Alexander Fuchs BSc

A neural network approach to Ni-Au nano-cluster modelling

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Technical Physics

submitted to

Graz University of Technology

Supervisor

Ass. Prof. Mag. phil. Dipl.-Ing. Dr. phil. Dr. techn. Andreas Hauser

Insitute of Experimental Phyics

Graz, March 2018

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

25.3.2018

Date

[illegible]

Signature

Kurzfassung

Der Einsatz quantenchemischer Methoden zur Modellierung und Verifikation experimenteller Messungen in der Molekül- und Clusterphysik hat in den letzten Jahrzehnten stark an Bedeutung dazugewonnen. Sie erweisen sich als wertvolles Instrument, um Erkenntnisse über physikalische Zusammenhänge der Nanowelt zu gewinnen, welche anders nur schwer greifbar wären. Trotz der stets steigenden Rechenleistung moderner Computer ist die Größe der mittels *ab initio*-Methoden berechenbaren Systeme limitiert. Als Alternativen bieten sich Näherungsverfahren auf der Basis von Kraftfeldern an, welche die Rechenzeit zwar um ein Vielfaches verringern können, allerdings aber auch Einbußen bei der Genauigkeit der Simulation mit sich bringen.
Der von Behler und Parinello[1] vorgeschlagene Ansatz, Kraftfelder über neuronale Netzwerke zu beschreiben, wurde bereits für einige Systeme erfolgreich umgesetzt. Er reduziert einerseits die benötigte Rechenzeit gegenüber *ab initio*-Berechnungen, andererseits konnte auch die Genauigkeit gegenüber herkömmlichen Kraftfeldansätzen gesteigert werden. Das Hauptziel dieser Diplomarbeit war es, einen lauffähigen Programmcode für ein neuronales Netzwerk nach der von Behler und Parinello vorgeschlagenen Struktur zu implementieren. Das Programm wurde anschließend optimiert und an die Aufgabe der Simulation von Ni-Au Nanopartikeln angepasst. Erste Ergebnisse für das bimetallische Ni-Au-System wurden mit anderen Methoden verglichen und bewertet.

Abstract

The use of computational chemistry methods for the modelling of experiments and the verification of measurements in molecular and cluster physics has gained popularity during the past years. It has become a valuable tool for the understanding of the nano-world and the underlying physical principles, which would be intangible otherwise. Despite the constant rise in computational power, the feasible system size of *ab initio* methods is still limited. To overcome this flaw, the introduction of approximations based on force fields can reduce the computational effort significantly. However, this also introduces larger errors to the simulation.
The ansatz proposed by Behler and Parinello[1] to describe force-fields via neuronal networks has been applied successfully to a variety of systems. Besides a large gain in evaluation speed compared to *ab initio* methods, it also outperforms conventional force-field methods in terms of chemical accuracy. The main focus of this thesis was the implementation of a running neural network code based on the structure proposed by Behler and Parinello. The program was further optimised and adapted for the simulation of Ni-Au nanoparticles. First results for the bimetallic Ni-Au-system were compared to other methods.

Acknowledgements

First and foremost I would like to thank my supervisor Ass.Prof. Mag. Dipl.-Ing. DDr. Andreas W. Hauser for his ongoing support and the opportunity to write this thesis. Together with him and my colleagues Ralf and Johannes I could gain deeper insights into this novel and interesting topic.

I also would like to thank the FWF Austria who made this thesis possible through the "Heterogeneous catalysis on metallic nanoparticles" project.

Last but not least I would also like to express my deep gratitude to my friends and family who supported me throughout my whole university career.

Contents

List of Figures

List of Tables

1. Introduction

Neuronal networks have become a valuable tool across a very large spectrum of disciplines. Their flexible functional form makes them also applicable to represent potential energy surfaces of molecules and other atomic compounds. This thesis will manly focus on the use of neural networks for predicting energies of metal cluster systems.

1.1. Motivation

Simulation of atomic and molecular processes has become a valuable tool for the prediction, verification and understanding of experimental results. Traditional approaches such as force-field based or *ab initio* methods have a large accuracy versus cost trade-off and can therefore only be applied to a limited field of problems. The force-field approach offers fast simulation of large scale systems, but for example fails to describe the making and breaking of bonds between the atoms. On the other side, the *ab initio* methods such as Hartree-Fock and Density Functional Theory are very accurate at describing molecules, small clusters and periodic systems, but are computationally rather expensive and it is difficult to achieve convergence for large metallic systems. The flexible and simple functional form of the neuronal networks offers a good compromise between accuracy and evaluation time of the model and should give a good representation of the system. The method still relies on training data produced by traditional *ab initio* methods. Therefore, the training data should be cheap in production, but also representative for the system under study.

1.2. Target

The target of this thesis is to develop a suitable neuronal network and a complementing framework which will then be applied to quantum chemistry problems and be compared to traditional methods and other neuronal network approaches. The main focus of this thesis lies on metal-cluster systems; in particular, on the Ni-Au system. During the thesis the following questions will be addressed:

- Force-Fields: Can the PES of metal clusters be represented by neuronal networks?
- Quantum Mechanical Interactions: Can quantum mechanical interactions be approximated by a neuronal network?
- Extrapolation capability: Are systems which are larger than those contained in the training set sufficiently well described by the neuronal net?
- Outlook on further improvements: How could the method be further improved? Is it possible to reduce the amount of training data without loosing accuracy?

1.3. Structure

Chapter 2 provides a short introduction into quantum-chemistry. The methods used for data generation will be explained. Furthermore, a general introduction about neuronal nets and the most common used neuronal net approaches for quantum chemistry will be discussed.
In **C**hapter 3 a detailed structure of the neuronal network and the Python framework will be presented. The most important methods and functionalities will be described and explained.
Chapter 4 contains details for the Ni-Au system. The model and the data generation for the Ni-Au cluster system will be discussed. Results of the method will be presented and compared to other methods.
In **C**hapter 5 this thesis will discuss the results of the used method and also give an outlook on further improvements.

1.4. Common Abbreviations

Abbreviation	**Meaning**
DFT	**D**ensity **F**unctional **T**heory
HF	**H**artree-**F**ock method
PES	**P**otential **E**nergy **S**urface
NN	**N**euronal **N**etwork
FNN	Feed **F**orward **N**eural **N**etwork
ANN	**A**tomic **N**eural **N**etwork

2. Theoretical background

2.1. Quantum-Chemistry

The following sections 2.1 -2.4 follow the book of F. Jensen entitled Introduction to Computational Chemistry.[1] The latter can be understood as the attempt to study chemical relevant processes by solving the underlying many-body problem containing electrons and nuclei with computational methods. The properties of molecular systems are dependent on the types of atoms involved, their number of electrons and the overall geometric arrangement. For a given system, quantum chemistry therefore attempts to calculate various properties such as:

- Equilibrium geometries for the system
- Relative energies of geometries
- Properties of a system (Dipole moment, polarizablility, etc.)
- Transition rates from one into another geometry
- Dynamic behaviour of the system over time
- Interaction of different atoms and molecules

2.2. Describing a system

To describe a sytem one needs four fundamental features:

- The system - What are the particles and where are they?
- Starting conditions - What are the particles starting conditions concerning position and velocity?
- Interaction - What is the mathematical form of the forces acting between the particles?
- Dynamics equation - How can the time evolution of the systems be described mathematically?

The interactions in a molecular system are determined by the underlying potential (**V**). The forces (**F**) on particles contained in that system are the result of a gradient in the potential at the position of the atom.

$$F = -\nabla V. \tag{2.1}$$

For interactions on atomic level the only relevant interaction is the electromagnetic interaction. In most cases, the simple form of the Coulomb potential is sufficient:

$$V_{\text{Coulomb}}(r_{ij}) = \frac{q_i q_j}{r_{ij}}. \tag{2.2}$$

[1]See reference 4, Chapter 1, pages 3-10.

In the picture of QED (Quantum Electro Dynamics) the Coulomb interaction is the zeroth-order term of the potential. For calculations of higher accuracy, higher order terms are are included to account for other effects such as electron-electron interactions.

$$V_{\text{elec}}(r_{12}) = \frac{1}{r_{12}} \left[1 - \frac{1}{2} \left(v_1 \cdot v_2 + \frac{(v_1 \cdot r_{12)} + (v_2 \cdot r_{12})}{r_{12}^2} \right) \right], \tag{2.3}$$

with v_i being the velocity vector of particle i.

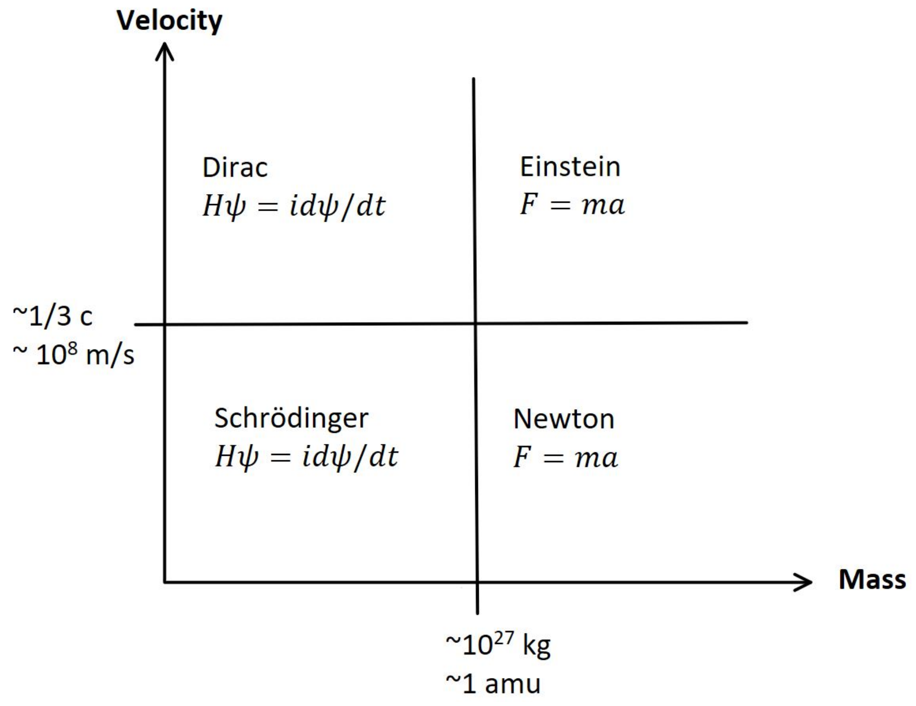


Figure 2.1.: Domains of dynamical equations.

The correct description of a system is dependent on its mass and the velocity of its particles. Figure 2.1 illustrates the domains of dynamic equations. For slowly moving heavy particles Newtonian mechanics applies (see 2.36), but as the velocity increases, relativistic effects become more important and one has to switch to relativistic mechanics and account for the relativistic mass (see 2.5).

$$F = \frac{dp}{dt} = m\frac{dv}{dt} = ma, \tag{2.4}$$

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}}. \tag{2.5}$$

The same applies to light particles. At slow speeds the Schrödinger equation applies very well(see eq.: 2.6 to 2.8), but it neglects relativistic effects which can become of relevance

for electrons close to the nuclei. Therefore, the Dirac-equation has to be introduced.
In the non-relativistic case the Schrödinger equation applies,

$$H\Psi = i\frac{\partial\Psi}{\partial t}, \tag{2.6}$$

with H being the Hamiltonian operator

$$H_{\text{Schrödinger}} = T + V, \tag{2.7}$$

V being the potential and the kinetic energy

$$T = \frac{p^2}{2m} = -\frac{1}{2m}\Delta. \tag{2.8}$$

In the case of a relativistic description, the Hamiltonian has the form

$$H_{\text{Dirac}} = (c \cdot \alpha \cdot p + \beta \cdot m \cdot c^2) + V, \tag{2.9}$$

with α and β being 4x4 matrices representing the relativistic time and the 3 space dimensions.
For a many-particle system, the Hamiltonian has to include the motion of all nuclei and electrons and their interactions with each other:

$$H = -\sum_{i=1}^{N}\frac{1}{2}\Delta_i^2 - \sum_{a=1}^{M}\frac{1}{2M_a}\Delta_a^2 - \sum_{i=1}^{N}\sum_{a=1}^{M}\frac{Z_a}{r_{ia}} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{r_{ij}} + \sum_{a=1}^{M}\sum_{b>a}\frac{Z_aZ_b}{R_{ab}}. \tag{2.10}$$

2.3. Hartree-Fock

Hartree-Fock[2] (HF) is a method for solving the time-independent Schrödinger equation for electrons in a given nuclear arrangement.

$$H\Psi = E\Psi \tag{2.11}$$

It is a mean field theory method relying on the variational principle and has to be solved iteratively.

2.3.1. Born-Oppenheimer approximation

An important simplification for solving the Schrödinger equation is the Born-Oppenheimer approximation, which assumes that the motion of the electrons is much faster than the motion of the nuclei. Therefore, the coupling between the nuclei and the electronic motion is neglected. At first the terms of the Hamiltonian 2.10 get restructured:

$$\begin{aligned} H_{\text{tot}} &= T_{\text{n}} + H_{\text{e}} + H_{\text{mp}}, \\ H_{\text{e}} &= T_{\text{e}} + V_{\text{ne}} + V_{\text{ee}} + V_{\text{nn}}. \end{aligned}$$

[2]See reference 4, Chapter 3, pages 80-92.

Here H_e is the electronic Hamiltonian and H_mp is the mass-polarization Hamiltonian,

$$H_\mathrm{mp} = -\frac{1}{2M_\mathrm{tot}}\left(\sum_i^{N_\mathrm{elec}} \nabla_i\right)^2 . \tag{2.12}$$

With M_tot the total mass of all nuclei.
Ψ_tot can be expanded in the complete set of electronic functions with the expansion coefficients being functions of the nuclear coordinates,

$$\Psi_\mathrm{tot}(R,r) = \sum_{i=1}^{\infty} \Psi_{ni}(R)\Psi_i(R,r). \tag{2.13}$$

Now H_e only depends on the nuclear positions, but not their momenta. Equation 2.13 inserted in 2.3 yields:

$$\sum_{i=1}^{\infty}(T_\mathrm{n} + H_\mathrm{e} + H_\mathrm{mp})\Psi_{ni}(R)\Psi_i(R,r) = E_\mathrm{tot}\sum_{i=1}^{\infty} \Psi_{ni}(R)\Psi_i(R,r). \tag{2.14}$$

By considering that H_e and H_mp only act on the electronic part of the wave function, and using the fact that Ψ_i is an exact solution of the electronic Schrödinger equation, one arrives at the expression

$$\begin{aligned}\sum_{i=1}^{\infty} \Psi_i(\nabla_n^2\Psi_{ni}) + 2(\nabla_n\Psi_i)(\nabla_n\Psi_{ni}) + \Psi_{ni}(\nabla_n^2\Psi_i)\\ + \Psi_{ni}E_i\Psi_i + \Psi_{ni}H_\mathrm{mp}\Psi_i = E_\mathrm{tot}\sum_{i=1}^{\infty}\Psi_{ni}\Psi_i.\end{aligned}$$

Using the orthonormality of Ψ_i and multiplying by Ψ_j^* from the left we arrive at:

$$\begin{aligned}\nabla_n^2\Psi_{nj} + E_j\Psi_{nj} + \sum_{i=1}^{\infty} 2\left\langle\Psi_j\right|\nabla_n\left|\Psi_i\right\rangle(\nabla_n\Psi_{ni})\\ + \left\langle\Psi_j\right|\nabla^2\left|\Psi_i\right\rangle\Psi_{ni} + \left\langle\Psi_j\right|H_\mathrm{mp}\left|\Psi_i\right\rangle\Psi_{ni} = E_\mathrm{tot}\Psi_{nj}.\end{aligned}$$

In the adiabatic expansion the wave function is restricted to a single electronic surface, resulting in the neglection of all the coupling terms:

$$(\nabla_n^2 + E_j + \left\langle\Psi_j\right|\nabla_n^2\left|\Psi_j\right\rangle + \left\langle\Psi_j\right|H_\mathrm{mp}\left|\Psi_j\right\rangle)\Psi_{nj} = E_\mathrm{tot}\Psi_{nj}.$$

After neglecting the mass-polarization and reintroducing the kinetic energy operator one arrives at:

$$(T_\mathrm{n} + E_j(R) + U(R))\Psi_{nj}(R) = E_\mathrm{tot}\Psi_{nj}(R).$$

In the Born-Oppenheimer approximation the diagonal correction $U(R)$ is neglected, resulting in

$$(T_\mathrm{n} + E_j(R))\Psi_{nj}(R) = E_\mathrm{tot}\Psi_{nj}(R). \tag{2.15}$$

Hence, in the Born-Oppenheimer picture the atoms move on a potential energy surface (PES) with their energy only depending on R and the kinetic energy.

