chemical Physics*, vol. 121, pp. 9776–9792, 2004.

[178] A. Heyden, A. T. Bell, and F. J. Keil, “Efficient methods for finding transition states in chemical reactions: Comparison of improved dimer method and partitioned rational function optimization method,” *The Journal of Chemical Physics*, vol. 123, p. 224101, 2005.

[179] J. Kästner and P. Sherwood, “Superlinearly converging dimer method for transition state search,” *The Journal of Chemical Physics*, vol. 128, p. 014106, 2008.

[180] I. Fdez. Galván, G. Raggi, and R. Lindh, “Restricted-Variance Constrained, Reaction Path, and Transition State Molecular Optimizations Using Gradient-Enhanced Kriging,” *Journal of Chemical Theory and Computation*, vol. 17, pp. 571–582, 2021.

[181] G. Mills and H. Jónsson, “Quantum and thermal effects in $H_2$ dissociative adsorption: Evaluation of free energy barriers in multidimensional quantum systems,” *Physical Review Letters*, vol. 72, pp. 1124–1127, 1994.

[182] G. Mills, H. Jónsson, and G. K. Schenter, “Reversible work transition state theory: application to dissociative adsorption of hydrogen,” *Surface Science*, vol. 324, pp. 305 – 337, 1995.

[183] H. Jónsson, G. Mills, and K. W. Jacobsen, *Nudged elastic band method for finding minimum energy paths of transitions*, pp. 385–404. World Scientific, Singapore, 1998.

[184] G. Henkelman and H. Jónsson, “Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points,” *The Journal of Chemical Physics*, vol. 113, pp. 9978–9985, 2000.

[185] G. Henkelman, B. P. Uberuaga, and H. Jónsson, "A climbing image nudged elastic band method for finding saddle points and minimum energy paths," *The Journal of Chemical Physics*, vol. 113, pp. 9901–9904, 2000.

[186] A. A. Peterson, "Acceleration of saddle-point searches with machine learning," *The Journal of Chemical Physics*, vol. 145, p. 074106, 2016.

[187] O.-P. Koistinen, F. B. Dagbjartsdóttir, V. Ásgeirsson, A. Vehtari, and H. Jónsson, "Nudged elastic band calculations accelerated with Gaussian process regression," *The Journal of Chemical Physics*, vol. 147, p. 152720, 2017.

[188] R. Meyer, K. S. Schmuck, and A. W. Hauser, "Machine Learning in Computational Chemistry: An Evaluation of Method Performance for Nudged Elastic Band Calculations," *Journal of Chemical Theory and Computation*, vol. 15, pp. 6513–6523, 2019.

[189] J. A. Garrido Torres, P. C. Jennings, M. H. Hansen, J. R. Boes, and T. Bligaard, "Low-Scaling Algorithm for Nudged Elastic Band Calculations Using a Surrogate Machine Learning Model," *Physical Review Letters*, vol. 122, p. 156001, 2019.

[190] O.-P. Koistinen, V. Ásgeirsson, A. Vehtari, and H. Jónsson, "Nudged Elastic Band Calculations Accelerated with Gaussian Process Regression Based on Inverse Interatomic Distances," *Journal of Chemical Theory and Computation*, vol. 15, pp. 6738–6751, 2019.

[191] A. Denzel, B. Haasdonk, and J. Kästner, "Gaussian Process Regression for Minimum Energy Path Optimization and Transition State Search," *The Journal of Physical Chemistry A*, vol. 123, pp. 9600–9611, 2019.

[192] A. C. Vaucher and M. Reiher, "Minimum Energy Paths and Transition States by Curve Optimization," *Journal of Chemical Theory and Computation*, vol. 14, pp. 3091–3099, 2018.

[193] F. H. Stillinger, "Exponential multiplicity of inherent structures," *Physical Review E*, vol. 59, pp. 48–51, 1999.

[194] C. J. Pickard and R. J. Needs, "Ab initio random structure searching," *Journal of Physics: Condensed Matter*, vol. 23, p. 053201, 2011.

[195] R. P. Bell and C. N. Hinshelwood, "The theory of reactions involving proton transfers," *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences*, vol. 154, pp. 414–429, 1936.

[196] M. Evans and M. Polanyi, "Further considerations on the thermodynamics of chemical equilibria and reaction rates," *Transactions of the Faraday Society*, vol. 32, pp. 1333–1360, 1936.

[197] S. Roy, S. Goedecker, and V. Hellmann, "Bell-Evans-Polanyi principle for molecular dynamics trajectories and its implications for global optimization," *Physical Review E*, vol. 77, 2008.

[198] S. Goedecker, "Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems," *The Journal of Chemical Physics*, vol. 120, pp. 9911–9917, 2004.

[199] Q. Tong, P. Gao, H. Liu, Y. Xie, J. Lv, Y. Wang, and J. Zhao, “Combining Machine Learning Potential and Structure Prediction for Accelerated Materials Design and Discovery,” *The Journal of Physical Chemistry Letters*, vol. 11, pp. 8710–8720, 2020.

[200] F. H. Stillinger and D. K. Stillinger, “Cluster optimization simplified by interaction modification,” *The Journal of Chemical Physics*, vol. 93, pp. 6106–6107, 1990.

[201] A. Laio and M. Parrinello, “Escaping free-energy minima,” *Proceedings of the National Academy of Sciences*, vol. 99, pp. 12562–12566, 2002.

[202] H. Grubmüller, “Predicting slow structural transitions in macromolecular systems: Conformational flooding,” *Physical Review E*, vol. 52, pp. 2893–2906, 1995.

[203] D. J. Wales and J. P. K. Doye, “Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms,” *The Journal of Physical Chemistry A*, vol. 101, pp. 5111–5116, 1997.

[204] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of State Calculations by Fast Computing Machines,” *The Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[205] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, pp. 671–680, 1983.

[206] R. H. Swendsen and J.-S. Wang, “Replica Monte Carlo Simulation of Spin-Glasses,” *Physical Review Letters*, vol. 57, pp. 2607–2609, 1986.

[207] D. J. Earl and M. W. Deem, “Parallel tempering: Theory, applications, and new perspectives,” *Physical Chemistry Chemical Physics*, vol. 7, pp. 3910–3916, 2005.

[208] J. Skilling, “Nested Sampling,” in *AIP Conference Proceedings*, AIP, 2004.

[209] L. B. Pártay, A. P. Bartók, and G. Csányi, “Efficient Sampling of Atomic Configurational Spaces,” *The Journal of Physical Chemistry B*, vol. 114, pp. 10502–10512, 2010.

[210] D. M. Deaven and K. M. Ho, “Molecular Geometry Optimization with a Genetic Algorithm,” *Physical Review Letters*, vol. 75, pp. 288–291, 1995.

[211] D. M. Daven, N. Tit, J. R. Morris, and K. M. Ho, “Structural optimization of Lennard-Jones clusters by a genetic algorithm,” *Chemical Physics Letters*, vol. 256, pp. 195–200, 1996.

[212] K. M. Ho, A. A. Shvartsburg, B. Pan, Z.-Y. Lu, C.-Z. Wang, J. G. Wacker, J. L. Fye, and M. F. Jarrold, “Structures of medium-sized silicon clusters,” *Nature*, vol. 392, pp. 582–585, 1998.

