Accurate Force- and Hessian Predictions from Neural Network Potentials

Von der Fakultät Chemie der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

April Mae Cooper

aus Ludwigsburg

Hauptberichter	Prof. Dr. Johannes Kästner
Mitberichter	Prof. Dr. Willem M. Klopper
Prüfungsvorsitzender	Prof. Dr. Frank Gießelmann

Tag der Mündlichen Prüfung: 03.07.2020

Institut für Theoretische Chemie
Universität Stuttgart

2020

I want to understand more about the world while I'm still here.

—Jostein Gaarder

Acknowledgments

Many people have enabled me to accomplish the work presented in this thesis and I want to say 'thank you for everything' to everyone, who supported me during my doctoral studies.

I would particularly like to thank Johannes Kästner for giving me the opportunity to work in the highly interesting and current field of machine learning. He enriched the work presented by frequent, lively discussions and fruitful ideas on the topic. I owe special thanks to Oliver Kohlbacher for agreeing to be the second examiner of this thesis and to Frank Gießelmann for assuming the chair of the examination board.

I am very grateful to Alexander Urban and Nongnuch Artrith for making my research stay at the university of St. Andrews possible. I further want to thank them for inspiring the successful collaborative research project on novel force training methods, whose results have become an integral part of this thesis.

Sincere thanks are given to all members of our research group, who made time for scientific exchange. I want to give special thanks to Sonia Álvarez Barcia for helping me out with diverse issues dealing with quantum chemical software and for lending a sympathetic ear to me, whenever scientific questions and challenges were getting exhausting. I further want to thank Alexander Denzel for many enlightening discussions on machine learning related topics.

I am grateful for the support by Björn Miksch and my family, who were always on hand with moral support and advice for me and without whom the work presented in this thesis would not have been possible to complete.

Abstract

On the basis of the Born–Oppenheimer approximation it is possible to define a potential energy surface (PES), which describes the potential energy of a chemical system dependent on the positions of the atoms.

Precise knowledge of the PES is crucial for many simulations in computational chemistry, as it defines, for example, equilibrium and transition structures, the energetics of chemical reactions as well as the forces acting on the atoms in a given structural configuration. Unfortunately, obtaining accurate information on the PES during a simulation requires computationally highly demanding quantum chemical computations, which increase the computational and time effort of a simulation drastically.

Thus, efficient approaches for obtaining an accurate description of the PES *a priori* are of high interest, since they significantly accelerate the simulation process and thereby enable the investigation of complex systems. A well established approach is to employ parameterized, physically motivated potentials like force fields. Alternatively it has become popular to employ a surrogate model of a PES which was constructed with machine learning methods. One approach which has become popular in current scientific studies is to employ artificial neural networks (NN) for this task. In contrast to force fields this approach isn't based on a physically motivated approximation of the PES, but it results in a solely mathematical model.

The process in which surrogate models are constructed with NNs on the basis of reference data is called training. During the training process highly flexible functions are adjusted such that the resulting model of the PES is optimal in the sense that it minimizes a pre-defined error measure of the approximation given by the so called loss function. The conventional approach to train these machine learned models is based on the accuracy of energy predictions made by the NN-PES alone. Thus, it will be called *energy training approach* in the remainder of this

thesis. This training approach is particularly well suited for applications in which the potential energy of a system is of high interest. However, the applicability of these potentials to simulations that require predictions of atomic forces or even Hessians is often limited due to a lack of accuracy.

In order to train NN-PESs with the energy training approach such that highly accurate predictions of derivatives of the underlying PES are possible, it is necessary to perform the training with a sufficiently large reference data set. This data set has to sample the area of phase space that is of interest for the subsequent simulation very thoroughly to allow for precise force- and Hessian predictions. On the one hand this introduces a significant computational overhead to the construction of NN-potentials as a large number of computationally demanding quantum chemical computations has to be performed in order to obtain the reference data set. On the other hand a thorough sampling of the phase-space is highly non-trivial for systems with a large number of atoms due to the high dimensionality of the PES.

Alternatively one can construct NN-PESs suited for the prediction of forces and Hessians by adding information on the derivatives of the PES to the reference data set and training on energy, force and, if required, Hessian information. In contrast to many other fields of machine learning, where including information on gradients in the reference data set increases the computational effort for the construction of the reference data set significantly, is the increase in computational effort for the machine learning problem at hand comparatively low. This is due to the fact that the effort for computing the gradient of the PES by quantum chemical calculations is comparable to the one for computing an energy at the same level of theory. Including information on the gradient, however, increases the information density in the reference data set significantly. Given a N dimensional PES such a reference data set contains for each chemical structure $N + 1$ pieces information that can be used for training as the gradient is a N dimensional vector and the energy a scalar quantity.

Due to the high information density, it is possible to construct precise NN-PESs on the basis of a reference data set that is significantly smaller than the ones required to reach similar accuracy by employing energy training. Thus, the computational effort for constructing these reference data sets is comparatively low since the reference data set is rather small and gradient information can be

obtained efficiently.

This thesis deals with training approaches for NN-potentials that allow for accurate approximations of PESs and the respective atomic forces and, if required, even Hessians. In this thesis it is demonstrated that including quality measures for energy, force and Hessian predictions of the NN-potential into the loss function yields excellent descriptions of the PES and its derivatives. NN-potentials constructed by this direct force training approach can even be employed in simulations which are highly sensitive to errors in the Hessian information, such as instanton reaction rate constant computations.

It is shown that the calculation of reaction rate constants by the instanton method on a coupled-cluster level of theory is possible when a NN-PES is employed in rate constant calculations. On the basis of reaction rate constants, which were obtained from the surrogate model for various deuteration patterns of the reaction $CH_3OH + H \rightarrow CH_2OH + H_2$, a good qualitative explanation for the unexpectedly high deuteration of methanol in the interstellar medium (ISM) can be found.

Moreover, a detailed derivation of an algorithm to perform direct force training for atomic neural networks is given.

Furthermore, it is shown that the predictions of forces cannot only be improved by direct force training, but also in a computationally significantly less demanding manner by employing a novel Taylor expansion based approach. This novel approach reduces the computational effort of training a NN with force information by introducing this information indirectly to the reference data. For this purpose, additional training data is generated from existing training examples. New structures are generated by slightly displacing atoms in a subset of the original training examples and extrapolating the energy of the newly generated structures with a first order Taylor series expansion scheme.

The predictive power of NN-potentials trained with this approach is compared to the predictive power of NN-PESs trained with the direct force training approach and the conventional energy training approach. As a test case predictions of potential energies and atomic forces for a multitude of structures of a cluster made up of six water molecules is investigated.

It is demonstrated that the Taylor expansion based approach yields a significant improvement of the predicted atomic forces in comparison to atomic forces

obtained by training NN-potentials with the conventional energy training approach. By comparing this approach to direct force training it is found that the direct force training approach yields the most accurate force predictions for the water cluster. However, employing the approximate Taylor expansion based force training approach yields about 50% of the improvement gained by direct force training at a significantly lower computational cost.

The software developed for the Taylor expansion based and direct force training approach is open source and was implemented in the software package ænet.

Zusammenfassung

Auf Basis der Born–Oppenheimer Näherung ist es möglich die Potentialenergiefläche (PES) eines Systems zu definieren. Die der PES zugrundeliegende Funktion beschreibt die potentielle Energie eines chemischen Systems abhängig von seiner Geometrie.

Eine exakte Kenntnis der PES ist für viele Simulationsanwendungen in der Computerchemie unabdingbar, da sie unter anderem die Energetik chemischer Reaktionen, sowie die atomaren Kräfte bestimmter Molekülgeometrien beschreibt. Darüber hinaus definieren die stationären Punkte der PES die Gleichgewichts- und Übergangszustands-Geometrien des Systems und ihre zugehörigen Energien.

Allerdings ist die exakte Bestimmung der PES während einer Computersimulation sehr rechenintensiv, da sie die Durchführung komplexer quantenchemischer Berechnungen erfordert. Daher wird durch dieses Vorgehen der mit der Simulation verbundene Rechen- und Zeitaufwand signifikant erhöht, was die Anwendbarkeit eines solchen Simulationsansatzes auf komplexe Systeme stark einschränkt, bis hin zur Unmöglichkeit der Beschreibung des Systems.

Aufgrund dessen ist es von großem Interesse Ansätze zu entwickeln, die *a priori* eine akkurate Beschreibung der PES erlauben, um den Rechenaufwand während der Simulation zu minimieren. Ein fest etablierter Ansatz dieser Problematik zu begegnen ist die Verwendung physikalisch motivierter und parametrisierter Potentiale wie beispielsweise die klassischen Kraftfelder, die in biochemischen Simulationen verwendet werden. Als Alternative zu diesem Vorgehen hat sich die Verwendung von Ersatzmodellen (engl. surrogate models) der PES etabliert, die mit Methoden des maschinellen Lernens konstruiert werden.

Ein typische Vorgehensweise die PES mittels eines solchen Ersatzmodells zu approximieren ist die Konstruktion von PESn mittels künstlicher neuronaler Netze (NN). Im Gegensatz zu klassischen Kraftfeldern ist dieser Ansatz rein mathematisch und

das Resultat ist nicht physikalisch motiviert.

Den Prozess in dem Ersatzmodelle mittels NNs auf Basis von Referenzdaten konstruiert werden bezeichnet man als *Training*. Während des Trainings werden sehr flexible Funktionen so angepasst, dass die Approximation der zugrundeliegenden PES optimal ist. Dabei wird dasjenige Modell als optimal betrachtet, das ein vorgegebenes Fehlermaß, die so genannte Kostenfunktion, minimiert. Die Kostenfunktion, die zur Konstruktion solcher NN-PESn in konventionellen Trainingsverfahren verwendet wird bewertet üblicherweise ausschließlich den Fehler, den das NN bei Energie-Vorhersagen macht. Aus diesem Grund sind die resultierenden NN-Potentiale zwar im Allgemeinen exzellent für Simulationsanwendungen für die die potentielle Energie von großem Interesse ist geeignet, allerdings ist ihre Verwendbarkeit für die Vorhersage atomarer Kräfte oder gar Hesse-Matrizen aufgrund mangelnder Genauigkeit stark eingeschränkt.

Um eine verlässliche Vorhersage von Ableitungen der PES, z.B. der Kräfte, mittels dieses Trainingsverfahrens zu erhalten, ist es im Allgemeinen nötig einen Referenzdatensatz für den Trainingsprozess zu verwenden, der den Bereich des Phasenraums, der für die spätere Simulationsanwendung relevant ist, detailiert beschreibt. Die resultierende große Anzahl an Referenzstrukturen und damit die große Anzahl quantenchemischer Berechnungen, die zur Generierung des Referenzdatensatzes benötigt werden, ist unvorteilhaft, da sie einen großen Mehraufwand bei der Konstruktion von NN-Potentialen bedeutet. Abgesehen davon erfordert eine hinreichend engmaschige Beschreibung des Phasenraums für komplexe chemische Systeme mit einer großen Anzahl von Atomen aufgrund der hohen Dimensionalität der PES komplexe Samplingverfahren.

Alternativ können NN-Potentiale, die sich zur Vorhersage von Kräften und Hesse-Matrizen eignen, konstruiert werden, indem man dem Referenzdatensatz Informationen zu den Ableitungen der PES hinzufügt und beim Training Informationen zu Energine, Kräften und, wenn nötig, auch Hesse-Matrizen berücksichtigt. Im Gegensatz zu vielen anderen Problemstellungen im Bereich des maschinellen Lernens, bei denen die Berechnung der Gradienten für die Generierung des Referenzdatensatzes mit einem hohen Rechenaufwand verbunden ist, ist der Mehraufwand für diese Problemstellung verhältnismäßig gering. Dies rührt daher, dass die quantenchemische Berechnung eines Gradienten mit einem Rechenaufwand verbunden

ist, der vergleichbar ist mit dem Aufwand für die Berechnung der Energie mit der selben quantenchemischen Methode. Das Erweitern des Referenzdatensatzen mit Informationen zum Gradienten erhöht jedoch dessen Informationsdichte drastisch. Angenommen die zu approximierende PES ist N dimensional, dann enthält der Referenzdatensatz für jede chemische Struktur $N + 1$ Informationen, die im Trainingsprozess genutzt werden können, da der Gradient ein N dimensionaler Vektor und die Energie eine skalare Größe ist.

Aufgrund der hohen Informationsdichte des Referenzdatensatzes ist es möglich akkurate NN-Potentiale auf Basis eines Referenzdatensatzes zu konstruieren, der deutlich kleiner ist als diejenigen, die bei der Anwendung des Energietrainings für eine vergleichbare Genauigkeit benötigt werden. Daher ist der Rechenaufwand für die Generierung eines solchen Referenzdatensatzes verhältnismäßig gering, da der Referenzdatensatz verhältnismäßig klein ist und Gradienten effizient berechnet werden können.

Diese Doktorarbeit beschäftigt sich mit Trainingsverfahren für NN-PESn, die exakte Approximationen der PES und der zugehörigen Kräfte und, falls gefordert, Hesse-Matrizen ermöglichen. Im Rahmen dieser Arbeit wird zunächst demonstriert, dass die Berücksichtigung der Energie-, Kraft- und Hesse-Matrix-Vorhersagen in der Kostenfunktion, eine exzellente Approximation der PES und ihrer Ableitungen durch NN-Potentiale ermöglicht. Verfahren dieser Art werden im Folgenden als *direkte Kraft-Trainingsverfahren* (engl. direct force training approach) bezeichnet. Diese NN-Potentiale, die durch direktes Trainieren der Kräfte und höheren Ableitungen konstruiert werden, können selbst für Simulationsanwendungen verwendet werden deren Vorhersagen sehr empfindlich für Fehler in der Beschreibung der Hesse-Matrix sind. Ein Beispiel für eine solche Simulationsanwendung ist die Berechnung von Reaktionsratenkonstanten mit der Instantonmethode.