When describing many-electron systems the total wave function has to be anti-symmetric, i.e. an anti-symmetrized product of one electron orbitals. A convenient way to create such a wave function is a by writing it as a Slater determinant:

$$\Phi_{SD} = \frac{1}{\sqrt{N}} \begin{vmatrix} \Phi_1(1) & \Phi_2(1) & \dots & \Phi_N(1) \\ \Phi_1(2) & \Phi_2(2) & \dots & \Phi_N(2) \\ \dots & \dots & \dots & \dots \\ \Phi_1(N) & \Phi_2(N) & \dots & \Phi_N(N) \end{vmatrix} ; \langle \Phi_i | \, | \Phi_j \rangle = \delta_{ij}. \tag{2.16}$$

2.3.2. Hartree-Fock equations

After evaluating all the terms in the many-body Hamiltonian 2.39, and considering permutations between particles, one arrives at following expression for the energy,

$$E = \sum_{i=1}^{N_{\text{elec}}} h_i + \frac{1}{2} \sum_{i=1}^{N_{\text{elec}}} \sum_{j=1}^{N_{\text{elec}}} (J_{ij} - K_{ij}) + V_{\text{nn}}, \tag{2.17}$$

with

$$V_{nn} = \sum_{a}^{N_{\text{nuclei}}} \sum_{b>a}^{N_{\text{nuclei}}} \frac{Z_a Z_b}{|R_a - R_b|},$$

$$h_i = -\frac{1}{2}\nabla_i^2 - \sum_{a}^{N_{\text{nuclei}}} \frac{Z_a}{|R_a - r_i|}.$$

Coulomb integrals, denoted as J_{ij}, represent the classical repulsion between two electrons. K_{ij} are the exchange integrals which have no classical equivalent. The target is to determine the set of molecular orbitals (MOs) that minimizes the energy, with the MOs remaining orthogonal and normalized. Such a constrained optimization can be done via Lagrange multipliers. The condition for the optimization is that the Lagrange function is stationary with respect to an orbital variation,

$$\delta L = \delta E - \sum_{ij}^{N_{\text{elec}}} \lambda_{ij} (\langle \delta\phi_i | \, | \phi_j \rangle - \langle \phi_i | \, | \delta\phi_j \rangle) = 0 \tag{2.18}$$

The variation of the energy is given by

$$\delta E = \sum_{i}^{N_{\text{elec}}} (\langle \delta\phi_i | \, h_i \, | \phi_i \rangle + \langle \phi_i | \, h_i \, | \delta\phi_i \rangle) + \sum_{ij}^{N_{\text{elec}}} (\langle \delta\phi_i | \, J_j - K_i \, | \phi_j \rangle + \langle \phi_i | \, J_j - K_j \, | \delta\phi_j \rangle). \tag{2.19}$$

Now we can introduce the Fock-operator F_i,

$$\delta E = \sum_{i}^{N_{\text{elec}}} (\langle \delta\phi_i | \, F_i \, | \phi_i \rangle + \langle \phi_i | \, F_i \, | \delta\phi_i \rangle), \tag{2.20}$$

an effective one-electron operator, describing the kinetic energy and the interaction with all nuclei and all other electrons:

$$F_i = h_i + \sum_j^{N_{\text{elec}}} (J_j - K_j). \tag{2.21}$$

Inserting the Fock-operator in equation 2.18 gives

$$\delta L = \sum_i^{N_{\text{elec}}} \langle \delta\phi_i | F_i | \phi_j \rangle \langle \phi_i | F_i | \delta\phi_j \rangle - \sum_{ij}^{N_{\text{elec}}} \lambda_{ij} (\langle \delta\phi_i | | \phi_j \rangle + \langle \phi_i | | \delta\phi_j \rangle \,,$$

using $\langle \phi | | \delta\phi \rangle = \langle \delta\phi | | \phi \rangle$ and $\langle \phi | F | \delta\phi \rangle = \langle \delta\phi | F | \phi \rangle$ one arrives at

$$\sum_{ij}^{N_{\text{elec}}} (\lambda_{ij} - \lambda_{ij}*) \langle \delta | | \phi_j \rangle = 0.$$

This means that $\lambda_{ij} = \lambda_{ji}^*$ and the Fock matrix is hermitian.
The final Hartree-Fock equations can be written as

$$F\phi_i = \sum_j^{N_{\text{elec}}} \lambda_{ij} \phi_j. \tag{2.22}$$

By choosing a unitary transformation, the Lagrange multiplier matrix becomes diagonal and the equations can be transformed into a standard Eigenvalue problem,

$$F_i \phi_i' = \epsilon \phi_i'. \tag{2.23}$$

Finally, the total energy can be written as

$$E = \sum_i^{N_{\text{elec}}} \epsilon_i - \frac{1}{2} \sum_{ij}^{N_{\text{elec}}} (J_{ij} - K_{ij}) + V_{\text{nn}}. \tag{2.24}$$

2.4. Density-Functional-Theory

Density functional theory[3] (DFT) is based on the proof by Hohenberg and Kohn that the electronic ground state energy is solely determined by the electron density ρ. In principle this means that the dimensionality of the problem can be reduced drastically from 4N to (three spatial one spin dimension) to 4 variables. The only problem of DFT is that the functional connecting the density ρ to the energy of the system is not know. Nevertheless, there are approaches to solve the problem by estimating the energies for the kinetic and exchange part of the total energy. The most common approach is to use Kohn-Sham theory, which reintroduces molecular orbitals in order to get a better estimate of the kinetic energy.

2.4.1. Kohn-Sham equations

The introduction of orbitals in DFT is the key behind today‘s use in computational chemistry, as orbital-free methods provide a insufficiency representation of the kinetic energy. Therefore, the Kohn-Sham formalism splits the kinetic energy into two parts, one which can be calculated exactly and a small correction term. The re-introduction of orbitals increased the complexity of the problems again, from 4 to 4N variables and the electron correlation term re-appears in the Hamiltonian:

$$H_\lambda = T + V_{\text{ext}}(\lambda) + \lambda \cdot V_{\text{ee}}, \tag{2.25}$$

with λ being a parameter proportional to the considered electron-electron interaction. For a non-interacting system with $\lambda = 0$ the kinetic energy has the form

$$T_S = \sum_{i=1}^{N_{\text{elec}}} \langle\phi_i| - \frac{1}{2}\nabla^2 |\phi_i\rangle . \tag{2.26}$$

By considering natural orbitals (NO) the exact kinetic energy can be calculated via

$$\begin{aligned} T[\rho_{\text{exact}}] &= \sum_{i=1}^{\infty} n_i \langle\phi_i^{\text{NO}}| - \frac{1}{2}\nabla^2 |\phi_i^{NO}\rangle , \\ \rho_{\text{exact}} &= \sum_{i=1}^{\infty} n_i |\phi_j^{\text{NO}}|^2, \\ N_{\text{elec}} &= \sum_{i=1}^{\infty} n_i. \end{aligned} \tag{2.27}$$

Since the exact density matrix and therefore the natural orbitals are not known, the approximate density can be written in terms of auxiliary one-electron functions.

$$\rho_{\text{approx}} = \sum_{i=1}^{N_{\text{elec}}} |\phi_i|^2 \tag{2.28}$$

[3] See reference 4, Chapter 6, pages 232-252.

By absorbing the remaining kinetic energy into the exchange-correlation term the general energy expression for DFT becomes

$$E_{\text{DFT}}[\rho] = T_{\text{S}}[\rho] + E_{\text{ne}}[\rho] + J[\rho] + E_{\text{XC}}[\rho]. \tag{2.29}$$

By setting E_{DFT} equal to the exact energy solution the correlation energy becomes

$$E_{\text{XC}}[\rho] = (T[\rho] - T_{\text{S}}[\rho]) + (E_{\text{ee}}[\rho]) - J[\rho]). \tag{2.30}$$

2.4.2. Exchange-Correlation Functionals

The various DFT methods differ only in their choice for the exchange-correlation energy functional. The explicit form of this potential has been elusive, except for special cases such as a uniform electron gas. However, it is possible to derive a number of properties such a functional would have to fulfil.

- The energy functional should be self-interaction-free.
- For a constant density, the uniform electron gas should be recovered.
- Rescaling the coordinates should result in a linear rescaling of the exchange energy: $\rho_\lambda(x, y, z) = \lambda^3 \rho_\lambda(\lambda x, \lambda y, \lambda z)$ and $E_{\text{X}}[\rho_\lambda] = \lambda E_{\text{X}}[\rho]$.
- When scaling the electron coordinates by a factor larger than 1 the magnitude of correlation should increase: $-E_{\text{C}}[\rho_\lambda] > -\lambda E_{\text{C}}[\rho]; \lambda > 1$.
- If the scaling parameter goes to infinity the correlation energy for a finite system has to approach a negative constant.
- A lower bound for the exchange-correlation energy relative to the LDA exchange energy is given by the Lieb-Oxford condition: $E_{\text{X}}[\rho] \geq E_{\text{XC}}[\rho] \geq 2.273 E_{\text{X}}^{\text{LDA}}[\rho]$.
- The exchange potential should have an asymptotic $\frac{1}{r}$ behaviour as $r \to \infty$.
- The correlation potential should have an asymptotic $-\frac{1}{2}\alpha r^{-4}$ behaviour, where α is the polarizability of the $N_{\text{elec}} - 1$ system.

2.4.3. Local Density Approximation

The Local Density Approximation assumes that the density is a slowly varying function. The exchange energy for this approximation is given by the Dirac formula:

$$E_{\text{X}}^{\text{LDA}}[\rho] = -C_{\text{X}} \int \rho^{\frac{4}{3}}(\vec{r}) d\vec{r}, \tag{2.31}$$

with the energy density

$$\epsilon_{\text{X}}^{\text{LDA}} = -C_{\text{X}} \rho^{\frac{1}{3}}. \tag{2.32}$$

When considering spin, the densities are split into separate densities ρ_α and ρ_β for each spin. This approximation is then called Local Spin Density Approximation (LSDA).

2.4.4. Gradient Corrected Methods

An obvious improvement to LDA is to add a dependence on derivatives of the density as well. The Generalized Gradient Approximation(GGA) methods include the first derivative of the density with the additional constraints that the integration over Fermi and Coulomb holes gives the required values of -1 and 0. A popular functional is the PBE (Perdew-Burke-Ernzerhof) functional. The exchange energy can be written as an enhancement factor multiplied onto the LSDA functional.

$$\begin{aligned} \epsilon_X^{\mathrm{PBE}} &= \epsilon_X^{\mathrm{LDA}} F(x), \\ F(x) &= 1 + a - \frac{a}{1+bx^2}. \end{aligned} \tag{2.33}$$

For the correlation part the correction is added to the LDA expression,

$$\begin{aligned} \epsilon_C^{\mathrm{PBE}} &= \epsilon_C^{\mathrm{LDA}} + H(t), \\ H(t) &= cf_3^3 ln \left[1 + dt^2 \left(\frac{1+At^2}{1+At^2+A^2t^4}\right)\right], \\ A &= d\left[\exp\left(-\frac{\epsilon_C^{\mathrm{LDA}}}{cf_3^3}\right) - 1\right]^{-1}, \\ f_3(\zeta) &= \frac{1}{2}\left[(1-\zeta)^{\frac{2}{3}} + (1-\zeta)^{\dfrac{2}{3}}\right], \\ t &= \left[2(3\pi^3)^{\frac{1}{3}} f_3\right]^{-1} x. \end{aligned} \tag{2.34}$$

The parameters a, b, c and d are not obtained by fitting to experimental data, but derived from the conditions mentioned in section 2.4.2.

2.4.5. Plane Wave Basis Functions

When modelling infinite systems with periodic boundary conditions, the appropriate choice of basis functions are functions with an infinite range, e.g. plane waves. Since the outer valence electrons in metals behave like free electrons, the solutions of the Schrödinger equation for free electrons can be used as a basis. The allowed values for $\vec{k}$ are given by the unit cell translational vector $\vec{t}$ with m being an integer.

$$\begin{aligned} \chi_k(\vec{r}) &= e^{i\vec{k}\vec{r}}, \\ E &= \frac{1}{2}|k|^2, \\ \vec{k}\vec{t} &= 2\pi m. \end{aligned} \tag{2.35}$$

With $\vec{k}$ being related to the energy by equations 2.35 the basis set size is defined by an energy cut-off. By using a very large unit cell, this basis set can also be used to describe non-periodic systems. This so called supercell approach requires a large basis set size to avoid self-interaction. This self-interaction stems from the periodicity of the system, with the system interacting with the atoms of the neighbouring unit cells. To reduce the k_{max} needed to describe the rapidly varying core electrons, this method is often used in combination with pseudopotentials (Section 2.4.6).

2.4.6. Pseudopotentials

For heavy elements with a large number of core electrons the number of basis functions to describe the electronic structure becomes inconveniently large. At the same time, core electrons do not participate in bonding and have only minimal influence on the valence structure. Therefore, the core electrons are often modelled by pseudopotentials.[4] This can be seen analogous to semi-empirical methods yielding good results at a fraction of the cost. The four major design steps for pseudopotentials are:

- Generate a suitable all-electron wave function for the atom using standard HF or relativistic Dirac-HF methods.
- Exchange the valence orbitals by a set of node-less pseudo-orbitals.
- Exchange the core electrons by a parametrized potential using expansions of analytical functions, such as Bessel or Gaussian functions.
- Obtain the suitable parameters for the potential by a fitting process such that the pseudo-orbitals solving the Schrödinger equation match the all-electron valence orbitals.

Projector Augmented Wave

The Projector Augmented Wave (PAW) methods also falls into the category of pseudopotentials, although it formally retains all the electrons. The PAW wave function can be written as a valence term expanded in a plane wave basis plus a contribution from the core radius of each nucleus. The core contribution expanded as the difference of the all electron density and the density from a node-less pseudo-atomic orbital density. This terms allows the wave function within the core region to adapt to different environments.

2.5. Force field methods for metal clusters

Molecular Dynamics methods based on force fields only use the atom positions and velocities as input information. The electrons are not considered as individual particles, which means that the bonding has to be included explicitly, rather than being calculated by solving the electronic Schrödinger equation. No effects of quantum nature are included in this type of calculations, which means the motion of the atoms is treated classically based on Newton‘s equations of motion,

$$F = m \cdot a = -\nabla\phi. \tag{2.36}$$

For time-independent problems, the calculation task reduces to finding energy minima on the potential energy surface (PES). Therefore, a potential for the system has to be constructed. The forces acting on the atoms are then calculated using formula 2.1. [4]

[4]See reference 4, Chapter 5, pages 222-225.

2.5.1. Pairwise potentials

Pairwise potentials describe the potential between two atoms. A simple example for a pair potential is the Morse potential:

$$E_{\text{Morse}}(r) = D(e^{-2\alpha(r-r_0)} - 2 \cdot e^{-\alpha(r-r_0)}), \quad (2.37)$$

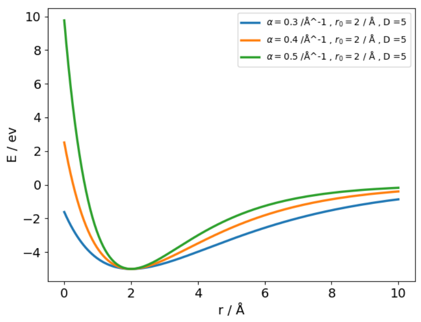


Figure 2.2.: Morse potential for different α.

with $\alpha = \sqrt{\dfrac{k}{2D}}$ and D as the dissociation energy.[5]
Another often used example for a pairwise potential is the Lennard-Jones potential,[6] which is computationally less expensive than potentials with an exponential dependence.

$$E_{\text{Lennard, Jones}} = 4\epsilon \left(\frac{r_0}{r^{12}} - \frac{r_0}{r^6} \right). \tag{2.38}$$

2.5.2. Many body potentials

For many body problems the modelled potential mostly consists of a pairwise interaction part and a three-body-interaction part. For a more precise description of the system higher order interactions can be considered and included in the potential. The appropriate order of truncation depends on the modelled material and the desired accuracy.[7]

$$E_{TOT} = \sum_{i,j}^{N} E(r_{ij}) + \sum_{i,j,k}^{N} E(r_{ijk}) + ... \tag{2.39}$$

2.5.3. Modified embedded atom model

The modified embedded atom model is a derivative of the embedded atom model. In this model, one atom is embedded in the charge density produced by its surrounding atoms via a properly chosen embedding function. The charge density itself is modelled by summing over two particle interactions. The total energy can be written as

$$E = \sum_i \left[F_i(\bar{\rho}_i) + \frac{1}{2} \sum_{i \neq j} \phi_{ij}(R_{ij}) \right], \tag{2.40}$$

with ϕ_{ij} being a pairwise interaction potential, ρ_i being the charge density and F_i the embedding function for atom i. For the modified embedded atom model ρ_i is augmented with additional angular dependencies.[8]

Sutton-Chen potential An often used potential for describing atoms of metallic systems is the Sutton-Chen potential, which is a variant of an embedded atom model. Here the embedding function F_i is chosen to be the $\sqrt{q_i}$, with $q_i = \sum_{j=1,i\neq j}^{N} \left(\frac{a}{r_{ij}}\right)^m$ and the pairwise interaction is a modified Coulomb Term[9]:

$$E_{\mathrm{TOT}} = \epsilon \left[\frac{1}{2} \sum_{i\neq j}^{N} \left(\frac{a}{r_{ij}}\right)^n - c \sum_{i=1}^{N} \sqrt{\sum_{j=1,i\neq j}^{N} \left(\frac{a}{r_{ij}}\right)^m} \right] \tag{2.41}$$

2.5.4. Second moment approximation models

Gupta potentials
The Gupta potential[10] is a second moment approximation to the tight binding model which was developed for transition metals. It uses a band model to estimate the attractive cohesive energy due to the d electrons. By neglecting the $d-d$ overlap and the transfer integrals for next nearest and higher neighbours and assuming exponentially varying transfer integrals one arrives at the expression

$$E_d = A \left(\sum_j \sqrt{e^{-2q(R_j - R_0)}} \right), \tag{2.42}$$

with $R_j - R_0$ being the displacement of atom j from its equilibrium position and q and A a fit parameter for the potential. The countervailing short-range repulsive force arises mainly from the s-electrons and is approximately of the form

$$E_s = B \sum_j e^{p(R_j - R_0)}. \tag{2.43}$$

By combining both terms and using $B = \frac{A}{\sqrt{N}} \frac{q}{p}$ one arrives at the complete Gupta-potential,

$$E_{\mathrm{TOT}} = A \left(\sum_j \sqrt{e^{-2q(R_j - R_0)}} \right) - \frac{A}{\sqrt{N}} \frac{q}{p} \sum_j e^{p(R_j - R_0)}. \tag{2.44}$$