[213] I. Rata, A. A. Shvartsburg, M. Horoi, T. Frauenheim, K. W. M. Siu, and K. A. Jackson, “Single-Parent Evolution Algorithm and the Optimization of Si Clusters,” *Physical Review Letters*, vol. 85, pp. 546–549, 2000.

[214] S. Darby, T. V. Mortimer-Jones, R. L. Johnston, and C. Roberts, “Theoretical study of Cu–Au nanoalloy clusters using a genetic algorithm,” *The Journal of Chemical Physics*, vol. 116, pp. 1536–1550, 2002.

[215] N. L. Abraham and M. I. J. Probert, "A periodic genetic algorithm with real-space representation for crystal structure and polymorph prediction," *Physical Review B*, vol. 73, p. 224104, 2006.

[216] S. T. Call, D. Y. Zubarev, and A. I. Boldyrev, "Global minimum structure searches via particle swarm optimization," *Journal of Computational Chemistry*, vol. 28, pp. 1177–1186, 2007.

[217] Y. Wang, J. Lv, L. Zhu, and Y. Ma, "Crystal structure prediction via particle-swarm optimization," *Physical Review B*, vol. 82, 2010.

[218] V. L. Deringer, C. J. Pickard, and G. Csányi, "Data-Driven Learning of Total and Local Energies in Elemental Boron," *Physical Review Letters*, vol. 120, p. 156001, 2018.

[219] V. L. Deringer, D. M. Proserpio, G. Csányi, and C. J. Pickard, "Data-driven learning and prediction of inorganic crystal structures," *Faraday Discussions*, vol. 211, pp. 45–59, 2018.

[220] R. Ouyang, Y. Xie, and D.-e. Jiang, "Global minimization of gold clusters by combining neural network potentials and the basin-hopping method," *Nanoscale*, vol. 7, pp. 14817–14821, 2015.

[221] S. Chiriki and S. S. Bulusu, "Modeling of DFT quality neural network potential for sodium clusters: Application to melting of sodium clusters (Na20 to Na40)," *Chemical Physics Letters*, vol. 652, pp. 130–135, 2016.

[222] S. Jindal, S. Chiriki, and S. S. Bulusu, "Spherical harmonics based descriptor for neural network potentials: Structure and dynamics of Au147 nanocluster," *The Journal of Chemical Physics*, vol. 146, p. 204301, 2017.

[223] S. Chiriki, S. Jindal, and S. S. Bulusu, "c-T phase diagram and Landau free energies of (AgAu)55 nanoalloy via neural-network molecular dynamic simulations," *The Journal of Chemical Physics*, vol. 147, p. 154303, 2017.

[224] S. Chiriki, S. Jindal, and S. S. Bulusu, "Neural network potentials for dynamics and thermodynamics of gold nanoparticles," *The Journal of Chemical Physics*, vol. 146, p. 084314, 2017.

[225] S. Jindal and S. S. Bulusu, "Structural evolution in gold nanoparticles using artificial neural network based interatomic potentials," *The Journal of Chemical Physics*, vol. 152, p. 154302, 2020.

[226] T. Jacobsen, M. Jørgensen, and B. Hammer, "On-the-Fly Machine Learning of Atomic Potential in Density Functional Theory Structure Optimization," *Physical Review Letters*, vol. 120, p. 026102, 2018.

[227] E. L. Kolsbjerg, A. A. Peterson, and B. Hammer, "Neural-network-enhanced evolutionary algorithm applied to supported metal nanoparticles," *Physical Review B*, vol. 97, p. 195424, 2018.

[228] M. Van den Bossche, H. Grönbeck, and B. Hammer, "Tight-Binding Approximation-Enhanced Global Optimization," *Journal of Chemical Theory and Computation*, vol. 14, pp. 2797–2807, 2018.

[229] S. Hajinazar, E. D. Sandoval, A. J. Cullo, and A. N. Kolmogorov, "Multitribe evolutionary search for stable Cu–Pd–Ag nanoparticles using neural network models," *Physical Chemistry Chemical Physics*, vol. 21, pp. 8729–8742, 2019.

[230] A. Thorn, J. Rojas-Nunez, S. Hajinazar, S. E. Baltazar, and A. N. Kolmogorov, "Toward ab Initio Ground States of Gold Clusters via Neural Network Modeling," *The Journal of Physical Chemistry C*, vol. 123, pp. 30088–30098, 2019.

[231] G.-F. Shao, J.-B. Yang, J.-F. Zhang, T.-D. Liu, and Y.-H. Wen, "GPU-based DPSO algorithm for structural optimization of Pt-Co bimetallic nanoparticles," *Physics Letters A*, vol. 383, pp. 3123–3133, 2019.

[232] E. V. Podryabinkin, E. V. Tikhonov, A. V. Shapeev, and A. R. Oganov, "Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning," *Physical Review B*, vol. 99, p. 064114, 2019.

[233] S. Chiriki, M.-P. V. Christiansen, and B. Hammer, "Constructing convex energy landscapes for atomistic structure optimization," *Physical Review B*, vol. 100, p. 235436, 2019.

[234] S. Hajinazar, A. Thorn, E. D. Sandoval, S. Kharabadze, and A. N. Kolmogorov, "MAISE: Construction of neural network interatomic models and evolutionary structure optimization," *Computer Physics Communications*, vol. 259, p. 107679, 2021.

[235] T. Yamashita, N. Sato, H. Kino, T. Miyake, K. Tsuda, and T. Oguchi, "Crystal structure prediction accelerated by Bayesian optimization," *Physical Review Materials*, vol. 2, p. 013803, 2018.

[236] M. S. Jørgensen, U. F. Larsen, K. W. Jacobsen, and B. Hammer, "Exploration versus Exploitation in Global Atomistic Structure Optimization," *The Journal of Physical Chemistry A*, vol. 122, pp. 1504–1509, 2018.

[237] N. Bernstein, G. Csányi, and V. L. Deringer, "De novo exploration and self-guided learning of potential-energy surfaces," *npj Computational Materials*, vol. 5, pp. 1–9, 2019.

[238] K. Gubaev, E. V. Podryabinkin, G. L. W. Hart, and A. V. Shapeev, "Accelerating high-throughput searches for new alloys with active learning of interatomic potentials," *Computational Materials Science*, vol. 156, pp. 148–156, 2019.

[239] M. Todorović, M. U. Gutmann, J. Corander, and P. Rinke, "Bayesian inference of atomistic structure in functional materials," *npj Computational Materials*, vol. 5, 2019.

[240] M. K. Bisbo and B. Hammer, "Efficient Global Structure Optimization with a Machine-Learned Surrogate Model," *Physical Review Letters*, vol. 124, p. 086102, 2020.

[241] M. S. Jørgensen, H. L. Mortensen, S. A. Meldgaard, E. L. Kolsbjerg, T. L. Jacobsen, K. H. Sørensen, and B. Hammer, "Atomistic structure learning," *The Journal of Chemical Physics*, vol. 151, p. 054111, 2019.

[242] M.-P. V. Christiansen, H. L. Mortensen, S. A. Meldgaard, and B. Hammer, “Gaussian representation for image recognition and reinforcement learning of atomistic structure,” *The Journal of Chemical Physics*, vol. 153, p. 044107, 2020.

[243] S. A. Meldgaard, H. L. Mortensen, M. S. Jørgensen, and B. Hammer, “Structure prediction of surface reconstructions by deep reinforcement learning,” *Journal of Physics: Condensed Matter*, vol. 32, p. 404005, 2020.