In dieser Doktorarbeit wird aufgezeigt, dass eine Berechnung von Reaktionsratenkonstanten mit der Instantonmethode auf Coupled-Cluster-Niveau möglich ist, wenn ein NN-Potential der Ratenkonstantenberechnung zugrunde gelegt wird. Darüber hinaus wird dargelegt, dass es möglich ist auf Basis von Reaktionsratenkonstanten, die für eine Vielzahl von Deuterierungsmustern der Reaktion $CH_3OH + H \rightarrow CH_2OH + H_2$ mittels NN-Potentialen berechnet wurden, eine gute qualitative Erklärung für die unerwartet starke Deuterierung von Metha-

nol in vielen Regionen des interstellaren Mediums (ISM) zu finden. Daraufhin erfolgt eine detaillierte Herleitung eines im Rahmen dieser Arbeit entwickelten Algorithmus für das direkte Kraft-Trainingsverfahren. Zudem wird demonstriert, dass die Vorhersage von Kräften nicht nur durch direktes Krafttraining, sondern auch mittels eines neuen, auf einer Taylorentwicklung der potentiellen Energie basierenden Verfahrens verbessert werden kann, das deutlich weniger rechenintensiv ist. Dieses Trainingsverfahren reduziert den mit dem Krafttraining verbundenen Rechenaufwand indem es Informationen zu den atomaren Kräften lediglich indirekt berücksichtigt. Um Kraftinformationen indirekt in den Trainingsdatensatz einzubeziehen, werden zusätzliche Trainingsdaten generiert. Dazu werden die Atompositionen von Strukturen des Referenzdatensatzes mit bekannter Energie und vollständiger atomarer Kraftinformation geringfügig verändert und die Energie der resultierenden Struktur wird mittels einer Taylorentwicklung erster Ordnung der potentiellen Energie der Ursprungsstruktur berechnet.

Im Rahmen der Diskussion des taylorentwicklungsbasierten Trainingsverfahrens werden die Energie- und Kraftvorhersagen, die man mittels dieses Trainingsverfahrens erhält mit Energie- und Kraft-Vorhersagen, die durch direktes Krafttraining oder mittels des konventionellen Energietrainings erhalten werden können, verglichen. Dabei beschreibt die zugrundeliegende PES, die durch alle Trainingsverfahren approximiert werden soll, ein Cluster von sechs Wassermolekülen.

Die Untersuchung des taylorentwicklungsbasierten Trainingsverfahrens zeigt, dass dieses Verfahren im Vergleich zum konventionellen, allein auf Energieinformation basierenden Trainingsverfahren, deutlich akkuratere Vorhersagen der atomaren Kräfte ermöglicht, wenn beide Verfahren den selben Referenzdatensatz zum Training verwenden. Ein Vergleich dieses Trainingsverfahrens mit direktem Krafttraining ergibt, dass direktes Krafttraining die akkuratesten Kraftvorhersagen ermöglicht. Allerdings können etwa 50% dieser durch direktes Krafttraining erreichten Verbesserung deutlich weniger rechenintensiv durch Anwendung des taylorentwicklungsbasierten Trainingsverfahrens erreicht werden.

Alle Algorithmen, die im Rahmen dieser Doktorarbeit entwickelt wurden, sind Open Source, d.h. quelltextoffen, und im Programmpaket ænet implementiert.

Outline

Part I - Introduction

In the beginning of **Chapter 1** a general introduction to theoretical chemistry is given. In the course of this introduction the concept of a potential energy surface (PES) is introduced and its properties as well as its applications in computational chemistry are discussed. At the end of this chapter a general motivation for fitting potential energy surfaces with machine learning methods is given.

Part II - Literature Review

Chapter 2 summarizes the fundamentals of instanton theory and how it can be used to compute reaction rate constants. A special focus is put on the discussion of the properties of an approximate PES that are necessary for accurate predictions of rate constants with the instanton method.

Chapter 3 is subdivided into two main parts. The first part focuses on neural networks (NNs) in general and their mathematical background. A general definition of machine learning is given and the basic terminology to describe NNs and their training is introduced. Further, the mathematical background of NNs is summarized. Thereafter, it is explained in detail how feedforward NNs can be trained and how the generalization of the predictions to structures unknown to the NN can be tested. In this context the concept of overfitting is introduced and a method to diagnose overfitting is discussed. In the second part of this chapter details on the application of NNs to the regression of PESs are discussed. Two different kinds of machine learning models, namely structure neural networks (SNNs) and

atomic neural networks (ANNs), are introduced. The chapter concludes with a detailed outline of work flows facilitating the design of appropriate neural network architectures. This includes a discussion of possible structural descriptors and loss functions as well as promising pre-conditioning techniques that can accelerate the training process.

Part III - Novel Methods and Algorithms

In **Chapter 4** an extensive description of the direct force and Hessian training approach is made for SNNs as well as ANNs. Further, the algorithm for direct force training with ANNs, that was developed and implemented into the software package ænet by the author is outlined. It is demonstrated how a formulation of the derivatives with matrix equations allows for an efficient computation of all additional derivatives of the loss function which are not required by conventional energy training approaches. All matrix equations defining the derivatives of the forces predicted by the NN with respect to the NN's weight and bias parameters are given for a NN with two hidden layers.

In **Chapter 5** the novel, Taylor expansion based force training approach, which was developed as part of the author's work presented in this thesis, is introduced. First the general idea behind the approach, which as developed in collaboration with N. Artrith and A. Urban from Columbia University, is discussed. Therein two possible variants of this training approach, that differ in the displacement scheme employed, are explained in detail. Lastly the workflow of training NNs with this indirect force training approach is described.

Part IV - Applications and Results

In **Chapter 6** it is demonstrated that employing NN-PESs trained with the direct force and Hessian training approach for the prediction of reaction rate constants with instanton theory is highly efficient and yields outstandingly accurate rate constant predictions. In the course of this chapter it is outlined how the unexpectedly

high deuteration of methanol which was observed in many regions of the interstellar medium can be qualitatively explained on the basis of reaction rate constants for various deuteration patterns of the reaction $CH_3OH + H \rightarrow CH_2OH + H_2$. Unimolecular reaction rate constants as well as the respective kinetic isotope effects are given down to temperatures of 30 K. Moreover, bimolecular reaction rate constants obtained from a microcanonical formalism are given in order to describe low-pressure bimolecular processes appropriately.

Chapter 7 shows that employing the novel Taylor expansion based force training approach drastically improves the predictions of atomic forces by NN-potentials, especially for small reference data sets, in comparison to the predictions obtained by conventional energy training. On top of that a comparison of the energy training approach to direct force training and the Taylor expansion based force training approach is made. On the basis of this comparison the optimal fields of application for the two force training approaches are discussed.

In **Chapter 8** a final discussion of all methods and applications discussed in **Chapters 4–7** is given.

Peer-Reviewed Publications

Articles included in this Thesis

- **A. M. Cooper**, P. P. Hallmen, and J. Kästner
  Potential Energy Surface Interpolation with Neural Networks for Instanton Rate Calculations, J. Chem. Phys. 148, 094106 (2018)

- **A. M. Cooper**, and J. Kästner
  Low Temperature Kinetic Isotope Effects in $CH_3OH+H \rightarrow CH_2OH+H_2$,
  J. Phys. Chem. A 123, 9061 (2019)

- **A. M. Cooper**, J. Kästner, A. Urban and N. Artrith
  Efficient Training of Accurate Neural Network Interatomic Potentials Including Atomic Force Information: Application to Water and Transition-Metal Oxides, npj Comput. Mater. 6, 54 (2020)

Contents

Abstract vi

Zusammenfassung x

Outline xiv

List of Abbreviations xxii

I. Introduction 1

1. The Potential Energy Surface in Theoretical Chemistry 2

1.1. Properties of the Potential Energy Surface 4

1.2. Motivation for Fitting Potential Energy Surfaces 5

II. Literature Review 8

2. Fundamentals of Instanton Theory 9

3. Artificial Neural Networks Defining Machine Learning Potentials 12

3.1. Machine Learning - A Definition 12

3.2. Basic Architecture of Neural Networks 13

3.3. Mathematical Background of Neural Networks 14

3.3.1. Notation 14

3.3.2. Maximum Likelihood 14

3.3.3. Linear Regression and Maximum Likelihood 15

3.3.4. The Kernel Trick and Neural Networks 17

3.4. Training Feedforward Neural Networks 19
3.4.1. The Forward Propagation Step 20
3.4.2. The Backpropagation Step 23
3.5. Testing Neural Networks and Overfitting 26
3.6. Neural Networks for Regression of Potential Energy Surfaces of Chemical Systems 28
3.6.1. Structure Neural Networks 29
3.6.2. Atomic Neural Networks 30
3.7. Design of Neural Network Frameworks 32
3.7.1. Descriptors 32
3.7.2. Activation Functions 39
3.7.3. Loss Functions 41
3.7.4. Defining Depth of the Network and the Width of Hidden Layers 44
3.7.5. Pre-Conditioning Techniques 45

III. Novel Methods and Algorithms 48

4. Direct Force and Hessian Training 49
4.1. Direct Force and Hessian Training for Structure Neural Networks . 49
4.2. Direct Force Training for Atomic Neural Networks 50
4.3. Notation 51
4.4. Expression for the Energy and its Gradient obtained from an Atomic Neural Network 53
4.5. The Loss Function and its Derivative with Respect to the Weight Parameters 54
4.6. Computing $\partial F_{c,\varphi}^{\mathrm{NN}} / \partial {}_t^l w_b^a$ for Atomic Neural Networks 56
4.7. Computing $\partial F_{c,\varphi}^{\mathrm{NN}} / \partial {}_t^l w_b^a$ for an Atomic Neural Network with Two Hidden Layers 58
4.7.1. Computing $\partial F_{c,\varphi}^{\mathrm{NN}} / \partial {}_t^l w_b^a$ Based on Matrix Equations 59
4.7.2. Computing $\partial F_{c,\varphi}^{\mathrm{NN}} / \partial {}_t^1 w_b^a$ 65

5. Taylor Expansion Based Force Training Approach **71**
5.1. Displacements along Cartesian Coordinate Axes 72
5.2. Random Displacements 73
5.3. Training Neural Networks with the Taylor Expansion Approach . . 73

IV. Applications and Results **76**

6. Reaction Rate Constants for $CH_3OH + H \rightarrow CH_2OH + H_2$ **77**
6.1. Introduction 77
6.2. Computational Details 80
6.2.1. Neural Network Potentials 80
6.2.2. Reaction Rate Constants 87
6.3. Results and Discussion 88
6.3.1. Bimolecular Canonical Reaction Rate Constants from Average Neural Network Potentials 88
6.3.2. Reaction Rate Constants and Kinetic Isotope Effects 94
6.3.3. Summary 105

7. Predicting Correct Forces for Small Water Clusters **106**
7.1. Introduction 106
7.2. Computational Details 108
7.2.1. Error Measures for Atomic Forces Predicted by Neural Networks 108
7.2.2. Neural Network Potentials 108
7.3. Results 112
7.3.1. Force Predictions from Neural Networks Trained with the Energy Training Approach 112
7.3.2. Force Predictions from Neural Networks Trained with the Taylor Expansion Based Approach 115
7.3.3. Force Predictions from Neural Networks Trained with the Direct Force Training Approach 120
7.4. Summary and Discussion 127

8. Conclusion **130**

V. Appendix 133

A. Direct Force and Hessian Training for Structure Neural Networks **134**

A.1. Gradient and Hessian of the Energy with Respect to the Input Coordinates 134

A.2. Derivatives of E, $\mathbf{g}$, $\mathbf{H}$ with Respect to Weights and Biases 135

Bibliography **144**

List of Abbreviations

ADF Atom centered angular distribution function

ANN Atomic neural networks

CC Coupled cluster theory

DFT Density functional theory

i.i.d. independent and identically distributed, mathematical property of random numbers

ISM Interstellar medium

KIE Kinetic isotope effect

L-BFGS Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm

MAE Mean absolute error

MD Molecular dynamics

MPI Message Passing Interface, communications protocol for parallel programs

MSE Mean square error

NN Artificial neural network

NN-PES Surrogate model of a PES based on a NN

NN-potential Surrogate model of a PES based on a NN

PES Potential energy surface

RDF Atom centered radial distribution function

ReLU Rectified linear unit

RMSE Root mean square error

RS Reactant state

SNN Structure NN

TS Transition structure, commonly imprecisely referred to as transition state

ZPE Zero-point vibrational energy

Part I.

Introduction

1. The Potential Energy Surface in Theoretical Chemistry

The aim of theoretical chemistry is to provide theoretical methods to explain chemical processes and experimental observations. However, theoretical chemistry is not limited to explaining experimental findings *a posteriori*. Its methods can, for example, be used to design novel compounds with a defined set of chemical and physical properties or to study chemical processes, like astrochemical processes, for which measurements are highly complicated and error-prone.

The methods of theoretical chemistry are very diverse. While some methods are based on the description of chemical systems by analytical mathematical theory as well theoretical physics alone, other methods make use of additional empirical parameters fitted to experimental results.

Computer simulations have become essential tools in theoretical chemistry since they enable solving problems which are due to their complexity unsolvable without computers.