SMATB with Born-Mayer type repulsion
The repulsive individual atomic contributions are composed of a Born-Mayer[11] type term, while the band energy of the atom is appoximated by the square root of the local density of states. With the fit parameters $A_{ij}, \zeta_{ij}, p_{ij}$ and q as well as the equilibrium distance $r_{0,ij}$ and the inter-atomic distance r_{ij} the resulting potential can be written as:

$$E_{\mathrm{TOT}} = \sum_i \left[\sum_j A_{ij} e^{-p_{ij}\left(\frac{r_{ij}}{r_{0,ij}} - 1\right)} - \sum_j \zeta_{ij} \sqrt{e^{-2q\left(\frac{r_{ij}}{r_{0,ij}} - 1\right)}} \right]. \tag{2.45}$$

2.6. Artificial Neuronal Networks

Artificial neuronal networks[12] have been an active field of research for the last decades, but it is only due to the increase of computational power and the development of new training methods that they are successfully applied to a variety of tasks, such as image and pattern recognition,[13] multi dimensional regression,[14] and Markov decision processes.[15] It can be shown that neural networks can be used as a universal appropriator for any finite dimensional function.[16]

2.6.1. General Introduction

A neural network is structured into layers. It typically consists of an input layer and several hidden layers, which are interconnected with each other, and the output layer (see Figure 2.3). Each of these layers is built up from nodes called artificial neurons. In fully connected networks each node is connected to every node in the previous layer and the following layer.[12]

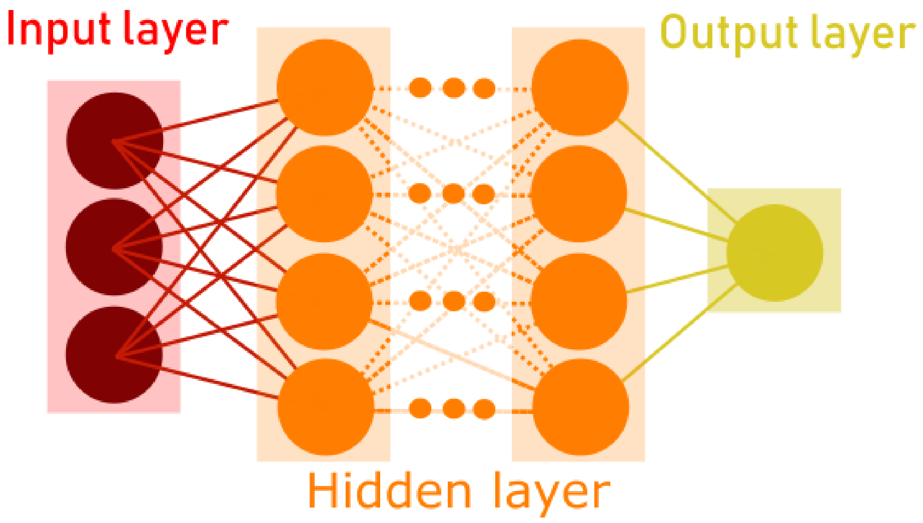


Figure 2.3.: Example for a neuronal network with N hidden layers

Usually each neuron is represented by a weight and a bias node plus a corresponding activation function. Every layer collects the activations of the previous layer as its input and propagates the resulting activation to the next layer until the output neuron is reached. For a network with one hidden layer and an arbitrary number of nodes the analytic expression can be written as:

$$y_i^j = f_i^j(b_i^j + \sum_{k=1}^{N_{j-1}} w_k^{j-1} \cdot y_k^{j-1}), \tag{2.46}$$

with y_i^j as the activation of the i^{th} node for the j^{th} layer, $w_k^{j-1}$the weight of the k^{th} node for the $j-1^{th}$ layer, b_i^j the bias of the i^{th} node for the j^{th} layer and f_i^j the activation function for the i^{th} node for the j^{th} layer.
Since the network has an analytic form its derivative can also be determined analytically.

2.6.2. Activation functions

Activation functions connect the nodes between the different layers of the network. Their goal is to model the activation of a biological neuron. The earliest examples for activation functions are sigmoid and hyperbolic tangent functions, which are manly used for wide-networks, consisting mostly of only one hidden layer with a large number of nodes.[17]

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.47}$$

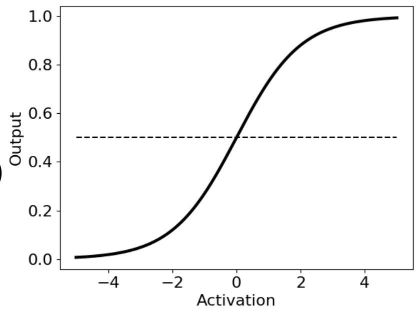


Figure 2.4.: Sigmoid activation function.

$$f(x) = \tanh(x) \tag{2.48}$$

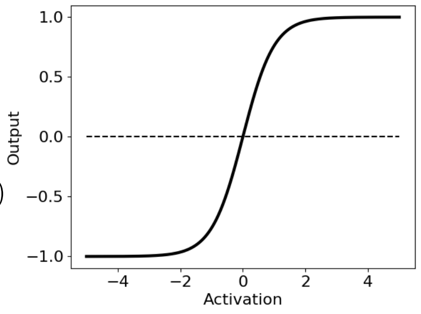


Figure 2.5.: Hyperbolic tangent activation function.

For more complex problems another activation function is found to be more efficient with respect to learning speed. The rectified linear unit (RELU) function and other functions derived from RELU are nowadays the most commonly used activation functions.[18]

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{2.49}$$

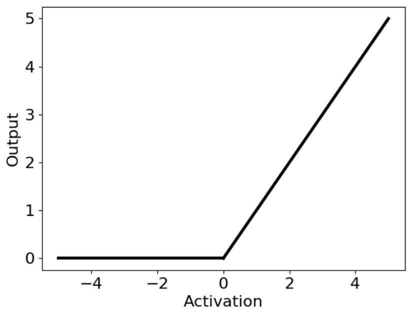


Figure 2.6.: RELU activation function.

The best performing derivative of the RELU function is the exponential linear unit (ELU) activation function which has exponential behaviour for activations smaller or equal to zero and a linear behaviour otherwise.[19]

$$f(x) = \begin{cases} e^x - 1 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{2.50}$$

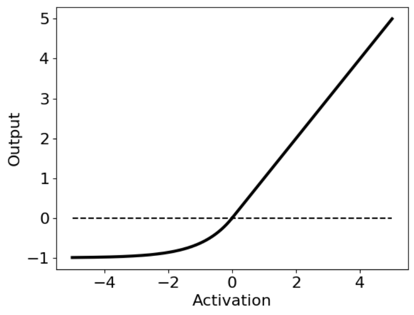


Figure 2.7.: ELU activation function.

The activation function of the last layer depends on the type of error function used for the training. For a regression task using a quadratic error function, or another function which has a linear derivative, the standard choice would be a linear activation function for the output layer.

2.6.3. Error function

The error function is minimized during the training. The type of error functions used during the training depends on the task specifics. For a regression task normally the quadratic error function is used. With $t^{(i)}$ being the target value and $y_w(x^{(i)})$ the prediction value of the network as a function of the input $x^{(i)}$ of the i^{th} sample in the batch. The total error E_{Q} is then calculated as the sum over all errors within the batch:

$$E_{\text{Q}} = \sum_{i=1}^{m}(t^{(i)} - y_w(x^{(i)})^2. \tag{2.51}$$

Due to the fact that, when fitting Gaussian distributed data, the maximum-likelihood fit, containing the weights w_{ML}, minimizes the quadratic error, the quadratic error function is used for regression tasks.[20]

$$\begin{aligned} p(t^{(i)}|x^{(i)}, w, \sigma^2) &= \prod_{i=1}^{n} \sqrt{(2\pi\sigma^2)} \cdot e^{-\frac{1}{2\sigma^2}\left(t^{(i)} - y_w(x^{(i)})\right)^2}, \\ w_{\text{ML}} &= \arg\max_w \sum_{i=1}^{m} \log(p(t^{(i)}|x^{(i)}, w, \sigma^2)), \\ \sum_{i=1}^{m} \log(p(t^{(i)}|x^{(i)}, w, \sigma^2)) &= -m \cdot \log(\sigma) - \frac{m}{2} log(2\pi) - \sum_{i=1}^{m} \frac{(t^{(i)} - y_w(x^{(i)}))^2}{2\sigma^2}, \\ w_{\text{ML}} &= \arg\min_w \sum_{i=1}^{m} (t^{(i)} - y_w(x^{(i)}))^2. \end{aligned} \tag{2.52}$$

However, the quadratic behaviour can cause instabilities during the training process due to outliers causing large errors and thus resulting in large changes for the weights.

2.6.4. Regularization

Regularization is a method to prevent the NN from over-fitting the data.[21] The main idea behind regularization is to include the value of the weights in the error function, so during the optimization process not only the error, but also the weights get minimized, leading to a better result for the optimization.

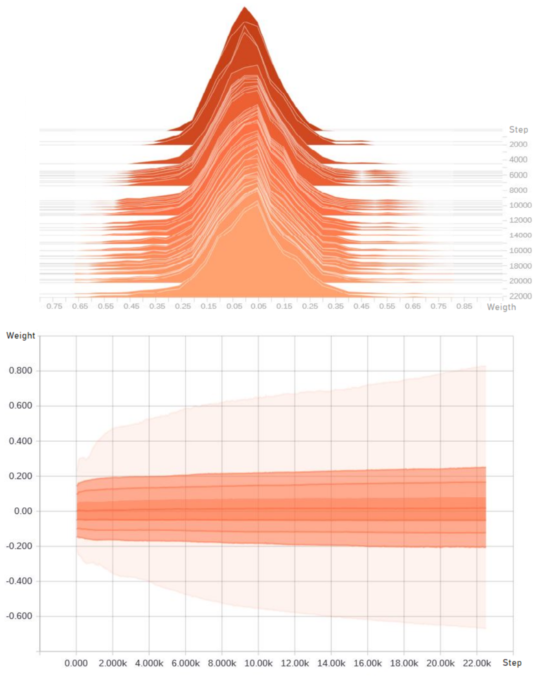


Figure 2.8.: Histogram of the evolution of the unregularized weights.

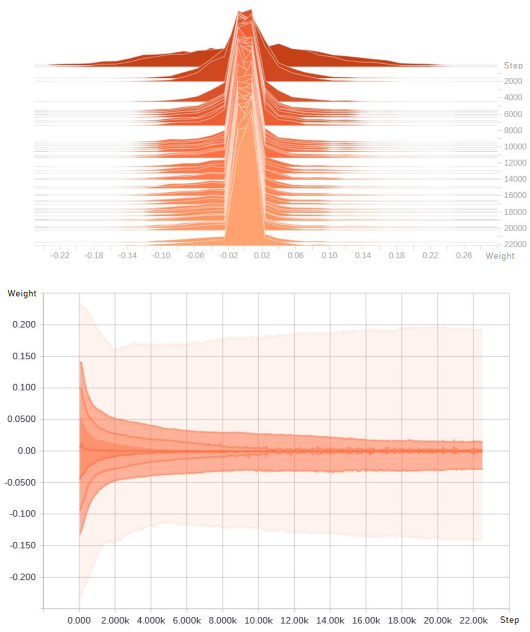


Figure 2.9.: Histogram of the evolution of the regularized weights.

This is due to the fact that only weights contributing a relevant information to the overall model survive the optimization, while the other weights minimize their absolute value (see Figures 2.8 and 2.9). By using this technique one gets a set of characteristic weights representing the main building blocks of the model, which are often referred to as features. These features can be seen analogous to eigenvector solutions to the trained task.
For the PES of any molecular system, these features can be visualized by choosing 2 degrees of freedom of the system and evaluating the neuron outputs for the specific layer. For simplicity, the chosen sample system will be a triatomic system with interatomic distances $r = r_1 = r_2$ and an angle ϕ as coordinates, which enforces at least $C_{2\text{V}}$ symmetry, see Figure 2.10.

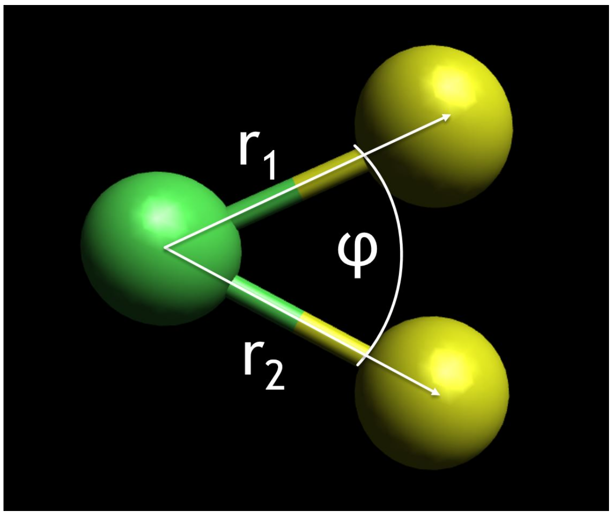


Figure 2.10.: System used to visualize the PES

This way one can visualize the features of a certain layer by removing the layers following and analysing the output of said layer. The results can be see in Figures 2.11-2.12:

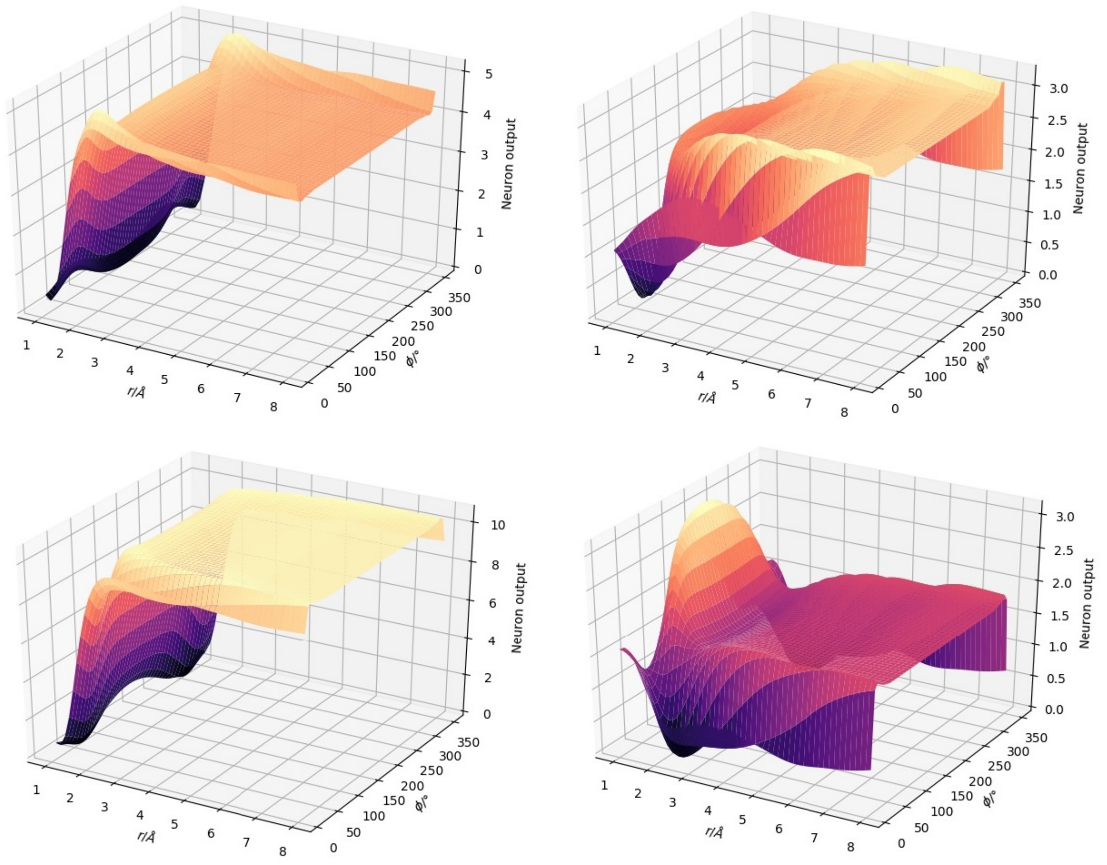


Figure 2.11.: Key features of the second hidden layer.

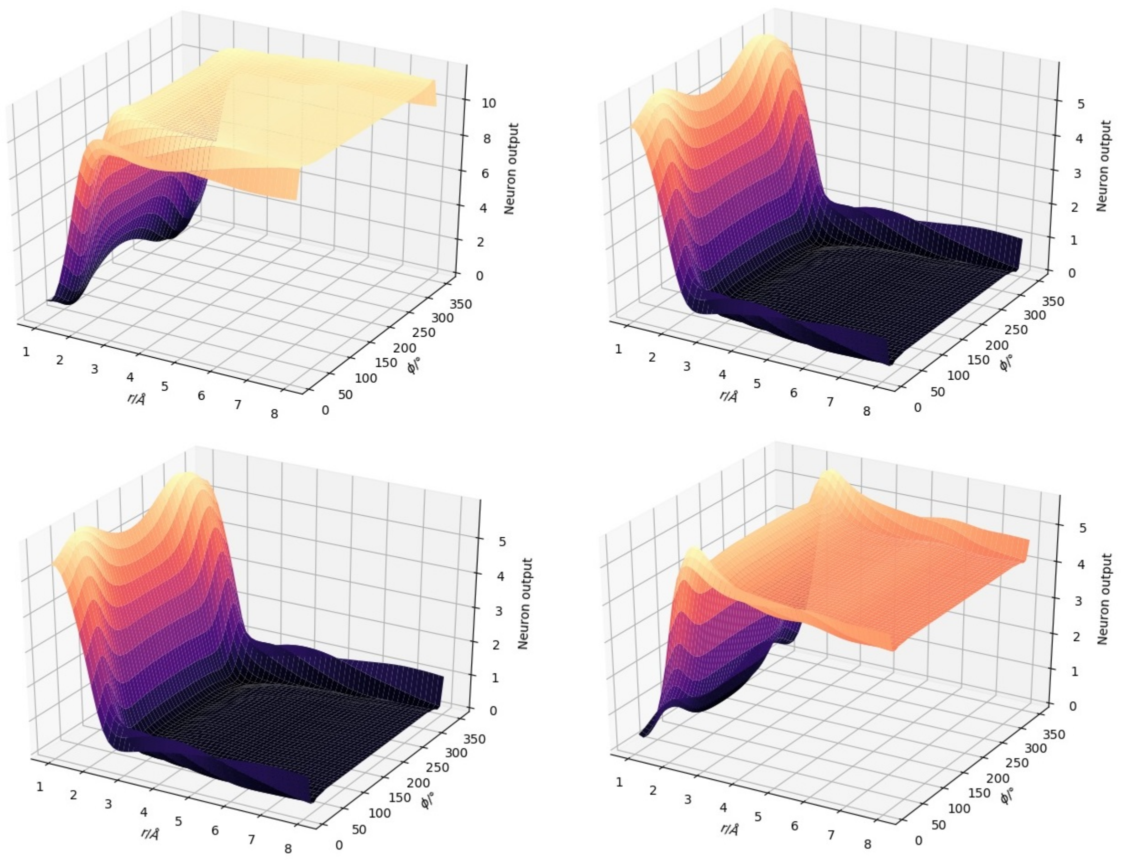


Figure 2.12.: Key features of the third hidden layer.