[244] H. L. Mortensen, S. A. Meldgaard, M. K. Bisbo, M.-P. V. Christiansen, and B. Hammer, “Atomistic structure learning algorithm with surrogate energy model relaxation,” *Physical Review B*, vol. 102, p. 075427, 2020.

[245] J. M. L. Ribeiro, P. Bravo, Y. Wang, and P. Tiwary, “Reweighted autoencoded variational Bayes for enhanced sampling (RAVE),” *The Journal of Chemical Physics*, vol. 149, p. 072301, 2018.

[246] W. Chen and A. L. Ferguson, “Molecular enhanced sampling with autoencoders: On-the-fly collective variable discovery and accelerated free energy landscape exploration,” *Journal of Computational Chemistry*, vol. 39, pp. 2079–2102, 2018.

[247] C. X. Hernández, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, and V. S. Pande, “Variational encoding of complex dynamics,” *Physical Review E*, vol. 97, 2018.

[248] C. Wehmeyer and F. Noé, “Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics,” *The Journal of Chemical Physics*, vol. 148, p. 241703, 2018.

[249] M. Schöberl, N. Zabaras, and P.-S. Koutsourelakis, “Predictive collective variable discovery with deep Bayesian models,” *The Journal of Chemical Physics*, vol. 150, p. 024109, 2019.

[250] J. Rogal, E. Schneider, and M. E. Tuckerman, “Neural-Network-Based Path Collective Variables for Enhanced Sampling of Phase Transformations,” *Physical Review Letters*, vol. 123, 2019.

[251] W. Chen, H. Sidky, and A. L. Ferguson, “Nonlinear discovery of slow molecular modes using state-free reversible VAMPnets,” *The Journal of Chemical Physics*, vol. 150, p. 214114, 2019.

[252] P. Ravindra, Z. Smith, and P. Tiwary, “Automatic mutual information noise omission (AMINO): generating order parameters for molecular systems,” *Molecular Systems Design & Engineering*, vol. 5, pp. 339–348, 2020.

[253] O. Valsson and M. Parrinello, “Variational Approach to Enhanced Sampling and Free Energy Calculations,” *Physical Review Letters*, vol. 113, 2014.

[254] L. Bonati, Y.-Y. Zhang, and M. Parrinello, “Neural networks-based variationally enhanced sampling,” *Proceedings of the National Academy of Sciences*, vol. 116, pp. 17641–17647, 2019.

[255] L. Bonati, V. Rizzi, and M. Parrinello, “Data-Driven Collective Variables for Enhanced Sampling,” *The Journal of Physical Chemistry Letters*, vol. 11, pp. 2998–3004, 2020.

[256] K. H. Sørensen, M. S. Jørgensen, A. Bruix, and B. Hammer, “Accelerating atomic structure search with cluster regularization,” *The Journal of Chemical Physics*, vol. 148, p. 241734, 2018.

[257] S. A. Meldgaard, E. L. Kolsbjerg, and B. Hammer, “Machine learning enhanced global optimization by clustering local environments to enable bundled atomic energies,” *The Journal of Chemical Physics*, vol. 149, p. 134104, 2018.

[258] M. S. Jørgensen, M. N. Groves, and B. Hammer, “Combining Evolutionary Algorithms with Clustering toward Rational Global Structure Optimization at the Atomic Scale,” *Journal of Chemical Theory and Computation*, vol. 13, pp. 1486–1493, 2017.

[259] J. Moult, J. T. Pedersen, R. Judson, and K. Fidelis, “A large-scale experiment to assess protein structure prediction methods,” *Proteins: Structure, Function, and Genetics*, vol. 23, pp. ii–iv, 1995.

[260] J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano, “Critical assessment of methods of protein structure prediction (CASP)-round XII,” *Proteins: Structure, Function, and Bioinformatics*, vol. 86, pp. 7–15, 2017.

[261] A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, and J. Moult, “Critical assessment of methods of protein structure prediction (CASP)—round XIII,” *Proteins: Structure, Function, and Bioinformatics*, vol. 87, pp. 1011–1020, 2019.

[262] S. Wang, S. Sun, and J. Xu, “Analysis of deep learning methods for blind protein contact prediction in CASP12,” *Proteins: Structure, Function, and Bioinformatics*, vol. 86, pp. 67–77, 2017.

[263] J. Schaarschmidt, B. Monastyrskyy, A. Kryshtafovych, and A. M. Bonvin, “Assessment of contact predictions in CASP12: Co-evolution and deep learning coming of age,” *Proteins: Structure, Function, and Bioinformatics*, vol. 86, pp. 51–66, 2017.

[264] W. Zheng, Y. Li, C. Zhang, R. Pearce, S. M. Mortuza, and Y. Zhang, “Deep-learning contact-map guided protein structure prediction in CASP13,” *Proteins: Structure, Function, and Bioinformatics*, vol. 87, pp. 1149–1164, 2019.

[265] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, “Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction (CASP13),” *Proteins: Structure, Function, and Bioinformatics*, vol. 87, pp. 1141–1148, 2019.

[266] J. Hou, T. Wu, R. Cao, and J. Cheng, “Protein tertiary structure modeling driven by deep learning and contact distance prediction in CASP13,” *Proteins: Structure, Function, and Bioinformatics*, vol. 87, pp. 1165–1178, 2019.

[267] S. M. Kandathil, J. G. Greener, and D. T. Jones, “Prediction of interresidue contacts with DeepMetaPSICOV in CASP13,” *Proteins: Structure, Function, and Bioinformatics*, vol. 87, pp. 1092–1099, 2019.

[268] Y. Li, C. Zhang, E. W. Bell, D.-J. Yu, and Y. Zhang, "Ensembling multiple raw coevolutionary features with deep residual neural networks for contact-map prediction in CASP13," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, pp. 1082–1091, 2019.

[269] J. Xu and S. Wang, "Analysis of distance-based protein structure prediction by deep learning in CASP13," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, pp. 1069–1081, 2019.

[270] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, pp. 706–710, 2020.

[271] "AlphaFold2." `https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology`. Accessed: 2021-01-18.

[272] M. S. Daw and M. I. Baskes, "Semiempirical, Quantum Mechanical Calculation of Hydrogen Embrittlement in Metals," *Physical Review Letters*, vol. 50, pp. 1285–1288, 1983.

[273] M. S. Daw and M. I. Baskes, "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals," *Physical Review B*, vol. 29, pp. 6443–6453, 1984.

[274] S. M. Foiles, M. I. Baskes, and M. S. Daw, "Embedded-atom-method functions for the fcc metals Cu, Ag, Au, Ni, Pd, Pt, and their alloys," *Physical Review B*, vol. 33, pp. 7983–7991, 1986.

[275] R. P. Gupta, "Lattice relaxation at a metal surface," *Physical Review B*, vol. 23, pp. 6265–6270, 1981.

[276] J. R. Smith and A. Banerjea, "New Approach to Calculation of Total Energies of Solids with Defects: Surface-Energy Anisotropies," *Physical Review Letters*, vol. 59, pp. 2451–2454, 1987.

[277] F. Ercolessi, M. Parrinello, and E. Tosatti, "Au(100) reconstruction in the glue model," *Surface Science*, vol. 177, pp. 314 – 328, 1986.

[278] J. Friedel, *Transition metals. Electronic structure of the d-band. Its role in the crystalline and magnetic structures.*, ch. 8. London: Cambridge U.P, 1969.