A realistic description of many chemical properties and processes, requires a quantum mechanical description. Further, purely quantum mechanical effects like atom tunneling can only be simulated by applying quantum theory. The basis of non-relativistic quantum chemical simulations is the *Schrödinger equation*. Theoretically any chemical property can be predicted from the analytical solution of the Schrödinger equation. Unfortunately, analytical solutions for this partial differential equation aren't readily available for most chemical systems. Today analytical solutions are only available for very simple chemical systems like atoms with a single electron such as the hydrogen atom or hydrogen-like ions like He^{+} or Li^{2+}.

In order to find at least approximate solutions of the Schrödinger equation, approximations are introduced. One fundamental approximation is that the motion of atomic nuclei and electrons can be treated separately since the time scales of their motion differ strongly due to the great difference in the mass of nuclei and electrons. This is known as the *Born–Oppenheimer approximation*. It can be used to rewrite the Schrödinger equation into two separate differential equations, one describing the electrons and one describing the nuclei. This enables solving the electronic Schrödinger equation first, which requires the positions of the nuclei as fixed parameters and then solving the Schrödinger equation for nuclear motion.

By applying the Born-Oppenheimer approximation and solving the electronic Schrödinger equation for varying nuclear positions one can define a mapping between the positions of the nuclei and the corresponding electronic energy. This mapping defines the effective potential in which the nuclei move and is therefore called potential energy surface (PES). For each electronic state of a chemical system such a PES can be defined, however, in the following the term potential energy surface will be used synonymously with potential energy surface of the electronic ground state.

The decoupling of nucleic and electronic motion drastically accelerates quantum chemical simulations. Unfortunately, solving the electronic Schrödinger equation even approximately can be very time consuming. Therefore, the time required for finding this approximate solution determines the problems that can be solved with computer simulations. Thus, the development of efficient methods for approximating solutions of this differential equation is one of the main focuses of computational chemistry.

Depending on the scientific problem, it is also possible to describe the motion of nuclei by classical or semi-classical equations of motion, which in general reduces the computational effort of the simulation. If, for example, the movement of a number of particles or molecules, like the folding or unfolding process of a protein, is to be studied, it is sufficient to describe the atomic motion fully classically by Newton's laws of motion. This is done in several simulation methods like molecular dynamics or in Monte Carlo simulations. These simulations, however, require information on how the energy of the system under study changes with the positions of the nuclei, as this determines the interactions and, thus, the movement of the atoms. In contrast

to standard quantum chemical simulations where this information is obtained by solving the electronic Schrödinger equation during the simulation, a molecular dynamics simulation commonly obtains this information from an empirical *force field*. These force fields are tailored to describe the potential energy of the chemical system well and are often constructed on the basis of data from quantum chemical computations as well as experimental findings. Thus, force fields themselves are not purely classical and implicitly describe quantum mechanical effects since they approximate the solution of the electronic Schrödinger equation, i.e. the PES.

Unfortunately, standard force fields require a fixed topology of the molecular structures simulated, which implies that it is impossible to describe bond formation or bond breaking with them. There is a class of force fields, the so called reactive force fields, that is capable of describing bond breaking and formation [1, 2]. However, usually reactive force fields, like most force fields, cannot be applied directly to a system of interest but have to be reparametrized to ensure an accurate description of the system.

Alternatively, it is also possible to employ a fitted PES in order to define a potential that allows for the description of bond formation and bond breaking. These fitted PESs can be constructed such that their transferability to other systems is straight forward and doesn't require a reparametrization. One example for easily transferable potentials are NN-PESs constructed with atomic neural networks, see section 3.6.2.

1.1. Properties of the Potential Energy Surface

The PES defines many physical and chemical properties which are of interest in computational chemistry. In the following some important examples for such properties defined are given. Minima on the PES correspond to equilibrium structures and saddle points of first order correspond to transition structures. A chemical reaction is determined by a set of equilibrium structures that define the reactants and products and the corresponding transition state structures. Thus, the potential energy surface determines the reaction kinetics, as the energy differences between reactants and transition states determines if and how fast a chemical reaction can take place. Further, the gradient of the PES with respect to the atomic positions

defines the forces acting on the atoms for a given atomic configuration. The information on the PES and its gradient with respect to the atomic positions is used for geometry optimizations, which are performed to determine equilibrium and transition structures. On top of that the atomic forces can be used to simulate the motion of the nuclei and the potential energy corresponding to a given spatial configuration of atoms in atomistic simulations like molecular dynamics or Monte Carlo simulations.

1.2. Motivation for Fitting Potential Energy Surfaces

The use of a fitted PES can greatly accelerate simulations. Evaluating such a PES for a given atomic configuration is significantly faster than solving the electronic Schrödinger equation in quantum chemical simulations. Therefore, once a fitted PES is available, quantum chemical simulations can be accelerated drastically by employing the fitted potential instead of solving the electronic Schrödinger equation during the simulation run. Given a potential surface is available, this enables the study of chemical systems that are too large or problems that are too complex for conventional simulation approaches. Normally defining the PES by a fit of energies obtained from quantum chemical simulations is more favorable than employing a force field as on the one hand it allows for the description of bond-formation and bond breaking and on the other hand the description of the potential energy is more precise.

However, if a PES is not already available for the chemical system of interest, fitting a PES introduces a large overhead to the simulation process. Obtaining a functional description of the PES requires solving the electronic Schrödinger equation for a dense grid of configurations and interpolating the energies between the grid points. The PES is for non-linear molecules a $3N - 6$ dimensional hypersurface with N being the number of nuclei. Due to the PES being so high-dimensional, a thorough description of a PES requires the solution of the electronic Schrödinger equation for a vast number of atomic configurations in order to sample the configurational space reasonably well. Therefore, the interpolation of a PES is computationally very expensive and the interpolation of global potential energy surfaces is only possible for elemental materials [3–11] and small molecules [12–18]. Even a local

description of a PES that only describes a confined area of configurational space requires significant computational effort, but can be done for chemical systems with hundreds of atoms [7]. A second disadvantage of classical mathematical methods for interpolating PESs is that many of the conventional methods require a lot of human effort and often do not allow for an easy transfer of the fitted model to similar systems.

The prospect of accelerating and improving the predictive power of chemical simulations by employing fitted PESs without standard mathematical fitting procedures inspired the approximation of PESs with machine leaning methods.

The idea is to use a machine learning algorithm to learn the potential energy surface from a *training data set* or *training set*, i.e. a set of chemical structures for which the potential energy was obtained by a quantum chemical simulation. One advantage of machine learning a PES is that the functions used for fitting are very flexible, which allows for a highly precise and, if required, complex description of the PES based on the training data. A second advantage is that a minimal human effort has to be put into the fitting process, since it is sufficient to define the general functional form used for the fit and a small number of parameters, like the optimizer used for the parameter optimization. The two most common classes of machine learning methods used to approximate PESs are kernel methods and artificial neural networks (NNs). These methods learn to approximate the PES by performing a regression of the training data. Detailed information on how NNs can be used to approximate PESs is given in chapter 3.

Unfortunately, even though energy predictions by NN-potentials trained with conventional NN training procedures are reliable, such NN-PESs only have limited applicability to simulations which require accurate force predictions. This is due to the fact that conventional training approaches measure the NNs performance on the basis of predictions of structural energies alone. Therefore, these training approaches will be referred to as *energy training approaches* in the following. In order to ensure reliable force predictions, it is usually required to sample the phase-space extensively and thereby generate a huge amount of reference data. Therefore, generating these reference data sets introduces a significant computational overhead to the NN training process, as for every reference structure a computationally expensive quantum chemical simulation has to be performed. The prediction of

Hessians and higher derivatives of the PES with respect to spatial coordinates require even more extensive sampling of the phase-space, which renders reliable predictions of these higher derivatives of the PES for most systems impossible.

To overcome these restrictions, it is usually beneficial to include force information and, if required, information on higher derivatives, like Hessians, into the training process to ensure reliable predictions. Employing these so called *force training* approaches also allows the use of smaller reference data sets in the training process in order to obtain reliable information on atomic forces and higher derivatives of the PES. This reduces the computational overhead of the NN training process drastically, however, force training approaches are in general computationally more expensive than the conventional energy training approaches.

Instead of incorporating information on the derivatives of the PES directly into the training process it is also possible to include force information indirectly. This idea was suggested for conventional molecular force filed optimization by Vlcek *et al.* [19], but was applied to NN-potentials for the first time in the work presented in this thesis. In this method the reference data set size is increased by displacing atoms in known reference structures and computing the corresponding energy not by quantum mechanical methods but on the basis of a first order Taylor expansion of the original structure's energy. This process implicitly includes force information into the training process by transforming force information to energy information.

The core research question discussed in this thesis is if direct or indirect force training approaches can be used to construct NN-potentials that can be employed in simulations that are highly sensitive to errors in atomic forces or Hessians. The second focus point of this thesis is to investigate the novel Taylor expansion based force training approach and compare it to conventional force training as well as energy training.

Part II.

Literature Review

2. Fundamentals of Instanton Theory

The quantum mechanical tunnel effect influences reaction rate constants. It allows for chemical reactions even if the system under study does not have the energy to overcome the respective barrier and thus, a reaction is impossible in classical transition state theory. While the influence of tunneling on the reaction rate constant is rather small at high temperatures, as reaction barriers are overcome by the system's thermal energy, it dominates the reaction rate constants at low temperatures. Therefore, it is vital to consider the tunnel effect in rate constant calculations for astrochemical reactions that take place in regions of the interstellar medium where temperatures are very low.

Instanton theory [20–27] is a semi-classical theory, that allows for the computation of reaction rate constants including the quantum mechanical tunnel effect. In instanton theory the Feynman path integral formulation [28] is employed. In order to compute reaction rate constants, first the instanton, i.e. the tunneling path with the highest statistical weight, is located by discretizing the Feynman path and finding a saddle point of first order in the space of discretized Feynman paths [29, 30]. The instanton can be determined even for high-dimensional systems highly efficiently by a modified Newton-Raphson method[29, 31], which converges quadratically to the solution. In order to compute the reaction rate constant, fluctuations around the instanton path are taken into account up to quadratic order. The reaction rate constant k_{inst} is given by [29, 31]:

$$k_{\mathrm{inst}} = \sqrt{\frac{S_0}{2\pi\hbar}}\sqrt{\frac{P}{\beta\hbar}}\frac{\prod_{l=N_0+1}^{NP}\sqrt{\lambda_l^{\mathrm{RS}}}}{\prod_{l=N_0+2}^{NP}\sqrt{|\lambda_l^{\mathrm{inst}}|}}\exp\left(-\frac{S_{\mathrm{E}}}{\hbar}\right). \tag{2.1}$$

Here $\beta = 1/k_{\mathrm{B}}T$ is the inverse temperature, with k_{B} being the Boltzmann constant. $\hbar$ is the reduced Planck's constant. N is the number of degrees of freedom and N_0 represents the number of translational and rotational degrees of freedom. P is the number of discretization points, also called images, of the Feynman path. Furthermore, S_{E} is the Euclidean action of the instanton path and S_0 is the corresponding shortened action [29]. $\lambda_l^{\mathrm{inst}}$ and λ_l^{RS} are the eigenvalues of the matrix of all second derivatives of the Euclidean action with respect to the coordinates of all images for the reactant state ($\lambda_l^R S$) and instanton ($\lambda_l^{\mathrm{inst}}$). Thereby, the second derivative matrix is given by

$$\frac{\partial^2 S_{\mathrm{E}}}{\partial y_i^c \partial y_j^d} = \frac{P}{\beta\hbar}\delta_{c,d}(2\delta_{i,j} - \delta_{i-1,j} - \delta_{i,j-1}) + \frac{\beta\hbar}{P}\delta_{i,j}\frac{\partial^2 E}{\partial y_i^c \partial y_j^d}, \tag{2.2}$$

where y_i^c stands for the mass-weighted coordinate component c of image i.

From the expression in equation (2.2) it can be seen that the second derivatives, i.e. the Hessians, of the potential energy with respect to all coordinates of each image i, $\nabla_i \nabla_i E = \partial^2 E/(\partial y_i^c \partial y_i^d)$, are required to compute k_{inst}. Unfortunately, computing these Hessian matrices on-the-fly during a simulation by quantum chemical methods is very time-consuming. Therefore, computing reaction rate constants with instanton theory is limited to rather small chemical systems and restricted to employing a computationally efficient quantum chemical method for the determination of the Hessian. Therefore, usually density-functional theory (DFT) is employed to keep the computational effort at bay.

The endeavor to describe the PES, and thus the Hessian, more accurately by employing computationally more expensive quantum chemical methods, like coupled cluster methods, as well as the aspiration to enable the application of instanton theory to large chemical systems inspired the use of NN-potentials. Employing NN-PESs can significantly accelerate instanton path optimizations and instanton reaction rate constant computations. This is due to the fact that Hessians of the PES don't have to be computed by quantum chemical methods during the computer simulation but can be determined by evaluating the NN-PES.

The time- and computational effort for evaluating a neural network is negligible in comparison to the effort of a standard quantum mechanical simulation of the electronic structure. In order to make reliable predictions for reaction rate constants

on the basis of a NN-PES, it is crucial that the NN-potential describes the Hessians accurately and as a smooth function of all coordinates.

How NN-PESs can be used to perform reliable instanton reaction rate computations on a coupled cluster level of theory is one of the main research questions studied in this thesis. A detailed discussion of this research question is given in chapter 6 .