From Figures 2.11 and 2.12 one can see that, as as you go deeper in the network, the features get more complex and begin to look more similar to the final PES. In the third layer the overall shape of the PES is more or less fixed. Therefore, the key features are negatives of each other containing only slight variations.
The total error E_{tot} is obtained by adding the regularization to the quadratic error E_{Q} (see equation 2.51). The most common types are the L1 regularization,

$$E_{\text{tot}} = E_{\text{Q}} + \lambda \sum_{i}^{k} |w_i|, \tag{2.53}$$

and the L2 regularization,

$$E_{\text{tot}} = E_{\text{Q}} + \lambda \sum_{i}^{k} (w_i)^2, \tag{2.54}$$

with λ being a small parameter weighting the regularization error.

2.6.5. Training

A training cycle consists of a forward and a backward step. During the forward step the inputs are presented to the network and propagated through each layer to produce a prediction value. In the backward step, the prediction produced by the network and the corresponding target value are inserted into the defined error function. The optimizer then changes the weights of each node corresponding to its error contribution:

$$\Delta wij^k = -\frac{dE_{\text{tot}}}{dw_{ij}^k}, \tag{2.55}$$

with wij^k being the weight matrix in layer k and E_{tot} the total error for the forward step.

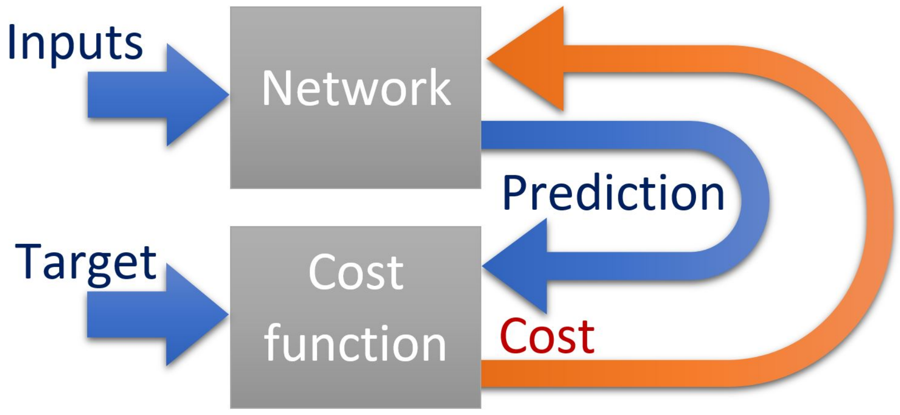


Figure 2.13.: Forward (blue) and backward (orange) step visualization.

Data preparation

To avoid getting trapped in local minima the training data is split into batches. These batches are all fed to the network and their corresponding errors are accumulated before the back-propagation step is performed.
For validation purposes, the total data contains training and validation data. The validation data is never used during the training and allows to evaluate the performance of the network concerning over-fitting and convergence. As shown in Figure 2.14, the optimal training point of the network is defined by the minimum of the validation error. After that, the training error and the validation error diverge and the network is over-fitting the training data.

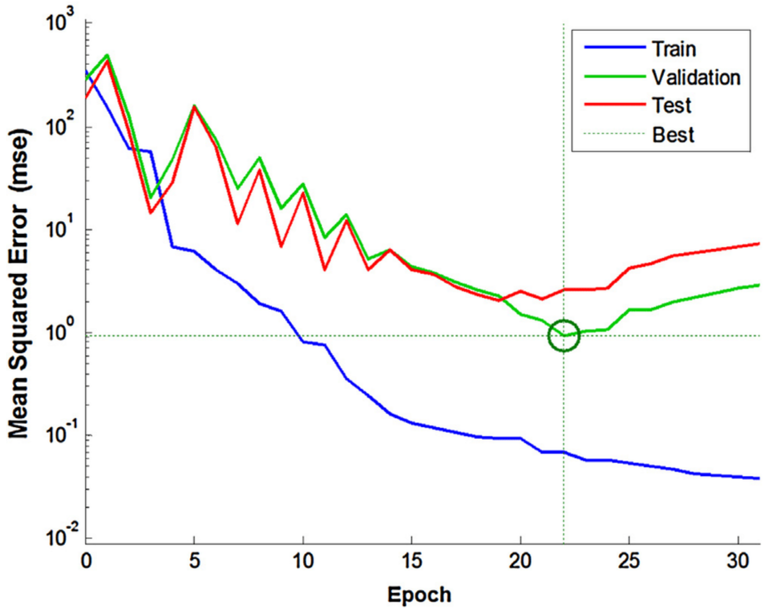


Figure 2.14.: Example error plot indicating the optimal stopping point.[2]

Optimizers

The optimizer computes the gradients for the error and applies them to the variables according to its type.

Gradient Descent Optimizer

The most basic optimizer is the gradient descent optimizer. It is a first order iterative optimizer which modifies the weights proportional to the negative gradient. The step width of the optimizer is defined by the so called learning rate η, which is a multiplicative factor in the weight change Δw_{ij}^k:

$$\Delta w_{ij}^k = -\eta \cdot \frac{\partial E_{\text{tot}}}{\partial w_{ij}^k}. \tag{2.56}$$

Adam Optimizer

The Adam algorithm[22] is a first-order gradient-based optimizer based on adaptive estimates of lower-order moments. It is a very efficient and stable algorithm, which outperformed the standard optimizers such as gradient descent or AdaGrad on a variety of tasks. The algorithm contains four predefined parameters: η the learning rate, β_1, the update parameter for the first moment m_t, β_2, the update parameter for the second moment v_t, and ϵ which prevents the update from being too sensitive to small gradients.

$$
\begin{aligned}
\Delta wij^k &= -\eta \cdot \frac{\hat{m_t}}{\sqrt{\hat{v_t}+\epsilon}}, \\
\hat{m_t} &= \frac{m_t}{1-\beta_1^t}, \\
\hat{v_t} &= \frac{v_t}{1-\beta_2^t}, \\
m_t &= \beta_1 m_{t-1} + (1-\beta_1)\frac{df}{dw_{ij}^k}, \\
v_t &= \beta_2 m_{t-1} + (1-\beta_2)\left(\frac{df}{dw_{ij}^k}\right)^2.
\end{aligned}
\tag{2.57}
$$

Levenberg-Marquardt
The Levenberg-Marquardt[23],[24] algorithm is a second-order optimizer and has an improved robustness compared to the Gauss-Newton algorithm.

$$
\Delta wij^k = (H + \lambda \cdot \mathrm{diag}[H])^{-1}\frac{\partial E_{\mathrm{tot}}}{dw_{ij}^k}, \tag{2.58}
$$

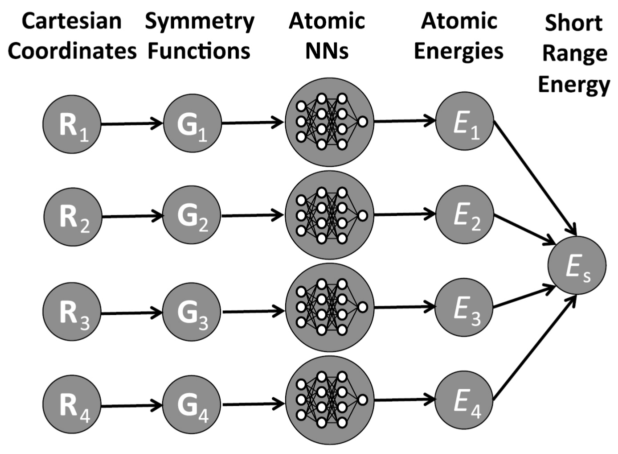
with H denoting the Hessian matrix and λ as a weighting parameter.

2.6.6. Behler-Type Networks

To create networks that are able to efficiently represent a molecular system they need to fulfil certain criteria:

- The number of input nodes has to stay constant and independent of system size or composition.
- The inputs need to represent a unique description of the molecular system.

In the model proposed by Behler and Parinello these requirements are met by de-constructing the total energy as a sum of atomic energies. Each atomic energy is produced by an **A**tomic **N**eural **N**et (**ANN**) represented by a feed forward network.



$$E_{\text{tot}} = \sum_{i}^{N} E_i \tag{2.59}$$

Figure 2.15.: Atomic neural networks as proposed by Behler and Parinello.[1]

The surroundings of each atom are mapped via symmetry adapted functions and are then fed to the network. Each atom type has a separate set of weights. This is necessary due to the fact that the net knows the atom position and types of its neighbours, but has no information about the atom it is representing.

2.7. Symmetry functions

Since Cartesian coordinates contain redundancies with respect to the rotational and the translational degrees of freedom of the total symmetry, they are a bad choice for the description of a molecular potential energy surface. A better choice could be internal coordinates. However, due to the fact that the internal coordinates can only describe one particular system, the NN would need to be trained on the target system, which would restrict its application to small systems only.
A more suitable choice are symmetry functions that transform the Cartesian coordinates into inputs for the NN fulfilling the following criteria:[25],[1]

- Rotation and translation invariance
- Invariance with respect to permutations within the same atomic species
- Unique description of the molecular system
- Constant number of inputs for the ANN

As a biological equivalent one could imagine the receptors of the eye, which serve as an input for the brain and are responsible for specific frequency ranges. Only by the combination of all three types of cones the human brain can distinguish between all different colors of the visible spectrum.

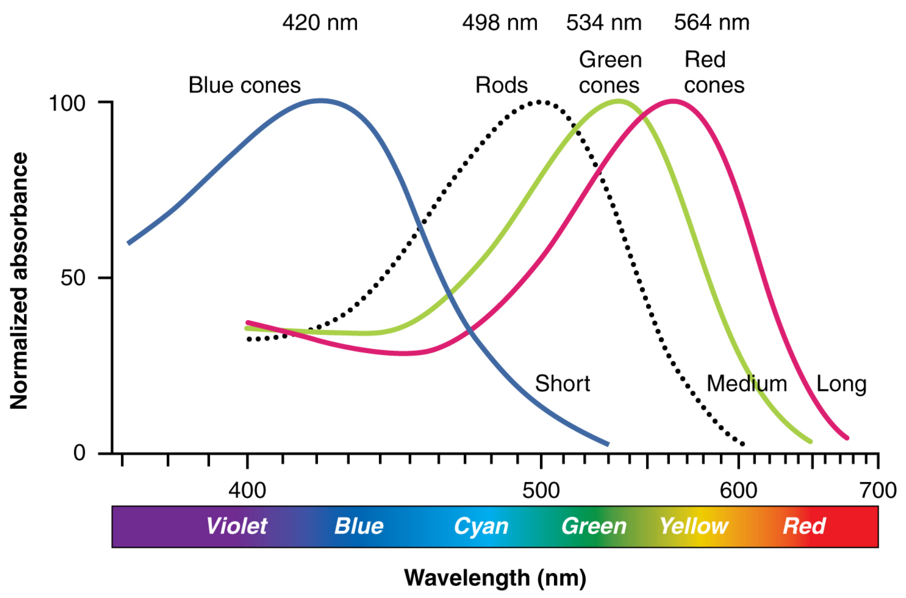


Figure 2.16.: Absorbance of the receptors of the human eye as a function of wavelength.[3]

2.7.1. Behler-Parinello symmetry functions

Behler and Parinello proposed symmetry functions which are separated into a radial (two-body) and an angular (three-body) interaction part.[1] To ensure that the potential goes to zero at the cut-off distance each symmetry function is multiplied with a radial cut-off function.

$$G^1_{ij} = f_c(R_{ij}). \tag{2.60}$$

One possible cut-off function is a cosine function which has zero value and slope at the cut-off radius.

$$f_{c,\cos}(R_{ij}) = \begin{cases} 0.5 \cdot \left[\cos\left(\dfrac{\pi R_i j}{R_c}\right) + 1\right] & \text{if } R_{ij} \leq R_c \\ 0 & \text{if } R_{ij} > R_c \end{cases} \tag{2.61}$$

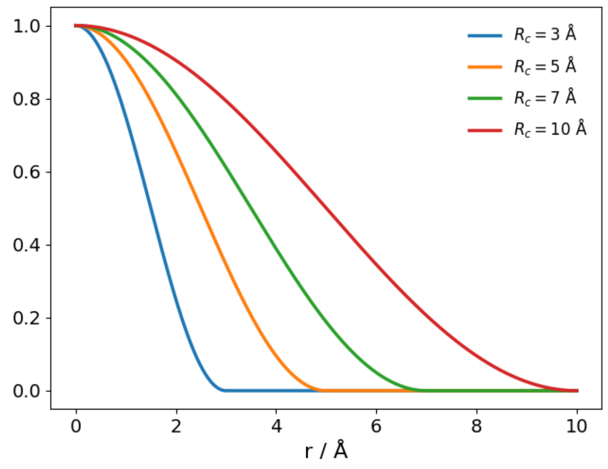


Figure 2.17.: The cosine cut-off function G^1 as a function of the inter-atomic distance r in Å for different cutoff radii R_c.

For the hyperbolic tangent, not only the first but also the second derivative goes to zero at the cut-off radius.

$$f_{c,\tanh}(R_{ij}) = \begin{cases} \tanh^3 \left[1 - \frac{R_ij}{R_c}\right] & \text{if } R_{ij} \leq R_c \\ 0 & \text{if } R_{ij} > R_c \end{cases} \tag{2.62}$$

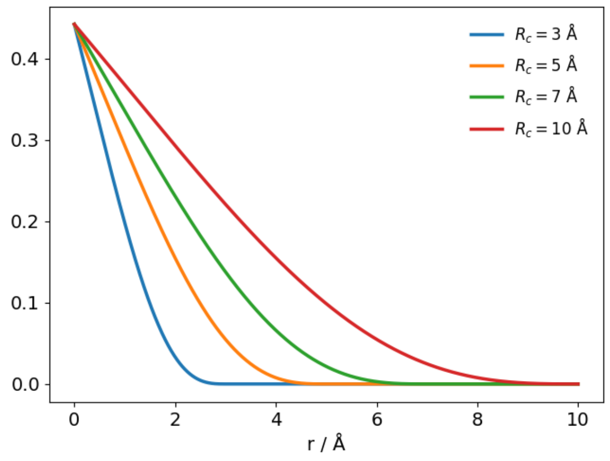


Figure 2.18.: Examples for the tanh cut-off function G^1.

Radial dependencies are captured by Gaussians with varying slopes and center positions:

$$G^2_{ij} = f_c(R_{ij}) \cdot e^{-\eta(R_{ij}-R_s)^2}. \tag{2.63}$$

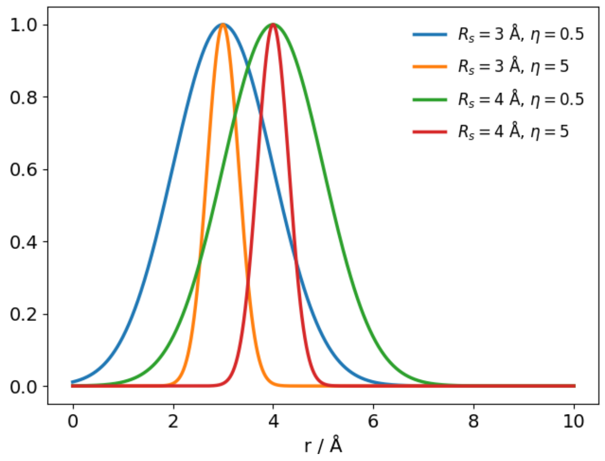


Figure 2.19.: G_2 functions as a function of the inter-atomic distance r in Å.

To get a smoother decay of the cut-off, the G^3_{ij} function can be used:

$$G^3_{ij} = f_c(R_{ij}) \cdot \left[\sum_k f_c(R_{ik}) + \sum_k f_c(R_{jk})\right]. \tag{2.64}$$

To explicitly include the rejective radial behaviour for three body interactions the G^4_{ij} functions can be included as well:

$$G^4_{ij} = f_c(R_{ij}) \cdot e^{-\eta R^2_{ij}} \left[\sum_k f_c(R_{ik}) \cdot e^{-\eta R^2_{ik}} + \sum_k f_c(R_{jk}) \cdot e^{-\eta R^2_{jk}}\right]. \tag{2.65}$$

To represent the angular dependencies of the system a modified cosine function can be used. The G^5_{ij} function also includes a radial dependency to emphasise the angular dependence of inter-atomic interactions at short range:

$$\begin{aligned} G^5_{ij} &= f_c(R_{ij}) \cdot e^{-\eta R^2_{ij}} \cdot 2^{1-\zeta} \cdot \\ &\sum_{\Theta_{kij}} \left[(1+\lambda \cdot \cos\Theta_{kij})^\zeta \cdot e^{-\eta(R^2_{ik}+R^2_{jk})} \cdot f_c(R_{ik}) \cdot f_c(R_{jk})\right]. \end{aligned} \tag{2.66}$$

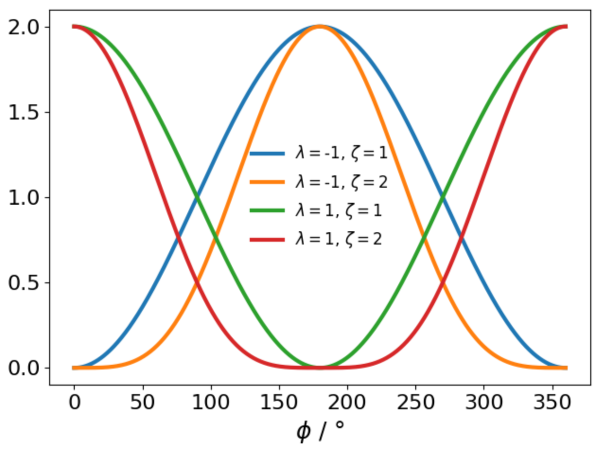


Figure 2.20.: G_5 functions as a function of the inter-atomic angle ϕ.