[279] D. Tománek, A. A. Aligia, and C. A. Balseiro, "Calculation of elastic strain and electronic effects on surface segregation," *Physical Review B*, vol. 32, pp. 5051–5056, 1985.

[280] F. Cleri and V. Rosato, "Tight-binding potentials for transition metals and alloys," *Physical Review B*, vol. 48, pp. 22–33, 1993.

[281] M. W. Finnis and J. E. Sinclair, "A simple empirical N-body potential for transition metals," *Philosophical Magazine A*, vol. 50, pp. 45–55, 1984.

[282] A. P. Sutton and J. Chen, “Long-range Finnis–Sinclair potentials,” *Philosophical Magazine Letters*, vol. 61, pp. 139–146, 1990.

[283] M. I. Baskes, “Application of the Embedded-Atom Method to Covalent Materials: A Semiempirical Potential for Silicon,” *Physical Review Letters*, vol. 59, pp. 2666–2669, 1987.

[284] M. I. Baskes, J. S. Nelson, and A. F. Wright, “Semiempirical modified embedded-atom potentials for silicon and germanium,” *Physical Review B*, vol. 40, pp. 6085–6100, 1989.

[285] M. I. Baskes, “Modified embedded-atom potentials for cubic materials and impurities,” *Physical Review B*, vol. 46, pp. 2727–2742, 1992.

[286] H. Bhattarai, K. E. Newman, and J. D. Gezelter, “Polarizable potentials for metals: The density readjusting embedded atom method (DR-EAM),” *Physical Review B*, vol. 99, p. 094106, 2019.

[287] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, “Neural network models of potential energy surfaces,” *The Journal of Chemical Physics*, vol. 103, pp. 4129–4137, 1995.

[288] D. F. R. Brown, M. N. Gibbs, and D. C. Clary, “Combining ab initio computations, neural networks, and diffusion Monte Carlo: An efficient method to treat weakly bound molecules,” *The Journal of Chemical Physics*, vol. 105, pp. 7597–7604, 1996.

[289] F. V. Prudente, P. H. Acioli, and J. J. S. Neto, “The fitting of potential energy surfaces using neural networks: Application to the study of vibrational levels of $H_3^+$,” *The Journal of Chemical Physics*, vol. 109, pp. 8801–8808, 1998.

[290] S. Lorenz, A. Groß, and M. Scheffler, “Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks,” *Chemical Physics Letters*, vol. 395, pp. 210 – 215, 2004.

[291] N. Artrith and J. Behler, “High-dimensional neural network potentials for metal surfaces: A prototype study for copper,” *Physical Review B*, vol. 85, 2012.

[292] N. Artrith, B. Hiller, and J. Behler, “Neural network potentials for metals and oxides – First applications to copper clusters at zinc oxide,” *physica status solidi (b)*, vol. 250, pp. 1191–1203, 2012.

[293] N. Artrith and A. M. Kolpak, “Grand canonical molecular dynamics simulations of Cu–Au nanoalloys in thermal equilibrium using reactive ANN potentials,” *Computational Materials Science*, vol. 110, pp. 20 – 28, 2015.

[294] L. Zhang, J. Han, H. Wang, W. Saidi, R. Car, and W. E, “End-to-end Symmetry Preserving Inter-atomic Potential Energy Model for Finite and Extended Systems,” in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 4436–4446, Curran Associates, Inc., 2018.

[295] C. Zeni, K. Rossi, A. Glielmo, Á. Fekete, N. Gaston, F. Baletto, and A. De Vita, “Building machine learning force fields for nanoclusters,” *The Journal of Chemical Physics*, vol. 148, p. 241739, 2018.

[296] C. Zeni, K. Rossi, A. Glielmo, and F. Baletto, “On machine learning force fields for metallic nanoparticles,” *Advances in Physics: X*, vol. 4, p. 1654919, 2019.

[297] R. Ferrando, J. Jellinek, and R. L. Johnston, “Nanoalloys: From Theory to Applications of Alloy Clusters and Nanoparticles,” *Chemical Reviews*, vol. 108, pp. 845–910, 2008.

[298] G. Wang, M. A. Van Hove, P. N. Ross, and M. I. Baskes, “Monte carlo simulations of segregation in Pt-Ni catalyst nanoparticles,” *The Journal of Chemical Physics*, vol. 122, p. 024706, 2005.

[299] G. Treglia and F. Ducastelle, “Is ordering in PtNi alloys induced by spin-orbit interactions?,” *Journal of Physics F: Metal Physics*, vol. 17, pp. 1935–1944, 1987.

[300] C. di Paola and F. Baletto, “Oxygen adsorption on small PtNi nanoalloys,” *Physical Chemistry Chemical Physics*, vol. 13, pp. 7701–7707, 2011.

[301] A. Radillo-Díaz, Y. Coronado, L. A. Pérez, and I. L. Garzón, “Structural and electronic properties of PtPd and PtNi nanoalloys,” *European Physical Journal D: Atomic, Molecular and Optical Physics*, vol. 52, pp. 127–130, 2009.

[302] D. Cheng, S. Yuan, and R. Ferrando, “Structure, chemical ordering and thermal stability of Pt–Ni alloy nanoclusters,” *Journal of Physics: Condensed Matter*, vol. 25, p. 355008, 2013.

[303] L. Cao, Z. Zhao, Z. Liu, W. Gao, S. Dai, J. Gha, W. Xue, H. Sun, X. Duan, X. Pan, T. Mueller, and Y. Huang, “Differential Surface Elemental Distribution Leads to Significantly Enhanced Stability of PtNi-Based ORR Catalysts,” *Matter*, vol. 1, pp. 1567 – 1580, 2019.

[304] G. Wang, H. Wu, D. Wexler, H. Liu, and O. Savadogo, “Ni@Pt core–shell nanoparticles with enhanced catalytic activity for oxygen reduction reaction,” *Journal of Alloys and Compounds*, vol. 503, pp. L1–L4, 2010.

[305] J. Wu, L. Qi, H. You, A. Gross, J. Li, and H. Yang, “Icosahedral Platinum Alloy Nanocrystals with Enhanced Electrocatalytic Activities,” *Journal of the American Chemical Society*, vol. 134, pp. 11880–11883, 2012.

[306] V. R. Stamenkovic, B. Fowler, B. S. Mun, G. Wang, P. N. Ross, C. A. Lucas, and N. M. Marković, “Improved Oxygen Reduction Activity on Pt3Ni(111) via Increased Surface Site Availability,” *Science*, vol. 315, pp. 493–497, 2007.

[307] J. Wu, J. Zhang, Z. Peng, S. Yang, F. T. Wagner, and H. Yang, “Truncated Octahedral Pt3Ni Oxygen Reduction Reaction Electrocatalysts,” *Journal of the American Chemical Society*, vol. 132, pp. 4984–4985, 2010.

[308] S.-I. Choi, S. Xie, M. Shao, J. H. Odell, N. Lu, H.-C. Peng, L. Protsailo, S. Guerrero, J. Park, X. Xia, J. Wang, M. J. Kim, and Y. Xia, “Synthesis and Characterization of 9 nm

Pt–Ni Octahedra with a Record High Activity of 3.3 A/mgPt for the Oxygen Reduction Reaction," *Nano Letters*, vol. 13, pp. 3420–3425, 2013.