An important restriction of instanton theory is that it is only applicable for the prediction of reaction rate constants at temperatures below the *crossover temperature* $T_c = \hbar\omega_{\mathrm{TS}}/2\pi k_B$, where ω_{TS} is the absolute value of the imaginary frequency at the transition structure. This restriction is especially of interest if several isotopologues are to be studied, since the crossover temperature is mass-dependent due to ω_{TS} being mass-dependent.

Further, instanton theory is intrinsically overestimating reaction rate constants at temperatures lower than but close to the crossover temperature [26]. However, methods to correct for this non-physical effect exist [32, 33] and were applied in the work presented in this thesis.

3. Artificial Neural Networks Defining Machine Learning Potentials

In this chapter a detailed review of neural networks (NNs) and how they can be employed as surrogate models for potential energy surfaces (PESs) is given.

Section 3.3, follows the line of thought of the book *Deep Learning* by Goodfellow, Bengio and Courville [34] and the description of atomic neural networks in section 3.6.2 summarizes the articles [3] and [35]. In both cases I summarized the explanations given and amended them by further aspects to facilitate comprehensibility.

3.1. Machine Learning - A Definition

In 1997 Mitchell gave in his book *Machine Learning* [36] a broad definition of machine learning:

Definition 1. *A computer program is said to learn from Experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E.*

Mitchell further states that a machine learning problem is well posed if and only if the three features E, T and P are well-defined.

A mathematically rigorous definition of these three features is non-trivial and does not facilitate the understanding of how machine learning is to be understood in the context of this thesis. To demonstrate that the machine learning problem

investigated in this thesis, i.e. regression of high dimensional functions, is well-posed, the meaning of E, T and P in this context will be discussed in the following.

In this thesis NNs are used to interpolate the PES for a given chemical system. Thus, the the task T is to approximate the PES, i.e. the function m^* that maps from the coordinates $\boldsymbol{x}$ of a chemical structure to the corresponding ground state energy $y \coloneqq m(\boldsymbol{x})$. The experience E is given by a data set of chemical structures for which the potential energy and its derivatives of interest are known. The performance measure P is given by the loss function of the NN. Often the root mean square error of the NN's predictions is used to define the loss function, but other performance measures are possible, see section 3.7.3.

3.2. Basic Architecture of Neural Networks

A NN is built up from simple compute units, the *neurons*. A schematic representation of a neuron is given in figure 3.3. Neurons that process data at the same time are combined into so called *layers*. Each layer is, in general, made up of several neurons, where the number of neurons in the layer defines its *width*.

A NN is then constructed by interconnecting multiple layers with each other and assigning parameters, the so called *weights*, to each connection. In general neural networks are structured as follows: In the first layer, the *input layer*, the reference data is represented. The following layers are called *hidden layers*, since the output values of the neurons, the so called *activations*, in these layers are *a priori* not known, i.e. hidden. The number of these hidden layers is a model parameter, that can be adjusted to the problem under study and to the computational resources available. The final layer is the *output layer* that contains the prediction that the NN makes for the given input. The total number of layers in an artificial NN defines the *depth* of the network. Figure 3.2 a) shows a feedforward NN with 2 hidden layers.

Depending on the way in which the layers are interconnected different kinds of NNs are obtained. In general there are no restrictions to the ways in which two layers can be interconnected. If all neurons from one layer are connected to all neurons in the subsequent layer then a NN is called *fully connected*. NNs that are defined by a mapping in which input information is processed unidirectional

from the input to the output and, thus, there are no feedback loops, are called *feedforward neural networks*. This kind of NNs is the most commonly used one for regression tasks and will therefore be discussed in detail in this chapter.

3.3. Mathematical Background of Neural Networks

In this section the mathematical background required to understand NNs and their training is presented. First some basic notation is introduced. Subsequently the concept of Maximum likelihood is explained. Then it is discussed how the principle of maximum likelihood for linear regression yields the mean square error as optimal performance measure P. Finally it is outlined how to extend the methods discussed to nonlinear models by applying the kernel trick and how the kernel trick can be employed for training NNs.

3.3.1. Notation

In the remainder of this thesis the following basic notation will be used: Vectors are given as lower-case bold and italic symbols $\boldsymbol{v}$. Matrices are given as upper case bold and italic symbols $\boldsymbol{A}$. Random variables are given as upright bold symbols $\mathbf{x}$ and the i-th value of a random variable shall be given by $\boldsymbol{x}^{(i)}$. Further, $||\cdot||$ indicates the L^2 norm of a vector.

3.3.2. Maximum Likelihood

Let $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ be a set of m independent training examples drawn from the probability distribution $p_{\text{data}}(\mathbf{x})$. Let further $p_{\text{model}}(\mathbf{x}, \boldsymbol{\theta})$ be a parametric family of probability distributions over the same space as $p_{\text{data}}(\mathbf{x})$, indexed by $\boldsymbol{\theta}$. This means that $p_{\text{model}}(\boldsymbol{x}, \boldsymbol{\theta})$ maps $\boldsymbol{x}$ to a real number that approximates $p_{\text{data}}(\boldsymbol{x})$.

With these definitions the maximum likelihood estimator for $\boldsymbol{\theta}$ is given by:

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \tag{3.1}$$

The product in equation (3.1) can cause numerical instabilities arising from under-

flow errors caused by the repeated multiplication of small numbers. Therefore, it is numerically more stable to solve the following equivalent optimization problem, which is obtained by taking the logarithm:

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log(p_{\mathrm{model}}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta})) \tag{3.2}$$

The maximum likelihood estimator can be generalized such that it estimates the conditional probability $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$.

For $\mathbf{X}$ representing all inputs and $\mathbf{Y}$ being the corresponding target values, the conditional maximum likelihood estimator is given by

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log(P(\boldsymbol{y}^{(i)}|x^{(i)};\boldsymbol{\theta})) \tag{3.3}$$

if the examples are independent and identically distributed (i.i.d.). This conditional maximum likelihood estimator is the basis of most supervised machine learning algorithms.

3.3.3. Linear Regression and Maximum Likelihood

Performing linear regression with a computer can be seen as a simple machine learning procedure. From one perspective the aim of linear regression is to build a model that maps a vector $\boldsymbol{x} \in \mathbb{R}^n$ to the scalar target value $y \in \mathbb{R}$, where the target value is a linear function of the input. We define $\hat{y}$ as the value that linear regression predicts for $\boldsymbol{x}$:

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x}, \tag{3.4}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of model parameters, the so called *weights*, used for the regression. Thus, $\hat{y}$ is an approximation of the target value y.

The task T of this machine learning problem is to predict $\hat{y} \approx y$ for $\boldsymbol{x}$ by employing the model $\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$. The experience E is defined by the observation of the *training set* $(\boldsymbol{X}^{(\mathrm{train})}, \boldsymbol{y}^{(\mathrm{train})})$. Here $\boldsymbol{X}^{(\mathrm{train})}$ is a matrix that contains all training examples $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(\mathrm{train})}$ is the vector of all respective target values $y^{(i)}$. Employing the mean square error of the training set ($\mathrm{MSE}_{\mathrm{train}}$) as the performance measure P

a well-posed machine learning problem is obtained. This machine learning problem can be solved by minimizing $\mathrm{MSE}_{\mathrm{train}}$ with respect to the weights $\boldsymbol{w}$. A closed form for the optimal weight parameters is given by the normal equations. From

$$\nabla_w \mathrm{MSE}_{\mathrm{train}} = 0 \tag{3.5}$$

the deduction of the optimal choice of $\boldsymbol{w}$, that minimizes $\mathrm{MSE}_{\mathrm{train}}$, is straight forward and yields the normal equations:

$$\boldsymbol{w} = (\mathbf{X}^{(\mathrm{train})^T}\mathbf{X}^{(\mathrm{train})})^{-1}\mathbf{X}^{(\mathrm{train})^T}\boldsymbol{y}^{(\mathrm{train})}. \tag{3.6}$$

The evaluation of equation (3.6) defines linear regression, a simple machine learning algorithm.

Another perspective on linear regression is to view it as a maximum likelihood procedure. From this standpoint linear regression is a model that produces a conditional distribution $p(y|\boldsymbol{x})$. The aim of the machine learning procedure is to fit $p(y|\boldsymbol{x})$ to all values y that are possible for the input $\boldsymbol{x}$. To derive the same linear algorithm as in the first part of this section, the posterior is defined by a normal distribution as $p(y|\boldsymbol{x}) = \mathcal{N}(y, \hat{y}(\boldsymbol{x}; \boldsymbol{w}), \sigma^2)$, where the variance σ^2 is assumed to be a user-defined constant and $\hat{y}(\boldsymbol{x}; \boldsymbol{w})$ is the normal distribution's mean.

If the m training examples in $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ are i.i.d. the conditional log-likelihood defined in equation (3.3) is given by:

$$\boldsymbol{W}_{\mathrm{ML}} = \sum_{i=1}^{m} \log(p(y^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{W})) \tag{3.7}$$

$$= -m\log(\sigma) - \frac{m}{2}\log(2\pi) - \sum_{i=1}^{m} \frac{||\hat{y}^{(i)} - y^{(i)}||^2}{2\sigma^2}. \tag{3.8}$$

From equation (3.8) it can be directly obtained that maximizing the conditional log-likelihood with respect to the weights $\boldsymbol{w}$ is equivalent to minimizing the mean square error of the training set given by

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{m}\sum_{i=1}^{m} ||\hat{y}^{(i)} - y^{(i)}||^2. \tag{3.9}$$

Thus, the optimal parameters $\boldsymbol{w}$ are obtained by maximizing the conditional log-likelihood, which is equivalent to minimizing $\text{MSE}_{\text{train}}$.

3.3.4. The Kernel Trick and Neural Networks

The learning algorithms discussed in this thesis are based on estimating $p(y|\boldsymbol{x})$ by employing a maximum likelihood estimation of the parameters $\boldsymbol{w}$ for $p(y|\boldsymbol{x};\boldsymbol{w})$, a parametric family of probability distributions.

As seen in section 3.3.3 linear regression corresponds to the parametric family

$$p(y|\boldsymbol{x} : \boldsymbol{w}) = \mathcal{N}(y; \boldsymbol{w}^T\boldsymbol{x}, \sigma^2). \tag{3.10}$$

In order to apply the methods introduced so far to the definition of nonlinear machine learning methods, the formalism has to be extended. One way to obtain nonlinear models is to employ the so called *kernel trick*. The kernel trick can only be applied if the machine learning algorithm can be rewritten such that it is exclusively composed of inner products between training examples.

Standard linear regression and even its extended form $\hat{y} = \boldsymbol{w}^T\boldsymbol{x} + b$ can be rewritten in such a way:

$$\hat{y} = \boldsymbol{w}^T\boldsymbol{x} + b = b + \sum_{i=1}^{m} \alpha_i \boldsymbol{x}^T\boldsymbol{x}^{(i)}, \tag{3.11}$$

where $\boldsymbol{\alpha}$ is a vector of the parameters α_i and $\boldsymbol{x}^{(i)}$ is the i-th training example.

In this formulation of linear regression $\boldsymbol{x}$ can be replaced by a *feature function* $\phi(\boldsymbol{x})$. The inner product in equation (3.11) is then given by a *kernel* $k(\boldsymbol{x}, \boldsymbol{x}^{(i)}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}^{(i)})$. The actual inner product that is used to define the kernel depends on the *feature space*. For example in infinite dimensional feature spaces inner products that are defined by integration have to be used instead of standard inner products for vectors in $\mathbb{R}^n$.

An intuitive interpretation of ϕ is that it is a function which defines a set of features that describe $\boldsymbol{x}$ and thereby provides a new representation of this training example.

In this formulation the machine learning algorithm can make predictions $\hat{y} = m(\boldsymbol{x})$

based on the following equation:

$$m(\boldsymbol{x}) = b + \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}^{(i)}). \tag{3.12}$$

This equation defines a machine learning model that is in general nonlinear in $\boldsymbol{x}$. However, the model defined by $m(\boldsymbol{x})$ is still linear in $\phi(\boldsymbol{x})$ and $\boldsymbol{\alpha}$. Thus, this nonlinear machine learning model is equivalent to first applying $\phi(\cdot)$ to all inputs $\boldsymbol{x}$ in a preprocessing step and subsequently learning a linear model in the space defined by $\phi(\cdot)$.

Under the assumption that ϕ is fixed, the kernel trick allows for the efficient training of nonlinear machine learning with convex optimizations techniques.

Having discussed the key ideas behind the kernel trick it is still unclear how one actually should define the feature function $\phi(\boldsymbol{x})$.

In principle ϕ can be manually chosen and fine tuned, which is a sophisticated and time consuming task. Another option is to choose a generic ϕ, which does not require a lot of human interaction, however often results in models that generalize poorly to examples not included in the training set.

The strategy used in NNs is to learn ϕ. The model that is used in deep learning is $y = m(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) =: \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$. This implies that we have two kinds of parameters: $\boldsymbol{w}$ that map from $\phi(\boldsymbol{x})$ to the target values y and $\boldsymbol{\theta}$ that are used to learn the feature function ϕ from a broad class of functions. This model represents a feedforward NN with one hidden layer, where $\boldsymbol{x}$ is the input to the NN and ϕ defines the hidden layer.

The disadvantage of learning ϕ is that the convexity of the machine learning problem is lost. Therefore, the efficient convergence of optimizing algorithms that are specially made for convex problems is not guaranteed anymore. Nevertheless, the benefits outweigh the harms since this approach to machine learning allows for the use of a very general family of feature functions $\phi(\boldsymbol{x}; \boldsymbol{\theta})$, which has a large capacity. This ansatz further enables the user to make use of prior knowledge on the properties that the resulting machine learning model must have by choosing a family of functions that has these respective properties.