3. Modelling of Metallic Nano Clusters

Since the network structure of Behler (see Section 2.6.6) is different from most other NN applications, many high level machine learning tools are not directly applicable to this approach. Therefore, a program package needed to be built from low level machine learning packages. The most versatile open source packages are offered for the Python programming language, which is used in combination with C-library extensions for time-critical calculations.

3.1. Python Framework

The code manly utilizes the program packages NumPy[26] and TensorFlow[27] for the construction, training and evaluation of the networks. To enable the generic generation of customized symmetry functions the SymPy[28] module is used. For optimization purposes the scipy.optimize[29] library is utilized to minimize implementation efforts. In the molecular dynamics simulation implementation large parts of the PyParticles[30] project code could be recycled and has been adapted to fit our needs. For the extraction of the training data from the files the Regular Expressions module re is utilized and the plots are done using the Matplotlib[31] module.

3.2. Implementation

The module is structured into the main module and 6 different sub-modules containing utilities for:

- Training
- Descriptors
- Data generation
- Network types
- Optimization
- MD-Simulation

The properties and methods of the sub-modules will be displayed in an UML diagram. In an UML diagram each class is represented by a box with the first section being the class name, the second section being the class properties and the third section displaying all methods implemented within this class.

3.2.1. Main

The main module consists of utilities for loading, training, saving and evaluating the networks. It accesses the sub-modules in `types` to build the networks and implements all settings used during training. It initializes the variable environment, specifies the optimizing procedure and loads the specified symmetry functions from the `descriptors` module.

3.2.2. Descriptors

The `descriptors` module is wrapping a C-library which represents the symmetry functions used for the network. New descriptors can be added to the library by modifying the corresponding text file `customSymFuns.txt` and recompiling the library via the setup.py file. This module uses the SymPy library to implement the generic generation of derivatives for any given symmetry function. The descriptors library can implement any type of two and three body interaction with a similar structure as the Behler-Parinello symmetry functions. This allows the assessment of different types of functions with respect to their ability to map the chemical environment onto neural networks.

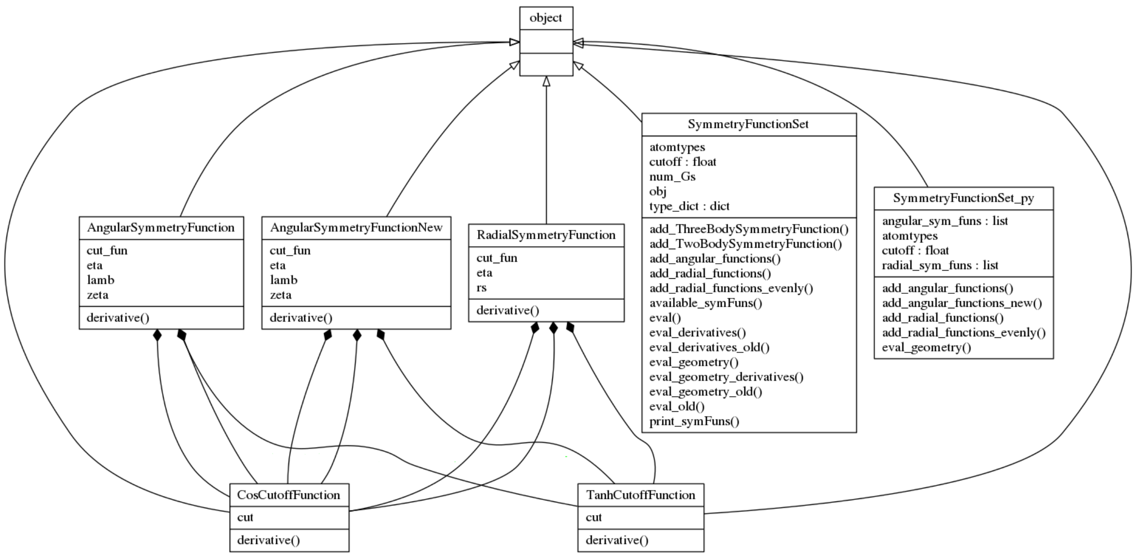


Figure 3.1.: Class structure of the `descriptors` sub-module.

3.2.3. Data generation

The purpose of this sub-module is to parse the data from the *ab initio* or force-field calculation output files into a suitable format. Currently, it is able to read Quantum-Espresso and LAMMPS output files.

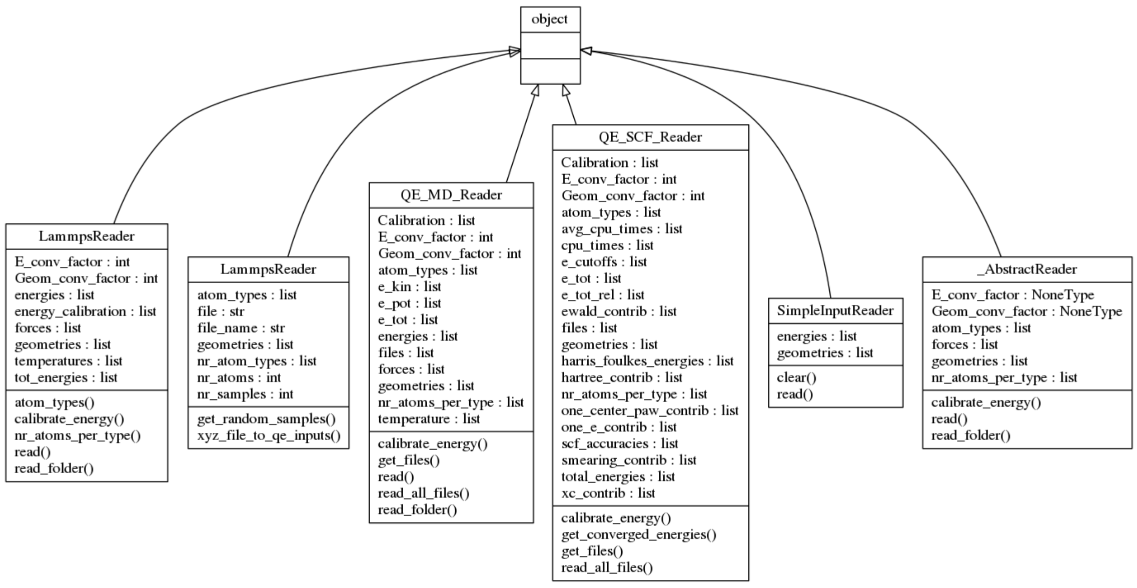


Figure 3.2.: Class structure of the `data_generation` sub-module.

3.2.4. Types

The `types` sub-module contains the utilities for the construction of the molecular neural networks and its force and energy tensors. It implements separate functions for serial and parallel construction of the networks. Due to the special structure of these networks (see Section 3.3) the training can only be done in a serial fashion. The evaluation, on the other hand, can also be done with a parallel structure.

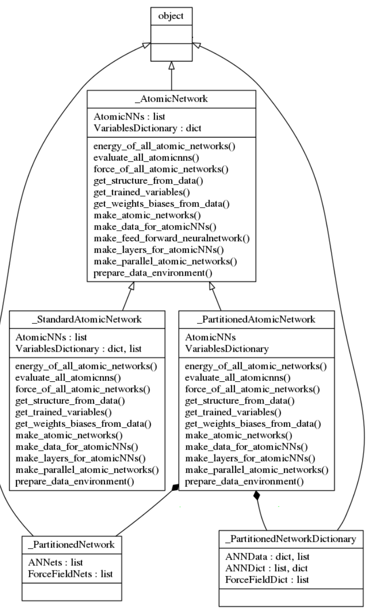

object
_AtomicNetwork
AtomicNNs : list
VariablesDictionary : dict
energy_of_all_atomic_networks()
evaluate_all_atomicnns()
force_of_all_atomic_networks()
get_structure_from_data()
get_trained_variables()
get_weights_biases_from_data()
make_atomic_networks()
make_data_for_atomicNNs()
make_feed_forward_neuralnetwork()
make_layers_for_atomicNNs()
make_parallel_atomic_networks()
prepare_data_environment()
_StandardAtomicNetwork
AtomicNNs : list
VariablesDictionary : dict, list
energy_of_all_atomic_networks()
evaluate_all_atomicnns()
force_of_all_atomic_networks()
get_structure_from_data()
get_trained_variables()
get_weights_biases_from_data()
make_atomic_networks()
make_data_for_atomicNNs()
make_layers_for_atomicNNs()
make_parallel_atomic_networks()
prepare_data_environment()
_PartitionedAtomicNetwork
AtomicNNs
VariablesDictionary
energy_of_all_atomic_networks()
evaluate_all_atomicnns()
force_of_all_atomic_networks()
get_structure_from_data()
get_trained_variables()
get_weights_biases_from_data()
make_atomic_networks()
make_data_for_atomicNNs()
make_layers_for_atomicNNs()
make_parallel_atomic_networks()
prepare_data_environment()
_PartitionedNetwork
ANNets : list
ForceFieldNets : list
_PartitionedNetworkDictionary
ANNData : dict, list
ANNDict : list, dict
ForceFieldDict : list

Figure 3.3.: Class structure of the `types` sub-module.

3.2.5. Optimization

The `optimize` sub-module is a wrapper for the scipy.optimize library, providing access to all optimization algorithms implemented in scipy.optimize.

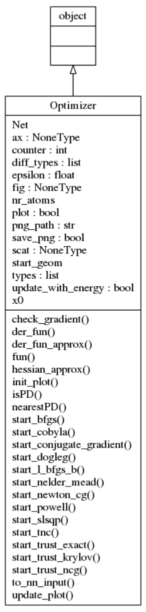


Figure 3.4.: Class structure of the `optimize` sub-module.

3.2.6. MD-Simulation

The md_utils sub-module is a modified and reduced version of PyParticles. The NNForce class wraps the force tensor evaluation from the main module to use it within the PyParticles particle set. Additionally this module implements a Berendsen[32] and a Langevin thermostat,[33],[34] as well as a logger for the xyz-format.

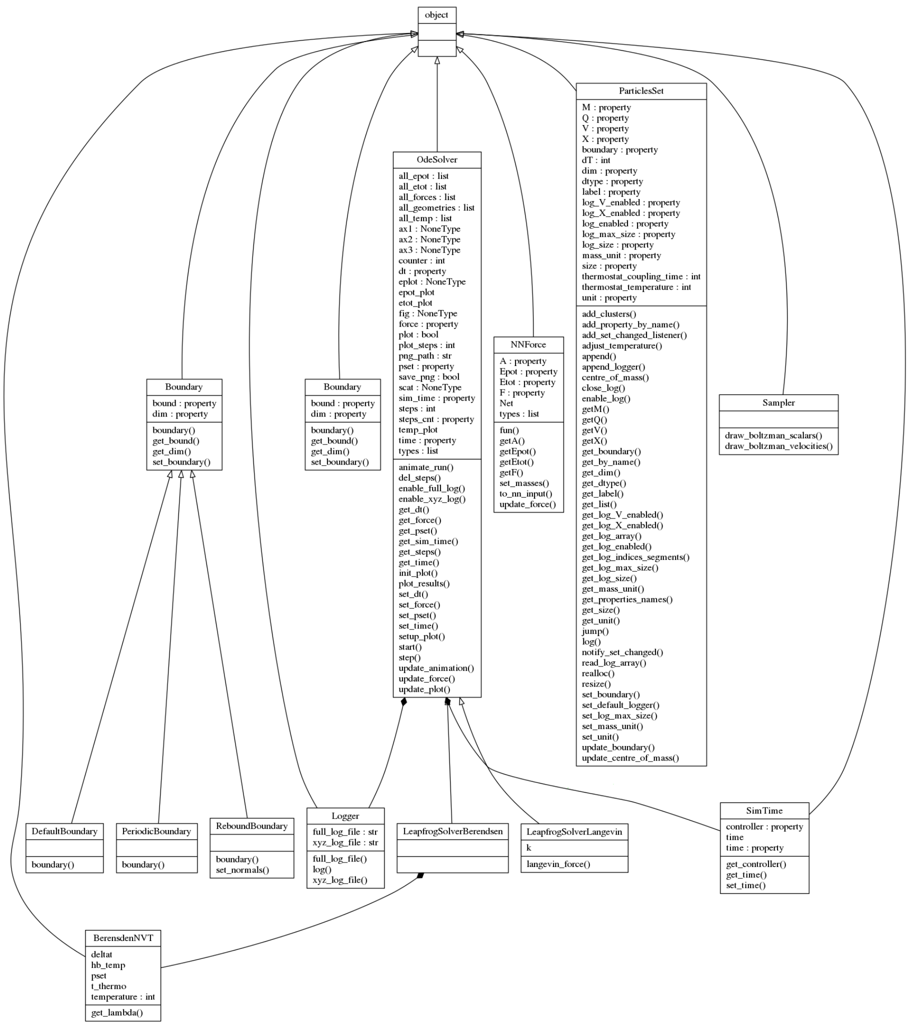


Figure 3.5.: Class structure of the md_utils sub-module.

3.3. NN Structure

For the training of a Behler-Parinello type network one set of variables is necessary for each atom type. Therefore, in the case of multiple atoms per type, the variable set needs to be evaluated and trained for each atom of the corresponding type. This corresponds to the situation of multiple geometries being presented to the same network.

3.3.1. Single composition system

For the situation of only one system size and composition the Tensorflow graph consists of a single energy prediction tensor and the corresponding error function. Figure 3.6 shows the structure of such a network for the Ni_2Au_2 system. The dark blue boxes represent the inputs, the green boxes the activation functions, the light blue boxes the atomistic energy predictions and the orange boxes the variables which are shared among the atomic neural networks of one type. The grey and dark blue boxes at the top represent the energy prediction and the target energy for the whole system, which then serve as an input for the error function.

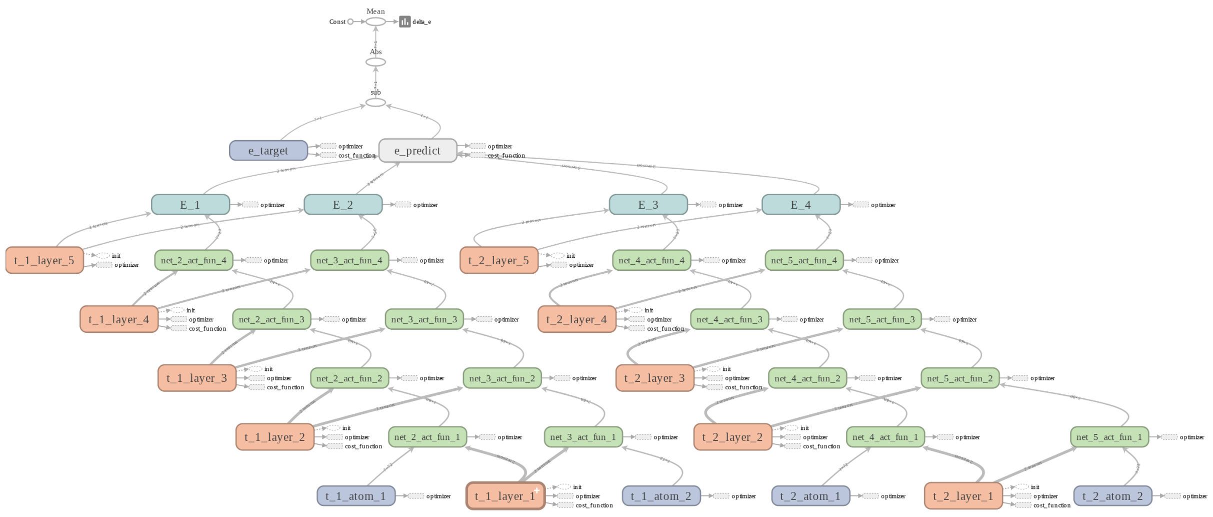


Figure 3.6.: The network structure for a Ni_2Au_2 system visualized in TensorBoard. The arrows are indicating how the tensors are combined during a forward step.

3.3.2. Multiple system sizes

Figure 3.7 shows the network structure for multiple system sizes and compositions. Each variant is represented by a separate energy prediction tensor and error function using the same structure as for the single composition network. For training, all these error functions are then summed up and fed to the optimizer. The trainable variables, represented as orange boxes in Figure 3.6, are shared between all variants so each variant relies on the same set of variables.