[309] C. Cui, L. Gan, M. Heggen, S. Rudi, and P. Strasser, "Compositional segregation in shaped Pt alloy nanoparticles and their structural behaviour during electrocatalysis," *Nature Materials*, vol. 12, pp. 765–771, 2013.

[310] G. Rossi and R. Ferrando, "Searching for low-energy structures of nanoparticles: a comparison of different methods and algorithms," *Journal of Physics: Condensed Matter*, vol. 21, p. 084208, 2009.

[311] M. Born and J. E. Mayer, "Zur Gittertheorie der Ionenkristalle," *Zeitschrift für Physik*, vol. 75, pp. 1–18, 1932.

[312] C. Kittel, *Introduction to Solid State Physics.* John Wiley and Sons Inc, 2004.

[313] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, p. 395502 (19pp), 2009.

[314] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. B. Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. D. Corso, S. de Gironcoli, P. Delugas, R. A. DiStasio, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. O. de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni, "Advanced capabilities for materials modelling with QUANTUM ESPRESSO," *Journal of Physics: Condensed Matter*, vol. 29, p. 465901, 2017.

[315] D. Vanderbilt, "Soft self-consistent pseudopotentials in a generalized eigenvalue formalism," *Physical Review B*, vol. 41, pp. 7892–7895, 1990.

[316] We used the pseudopotentials Pt.pbe-n-rrkjus_psl.0.1.UPF and Ni.pbe-n-rrkjus_psl.0.1.UPF from `http://www.quantum-espresso.org`.

[317] M. Methfessel and A. T. Paxton, "High-precision sampling for Brillouin-zone integration in metals," *Physical Review B*, vol. 40, pp. 3616–3621, 1989.

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

[319] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg,

M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” 2015. Software available from tensorflow.org.

[320] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” 2014. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[321] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 249–256, PMLR, 2010.

[322] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Efficient BackProp*, pp. 9–48. Springer Berlin Heidelberg, 1998.

[323] H. J. Monkhorst and J. D. Pack, “Special points for Brillouin-zone integrations,” *Physical Review B*, vol. 13, pp. 5188–5192, 1976.

[324] P. Pulay, G. Fogarasi, F. Pang, and J. E. Boggs, “Systematic ab initio gradient calculation of molecular geometries, force constants, and dipole moment derivatives,” *Journal of the American Chemical Society*, vol. 101, pp. 2550–2560, 1979.

[325] J. D. Head, “Partial optimization of large molecules and clusters,” *Journal of Computational Chemistry*, vol. 11, pp. 67–75, 1990.

[326] P. Pulay and G. Fogarasi, “Geometry optimization in redundant internal coordinates,” *The Journal of Chemical Physics*, vol. 96, pp. 2856–2860, 1992.

[327] G. Fogarasi, X. Zhou, P. W. Taylor, and P. Pulay, “The calculation of ab initio molecular geometries: efficient optimization by natural internal coordinates and empirical correction by offset forces,” *Journal of the American Chemical Society*, vol. 114, pp. 8191–8201, 1992.

[328] J. Baker, “Techniques for geometry optimization: A comparison of cartesian and natural internal coordinates,” *Journal of Computational Chemistry*, vol. 14, pp. 1085–1100, 1993.

[329] J. Baker, A. Kessi, and B. Delley, “The generation and use of delocalized internal coordinates in geometry optimization,” *The Journal of Chemical Physics*, vol. 105, pp. 192–212, 1996.

[330] J. Baker and F. Chan, “The location of transition states: A comparison of Cartesian, Z-matrix, and natural internal coordinates,” *Journal of Computational Chemistry*, vol. 17, pp. 888–904, 1996.

[331] S. R. Billeter, A. J. Turner, and W. Thiel, “Linear scaling geometry optimisation and transition state search in hybrid delocalised internal coordinates,” *Physical Chemistry Chemical Physics*, vol. 2, pp. 2177–2186, 2000.

[332] B. Paizs, J. Baker, S. Suhai, and P. Pulay, “Geometry optimization of large biomolecules in redundant internal coordinates,” *The Journal of Chemical Physics*, vol. 113, pp. 6566–6572, 2000.

[333] J. Andzelm, R. King-Smith, and G. Fitzgerald, “Geometry optimization of solids using delocalized internal coordinates,” *Chemical Physics Letters*, vol. 335, pp. 321 – 326, 2001.

[334] J. U. Reveles and A. M. Köster, "Geometry optimization in density functional methods," *Journal of Computational Chemistry*, vol. 25, pp. 1109–1116, 2004.

[335] K. Németh, M. Challacombe, and M. Van Veenendaal, "The choice of internal coordinates in complex chemical systems," *Journal of Computational Chemistry*, vol. 31, pp. 2078–2086, 2010.

[336] L.-P. Wang and C. Song, "Geometry optimization made simple with translation and rotation coordinates," *The Journal of Chemical Physics*, vol. 144, p. 214108, 2016.

[337] J. Baker and P. Pulay, "Geometry optimization of atomic microclusters using inverse-power distance coordinates," *The Journal of Chemical Physics*, vol. 105, pp. 11100–11107, 1996.

[338] C. C. Pye and R. A. Poirier, "Graphical approach for defining natural internal coordinates," *Journal of Computational Chemistry*, vol. 19, pp. 504–511, 1998.

[339] M. von Arnim and R. Ahlrichs, "Geometry optimization in generalized natural internal coordinates," *The Journal of Chemical Physics*, vol. 111, pp. 9183–9190, 1999.

[340] M. Swart and F. Matthias Bickelhaupt, "Optimization of strong and weak coordinates," *International Journal of Quantum Chemistry*, vol. 106, pp. 2536–2544, 2006.

[341] P. Császár and P. Pulay, "Geometry optimization by direct inversion in the iterative subspace," *Journal of Molecular Structure*, vol. 114, pp. 31 – 34, 1984.

[342] K. Németh and M. Challacombe, "The quasi-independent curvilinear coordinate approximation for geometry optimization," *The Journal of Chemical Physics*, vol. 121, pp. 2877–2885, 2004.

[343] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, and P. Gumbsch, "Structural Relaxation Made Simple," *Physical Review Letters*, vol. 97, p. 170201, 2006.

[344] D. Sheppard, R. Terrell, and G. Henkelman, "Optimization methods for finding minimum energy paths," *The Journal of Chemical Physics*, vol. 128, p. 134106, 2008.

[345] Z. D. Pozun, K. Hansen, D. Sheppard, M. Rupp, K.-R. Müller, and G. Henkelman, "Optimizing transition states via kernel-based machine learning," *The Journal of Chemical Physics*, vol. 136, p. 174101, 2012.

[346] G. Laude, D. Calderini, D. P. Tew, and J. O. Richardson, "Ab initio instanton rate theory made efficient using Gaussian process regression," *Faraday Discussions*, vol. 212, pp. 237–258, 2018.

[347] A. M. Cooper, P. P. Hallmen, and J. Kästner, "Potential energy surface interpolation with neural networks for instanton rate calculations," *The Journal of Chemical Physics*, vol. 148, p. 094106, 2018.

[348] S. R. McConnell and J. Kästner, "Instanton rate constant calculations using interpolated potential energy surfaces in nonredundant, rotationally and translationally invariant coordinates," *Journal of Computational Chemistry*, vol. 40, pp. 866–874, 2019.