Now that the mathematical background of feedforward NNs with one hidden

layer has been discussed in detail, the evaluation of feedforward NNs with an arbitrary number of hidden layers and their training will be discussed.

The following sections focus less on a statistical view on NNs but more on a description of NNs from a computer-science perspective, since this view facilitates the understanding of the general functional principles of NNs.

3.4. Training Feedforward Neural Networks

NNs are trained by employing an iterative optimization scheme. A flow chart describing the training process is given in figure 3.1.

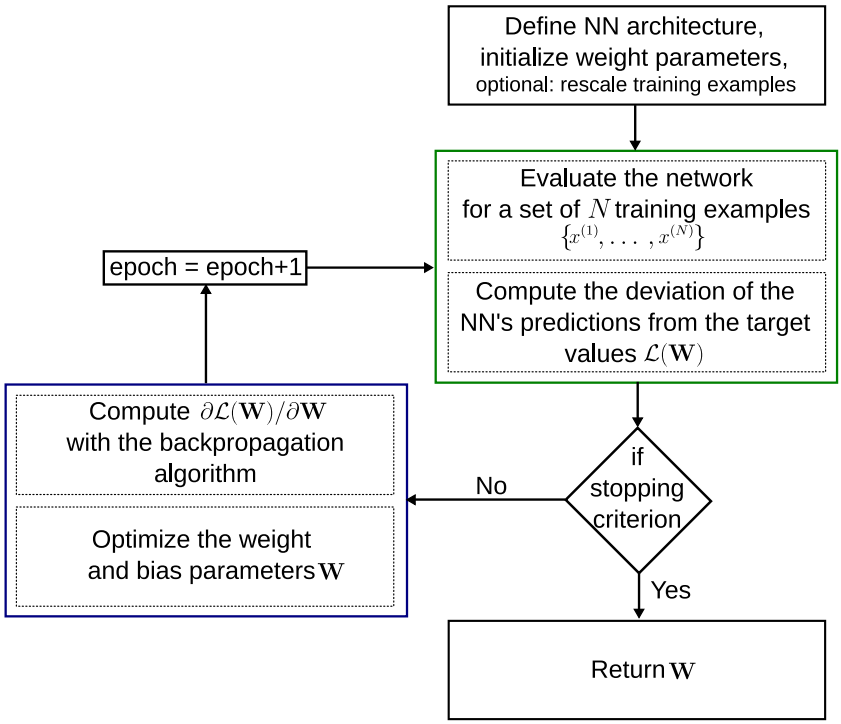


Figure 3.1.: **Flowchart of the training process.** The forward propagation step is shown in green, the backpropagation step is shown in blue.

The iterative optimization is usually done in two phases: during the first phase, the *forward propagation step*, the NN is evaluated for a set of training examples and the error of the NN's predictions, the so called *loss*, is calculated. In the second phase, the *backpropagation step*, the weight parameters are optimized such that

the loss is expected to decrease in the next iteration. The execution of a forward propagation and a subsequent backward propagation step together defines one full iteration of the training process, which is commonly called a *training epoch*. The NN is trained for several epochs until either a convergence criterion or a predefined number of training epochs is reached. Possible convergence criteria are that the NN's loss falls below a certain threshold $\mathcal{L}_{\text{thresh}}$ or that the loss has not changed for a given number of epochs, which suggests that further optimization of the weights will not improve the NN's predictions anymore.

The following sections give a deeper insight into the two phases of a training epoch. In section 3.4.1 the forward propagation step is described, the backpropagation step is explained in section 3.4.2.

3.4.1. The Forward Propagation Step

Assume that the quantum mechanical PES is given by $y = m^*(\boldsymbol{x})$. The feedforward neural network, which is employed as a surrogate model of the PES, defines a mapping $\hat{y}_{\text{NN}} = m(\boldsymbol{x}, \boldsymbol{W})$. The NN-PES maps a chemical structure $\boldsymbol{x}$ to an approximation of the corresponding potential energy $\hat{y}_{\text{NN}}$. The weight parameters or *weights*, which are assumed to be collected in the matrix $\boldsymbol{W}$, are optimized during training and are chosen such that $m(\boldsymbol{x}, \boldsymbol{W})$ is a good approximation to $m^*(\boldsymbol{x})$. This section explains how this mapping $m(\boldsymbol{x}, \boldsymbol{W})$ is done by applying *forward propagation*.

The mapping $m(\boldsymbol{x}; \boldsymbol{W})$ defined by a feedforward NN can be rewritten as a composite function. Assume the depth of the NN is L, then the mapping can be written by a composition of $L - 1$ functions:

$$m(\boldsymbol{x}, \boldsymbol{W}) = (m^{(L-1)}(m^{(L-2)}(\dots m^{(1)}(\boldsymbol{x}; {}^{1}\boldsymbol{W}); {}^{L-2}\boldsymbol{W}); {}^{L-1}\boldsymbol{W}). \tag{3.13}$$

where $m^{(l)}(\cdot, {}^{l}\boldsymbol{W})$, $l \in \{1, 2, \dots, L-1\}$, defines the *activations*, i.e. the output, of layer $l + 1$, and ${}^{l}\boldsymbol{W} \in \mathbb{R}^{D^{(l-1)} \times D^{(l)}}$ are the weight parameters of the connections pointing to layer $l + 1$, which are learned from the training set $\mathbb{X}$. Here $D^{(l)}$ defines the width of layer l.

The mapping $m(\boldsymbol{x}, \boldsymbol{W})$ can be associated with a directed acyclic graph which describes the composition of the functions $m^{(l)}$. In figure 3.2 a) the graph for a

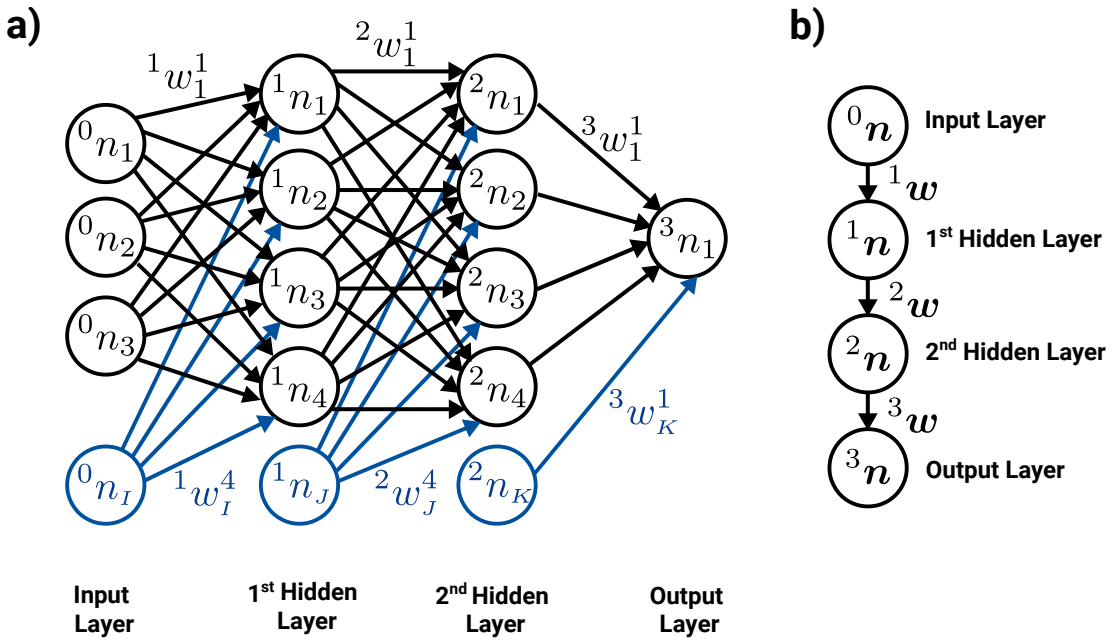


Figure 3.2.: **Schematic representations of a feedforward neural network.** **a)** shows a detailed representation of a feedforward neural network with 2 hidden layers where each neuron is shown as one node in the graph. Bias neurons and their corresponding weights are shown in blue. **b)** shows a simplified representation of the same NN where each node represents a whole layer.

fully connected feedforward NN with two hidden layers is given. Every node of this graph represents a neuron and the weights on the edges of the graph are defined by the weight parameters of the neural network. In figure 3.2 b) a more compact representation of the same NN is given by a graph where every node represents the activation of a layer. This representation is particularly useful for medium sized and large NNs for which a detailed graph showing all neurons would be confusing.

In order to define the functions $m^{(l)}$, first the evaluation of a single neuron will be described in detail. The following description is valid for neurons in the hidden layers and the output layer. The value of neurons in the input layer is defined by the inputs $\boldsymbol{x}$ to the NN alone.

A schematic representation of a neuron and its functional principle is given in figure 3.3. Since the feedforward NN is fully connected each neuron k in layer $l+1$ is connected to all neurons in layer l. Neuron k in layer $l+1$ obtains from each of

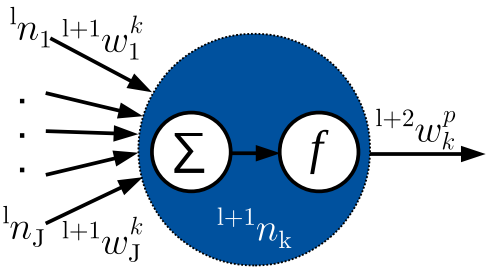


Figure 3.3.: **Schematic representation of a neuron.**

the neurons in layer l an input value ${}^{l}n_\alpha$. These input values are multiplied by the weight ${}^{l}w_\alpha^k$ of the corresponding edge of the graph and subsequently the sum of all weighted inputs, $\sum_\alpha {}^{l}n_\alpha \, {}^{l}w_\alpha^k$, is calculated. This aggregated sum is also called *preactivation*.

The output of neuron k, ${}^{l+1}n_k$, is obtained in the *activation* step by the evaluation of a so called *activation function* or *transfer function* ${}^{l+1}f(\cdot)$ for the respective layer: ${}^{l+1}n_k := {}^{l+1}f(\sum_\alpha {}^{l}n_\alpha \, {}^{l}w_\alpha^k)$. The activation function is a mathematical function that adds non-linearity to the NN, i.e. it has the role of a feature function ϕ as seen in section 3.3.4. A detailed list of possible activation functions is given in section 3.7.2. For all neuron under consideration that are not in the output layer, the neuron's activation ${}^{l+1}n_k$ is sent to all neurons in layer $l+2$.

In order to allow for a more flexible mapping,a standard approach is to introduce additional *bias neurons*. Usually one adds one neuron per hidden layer and in the input layer. The activation of bias neurons is kept constant at 1 during training. In a fully connected NN bias neurons are, as any other neuron, connected to all neurons in the subsequent layer. The bias neurons introduce additional weight parameters, that can be used to shift the preactivation and thereby influence a neuron's activation $f(1 \cdot {}^{l}w_{\text{bias}}^k + \sum_\alpha {}^{l}n_\alpha \, {}^{l}w_\alpha^k)$. In the following bias neurons will be denoted as ${}^{l}n_{D^{(l)}}$ and the respective bias parameter pointing to the neuron with index k in layer $l+1$ will be called ${}^{l+1}w_{D^{(l)}}^k$.

Now the activations $m^{(l)}$ can be defined on the basis of the activations of individual neurons:

Let the function value of $m^{(l)} : \mathbb{R}^{D^{(l-1)} \times (D^{(l-1)} \times D^{(l-1)})} \to \mathbb{R}^{D^{(l)}}$ be

$$ {}^{l}\boldsymbol{n} := m^{(l)}(m^{(l-1)}; {}^{l-1}\boldsymbol{W}). \tag{3.14}$$

The vector of activations is defined as ${}^{l}\boldsymbol{n} \coloneqq \left({}^{l}n_1, \ldots, {}^{l}n_{D^{(l)}}\right)^T$ by the outputs of the neurons ${}^{l}n_k$ in layer l.

With the expression for the activations of a layer l given in equation (3.14), equation (3.13) can be used to evaluate the NN by subsequently calculating the activations of individual layers starting from the first hidden layer to the output layer.

Since the output of the NN is obtained by propagating the input information forward, i.e. from the input to the output layer, through the NN, this process is called *forward propagation*.

In the *forward propagation step* of a training epoch the NN is evaluated for a subset of the training data set $\tilde{\mathbb{X}}_{\text{train}} \subseteq \mathbb{X}_{\text{train}}$. The errors in the predictions that the NN makes for each of the training examples $\boldsymbol{x}_{\text{train}} \in \tilde{\mathbb{X}}_{\text{train}}$ are quantified by the *loss function* $\mathcal{L}(\boldsymbol{W})$. After the forward propagation step is completed, the backpropagation step is executed. For evaluating a trained NN-potential in order to obtain the energy for a given molecular structure , only a single forward propagation step is required. If on top of the energy also atomic forces or higher derivatives of the PES are required, they have to be computed by backpropagation.

3.4.2. The Backpropagation Step

The aim of a training algorithm for NNs is to optimize the weight parameters $\boldsymbol{W}$ such that the loss function $\mathcal{L}(\boldsymbol{W})$ is minimized. For the minimization of the loss function most optimization algorithms require its the gradient with respect to all weight parameters $\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})$.

Thus, the backpropagation step can be subdivided into two phases: the first phase in which the actual backpropagation of information is performed and a second phase in which the weights and biases are optimized.

During the backpropagation phase information on the loss function is propagated backwards from the output layer through all hidden layers to the input layer in order to calculate the gradient of the loss function with respect to all weight parameters $\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})$.