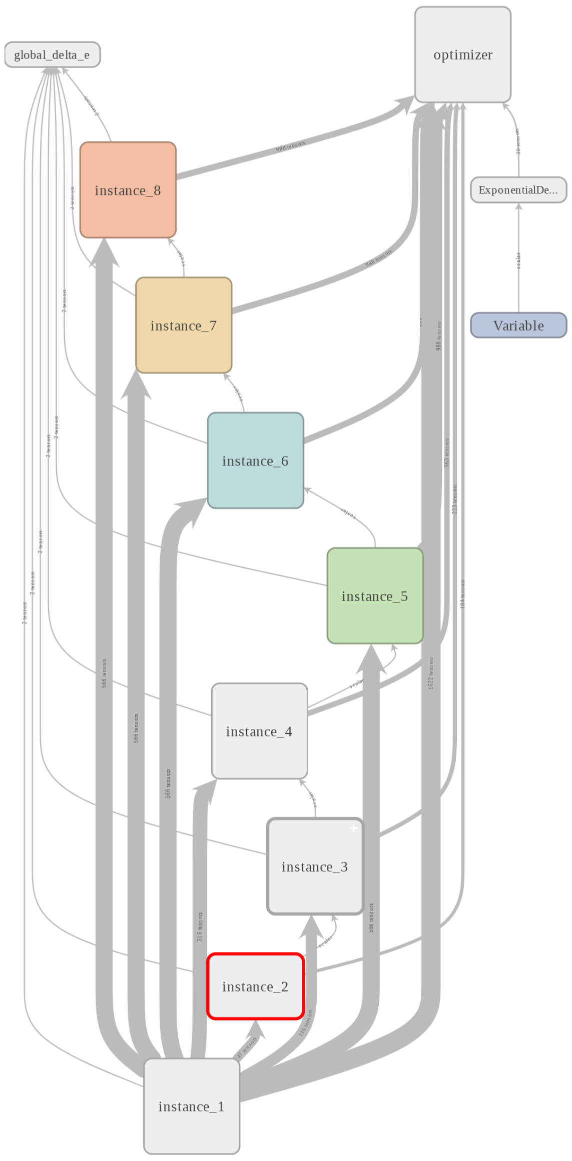


Figure 3.7.: Network structure for eight different system composition visualized in TensorBoard.

3.3.3. Data generation

For the generation of training data it is crucial to have a good coverage of the whole phase space of a system. It is also vital to include different system sizes and compositions into the training set. This can be achieved either via MD-simulations at high temperature, by sampling techniques such as Meta-Dynamics simulations, or by a combination of both techniques.

MD-simulations
The MD-simulations should be carried out well above the melting point of the material of interest. This way, the potential will also be sampled in regions far off the equilibrium structure.

Meta-Dynamics
Meta-Dynamics, as implemented in the program package PLUMED,[35] offers phase space sampling with less redundancy. It is also based on an MD-simulation, but constantly modifies the potential by giving a penalty for previously visited regions of the phase space. Therefore, the MD-run is less likely to revisit the penalized regions and drives the system into phase space regions a standard simulation would only reach at very high temperatures. This is an effective method for a systematic sampling of a system. However, for larger system the implementation of a Meta-Dynamics simulation becomes more and more challenging and time consuming, and only for small systems this method will offer a time benefit over MD-simulations.

4. The Ni-Au system

4.1. Ni-Au Metal Clusters

Due to their unique physical, optical and chemical properties bimetallic nano clusters are an interesting new class of materials for a variety of applications.[36] Ni is a ferromagnetic material, which could be used in medical applications such as drug delivery, DNA separation or magnetic resonance imaging enhancement.[37],[38] However, pure particles of Ni oxidise very easily due to their large surface to volume ratio, which reduces their magnetic moment. Therefore, bimetallic core-shell structures using Au as a coating for the Ni core, could offer a solution to prevent this oxidation. Also, for applications as a catalyst, mixed metallic nanoparticles offer interesting possibilities such as tunable reactivities,[39] bifunctional activity[40] and additional stabilization of sensitive catalysts via a co-metal partner.

4.2. NN for Ni-Au-Cluster

Since the simulation of Ni-Au clusters with *ab initio* methods is a time extensive and often difficult task, the use of NN could widen the range of possible computer experiments. Especially for applications in catalysis a faster method could enable simulations of surface reactions on a large scale. However, a first test of our NN approach will be the correct description of structural features of the pristine clusters.

4.3. Training data

The most time consuming and sensitive task when training a NN potential is the generation of training data. A choice of representative model systems is crucial. For our training dataset we choose $\mathrm{Ni}_x\mathrm{Au}_{13-x}$ and the $\mathrm{Ni}_x\mathrm{Au}_{55-x}$ systems. All calculations for the generation of the training data are carried out with Quantum Espresso,[41] using a projector augmented wave ansatz (see Section 2.4.6) with the PBE[42] functional for the exchange/correlation energy. The exact parametrisation can be looked up in the Appendix.

4.4. Investigations

4.4.1. Structure

The feed forward network was designed with the basic concept of a pyramid-shaped structure for the hidden layers.[43] For the standard network, the input and the hidden layers are connected via ELU activation functions and the output layer via a linear activation.

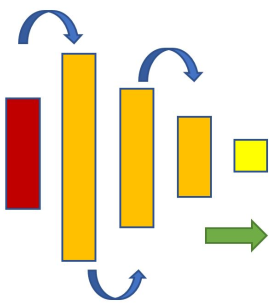


Figure 4.1.: Pyramid-shaped network structure.

The network size is determined empirically. A coarse hyper-parameter search was conducted to find the optimal structure.

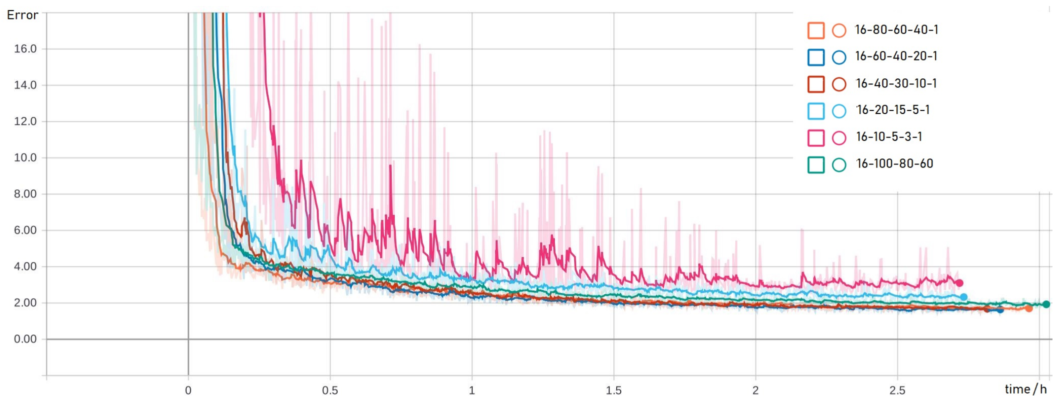


Figure 4.2.: Error as a function of time for different network structures.

While the **16-60-40-20-1** structure reaches the lowest error of all structures, it has a comparable slow convergence behaviour. The **16-80-60-40-1** structure shows the fastest and best overall error convergence. Therefore, all further investigations were carried out using this network structure.

4.4.2. Symmetry functions

For a better comparability with the **M**odified **E**mbedden **A**tom **M**odel (**MEAM**) and the **S**econd **M**oment **A**pproximation to **T**ight **B**inding model (**SMATB**) only radial symmetry functions are considered for the training. In this case all Gaussian functions are centered at $R_s = 0$ Å and the slope η is varied as specified in Table 4.1. The cutoff-radius R_c was set to be at 7 Å and the cutoff function was multiplied onto the Gaussian function as visualized in Figure 4.3.

Table 4.1.: Parameters of the used radial symmetry functions.

Nr.	R_s /Å	R_c /Å	η /Å^{-2}
1	0.0	7.0	1.428
2	0.0	7.0	0.714
3	0.0	7.0	0.357
4	0.0	7.0	0.214
5	0.0	7.0	0.124
6	0.0	7.0	0.071
7	0.0	7.0	0.036
8	0.0	7.0	0.003

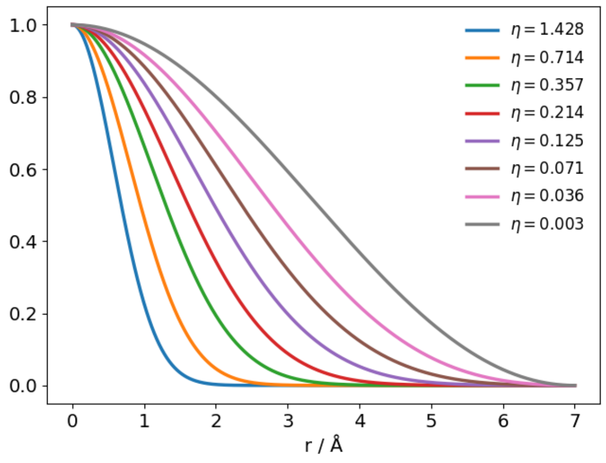


Figure 4.3.: Radial dependence of the used symmetry functions.

4.4.3. Optimizer

For the optimization, the first order ADAM optimizer (see. 2.6.5) was selected using the following standard parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. This is due to the possibility to perform batch training, which is not an option in other second-order optimizers such as Levenberg-Marquardt or Kalman-Filter.[44] Without the use of batch-learning, the optimizations are more likely to end up in local minima.[45] Second-order methods also have the downside of limited network sizes, because of the need for a matrix inversion during the optimization step.

4.4.4. Learning rate

It is also advisable to let the learning rate decay as the training progresses. For this task an exponential decay was chosen,

$$lr_{t+1} = lr_t \cdot k^{\frac{t}{t_0}}, \tag{4.1}$$

with k being the decay rate and t_0 as the decay step. The parameters chosen for the training are $k = 0.96$, $t_0 = 500$.

4.4.5. Error function

In general training data will be noisy, resulting in a Gaussian distribution for the data around the exact solution μ with a standard deviation of σ. For our modelling we assume that the *ab initio* energies represent the exact solution μ for our system, without having any uncertainty for the data. Therefore, $\sigma \to 0$ and the maximum-likelihood is not defined, making the choice of error functions depending more strongly on other criteria like robustness. For large errors, the steep gradient of the quadratic error makes the training converge fast towards a better model parametrisation, but as the training error approaches lower values, the absolute error function becomes more favourable because of its robustness against outliers.[46] Therefore, the following error function was introduced, which continuously transforms from a quadratic to an absolute error function in the range of $\Delta E = |10^{-1} - 10^{1}|\ eV$:

$$E_{\text{tot}} = \frac{1}{2} \sum_{i}^{\text{Batchsize}} (\Delta E)^2 \cdot \left(\sigma(|\Delta E|) - \frac{1}{2}\right) + |\Delta E| \cdot \left(\sigma(|\Delta E|) + \frac{1}{2}\right), \tag{4.2}$$

with σ denoting the sigmoid function (see 2.6.2). By modifying the function $\sigma(|\Delta E|)$ to $\sigma(|\Delta E| \cdot k + d)$ the transition can be adjusted to the specific problem.

To compare the convergence behaviour of the networks for the quadratic and the adaptive error function we train first with a Ni_5Au_5 dataset and equal parametrization with a learning rate of $l_r = 0.001$.

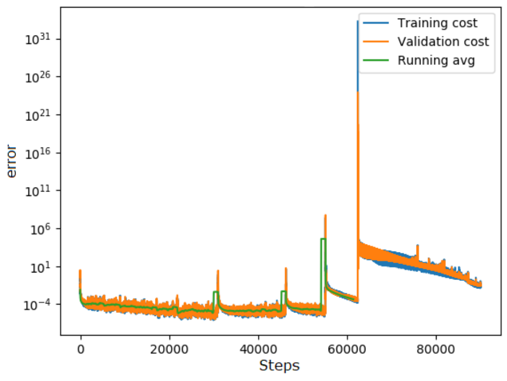


Figure 4.4.: Quadratic error as a function of the training steps.

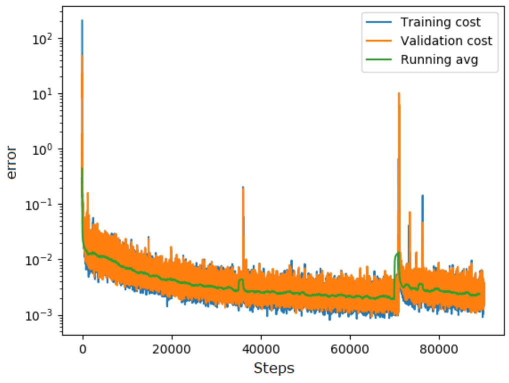


Figure 4.5.: Adaptive error as a function of the training steps.

Table 4.2.: Comparison between the quadratic and the adaptive error function w.r.t. the energy root mean square error E_{RMSE} in meV for $l_r = 0.001$.

	E_{RMSE} training / meV	E_{RMSE} validation /meV
Quadratic	261.49	281.79
Adaptive	4.51	4.85

Figure 4.4 shows that the quadratic error function suffers from large instabilities during the training process, being the result of outliers in the training data. Outliers are far off the predicted values and therefore produce large errors, which are enhanced by the quadratic behaviour of the error function. Although Figure 4.5 also shows spikes produced by outliers in the dataset, only the model using the quadratic error function is affected significantly, resulting in a much larger energy root mean square error. The quadratic term is less dominant in the adaptive error function, resulting in smaller weight changes during the back-propagation and a smoother convergence behaviour.

Using a smaller learning rate of $l_r = 1e^{-4}$, both error charts converge smoothly towards the minimum. However, the adaptive error function reaches a significantly lower root mean square error after the same number of training steps without over-fitting the training data points.

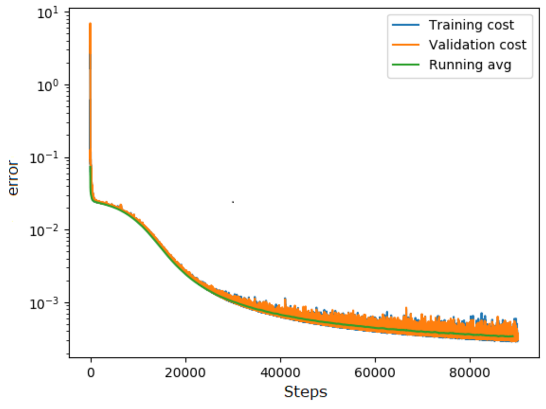


Figure 4.6.: Quadratic error as a function of the training steps.

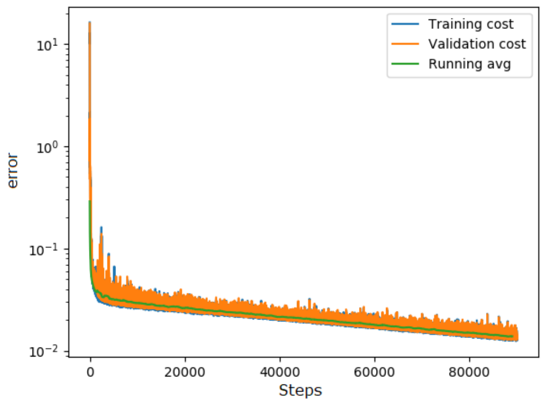


Figure 4.7.: Adaptive error as a function of the training steps.

Table 4.3.: Comparison between the quadratic and the adaptive error function with respect to the energy root mean square error E_{RMSE} in meV for $l_r = 1e^{-4}$.

	E_{RMSE} training / meV	E_{RMSE} validation /meV
Quadratic	11.85	11.71
Adaptive	4.13	4.96

By looking at the convergence behaviour of the training runs in Figures 4.6 - 4.7 , it can be stated that the adaptive error function outperforms the standard quadratic error function even for a smaller learning rate. This is an interesting finding, because neither of the error plots show any major spikes. Nevertheless, the adaptive error function seems to be able to further reduce the error, while the quadratic error function has almost converged after the specified number of training steps. This might be a result of the not normally distributed nature of our training data.

4.4.6. Activation functions

To connect the input and the hidden layers the ELU (see 2.6.2) activation function is used. It allows for an efficient training of deep neural networks and also provides a smooth energy landscape for the model. For the output layer a linear activation is used in order to have an unrestricted value range for the energy output. A series of tests revealed that the training speed and the RMSE of the model can be improved significantly by exchanging one of the ELU activation functions with a Morse potential-like activation function. By forcing the network into this structure, we can assume that the input of the Morse layer has to represent an adapted distance. The part of the network following the Morse layer then post-processes the output to the fit the training potential. For the activation function all parameters of the Morse potential are chosen to be one, yielding the function

$$f(x) = e^{-2(x-1)} - 2e^{-(x-1)}. \tag{4.3}$$

This physically motivated activation function also improves the asymptotic behaviour of the potential for small inter-atomic distances and lowers the number of training points needed in this critical section of the phase space. This way, we obtain a more robust model, e.g. in combination with MD-simulations.

Comparison between different network architectures.
The training was performed for 4 different network architectures, with the Morse activation function replacing the ELU activation function between different layers. By analysing the convergence behaviour, the root mean square error, and the shape of the potential, the overall performance of the variants will be assessed.

Table 4.4.: Different network architectures.

Run Nr.	
1	Morse activation function between layer 1 and 2
2	Morse activation function between layer 2 and 3
3	Morse activation function between layer 3 and 4
4	Only ELU activation functions

Training with energies
For visualization purposes, the error graphs were made using the smoothing filter of TensorBoard with the smoothing factor f. Figures 4.8 - 4.10 show that for the training on energies only, run 3 performs well at the beginning of the training, but fails to reduce the error below a certain threshold. The best overall performance was delivered in runs 1 and 2. Note that run 1 achieves the lowest E_{RMSE}, but not the lowest error. This can be a result of difficulties in the regularization for this configuration. In general, run 2 has a smoother decay of the error function and the best overall performance. All variants performed better then the pure mathematical model using only ELU activation functions.