[349] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—a Python library for working with atoms," *Journal of Physics: Condensed Matter*, vol. 29, p. 273002, 2017.

[350] C. G. Broyden, "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations," *IMA Journal of Applied Mathematics*, vol. 6, pp. 76–90, 1970.

[351] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, pp. 317–322, 1970.

[352] D. Goldfarb, "A Family of Variable-Metric Methods Derived by Variational Means," *Mathematics of Computation*, vol. 24, pp. 23–26, 1970.

[353] D. F. Shanno, "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computation*, vol. 24, pp. 647–656, 1970.

[354] B. Matérn, *Spatial variation*, vol. 36 of *Lecture notes in statistics*. New York: Springer, 2nd ed., 1986.

[355] B. Minasny and A. B. McBratney, "The Matérn function as a general model for soil variograms," *Geoderma*, vol. 128, pp. 192 – 207, 2005. Pedometrics 2003.

[356] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. New York, NY, USA: Cambridge University Press, 2002.

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

[358] S. Califano, *Vibrational States*. Wiley, London, 1976.

[359] A. Baiardi, J. Bloino, and V. Barone, "General formulation of vibronic spectroscopy in internal coordinates," *The Journal of Chemical Physics*, vol. 144, p. 084114, 2016.

[360] E. B. Wilson Jr., *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra*. Dover Publications, 1980.

[361] C. Peng, P. Y. Ayala, H. B. Schlegel, and M. J. Frisch, "Using redundant internal coordinates to optimize equilibrium geometries and transition states," *Journal of Computational Chemistry*, vol. 17, pp. 49–56, 1996.

[362] F. Eckert, P. Pulay, and H.-J. Werner, "Ab initio geometry optimization for large molecules," *Journal of Computational Chemistry*, vol. 18, pp. 1473–1483, 1997.

[363] A. V. Mitin, "Use of symmetric rank-one hessian update in molecular geometry optimization," *Journal of Computational Chemistry*, vol. 19, pp. 1877–1886, 1998.

[364] R. Lindh, A. Bernhardsson, and M. Schütz, "Force-constant weighted redundant coordinates in molecular geometry optimizations," *Chemical Physics Letters*, vol. 303, pp. 567 – 575, 1999.

[365] O. Farkas and H. B. Schlegel, "Methods for optimizing large molecules. II. Quadratic search," *The Journal of Chemical Physics*, vol. 111, pp. 10806–10814, 1999.

[366] V. Bakken and T. Helgaker, "The efficient optimization of molecular geometries using redundant internal coordinates," *The Journal of Chemical Physics*, vol. 117, pp. 9160–9174, 2002.

[367] W. J. Hehre, R. F. Stewart, and J. A. Pople, "Self-Consistent Molecular-Orbital Methods. I. Use of Gaussian Expansions of Slater-Type Atomic Orbitals," *The Journal of Chemical Physics*, vol. 51, pp. 2657–2664, 1969.

[368] W. J. Hehre, R. Ditchfield, R. F. Stewart, and J. A. Pople, "Self-Consistent Molecular Orbital Methods. IV. Use of Gaussian Expansions of Slater-Type Orbitals. Extension to Second-Row Molecules," *The Journal of Chemical Physics*, vol. 52, pp. 2769–2773, 1970.

[369] Y. Shao, Z. Gan, E. Epifanovsky, A. T. B. Gilbert, M. Wormit, J. Kussmann, A. W. Lange, A. Behn, J. Deng, X. Feng, D. Ghosh, M. Goldey, P. R. Horn, L. D. Jacobson, I. Kaliman, R. Z. Khaliullin, T. Kús, A. Landau, J. Liu, E. I. Proynov, Y. M. Rhee, R. M. Richard, M. A. Rohrdanz, R. P. Steele, E. J. Sundstrom, H. L. Woodcock III, P. M. Zimmerman, D. Zuev, B. Albrecht, E. Alguire, B. Austin, G. J. O. Beran, Y. A. Bernard, E. Berquist, K. Brandhorst, K. B. Bravaya, S. T. Brown, D. Casanova, C.-M. Chang, Y. Chen, S. H. Chien, K. D. Closser, D. L. Crittenden, M. Diedenhofen, R. A. DiStasio Jr., H. Dop, A. D. Dutoi, R. G. Edgar, S. Fatehi, L. Fusti-Molnar, A. Ghysels, A. Golubeva-Zadorozhnaya, J. Gomes, M. W. D. Hanson-Heine, P. H. P. Harbach, A. W. Hauser, E. G. Hohenstein, Z. C. Holden, T.-C. Jagau, H. Ji, B. Kaduk, K. Khistyaev, J. Kim, J. Kim, R. A. King, P. Klunzinger, D. Kosenkov, T. Kowalczyk, C. M. Krauter, K. U. Lao, A. Laurent, K. V. Lawler, S. V. Levchenko, C. Y. Lin, F. Liu, E. Livshits, R. C. Lochan, A. Luenser, P. Manohar, S. F. Manzer, S.-P. Mao, N. Mardirossian, A. V. Marenich, S. A. Maurer, N. J. Mayhall, C. M. Oana, R. Olivares-Amaya, D. P. O'Neill, J. A. Parkhill, T. M. Perrine, R. Peverati, P. A. Pieniazek, A. Prociuk, D. R. Rehn, E. Rosta, N. J. Russ, N. Sergueev, S. M. Sharada, S. Sharmaa, D. W. Small, A. Sodt, T. Stein, D. Stück, Y.-C. Su, A. J. W. Thom, T. Tsuchimochi, L. Vogt, O. Vydrov, T. Wang, M. A. Watson, J. Wenzel, A. White, C. F. Williams, V. Vanovschi, S. Yeganeh, S. R. Yost, Z.-Q. You, I. Y. Zhang, X. Zhang, Y. Zhou, B. R. Brooks, G. K. L. Chan, D. M. Chipman, C. J. Cramer, W. A. Goddard III, M. S. Gordon, W. J. Hehre, A. Klamt, H. F. Schaefer III, M. W. Schmidt, C. D. Sherrill, D. G. Truhlar, A. Warshel, X. Xua, A. Aspuru-Guzik, R. Baer, A. T. Bell, N. A. Besley, J.-D. Chai, A. Dreuw, B. D. Dunietz, T. R. Furlani, S. R. Gwaltney, C.-P. Hsu, Y. Jung, J. Kong, D. S. Lambrecht, W. Liang, C. Ochsenfeld, V. A. Rassolov, L. V. Slipchenko, J. E. Subotnik, T. Van Voorhis, J. M. Herbert, A. I. Krylov, P. M. W. Gill, and M. Head-Gordon, "Advances in molecular quantum chemistry contained in the Q-Chem 4 program package," *Molecular Physics*, vol. 113, pp. 184–215, 2015.

[370] D. Packwood, J. Kermode, L. Mones, N. Bernstein, J. Woolley, N. Gould, C. Ortner, and G. Csányi, "A universal preconditioner for simulating condensed phase materials," *The Journal of Chemical Physics*, vol. 144, p. 164109, 2016.

[371] L. Mones, C. Ortner, and G. Csányi, "Preconditioners for the geometry optimisation and saddle point search of molecular systems," *Scientific Reports*, vol. 8, p. 13991, 2018.

[372] J. N. Wei, D. Duvenaud, and A. Aspuru-Guzik, “Neural Networks for the Prediction of Organic Chemistry Reactions,” *ACS Central Science*, vol. 2, pp. 725–732, 2016.