The calculation of $\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})$ by backpropagation is straight forward and computationally efficient as it is based on the generalized chain rule of calculus. Assuming

that $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$ and that the functions f and g are defined as follows $g : \mathbb{R}^m \to \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ with $f(\boldsymbol{y}) = z$, $g(\boldsymbol{x}) = \boldsymbol{y}$ the chain rule defines the gradient of z with respect to $\boldsymbol{x}$

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^T \nabla_{\boldsymbol{y}} z, \tag{3.15}$$

where $\partial \boldsymbol{y}/\partial \boldsymbol{x}$ is the Jacobi matrix of g.

In backpropagation one usually has to deal with tensors instead of vectors. Therefore, it is a common approach to reformulate the problem as a vector valued problem. Then one can apply the chain rule as defined in equation (3.15) and subsequently reshape the obtained gradient to match the original tensor.

$\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W})$ depends on the functional form of the loss function used in the training process. However, independent of the loss function's functional form, the computation of $\partial \hat{y}_{\mathrm{NN}}/\partial^{\,l} \boldsymbol{W}$ with a backpropagation algorithm is always an integral part of computing the loss function's gradient with respect to the weights. For a NN with depth L $\partial \hat{y}_{\mathrm{NN}}/\partial^{\,l} \boldsymbol{W}$ is computed by applying the generalized chain rule for all layers l, starting for the weights defining the output layer's activation ($l = L - 1$) and subsequently moving layer by layer backward through the NN, i.e. they are calculated in the order $\partial \hat{y}_{\mathrm{NN}}/\partial^{\,L-1} \boldsymbol{W}$, $\partial \hat{y}_{\mathrm{NN}}/\partial^{\,L-2} \boldsymbol{W}, \ldots,$ $\partial \hat{y}_{\mathrm{NN}}/\partial^{\,1} \boldsymbol{W}$.

In the following it is shown how $\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W})$ can be calculated for a NN with one hidden layer.

Example: $\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W})$ for a Neural Network with one hidden layer

This section gives an example how backpropagation can be used to compute the derivative of the loss function with respect to the weights. Assume that the NN has one hidden layer, i.e. it's depth L is 3, and that the mean average error for a training set with m examples is used as loss function:

$$\mathcal{L}(\boldsymbol{W}) = \frac{1}{m} \sum_{i=1}^{m} |y^{(i)} - \hat{y}_{\mathrm{NN}}^{(i)}|,$$

where $\hat{y}_{\mathrm{NN}}^{(i)}$ is the prediction made by the NN for the training example $x^{(i)}$ and $y^{(i)}$ is the corresponding target value. Then the derivative of the loss function with

respect to a weight $w \in \boldsymbol{W}$ is given by

$$\frac{\partial \mathcal{L}(\boldsymbol{W})}{\partial w} = -\frac{1}{m}\sum_{i=1}^{m} \frac{(y^{(i)} - \hat{y}_{\mathrm{NN}}^{(i)})}{|y^{(i)} - \hat{y}_{\mathrm{NN}}^{(i)}|} \frac{\partial \hat{y}_{\mathrm{NN}}^{(i)}}{\partial w}$$

Thus, computing $\partial \hat{y}_{\mathrm{NN}}^{(i)} / \partial w$ for all weights w is essential for computing $\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W})$. With

$$\begin{aligned} \hat{y}_{\mathrm{NN}}^{(i)} = {}^{2}n_1 &= {}^{2}f_1 \left(\sum_{k=1}^{D^{(1)}} {}^{2}w_k^1 \, {}^{1}n_k \right), \\ {}^{1}n_k &= {}^{1}f_k \left(\sum_{j=1}^{D^{(0)}} {}^{1}w_j^k \, {}^{0}n_j \right), \\ {}^{0}n_j &= x_j^{(i)}, \end{aligned}$$

where ${}^{l}f_k$ is the activation function of node k in layer l, and $x_j^{(i)}$ is the j-th component of the feature vector describing training example $\boldsymbol{x}^{(i)}$, the derivatives of $\hat{y}_{\mathrm{NN}}^{(i)}$ with respect to the weights can be calculated:

$$\begin{aligned} \frac{\partial \hat{y}_{\mathrm{NN}}^{(i)}}{\partial \, {}^{2}w_\alpha^1} &= {}^{2}f_1' \, {}^{1}n_\alpha \\ \frac{\partial \hat{y}_{\mathrm{NN}}^{(i)}}{\partial \, {}^{1}w_\alpha^\beta} &= {}^{2}f_1' \, {}^{2}w_\beta^1 \, {}^{1}f_\beta' x_\alpha^{(i)}, \end{aligned}$$

where α and β are indices of neurons.

Once the gradient $\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W})$ is obtained by backpropagation it can be used to minimize the loss function with standard optimizers. Often Quasi-Newton methods like the L-BFGS algorithm [37–39], or methods that depend on the gradient alone like stochastic gradient descent [40] are employed. There are even optimizers like Adam [41] or AMSGrad [42] that were specifically designed for training NNs. However, the choice of the optimizer usually depends on the problem at hand their availability in the software package used for training.

3.5. Testing Neural Networks and Overfitting

In the last two sections it was discussed how to train a NN to make reliable predictions for training examples in a given reference data set $\mathbb{X}_{\text{train}}$. However, a good performance for the training data does not necessarily imply that the NN will make reliable predictions for examples that were not included in the training process, since NNs are known to be particularly unreliable extrapolators. Therefore, accurate predictions can in general only be expected for examples that are reasonably close to training examples. In order to estimate how well the NN generalizes to unknown examples which are reasonably close to the training examples, the trained NN is evaluated for a set of test examples $\mathbb{X}_{\text{test}}$ that follows the same probability distribution as the training data set but is independent of $\mathbb{X}_{\text{train}}$, i.e. $\mathbb{X}_{\text{test}} \cap \mathbb{X}_{\text{train}} = \emptyset$. For each test example in the test set $\mathbb{X}_{\text{test}}$ the loss is computed via the loss function used for training.

If the loss of the test set $\mathcal{L}_{\text{test}}(\boldsymbol{W})$ is similar to the loss obtained for the training examples, it can be assumed that the NN generalizes well to unknown structures, which are reasonably similar to the training examples. If the error of the predictions for the test examples is significantly larger than the loss of the training set, *overfitting* has occurred. This means that the training examples were fitted very exactly at the cost of the loss of generalization for unseen structures. Overfitting can occur, for example, if the reference data set is very small or noisy. Moreover, employing a too complex model, i.e. a NN with too many weights, or a descriptor describing features that are irrelevant for the machine learning problem are common issues that cause overfitting.

In figure 3.4 an example for overfitting is given. Assume that the target values of the training examples should linearly increase with the quantity given on the x axis of the graph and the exact description of the reference data is given by the green straight line. Furthermore assume that the training examples were obtained by a method that introduces some noise to the target values.

If the model obtained by the machine learning algorithm is given by the high-order polynomial shown in blue, the loss of the training set is zero, as all training examples are reproduced exactly by the model. It is obvious that the polynomial is overfitting the training data since the loss of a general test set can be assumed to

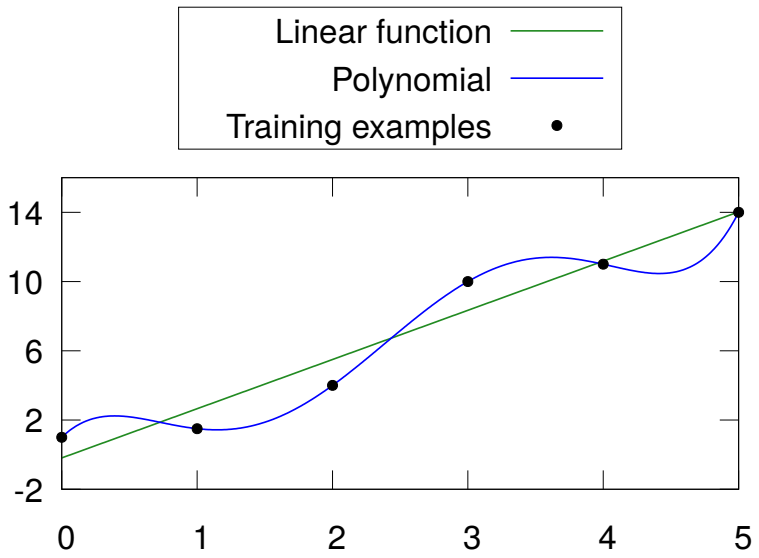


Figure 3.4.: **Example for overfitting.** Training data (black points) that behaves nearly linear is fitted by a linear model (green) and a high-order polynomial (blue). The polynomial is overfitting the data strongly by meeting all training points exactly but compromising the predictive quality of the model for points not included in the fitting process.

be notably lager than zero due to the fact that the high-order polynomial is not a good approximation of the straight line that describes the noise free data exactly.

If the loss of test set $\mathcal{L}_{\text{test}}$ is computed during the training process, one can identify overfitting by comparing the *learning curves* of the training- and the test set. The learning curves are defined by the loss as a function of the training epoch. The model is overfitting if during training $\mathcal{L}_{\text{test}}$ increases but the loss of the training examples $\mathcal{L}_{\text{train}}$ continues to decrease. An example for learning curves for a training process in which overfitting occurs is given in figure 3.5. In this graph it can be seen that overfitting occurs after about 1700 training epochs as $\mathcal{L}_{\text{test}}$ starts increasing while $\mathcal{L}_{\text{train}}$ continues decreasing.

It is to be expected that a NN reproduces the training examples better than unknown examples. Therefore, it is non-trivial to formally define which deviations of $\mathcal{L}_{\text{train}}(\boldsymbol{W})$ from $\mathcal{L}_{\text{test}}(\boldsymbol{W})$ should be considered as overfitting. This is especially difficult if the loss of the test set does not start to increase but continues decreasing at a significantly lower rate in comparison to the loss of the training set. This

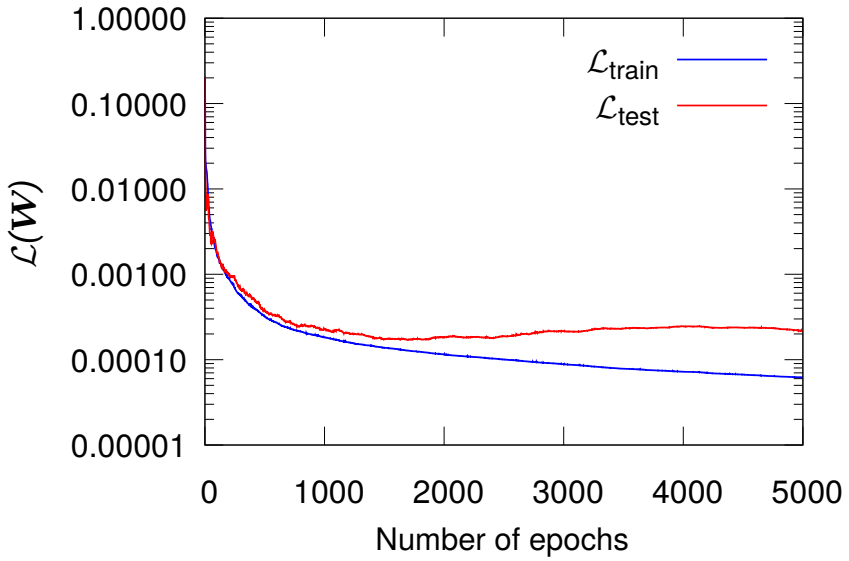


Figure 3.5.: **Diagnosing overfitting with learning curves.** Overfitting occurs after about 1700 training epochs where $\mathcal{L}_{\text{test}}$ starts increasing but $\mathcal{L}_{\text{train}}$ decreases monotonously.

behavior can still lead to a huge discrepancy of the NN's predictive power for training- and test examples.

If several NNs are available for the problem at hand, the NN whose weights $\boldsymbol{W}$ minimize $\mathcal{L}_{\text{test}}(\boldsymbol{W})$ should be used to allow for optimal generalization. Nevertheless, it has to be verified that for this seemingly optimal choice of weights overfitting is negligible by comparing the losses of the training- and test set.

3.6. Neural Networks for Regression of Potential Energy Surfaces of Chemical Systems

So far the mathematical background of NNs and the general principles of training feedforward NNs was explained in detail. In the following it will be shown how feedforward NNs can be used for the regression of PESs. Since the PES defines a mapping of chemical structure to the corresponding potential energy, the NN should perform the same mapping.

A naive approach to construct NN-PESs is explained in section 3.6.1. In this

approach an entire chemical structure is mapped to its potential energy. An alternative and more flexible approach is to construct NN-potentials for each chemical element of interest and to map each atom and its environment to the energy this atom contributes to the total potential energy of the respective chemical structure. Details on this approach are given in section 3.6.2.

3.6.1. Structure Neural Networks

This section deals with NNs that map an entire chemical structure to its respective ground state potential energy. This class of NNs will be in the following referred to as *structure neural networks* (SNNs). This approach to training a NN is excellently suited for the description of PESs of small molecules. The first PES fitting with a structure NN was done by Doren *et al.* in 1995 [43]. Since then this method was used for the description of many chemical systems.

In the context of SNNs it is common to use internal coordinates to describe chemical structures in the input layer. For this choice of a structural *descriptor*, however, the computational effort for the evaluation, and thus the training, of a SNN increases strongly with the number of atoms in the system. The reason for the strong increase is that the dimension of the descriptor, depends linearly on the number of atoms in the structure. This implies, however, that the width of the input layer scales linearly with the number of atoms in the structure. This is due to the fact that for each element of the descriptor vector there has to be a neuron in the input layer representing it. Since the network is fully connected this implies a quadratic increase in the number of weights between the input and the first hidden layer. This effect is amplified further as a wider input layer generally implies that the width of the hidden layers has to be increased in order to allow for sufficient flexibility of the NN model to fit the training data well.