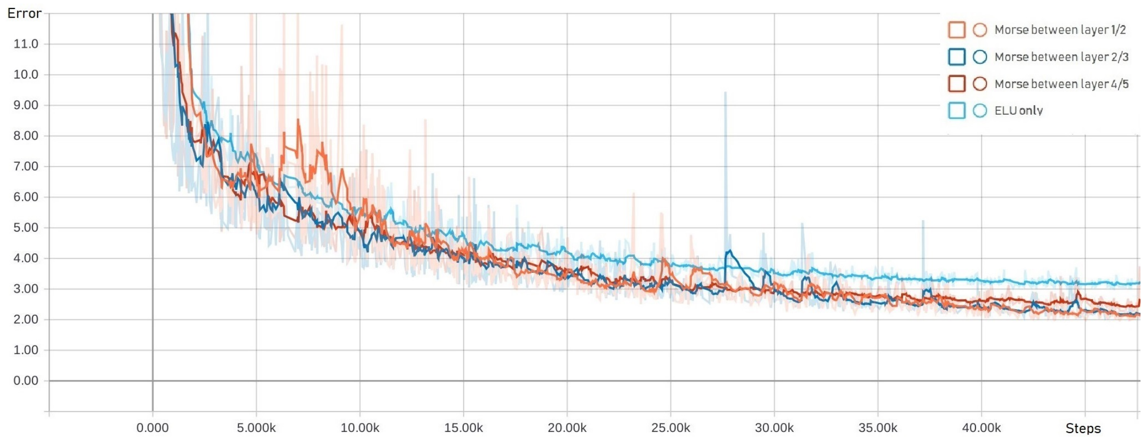


Figure 4.8.: Comparison between the errors for the different network architectures trained on energies as a function of the training steps ($f = 0.85$).

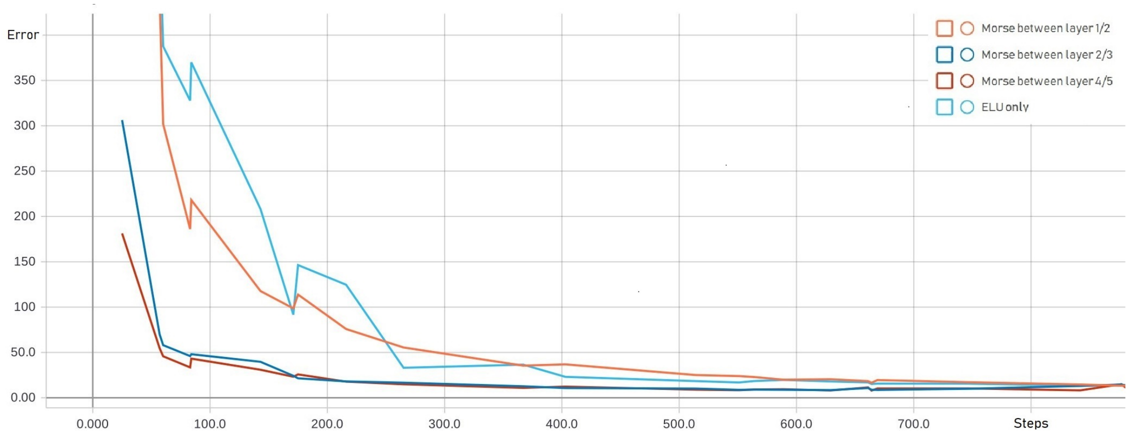


Figure 4.9.: Comparison between the errors for the different network architectures trained on energies for the first 800 training steps ($f = 0.00$).

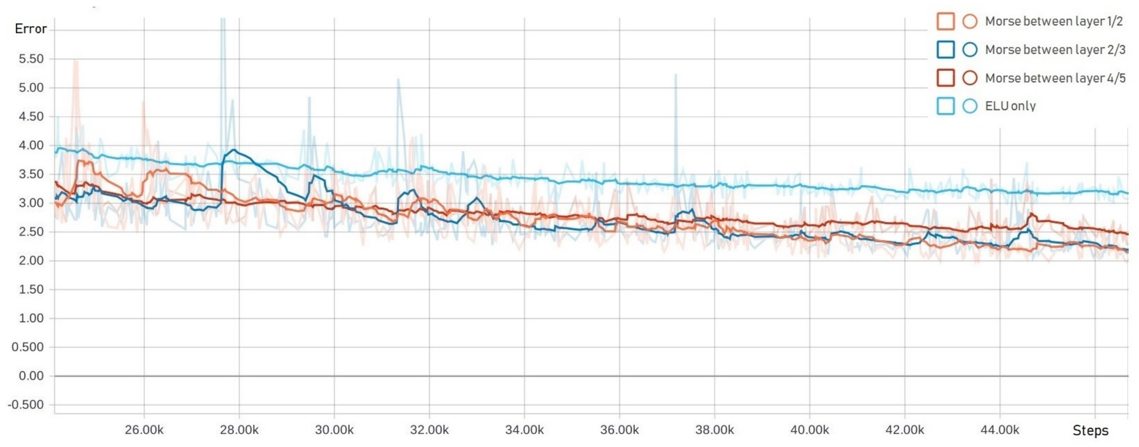


Figure 4.10.: Comparison between the errors for the different network architectures trained on energies for the last 25000 training steps ($f = 0.85$).

Table 4.5.: Comparison of the energy root mean square error E_{RMSE} in meV for the different network architectures trained on energies.

Run Nr.	E_{RMSE} training /meV	E_{RMSE} validation /meV
1	6.87	6.51
2	7.57	7.37
3	10.61	10.57
4	11.99	12.76

Training with energies and forces

By including gradients into the training the error converges much faster. As Figures 4.8 - 4.10 show, run number 1 again achieves the lowest E_{RMSE} and run 2 the lowest error. From the graph one can see that the error for run 1 is oscillating very strongly and therefore the results may vary significantly depending on the number of steps used for training.

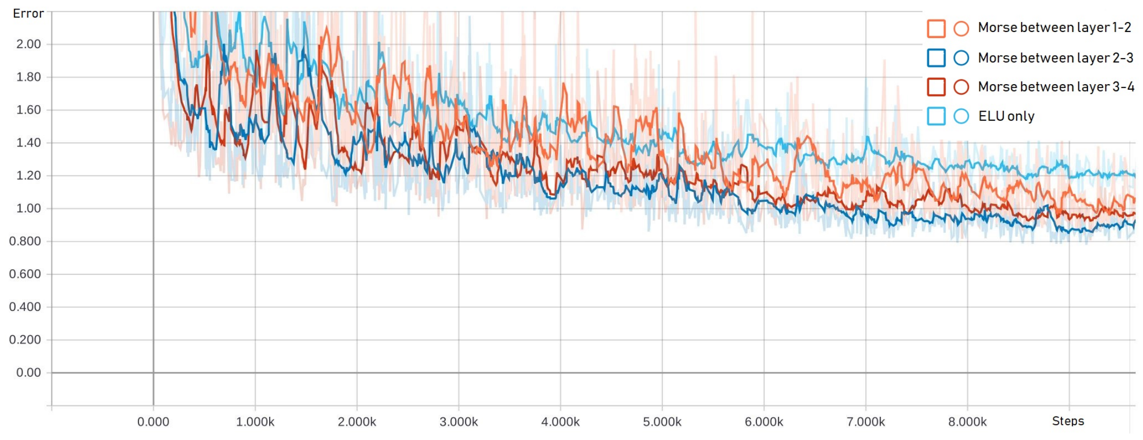


Figure 4.11.: Comparison between the errors for the different network architectures trained on energies and forces as a function of the training steps ($f = 0.85$).

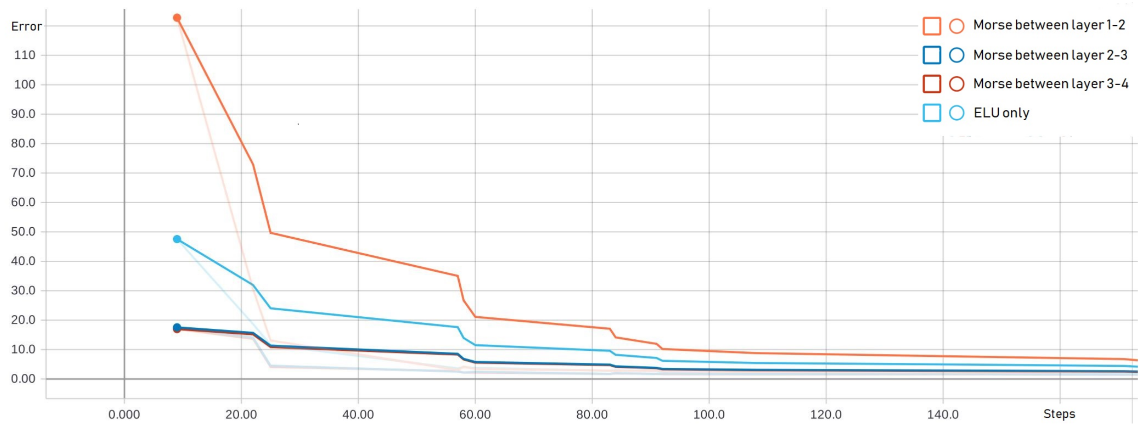


Figure 4.12.: Comparison between the errors for the different network architectures trained on energies and forces for the first 150 training steps ($f = 0.5$).

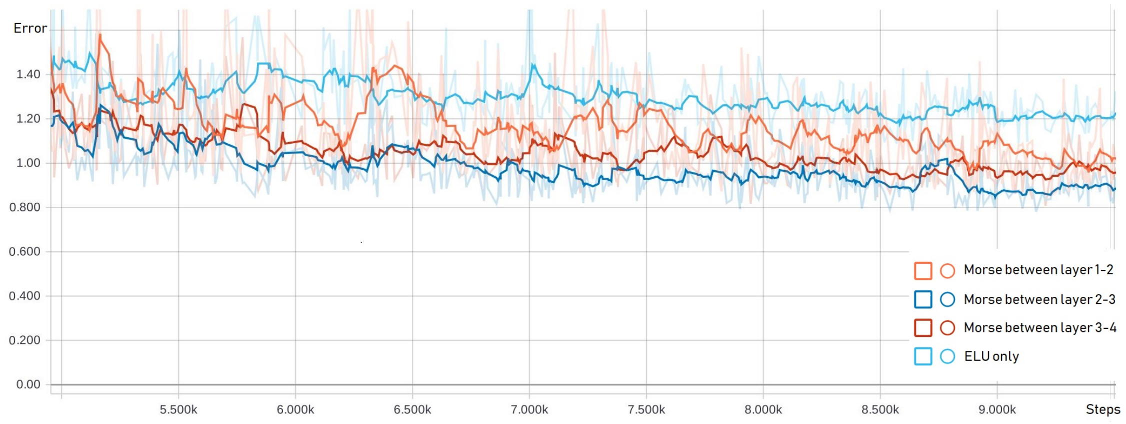


Figure 4.13.: Comparison between the errors for the different network architectures trained on energies and forces for the last 5000 training steps ($f = 0.85$).

Table 4.6.: Comparison of the energy root mean square error E_{RMSE} in meV for the different network architectures trained on energies and forces.

Run Nr.	E_{RMSE} training /meV	E_{RMSE} validation /meV
1	5.33	5.39
2	7.06	6.88
3	8.79	8.63
4	11.88	11.75

Comparision of the potential shapes for the Ni_2Au system

For visualisation, the inter-atomic distance $r = r_1 = r_2$ and the angle ϕ between the atoms are chosen, which enforces at least C_{2v} symmetry in the triatomic system. The results are summarized in Figure 2.10.

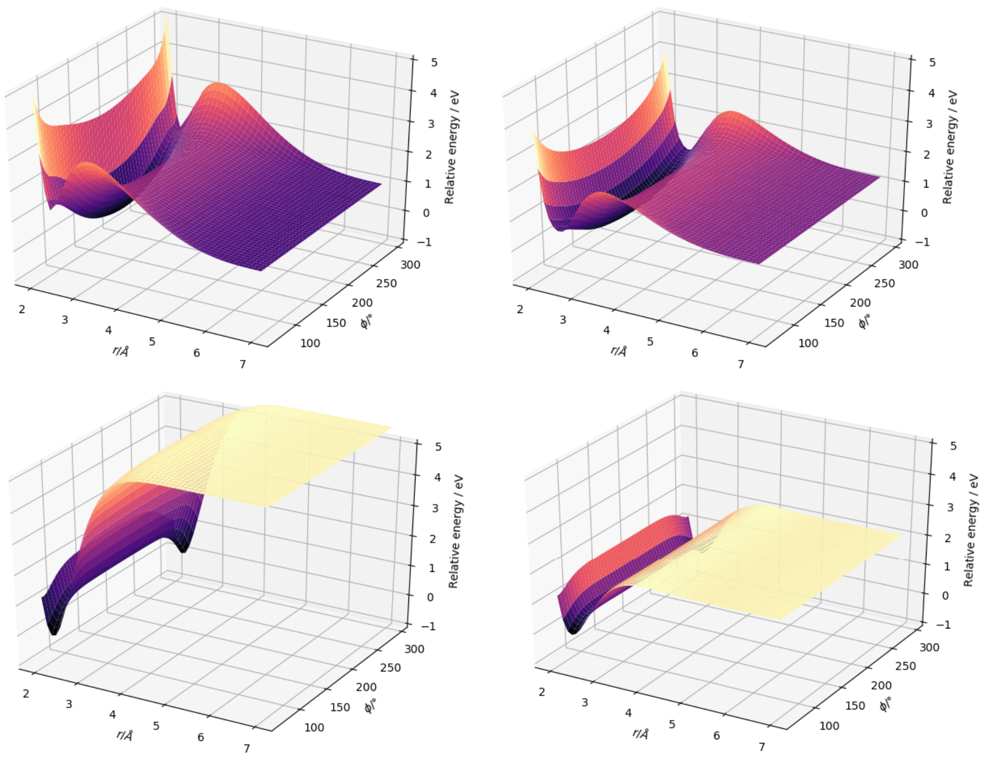


Figure 4.14.: Cut through the PES of the Ni_2Au system for the different runs.

By looking at the PES for this simple system one can observe that the runs 1 and 2 are able to reproduce the correct equilibrium distance as well as the exponential dependence for small radii and angles. Interestingly, run 3 fails to get the correct shape of the PES. This may be a result of the low remaining flexibility of the network after the Morse layer. In run 4, the network manages to reproduce the PES close to the equilibrium distance, but shows a non-physical behaviour for small angles. This is due to a lack of data points in this region of the phase space.

4.4.7. Comparison of the error for different methods

Modified Embedded Atom Model (MEAM)

The modified embedded atom model was evaluated with the LAMMPS[47],[48] program package using the parameters as suggested by Baskes.[8]

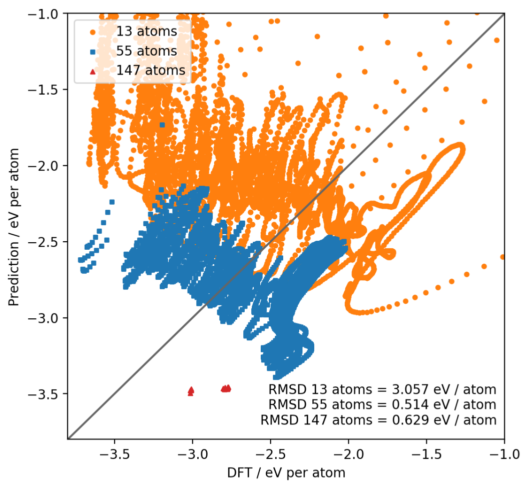


Figure 4.15.: Comparison between the energies predicted by the MEAM, and the calculated DFT data.

Unsurprisingly, the MEAM fails for small cluster sizes. However, it also performs badly for the largest clusters in the dataset consisting of 147 atoms.

Second Moment Approximation to Tight Binding (SMATB)

The SMATB model (see eq.2.45) was fitted to the Ni-Au data also used for the training of the neural network (see Section 4.3).

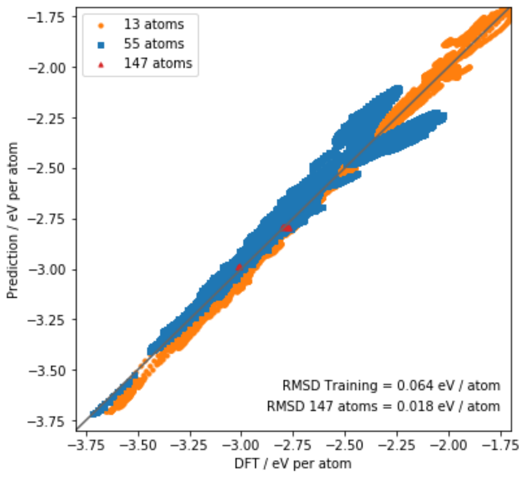


Figure 4.16.: Comparison between the energies predicted by the SMATB model, and the calculated DFT data.

The used parameters can be found in Table 4.7.

Table 4.7.: Fitted parameters for the SMATB model.

	Ni-Ni	Ni-Au	Au-Au
A_{ij} /eV	0.20	0.20	0.24
ζ_{ij} /eV	1.84	1.60	1.40
p_{ij}	9.94	9.93	9.78
q_{ij}	2.49	3.54	4.30
$r_{0,ij}$ /Å	2.49	2.69	2.88

Although this method manages to get a significantly lower error for the larger clusters than the MEAM, it is still too inaccurate for smaller cluster sizes.

Neural network approach

The network consists of a **16-80-60-40-1** structure using ELU activation functions except between layer two and three, where a Morse potential like function is used.

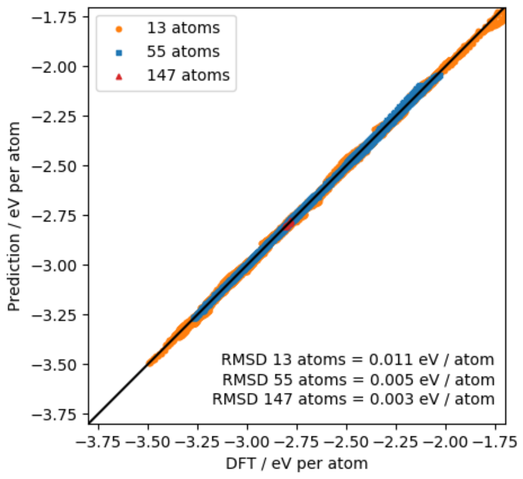


Figure 4.17.: Comparison between the energies predicted by the NN trained on energies, and the calculated DFT data.