[373] R. Gómez-Bombarelli, J. Aguilera-Iparraguirre, T. D. Hirzel, D. Duvenaud, D. Maclaurin, M. A. Blood-Forsythe, H. S. Chae, M. Einzinger, D.-G. Ha, T. Wu, G. Markopoulos, S. Jeon, H. Kang, H. Miyazaki, M. Numata, S. Kim, W. Huang, S. I. Hong, M. Baldo, R. P. Adams, and A. Aspuru-Guzik, “Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach,” *Nature Materials*, vol. 15, p. 1120, 2016.

[374] M. Gastegger, J. Behler, and P. Marquetand, “Machine learning molecular dynamics for the simulation of infrared spectra,” *Chemical Science*, vol. 8, pp. 6924–6935, 2017.

[375] T. D. Huan, R. Batra, J. Chapman, S. Krishnan, L. Chen, and R. Ramprasad, “A universal strategy for the creation of machine learning-based atomistic force fields,” *npj Computational Materials*, vol. 3, p. 37, 2017.

[376] A. Goodrow, A. T. Bell, and M. Head-Gordon, “Transition state-finding strategies for use with the growing string method,” *The Journal of Chemical Physics*, vol. 130, p. 244108, 2009.

[377] A. Behn, P. M. Zimmerman, A. T. Bell, and M. Head-Gordon, “Efficient exploration of reaction paths via a freezing string method,” *The Journal of Chemical Physics*, vol. 135, p. 224108, 2011.

[378] P. M. Zimmerman, “Growing string method with interpolation and optimization in internal coordinates: Method and examples,” *The Journal of Chemical Physics*, vol. 138, p. 184102, 2013.

[379] G. Henkelman, G. Jóhannesson, and H. Jónsson, *Methods for Finding Saddle Points and Minimum Energy Paths*, pp. 269–302. Dordrecht: Springer Netherlands, 2002.

[380] S. Smidstrup, A. Pedersen, K. Stokbro, and H. Jónsson, “Improved initial guess for minimum energy path calculations,” *The Journal of Chemical Physics*, vol. 140, p. 214106, 2014.

[381] N. Artrith and A. M. Kolpak, “Understanding the Composition and Activity of Electrocatalytic Nanoalloys in Aqueous Solvents: A Combination of DFT and Accurate Neural Network Potentials,” *Nano Letters*, vol. 14, pp. 2670–2676, 2014.

[382] A. W. Hauser, J. Gomes, M. Bajdich, M. Head-Gordon, and A. T. Bell, “Subnanometer-sized Pt/Sn alloy cluster catalysts for the dehydrogenation of linear alkanes,” *Physical Chemistry Chemical Physics*, vol. 15, pp. 20727–20734, 2013.

[383] A. W. Hauser, P. R. Horn, M. Head-Gordon, and A. T. Bell, “A systematic study on Pt based, subnanometer-sized alloy cluster catalysts for alkane dehydrogenation: effects of intermetallic interaction,” *Physical Chemistry Chemical Physics*, vol. 18, pp. 10906–10917, 2016.

[384] A. E. Green, J. Justen, W. Schöllkopf, A. S. Gentleman, A. Fielicke, and S. R. Mackenzie, “IR Signature of Size-Selective CO2 Activation on Small Platinum Cluster Anions, $Pt_n^-$ (n=4–7),” *Angewandte Chemie International Edition*, vol. 57, pp. 14822–14826, 2018.

[385] R. Ditchfield, W. J. Hehre, and J. A. Pople, "Self-Consistent Molecular-Orbital Methods. IX. An Extended Gaussian-Type Basis for Molecular-Orbital Studies of Organic Molecules," *The Journal of Chemical Physics*, vol. 54, pp. 724–728, 1971.

[386] W. J. Hehre, R. Ditchfield, and J. A. Pople, "Self-Consistent Molecular Orbital Methods. XII. Further Extensions of Gaussian-Type Basis Sets for Use in Molecular Orbital Studies of Organic Molecules," *The Journal of Chemical Physics*, vol. 56, pp. 2257–2261, 1972.

[387] P. C. Hariharan and J. A. Pople, "The influence of polarization functions on molecular orbital hydrogenation energies," *Theoretica Chimica Acta*, vol. 28, pp. 213–222, 1973.

[388] T. Clark, J. Chandrasekhar, G. W. Spitznagel, and P. V. R. Schleyer, "Efficient diffuse function-augmented basis sets for anion calculations. III. the 3-21+G basis set for first-row elements, Li-F," *Journal of Computational Chemistry*, vol. 4, pp. 294–301, 1983.

[389] J. P. Perdew, "Density-functional approximation for the correlation energy of the inhomogeneous electron gas," *Physical Review B*, vol. 33, pp. 8822–8824, 1986.

[390] T. H. Dunning, "Gaussian Basis Functions for Use in Molecular Calculations. I. Contraction of (9s5p) Atomic Basis Sets for the First-Row Atoms," *The Journal of Chemical Physics*, vol. 53, pp. 2823–2833, 1970.

[391] T. H. Dunning and P. J. Hay, *Gaussian Basis Sets for Molecular Calculations*, pp. 1–27. Boston, MA: Springer US, 1977.

[392] D. Andrae, U. Häußermann, M. Dolg, H. Stoll, and H. Preuß, "Energy-adjusted ab initio pseudopotentials for the second and third row transition elements," *Theoretica Chimica Acta*, vol. 77, pp. 123–141, 1990.

[393] A good argument for staying with Cartesian representations is the straight-forward interpretation of spring forces and a treatment of all degrees of freedom on the same footing.

[394] J. P. Perdew and A. Ruzsinszky, "Fourteen easy lessons in density functional theory," *International Journal of Quantum Chemistry*, vol. 110, pp. 2801–2807, 2010.

[395] A. D. Becke, "Perspective: Fifty years of density-functional theory in chemical physics," *The Journal of Chemical Physics*, vol. 140, p. 18A301, 2014.

[396] W. Kohn and L. J. Sham, "Self-Consistent Equations Including Exchange and Correlation Effects," *Physical Review*, vol. 140, pp. A1133–A1138, 1965.

[397] L.-W. Wang and M. P. Teter, "Kinetic-energy functional of the electron density," *Physical Review B*, vol. 45, pp. 13196–13220, 1992.

[398] E. Smargiassi and P. A. Madden, "Orbital-free kinetic-energy functionals for first-principles molecular dynamics," *Physical Review B*, vol. 49, pp. 5220–5226, 1994.

[399] F. Perrot, "Hydrogen-hydrogen interaction in an electron gas," *Journal of Physics: Condensed Matter*, vol. 6, pp. 431–446, 1994.

[400] Y. A. Wang, N. Govind, and E. A. Carter, "Orbital-free kinetic-energy functionals for the nearly free electron gas," *Physical Review B*, vol. 58, pp. 13465–13471, 1998.

[401] Y. A. Wang, N. Govind, and E. A. Carter, "Orbital-free kinetic-energy density functionals with a density-dependent kernel," *Physical Review B*, vol. 60, pp. 16350–16358, 1999.

[402] C. Huang and E. A. Carter, "Nonlocal orbital-free kinetic energy density functional for semiconductors," *Physical Review B*, vol. 81, p. 045206, 2010.