The consequential increase in the number of weight parameters makes training more difficult as the dimensionality of the weight optimization problem increases. Due to the high complexity of the optimization problem, any convergence criterion on the loss of the network will be met after a larger number of epochs than for the optimization of a simpler model. This issue of the model complexity increasing with system size can be solved by employing descriptors of fixed dimension.

Another disadvantage of SNNs is that the knowledge learned for a specific chemical system cannot be transferred to other systems. Assume that a SNN was trained to predict the potential energy for a water dimer. This SNN-PES cannot be used to describe other, even very similar systems like water trimers. This is due to the fact that the dimension of the descriptor for a water trimer differs from the one for the water dimer and thus, the NN-PES can't be evaluated. This non-transferability of obtained knowledge is very undesirable since the slightest change in the chemical system under study implies that a new NN has to be trained from scratch in a time consuming manner.

3.6.2. Atomic Neural Networks

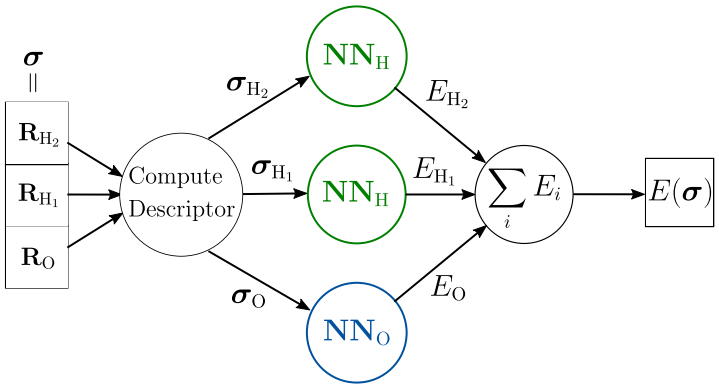


Figure 3.6.: **Schematic representation of an atomic neural network describing one water molecule.** For each atom i the descriptor $\boldsymbol{\sigma}_i$ is computed. The $\boldsymbol{\sigma}_i$ are then used as an input for the atomic NNs for the respective element. There is a unique NN for each chemical species, $\mathbf{NN}_{\mathrm{O}}$ for oxygen (blue) and $\mathbf{NN}_{\mathrm{H}}$ for Hydrogen (green). For atoms of the same chemical species identical NNs are used.

The motivation behind the use of *atomic neural networks* (ANNs) is twofold. On one hand ANNs allow for the regression of high-dimensional PESs describing large molecules and on the other hand they enable the straight-forward transfer of knowledge obtained by the ANNs to other chemical systems with similar atomic interaction. ANNs were introduced by Behler and Parinello in 2007 [3].

To employ ANNs it has to be assumed that the total potential energy of a chemical

system E_{tot} can be represented exactly by a sum of atomic energy contributions

$$E_{\text{tot}} = \sum_i E_i. \tag{3.16}$$

This is an assumption that is not unique to this ansatz but is commonly used to define classical force fields. Making use of this assumption an individual NN is constructed for each chemical element that is observed in the system under study. These NNs predict the energy contribution of an atom of this species to the total energy of the training example. The architecture of the NNs is fixed for each species, but can differ from element to element.

Training individual NNs for different chemical species is motivated by the observation that the interaction between two atoms is always the same under a fixed set of outside conditions. Thus, it should be possible to transfer the interaction of an atom i of element e_i in a given chemical environment, which is defined by the positions and species of surrounding atoms that interact with atom i, to another atom j of the same element in an equal or at least similar chemical environment.

Atomic NNs map a description of an atom and its chemical environment to the atom's energy contribution E_i to the total energy of the chemical system under study. The atom i for which the ANN is evaluated in the forward propagation step is often referred to as the *central atom*. It is called the central atom since the central atom's environment is defined by all atoms within a cutoff sphere around it. All atoms within this sphere are considered to be relevant to describe the interaction of the central atom with its surroundings, while all atoms not included in this cutoff sphere are neglected.

Due to the fixed architecture of the ANNs the width of the input layer has to be independent of the coordination of the central atom in order to allow for making predictions for all atoms of this element. This independence can be ensured by using a descriptor that has a fixed dimension, defining the width of the ANN's input layer. Details on descriptors that can be used in this context are given in section 3.7.1.

The dimensionality of descriptors for atomic environments can vary for different atomic species. The fixed architecture of ANNs allows for an efficient training with large chemical structures. The computational effort for training scales linearly with

system size as in each epoch for each atom the respective ANN has to be evaluated.

The fixed architectures of the ANNs also allow for a straight-forward transfer of previously obtained knowledge to other systems if an appropriate descriptor is employed.

The training of ANNs is done in accordance to the description given in section 3.4. However, the workflow of the training process has to be altered slightly, since the atomic energy contributions E_i are in general not known from *ab initio* computations. Therefore, first the total potential energy of the structure as predicted by the ANNs, $E_{\text{tot}}^{\text{NN}}$, is computed from the atomic energy contributions E_i and subsequently the loss function is evaluated. The weights of ANNs describing different elements are independent from each other and therefore the backpropagation step is analogous to the description given in section 3.4.

3.7. Design of Neural Network Frameworks

To complete the picture on how NNs can be used in theoretical chemistry to regress PESs, this sections gives detailed insights into the design process of a neural network framework. First three standard descriptors are introduced, then common choices for activation functions as well as the loss function are presented. Subsequently the process employed for the definition of the width and depth of the NN is described. This section concludes with a summary of preconditioning techniques that can be used to accelerate the training process.

3.7.1. Descriptors

In order to successfully train a NN it is required to extract features from the input data that contain information which is relevant for the task that the NN should perform. The features that describe a training example are collected in a *feature vector*, which will be synonymously referred to as *descriptor* in the following.

In the context of regressing PESs a descriptor defines a set of coordinates. A good descriptor makes use of physical properties like invariances or symmetries in order to facilitate learning and improve the transferability of the model. Important invariances that should be accounted for by a descriptor are the invariance of the

energy with respect to rotations and translations of the whole system. Further, it is important that exchanging the positions of two atoms of the same element does not change the descriptor as this is an operation under which the total energy remains unchanged. Descriptors also have to be continuously differentiable functions of the spatial coordinate in order to allow for a calculation of forces on the PES. On top of that it is beneficial if descriptors decay to zero for large interatomic distances, since physical interactions between the atoms will be negligible for large distances. It is advantageous if the feature vector is independent of the order in which the atoms are given in the training or test examples, because the order in which the atoms are given is arbitrary. If a descriptor that does not have this property is used, it has to be ensured that the atoms are always given in the same order, which makes a transfer of the learned information to other chemical systems significantly more difficult as the correct order of the atoms has to be known. For small chemical systems the incorporation of symmetry information into the feature vector can accelerate the training process. However, since most large systems have C_1 symmetry, including symmetry into descriptors is not needed for most tasks. An important property of a good descriptor for molecular structures is that similar structures should be described by similar descriptors, but descriptors of chemically different structures should significantly differ from each other. If this property is not given, it is practically impossible for the NN to learn the difference in energy between those chemically different structures for which the descriptors is similar as contradictory data is used to train the NN.

For small chemical systems with up to about 20 degrees of freedom it is the standard approach to use internal coordinates as a descriptor, since they are invariant with respect to translations and rotations of the whole system. Common choices are either a description by elongations along normal modes or descriptions based on a set of interatomic distances. For larger chemical systems such a description is infeasible because the number of internal degrees of freedom, and thus the dimension of the feature vector, increases at least linearly with system size. A high-dimensional descriptor, however, increases the computational effort of training and evaluating the NN. Employing ANNs reduces the computational effort for training and evaluating NN-potentials for medium sized and large systems drastically. ANNs require a description of an atom's environment as an input. In the context

of ANNs it is common to use a localized description of an atom's environment by only considering interactions within a spherical cutoff around the central atom. Any interactions with atoms outside this cutoff sphere are neglected. Especially for direct force training this approach reduces the computational effort drastically. For large systems the scaling of the scaling of the computational effort is reduced from a quadratic scaling to a linear scaling in the number of atoms. However, the the prefactor $\mathcal{N}_{\mathrm{loc}}$ that is defined by the average number of atoms within two times the cutoff radius [44] is still large. ANNs require an input of constant size, which implies that the dimension of the descriptor has to be independent of the number of atoms in the cutoff sphere.

In the following two state of the art descriptors for ANNs will be described in detail. First the Behler and Parinello *symmetry function* descriptor [3] will be discussed and subsequently the more recently developed Chebyshev polynomial based descriptor by Artrith *et al.* [35] will be described in detail.

Symmetry Functions

Behler and Parinello developed ANNs and the symmetry function descriptor was constructed specifically for this application [3, 6].

The aim of symmetry functions is to describe an atom and its environment, i.e. the position and elements of all neighboring atoms within the cutoff sphere. The environment is described by a constant number of functions M, which ensures that the dimensionality of the descriptor is independent of the number of atoms in the cutoff sphere. These functions are many body functions that, in general, depend on all atoms in the cutoff sphere.

There are two kinds of symmetry functions: radial- and angular symmetry functions. Radial symmetry functions are two body functions that describe the radial distribution of neighboring atoms, whereas angular symmetry functions are three body functions which specify the angular arrangement of atoms.

Behler and Parinello proposed in [6] three radial ($G_i^1 - G_i^3$) and two angular symmetry functions ($G_i^4 - G_i^5$). These functions are employed as a basis to span the feature space.

$$G_i^1 = \sum_j f_c(R_{ij}) \tag{3.17}$$

$$G_i^2 = \sum_j \exp(-\eta(R_{ij} - R_s)^2) f_c(R_{ij}) \tag{3.18}$$

$$G_i^3 = \sum_j \cos(\kappa R_{ij}) f_c(R_{ij}) \tag{3.19}$$

$$\begin{aligned} G_i^4 =& 2^{1-\zeta} \sum_{j,k\neq i} (1 + \lambda \cos(\theta_{ijk})^\zeta) \exp(-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)) \\ & \cdot f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \end{aligned} \tag{3.20}$$

$$G_i^5 = 2^{1-\zeta} \sum_{j,k\neq i} (1 + \lambda \cos(\theta_{ijk}))^\zeta \exp(-\eta(R_{ij}^2 + R_{ik}^2)) f_c(R_{ij}) f_c(R_{ik}) \tag{3.21}$$

where η, κ and $\zeta \in \mathbb{R}$ as well as $\lambda \in \{-1, 1\}$ are tunable parameters. Further, θ_{ijk} is defined as $\theta_{ijk} \coloneqq \arccos(\boldsymbol{R}_{ij} \cdot \boldsymbol{R}_{ik}/(R_{ij} \cdot R_{ik}))$. The cutoff function f_c which was used in the original definition of the symmetry functions is

$$f_c(\boldsymbol{R}_{ij}) = \begin{cases} \frac{1}{2}\left(\cos(\frac{\pi R_{ij}}{R_c}) + 1\right) & , R_{ij} \leq R_c \\ 0 & , R_{ij} > R_c. \end{cases} \tag{3.22}$$

Symmetry functions meet all requirements for a good descriptor that were stated before. Firstly symmetry functions are continuously differentiable functions that decay to zero for large distances. This descriptor is defined by the distance of the neighboring atoms to the central atom as well as the elements of these atoms alone. Therefore, translations and rotations of the whole molecular structure do not change the descriptor. Further, swapping the positions of two atoms of the same element does not change the descriptor either and the order at which the atoms are given in the training or test examples is irrelevant. Symmetry functions and their first derivatives decay to zero for large interatomic distances. However, the proposed cutoff function has the disadvantage that its second derivative with respect to the interatomic distance at $r = R_c$ is discontinuous. This discontinuity implies that the atomic forces might not be a continuous function of the interatomic distances for atoms that enter or leave the cutoff sphere. These jumps in the force

values violate energy conservation, which can, in principle, impair the reliability of simulations significantly. The discontinuity of the forces, however, becomes less significant if sufficiently large cutoff radii are used. The effect of the discontinuity in the forces is usually negligible for cutoff radii bigger than 6 Å [45].

The symmetry function descriptor was extended by Smith *et al.* to allow for a more flexible description of molecular structure by introducing more versatile angular symmetry functions [46].

In order to construct a descriptor for a chemical system of interest, one transforms the Cartesian coordinates of the reference structures into a basis defined by M symmetry functions. This selection of M symmetry functions and the choice of their intrinsic parameters can be verified by expanding the atomic radial and angular distribution functions in the basis of a these symmetry functions. An expansion that represents the radial and angular distribution functions well can be assumed to be a good descriptor of the system. Unfortunately, the number of symmetry functions needed to define a reliable descriptor increases strongly with the number of species in the chemical system under study, since the interaction of each pair (in radial symmetry functions) or triplet (in angular symmetry functions) of atomic species has to be treated individually. Due to this restriction it is computationally very inefficient to use symmetry functions as a descriptor for large systems with more than 4 species [44].

Expansion of the Radial and Angular Distribution Function in a Complete Basis

Due to the strong scaling of its complexity with the number of species, the application of the symmetry function descriptor is limited to chemical systems with a small number of chemical species. In order to allow for the description of chemical systems with a larger number of chemical species, a descriptor that scales less strongly with the number of chemical species is required.