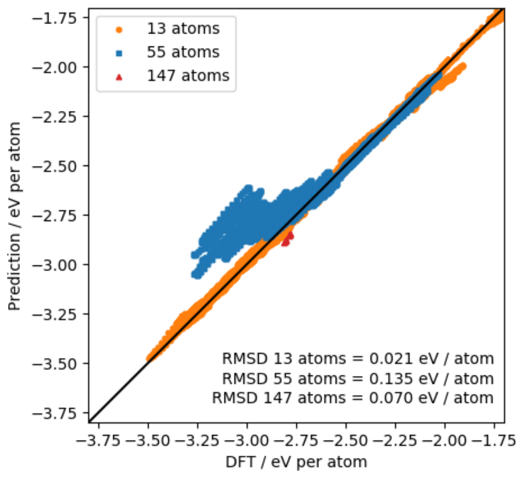


Figure 4.18.: Comparison between the energies predicted by the NN trained on energies and forces, and the calculated DFT data.

The network trained solely on energies is able to reproduce the absolute energies for the systems within the training dataset with a high accuracy and also has a comparable error for the extrapolation to larger clusters (see Figure 4.17). By including forces to the training the overall error rises for all cluster sizes. It has to be noted that due to memory issues not all geometries were included in the training set, which results in a significantly larger error for these geometries (see Figure 4.18).

4.4.8. Prediction of relative energies

Having a comparable E_{RMSE} for the NiAu_{146} clusters in the test set, the SMATB model and the neural network approach are being compared in their ability to predict relative energies. Seven NiAu_{146} icosahedra are selected (see Figure 4.19) for the assessment of the reproduction quality of relative energies. Both methods, the SMATB and the neural network approach are evaluated for the seven geometries and the DFT energies are calculated accordingly. Afterwards, the resulting energies relative to the global optimum structure (Ni located in the center of the cluster) are calculated.

Neural network trained on energies

The relative energies E_{DFT}, E_{SMATB} and E_{NN} are calculated with respect to the lowest energy solution $E_{i,0}$ of the specific method.

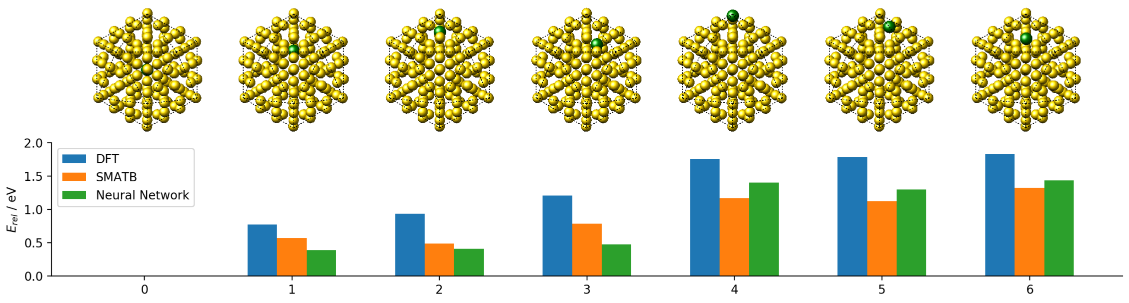


Figure 4.19.: Evaluation of the relative energies of different geometries for a neural network being trained on energies.

Table 4.8.: Comparison between SMATB, DFT and the NN relative energy results, for the NN being trained on energies.

Geometry Nr.	E_{DFT} /eV	E_{SMATB} /eV	E_{NN} /eV
0	0.00	0.00	0.00
1	0.77	0.57	0.39
2	0.93	0.48	0.40
3	1.20	0.78	0.47
4	1.75	1.17	1.40
5	1.79	1.12	1.29
6	1.83	1.32	1.43

While both methods manage to predict the global optimum structure correctly, they also fail to reproduce the correct energy ordering for the different geometries. Therefore, none of the above methods is preferable over the other for this task.

Neural network trained on energies and forces

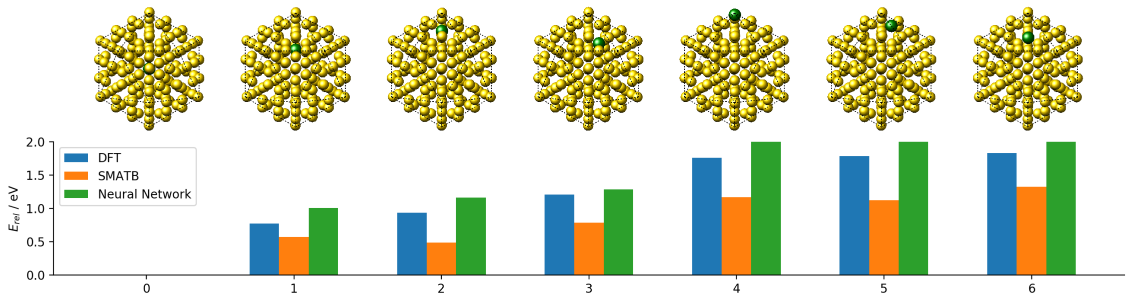


Figure 4.20.: Evaluation of the relative energies of different geometries for a neural network being trained on energies and forces.

Table 4.9.: Comparison between SMATB, DFT and the NN relative energy results, for the NN being trained on energies and forces.

Geometry Nr.	E_{DFT} / eV	E_{SMATB} / eV	E_{NN} / eV
0	0.00	0.00	0.00
1	0.77	0.57	1.00
2	0.93	0.48	1.16
3	1.20	0.78	1.28
4	1.75	1.17	2.08
5	1.79	1.12	2.11
6	1.83	1.32	2.22

The inclusion of forces into the training improves the relative energies of the neural network approach significantly. It reproduces the correct order for all geometries. Therefore, this network seems applicable to geometry optimizations.

5. Conclusion and Outlook

The use of neural networks for the calculation of metallic nanoparticles has shown to be a promising but also quite limited approach. The quality of the network is strongly dependent on the used training data. Extensive amounts of ab inito calculations need to be performed in a preliminary step, in order to cover sufficiently large regions of the phase space. This reduces the calculation time benefit against pure *ab initio* methods for small cluster sizes. However, the neural network method proved to be a valuable alternative to other force field methods for the application on metal clusters by drastically reducing the error with respect to the DFT calculations. In geometry optimizations, the neural network potential could improve on the force field potentials as well. When trained on energies and forces, it was able to reproduce the *ab initio* results qualitatively. This may also enable the use in MD-calculations and geometry optimizations of large metallic clusters, where the convergence of the DFT iteration is difficult to achieve.
Since the relative energy error using only radial symmetry function was low enough to reproduce the expected energy ordering between the test geometries, angular interactions do not seem to contribute significantly for the investigated Ni-Au system. Once the neural network potential is trained and validated, the calculation time per geometry is mostly dependent on the translation of Cartesian coordinates to symmetry function activations. By utilizing only radial two-body interactions, the computational effort can be reduced from N^3 to N^2, which is an significant improvement for larger clusters. To further improve the efficiency of the translation from Cartesian coordinates into symmetry functions, the number and parameters of the Gaussian functions could be optimized in advance to present the data to the neural network in an optimal way. This could either be done by empirical regression on the target system or by finding an analytic expression for the distribution of symmetry functions based on a set of optimized parameters.
Additionally, a new error function was introduced to improve the convergence behaviour during the training. The error function was shown to be less effected by outliers during the training and reached a lower error on both the validation and the training dataset. This new error function allows for larger learning rates and therefore shorter total training times. In future investigations, the error function should be tested using other optimizers, including second order optimizers to verify the convergence improvements for these methods. The general idea of making an adaptive error function can also be extended and tested for other applications.
Furthermore, it was shown that, by including a Morse potential-like activation function in one layer of the network, the convergence as well as the asymptotic behaviour of the neural network potential could be improved significantly. This measure not only reduced the error of the method, but also fixed the undefined and often non-physical behaviour of the network in formerly unexplored parts of the phase space. It was shown that also the position of the Morse-activation function within the network played an important role in the networks convergence behaviour, with the most reliable results being achieved by

placing the Morse-activation function between layer 2 and 3. Future investigations should try to verify these improvements also for organic systems when three-body interactions are included. Also, more complex activation functions can be thought of and tested to force the neural network expression to be of a certain form and include further physically meaningful information right from the start.
Finally, to ultimately test this method, measurable physical properties need to be calculated and compared to experimental results. Using the molecular dynamics module, a phase diagram could be constructed and diffusion processes could be modelled for selected metal combinations and cluster sizes. Furthermore, in combination with other methods, the potential could be used for transition state searches as for example in catalysis applications. Nevertheless, it has to be stated that the generation of training data is an immensely time consuming task, which grows in complexity as the number of different elements increases. Therefore, this approach seems to be currently limited to a very specific class of computational chemistry problems.

A. Appendix

Sample input file for Quantum Espresso

```
&CONTROL
 calculation  = 'md',
 restart_mode = 'from_scratch',
 prefix      = 'Ni6Au7_md_0',
 pseudo_dir   = '/home/…',
 outdir      = '/home/…',
 dt = 50,
 nstep = 500
/
&SYSTEM
 ibrav    = 1,
 celldm(1) = 45
 nat      = 13,
 ntyp     = 2,
 ecutwfc   = 30.D0,
 ecutrho   = 300.D0,
 nosym = .true.,
 occupations='smearing', smearing='methfessel-paxton',degauss=0.01
/
&ELECTRONS
 conv_thr    = 1.D-4,
 mixing_beta = 0.1D0,
 electron_maxstep = 400
/
&IONS
 ion_dynamics = 'verlet',
 ion_temperature ='berendsen'
 tempw = 2000.D0
 pot_extrapolation='second_order'
 wfc_extrapolation='second_order'
/
```

```
ATOMIC_SPECIES
Ni 58.69 Ni.pbe-n-kjpaw_psl.0.1.UPF
Au 197.00 Au.pbe-dn-kjpaw_psl.0.1.UPF
ATOMIC_POSITIONS {{angstrom}}
Ni   0.00000000000    0.00000000000    0.00000000000
Ni   0.00000000000    1.15660844666    1.87143177837
Ni   1.15660844666    1.87143177837    0.00000000000
Ni   1.87143177837    0.00000000000    1.15660844666
Ni  -1.15660844666    1.87143177837    0.00000000000
Ni  -1.87143177837    0.00000000000    1.15660844666
Au   0.00000000000   -1.15660844666    1.87143177837
Au  -1.15660844666   -1.87143177837    0.00000000000
Au   1.15660844666   -1.87143177837    0.00000000000
Au   1.87143177837    0.00000000000   -1.15660844666
Au   0.00000000000    1.15660844666   -1.87143177837
Au  -1.87143177837    0.00000000000   -1.15660844666
Au   0.00000000000   -1.15660844666   -1.87143177837
K_POINTS Gamma
```

Bibliography

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

[2] T. Shaikhina and N. A. Khovanova, "Handling limited datasets with neural networks in medical applications: A small-data approach," *Artificial Intelligence in Medicine*, vol. 75, pp. 51 – 63, 2017.

[3] O. College., *Anatomy & physiology.* Houston, TX: OpenStax CNX. Retrieved from http://cnx.org/content/col11496/latest/, 2013.

[4] F. Jensen, *Introduction to Computational Chemistry.* John Wiley & Sons, 2006.

[5] P. M. Morse, "Diatomic molecules according to the wave mechanics. ii. vibrational levels," *Phys. Rev.*, vol. 34, pp. 57–64, Jul 1929.

[6] J. E. Lennard-Jones, "Cohesion," *Proceedings of the Physical Society*, vol. 43, no. 5, p. 461, 1931.

[7] M. W. Finnis and J. E. Sinclair, "A simple empirical n-body potential for transition metals," *Philosophical Magazine A*, vol. 50, no. 1, pp. 45–55, 1984.

[8] M. I. Baskes, "Modified embedded-atom potentials for cubic materials and impurities," *Phys. Rev. B*, vol. 46, pp. 2727–2742, Aug 1992.

[9] A. P. Sutton and J. Chen, "Long-range finnissinclair potentials," *Philosophical Magazine Letters*, vol. 61, no. 3, pp. 139–146, 1990.

[10] R. P. Gupta, "Lattice relaxation at a metal surface," *Phys. Rev. B*, vol. 23, pp. 6265–6270, Jun 1981.

[11] M. Born and J. E. Mayer, "Zur Gittertheorie der Ionenkristalle," *Zeitschrift für Physik*, vol. 75, pp. 1–18, Jan 1932.

[12] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14, pp. 2627 – 2636, 1998.

[13] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of Electronic Imaging*, vol. 16, 2007.

[14] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, pp. 568–576, Nov 1991.

[15] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994* (W. W. Cohen and H. Hirsh, eds.), pp. 157 – 163, San Francisco (CA): Morgan Kaufmann, 1994.

[16] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359 – 366, 1989.

[17] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, Dec 1989.

[18] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[19] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015.

[20] J. Kiefer and J. Wolfowitz, "Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters," *The Annals of Mathematical Statistics*, vol. 27, no. 4, pp. 887–906, 1956.

[21] H. Adeli and M. Wu, "Regularization neural network for construction cost estimation," *Journal of Construction Engineering and Management*, vol. 124, no. 1, pp. 18–24, 1998.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[23] K. Levenberg, "A method for the solution of certain non-linear problems in least squares.," *Q. Appl. Math.*, vol. 2, pp. 164–168, 1944.

[24] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[25] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," *The Journal of Chemical Physics*, vol. 134, no. 7, p. 074106, 2011.

[26] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, pp. 22–30, March 2011.

[27] 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.

[28] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "Sympy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017.

[29] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 27.02.2018].

[30] S. R., "PyParticles: Open source scientific tools for Python," 2012 –. [Online; accessed 27.2.2018].

[31] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[32] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," *The Journal of Chemical Physics*, vol. 81, no. 8, pp. 3684–3690, 1984.

[33] T. Schneider and E. Stoll, "Molecular-dynamics study of a three-dimensional one-component model for distortive phase transitions," *Phys. Rev. B*, vol. 17, pp. 1302–1322, Feb 1978.

[34] B. DNWEG and W. PAUL, "Brownian dynamics simulations without gaussian random numbers," *International Journal of Modern Physics C*, vol. 02, no. 03, pp. 817–827, 1991.

[35] M. Bonomi, D. Branduardi, G. Bussi, C. Camilloni, D. Provasi, P. Raiteri, D. Donadio, F. Marinelli, F. Pietrucci, R. Broglia, and M. Parrinello, "Plumed: A portable plugin for free-energy calculations with molecular dynamics," vol. 180, pp. 1961–1972, 10 2009.

[36] M. Schnedlitz, M. Lasserus, R. Meyer, D. Knez, F. Hofer, W. E. Ernst, and A. W. Hauser, "Stability of coreshell nanoparticles for catalysis at elevated temperatures: Structural inversion in the NiAu system observed at atomic resolution," *Chemistry of Materials*, vol. 30, no. 3, pp. 1113–1120, 2018.

[37] J. Gao, H. Gu, and B. Xu, "Multifunctional magnetic nanoparticles: design, synthesis, and biomedical applications," *Accounts of chemical research*, vol. 42, no. 8, pp. 1097–1107, 2009.

[38] C. Sun, J. S. Lee, and M. Zhang, "Magnetic nanoparticles in MR imaging and drug delivery," *Advanced drug delivery reviews*, vol. 60, no. 11, pp. 1252–1265, 2008.

[39] S. Alayoglu, A. U. Nilekar, M. Mavrikakis, and B. Eichhorn, "Ru-Pt core-shell nanoparticles for preferential oxidation of carbon monoxide in hydrogen," *Nat Mater*, vol. 7, pp. 333–338, 04 2008.

[40] D. C. Papageorgopoulos and F. A. de Bruijn, “Examining a potential fuel cell poison: A voltammetry study of the influence of carbon dioxide on the hydrogen oxidation capability of carbon-supported Pt and PtRu anodes,” *Journal of The Electrochemical Society*, vol. 149, no. 2, pp. A140–A145, 2002.

[41] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. D. 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, no. 39, p. 395502, 2009.

[42] J. P. Perdew, K. Burke, and M. Ernzerhof, “Generalized gradient approximation made simple,” *Phys. Rev. Lett.*, vol. 77, pp. 3865–3868, Oct 1996.

[43] J. S. Smith, O. Isayev, and A. E. Roitberg, “Ani-1: an extensible neural network potential with dft accuracy at force field computational cost,” *Chem. Sci.*, vol. 8, pp. 3192–3203, 2017.

[44] M. Gastegger and P. Marquetand, “High-dimensional neural network potentials for organic reactions and an improved training algorithm,” *Journal of Chemical Theory and Computation*, vol. 11, no. 5, pp. 2187–2198, 2015. PMID: 26574419.

[45] R. Ge, F. Huang, C. Jin, and Y. Yuan, “Escaping from saddle points - online stochastic gradient for tensor decomposition,” *CoRR*, vol. abs/1503.02101, 2015.

[46] F. H. Thanoon, “Robust regression by least absolute deviations method,” *International Journal of Statistics and Applications, Vol. 5 No. 3,pp. 109-112.*, 2015,.

[47] S. Plimpton, “Fast parallel algorithms for short-range molecular dynamics,” *Journal of Computational Physics*, vol. 117, no. 1, pp. 1 – 19, 1995.

[48] S. J. Plimpton and A. P. Thompson, “Computational aspects of many-body potentials,” *MRS Bulletin*, vol. 37, no. 5, p. 513521, 2012.

[49] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the Marquardt algorithm,” *IEEE Transactions on Neural Networks*, vol. 5, pp. 989–993, Nov 1994.

[50] R. LeSar, *Introduction to Computational Materials Science: Fundamentals to Applications.* Cambridge University Press, 2013.

[51] B.-J. Lee, W.-S. Ko, H.-K. Kim, and E.-H. Kim, “The modified embedded-atom method interatomic potentials and recent progress in atomistic simulations,” *Calphad*, vol. 34, no. 4, pp. 510 – 522, 2010.

[52] M. S. Daw and M. I. Baskes, “Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals,” *Phys. Rev. B*, vol. 29, pp. 6443–6453, Jun 1984.