[403] W. Mi, A. Genova, and M. Pavanello, "Nonlocal kinetic energy functionals by functional integration," *The Journal of Chemical Physics*, vol. 148, p. 184107, 2018.

[404] L. Hung and E. A. Carter, "Accurate simulations of metals at the mesoscale: Explicit treatment of 1 million atoms with quantum mechanics," *Chemical Physics Letters*, vol. 475, pp. 163–170, 2009.

[405] M. Chen, X.-W. Jiang, H. Zhuang, L.-W. Wang, and E. A. Carter, "Petascale Orbital-Free Density Functional Theory Enabled by Small-Box Algorithms," *Journal of Chemical Theory and Computation*, vol. 12, pp. 2950–2963, 2016.

[406] X. Shao, W. Mi, Q. Xu, Y. Wang, and Y. Ma, "O ( N log N ) scaling method to evaluate the ion–electron potential of crystalline solids," *The Journal of Chemical Physics*, vol. 145, p. 184110, 2016.

[407] X. Shao, Q. Xu, S. Wang, J. Lv, Y. Wang, and Y. Ma, "Large-scale ab initio simulations for periodic system," *Computer Physics Communications*, vol. 233, pp. 78–83, 2018.

[408] A. Aguado, J. M. López, J. A. Alonso, and M. J. Stott, "Melting in Large Sodium Clusters: An Orbital-Free Molecular Dynamics Study," *Journal of Physical Chemistry B*, vol. 105, pp. 2386–2392, 2001.

[409] A. Aguado and J. M. López, "Molecular dynamics simulations of the meltinglike transition in $Li_{13}Na_{42}$ and $Na_{13}Cs_{42}$ clusters," *Physical Review B*, vol. 71, p. 075415, 2005.

[410] V. Gavini, J. Knap, K. Bhattacharya, and M. Ortiz, "Non-periodic finite-element formulation of orbital-free density functional theory," *Journal of the Mechanics and Physics of Solids*, vol. 55, pp. 669–696, 2007.

[411] J. Xia and E. A. Carter, "Density-decomposed orbital-free density functional theory for covalently bonded molecules and materials," *Physical Review B*, vol. 86, p. 235109, 2012.

[412] J. Xia, C. Huang, I. Shin, and E. A. Carter, "Can orbital-free density functional theory simulate molecules?," *The Journal of Chemical Physics*, vol. 136, p. 084102, 2012.

[413] L. Li, J. C. Snyder, I. M. Pelaschier, J. Huang, U.-N. Niranjan, P. Duncan, M. Rupp, K.-R. Müller, and K. Burke, "Understanding machine-learned density functionals," *International Journal of Quantum Chemistry*, vol. 116, pp. 819–833, 2016.

[414] J. M. Alred, K. V. Bets, Y. Xie, and B. I. Yakobson, "Machine learning electron density in sulfur crosslinked carbon nanotubes," *Composites Science and Technology*, vol. 166, pp. 3–9, 2018.

[415] A. Grisafi, A. Fabrizio, B. Meyer, D. M. Wilkins, C. Corminboeuf, and M. Ceriotti, "Transferable Machine-Learning Model of the Electron Density," *ACS Central Science*, vol. 5, pp. 57–64, 2019.

[416] A. Fabrizio, A. Grisafi, B. Meyer, M. Ceriotti, and C. Corminboeuf, "Electron density learning of non-covalent systems," *Chemical Science*, vol. 10, pp. 9424–9432, 2019.

[417] A. Chandrasekaran, D. Kamal, R. Batra, C. Kim, L. Chen, and R. Ramprasad, "Solving the electronic structure problem with machine learning," *npj Computational Materials*, vol. 5, p. 22, 2019.

[418] A. T. Fowler, C. J. Pickard, and J. A. Elliott, "Managing uncertainty in data-derived densities to accelerate density functional theory," *Journal of Physics: Materials*, vol. 2, p. 034001, 2019.

[419] J. Schmidt, C. L. Benavides-Riveros, and M. A. L. Marques, "Machine Learning the Physical Nonlocal Exchange–Correlation Functional of Density-Functional Theory," *Journal of Physical Chemistry Letters*, vol. 10, pp. 6425–6431, 2019.

[420] S.-C. Lin and M. Oettel, "A classical density functional from machine learning and a convolutional neural network," *SciPost Physics*, vol. 6, p. 25, 2019.

[421] R. Bhavsar and R. Ramakrishnan, "Machine learning modeling of Wigner intracule functionals for two electrons in one-dimension," *The Journal of Chemical Physics*, vol. 150, p. 144114, 2019.

[422] S.-C. Lin, G. Martius, and M. Oettel, "Analytical classical density functionals from an equation learning network," *The Journal of Chemical Physics*, vol. 152, p. 021102, 2020.

[423] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 3 ed., 2007.

[424] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[425] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[426] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[427] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, IEEE, 2016.

[428] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating Second-Order Functional Knowledge for Better Option Pricing," in *Advances in Neural Information Processing Systems 13* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), pp. 472–478, MIT Press, 2001.

[429] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 2011.

[430] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[431] H. Jiang and W. Yang, "Conjugate-gradient optimization method for orbital-free density functional calculations," *The Journal of Chemical Physics*, vol. 121, pp. 2030–2036, 2004.

[432] V. Vapnik, S. E. Golowich, and A. J. Smola, "Support Vector Method for Function Approximation, Regression Estimation and Signal Processing," in *Advances in Neural Information Processing Systems 9* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), pp. 281–287, MIT Press, 1997.

[433] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support Vector Regression Machines," in *Advances in Neural Information Processing Systems 9* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), pp. 155–161, MIT Press, 1997.

[434] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[435] A. J. Smola and P. L. Bartlett, "Sparse Greedy Gaussian Process Regression," in *Advances in Neural Information Processing Systems 13* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), pp. 619–625, MIT Press, 2001.

[436] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using Pseudo-inputs," in *Advances in Neural Information Processing Systems 18* (Y. Weiss, B. Schölkopf, and J. C. Platt, eds.), pp. 1257–1264, MIT Press, 2006.

[437] M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics* (D. van Dyk and M. Welling, eds.), vol. 5 of *Proceedings of Machine Learning Research*, (Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA), pp. 567–574, PMLR, 2009.

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

[439] H. Henderson and S. Searle, "On Deriving the Inverse of a Sum of Matrices," *SIAM Review*, vol. 23, pp. 53–60, 1981. Equation (13).

[440] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[441] W. Duncan, "LXXVIII. Some devices for the solution of large sets of simultaneous linear equations," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 35, pp. 660–670, 1944.

[442] L. Guttman, "Enlargement Methods for Computing the Inverse Matrix," *The Annals of Mathematical Statistics*, vol. 17, pp. 336–343, 1946. Equation (14).

# Acknowledgments

First and foremost I want to thank my supervisor Andreas Hauser who granted me complete "creative freedom" in my research while also setting the deadlines my perfectionist self so desperately needs. I am also thankful for the fruitful discussions and great advice provided by Wolfgang Ernst and the memorable time with the master's students I had the pleasure to work with.

I want to thank my friends and fellow PhD-candidates for our weekly therapeutic meetings, helping me through the ups and downs of the academic life. Special thanks go to Anna for offering a shoulder to lean on and her patience even through the numerous late night work sessions.

Finally, I want to thank my parents who supported me throughout my studies, my brother who provided invaluable computer science advice, and my grandparents who I wish could celebrate this moment with me.