One descriptor that can be used for systems with many chemical species was proposed by Artrith *et al.*. This descriptor is based on the expansion of the atomic radial and angular distribution functions in a complete basis [35]. They could demonstrate that the computational effort for training ANNs does not necessarily

have to increase with the number of chemical species in the system. In their work they show that their descriptor allows the description of transition-metal oxide compounds as well as biomolecules by a feature vector of constant dimension for systems with 3-11 species. This descriptor defines the local atomic environment $\sigma^i \subset \sigma$ of an atom i in a chemical structure σ by two invariant sets of coordinates: one describing the atomic positions $\mathbf{R}$ of the atoms within the cutoff sphere, ${}^{\mathbf{R}}\sigma^i$, and one describing the chemical species $\mathbf{t}$ of the atoms within the cutoff radius, ${}^{\mathbf{t}}\sigma^i$. The descriptor of the local atomic environment is defined as the union of these two sets of coordinates: $\sigma^i = {}^{\mathbf{R}}\sigma^i \cup {}^{\mathbf{t}}\sigma^i$.

The structural descriptor ${}^{\mathbf{R}}\sigma^i$ is constructed from expansion coefficients obtained by expanding the atom centered radial (RDF) and angular (ADF) distribution functions in a complete basis $\{\varphi_\alpha\}$ for all atoms in the reference data set:

$$\mathrm{RDF}_i(r) = \sum_\alpha c_\alpha^r \varphi_\alpha(r), \qquad 0 \leq r \leq R_c \tag{3.23}$$

$$\mathrm{ADF}_i(\theta) = \sum_\alpha c_\alpha^a \varphi_\alpha(\theta), \qquad 0 \leq \theta \leq \pi. \tag{3.24}$$

${}^{\mathbf{t}}\sigma^i$ is defined by the expansion coefficients of the RDF and ADF as well. However, the individual atomic contributions are weighted by a weight factor depending on the atom's species.

Rewriting the RDF in terms of discrete delta functions that are centered at the bond lengths R_{ij} between the central atom i and each of its neighboring atoms j and reformulating the ADF employing discrete delta functions centered at the bond angle θ_{ijk} enclosed by $\mathbf{R}_{ij}$ and $\mathbf{R}_{ik}$ one obtains

$$\mathrm{RDF}_i(r) = \sum_{\boldsymbol{R}_j \in \sigma^i} \delta(r - R_{ij}) f_c(R_{ij}) \omega_{t_j} \tag{3.25}$$

$$\mathrm{ADF}_i(\theta) = \sum_{\boldsymbol{R}_j, \boldsymbol{R}_k \in \sigma^i} \delta(\theta - \theta_{ijk}) f_c(R_{ij}) f_c(R_{ik}) \omega_{t_j} \omega_{t_k}. \tag{3.26}$$

The cutoff function is given in equation (3.22).

The weight parameters ω_{t_j} and ω_{t_k} are defined such that their values are dependent on the species of the atom. For the structural descriptor all weight parameters are 1. Artrith *et al.* suggest using a pseudospin convention to define the weight

parameters for the compositional descriptor ${}^{\mathbf{t}}\sigma^i$: $\omega_{t.} = 0, \pm 1, \pm 2, \pm 3, \ldots$. The value 0 is omitted if the number of species is even.

Choosing a complete and orthonormal basis, the expansion coefficients can be readily calculated:

$$c_\alpha^r = \sum_{\boldsymbol{R}_j \in \sigma^i} \varphi_\alpha(R_{ij}) f_c(R_{ij}) \omega_{t_j} \tag{3.27}$$

$$c_\alpha^a = \sum_{\boldsymbol{R}_j, \boldsymbol{R}_k \in \sigma^i} \varphi_\alpha(\theta_{ijk}) f_c(R_{ij}) f_c(R_{ik}) \omega_{t_j} \omega_{t_k}. \tag{3.28}$$

The expansion of the RDF is truncated at order N^r and the expansion of the ADF is truncated at order N^a. The order of the expansion determines the resolution of the feature vector as well as its dimension:

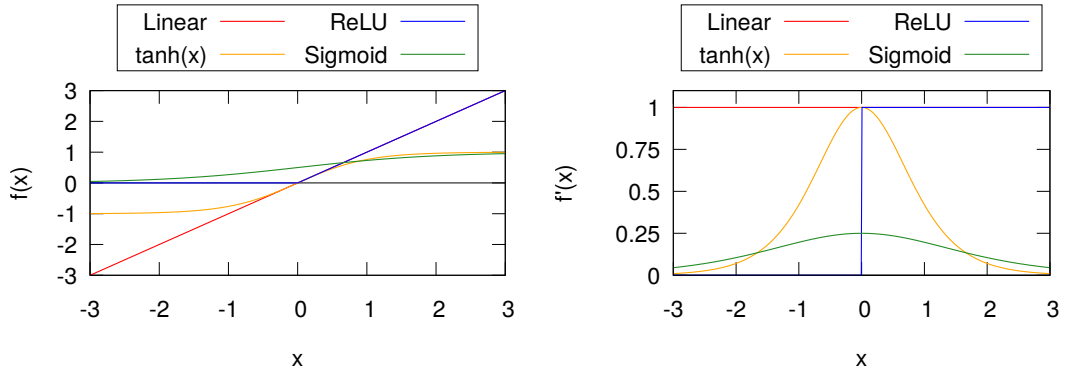
$$\sigma^i = (c_1^{r,\mathbf{R}}, \ldots, c_{N^r}^{r,\mathbf{R}}, c_1^{a,\mathbf{R}}, \ldots, c_{N^a}^{a,\mathbf{R}}, c_1^{r,\mathbf{t}}, \ldots, c_{N^r}^{r,\mathbf{t}}, c_1^{a,\mathbf{t}}, \ldots, c_{N^a}^{a,\mathbf{t}}), \tag{3.29}$$

where $c_i^{r,\mathbf{R}}$ and $c_i^{a,\mathbf{R}}$ are the expansion coefficients for the structural descriptor ${}^{\mathbf{R}}\sigma^i$ and $c_i^{r,\mathbf{t}}$ as well as $c_i^{a,\mathbf{t}}$ are the expansion coefficients for the compositional descriptor ${}^{\mathbf{t}}\sigma_i$. Since the feature vector is constructed from both the information of the structural descriptor and the information of the compositional descriptor, it is $2(N^r + N^a)$ dimensional.

The advantage of this descriptor is that it can be systematically improved by in creasing the order of the expansion N^r or N^a, which increases the descriptors resolution, since the basis used for the expansion of the RDF and ADF is complete.

In principle any orthonormal basis functions that form a complete basis could be used to construct a descriptor by expanding the atom centered RDF and ADF. However, basis functions that can be evaluated with minimal computational cost and for which the derivative with respect to spatial coordinates can be computed efficiently are to be preferred to keep the computational effort minimal. Therefore, Artrith *et al.* suggest the use of normalized Chebyshev polynomials of the first kind because the polynomials as well as their derivatives can be computed efficiently from a simple recurrence relation.

3.7.2. Activation Functions



(a) **Activation functions.** (b) **Derivatives of the activation functions.**

Figure 3.7.: **Common activation functions and their derivatives**

Since the 1950s when NNs were introduced, a vast multitude of activation functions for neurons has been employed and tested.

A selection of four of the most common activation functions used in the context of regressing high-dimensional functions like a PES with NNs is given in figure 3.7.

Even though there are many choices for activation functions available, most activation functions have the following two properties: Firstly, activation functions usually are differentiable, as this is required if the backpropagation algorithm is to be applied. Secondly, often monotonic increasing activation functions are used as these lead in many cases to a faster convergence of the weights during training.

It was shown that for NNs with one hidden layer the use of a monotonic activation function implies that the loss $\mathcal{L}(\boldsymbol{W})$ is convex [47], which allows for an efficient minimization of the loss. However, the use of non-monotonic functions is possible.

The functional form of individual activation functions is driven by problem specific or intuitive restrictions.

The hyperbolic tangent or the sigmoid function $1/(1+\exp(-x))$ were initially chosen as activation functions with the idea of a biological neuron in mind, which has a defined maximum signal strength and only sends a signal to other neurons if a certain activation potential is reached in the neuron.

However, the saturation of the hyperbolic tangent or the sigmoid function can cause the gradients of the loss function to vanish. If a neuron's activation is well in the saturation regime of the activation function, the neuron responds significantly less sensitively to changes in the preactivation induced by an optimization of the weights pointing to this neuron. On top of that the saturation of the hyperbolic tangent and the sigmoid function restricts the values for the activation of the neurons to a fixed range ($f_{\text{sigmoid}}(x) \in [0, 1]$, $f_{\tanh}(x) \in [-1, 1]$).

Both activation functions are not used in deep NNs as they in general reduce the gradient of the loss with respect to the weight parameters due to the fact that $f'(x) < 1$ for almost all preactivations x. This property of the activation function's derivative leads to a vanishing of the gradient which aggravates with increasing depth of a NN. Therefore, activation functions which saturate are commonly solely used for comparatively shallow NNs with just a few hidden layers, where the vanishing of the gradient is minimal and does not affect the training procedure significantly.

In order to overcome the *vanishing gradient problem* often the rectified linear unit (ReLU) is used as an activation function as its derivative for preactivations $x > 0$ is 1. The advantages of employing this activation function are that both the activation function and its derivative are monotonically increasing and that the activation function is straight-forward to evaluate, as it is defined by a simple relation: $f(x) = \max(0, x)$. The ReLU activation function is differentiable at all points x except from $x = 0$. Although this activation function is not differentiable at $x = 0$, it is successfully used in training procedures where backpropagation is employed by defining $f'(x = 0)$ arbitrarily. The ReLU allows for activations of arbitrarily high value, but its minimum value is 0, which restricts the activations to non-negative values.

Whenever arbitrary activations are required, employing a non-saturating activation function is necessary. This requirement applies for example to the neurons in the output layer if the target values can take on arbitrarily small or large values. In order to predict arbitrary values an activation function that maps to all real numbers is needed. One such activation function that is commonly used is a linear activation function $f(x) = x$. Since it has a non-vanishing derivative $f'(x) = 1$ and its evaluation and differentiation does not involve any computational cost, it

is an activation function that allows for the efficient training of NNs. However, since the derivative is constant, backpropagation leads to a constant gradient of the loss with respect to the weights that point to a neuron with a linear activation function, which impedes weight optimization. Further, it is not possible to use only linear activation functions if a non-linear model shall be trained, since NNs that use solely linear activation functions always perform a linear transformation of the input, no matter how many hidden layers are used. Using non-linear activation functions like the hyperbolic tangent for the neurons in the hidden layers of a NN and applying a linear activation function in the output layer's neurons allows for training a non-linear model that can regress target values of arbitrary absolute value.

3.7.3. Loss Functions

Loss functions are used as a performance measure for NNs. For the regression of PESs the computational demand of computing the loss function depends on the physical quantities included in the loss function. If only energy information is used for training, the loss function measures the deviation of the energy predicted for a set of reference structures $\{\hat{y}_{\mathrm{NN}}^{(i)}\}$ from the corresponding reference energies given in the training set $\{\hat{y}^{(i)}\}$. In this case the loss function can be computed by a single forward propagation step and the computational cost for training scales linearly with the number of atoms (ANNs) or structures (SNNs).

However, if not only energy information but also information on the atomic forces or Hessians is used for training, the computation of the loss function requires a forward propagation step and an additional backward propagation step. The additional backward propagation step is needed, since the atomic forces or the Hessian aren't outputs of the NN, but are calculated by deriving the predicted energy analytically by the spatial coordinates. This computation increases the computational demand for calculating the loss function. Formally computing the atomic forces, i.e. the negative gradient of the energy with respect to the spatial coordinates, scales quadratic with the number of atoms in a given reference structure. This, however, implies that the computational demand for training scales quadratically with the number of atoms in the reference data set. If a localized descriptor is used, the

quadratic scaling can be reduced for large or periodic structures to a linear scaling with a potentially large prefactor, see section 3.7.1.

The loss functions used for the studies described in this thesis are given in the computational details sections of the respective projects.

In the following two of the most commonly used loss functions used for the regression of high-dimensional functions will be described and compared. The discussion of the advantages and disadvantages of the methods summarizes the findings in [48].

Mean Absolute Error

The mean absolute error

$$\mathcal{L}_{\mathrm{MAE}}(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}^{(i)} - \hat{y}_{\mathrm{NN}}^{(i)}| \tag{3.30}$$

is a loss function which is used if the focus of the training approach is on reducing the average error made by the machine learning model fast, and outliers, i.e. training examples with large errors, are of less importance than reducing the average error. This means that the maximum error of a model trained with $\mathcal{L}_{\mathrm{MAE}}(\boldsymbol{W})$ can be significantly higher than a model trained with a loss that penalizes large deviations stronger like $\mathcal{L}_{\mathrm{RMSE}}(\boldsymbol{W})$. An important advantage of the mean absolute error is it's straight forward interpretation as, in contrast to other common loss functions like the mean squared error or the root mean squared error, it mirrors the average performance of the model directly.

A disadvantage of using the mean absolute error as a loss function is that it is not differentiable at its minimum, which can hinder the minimization of the loss significantly.

Mean Square Error and Root Mean Square Error

For regression problems it is common to use the root mean square error $\mathcal{L}_{\text{RMSE}}(\boldsymbol{W})$ or the mean squared error $\mathcal{L}_{\text{MSE}}(\boldsymbol{W})$ to measure the model's loss.

$$\mathcal{L}_{\text{RMSE}}(\boldsymbol{W}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |\hat{y}^{(i)} - \hat{y}_{\text{NN}}^{(i)}|^2} \tag{3.31}$$

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}^{(i)} - \hat{y}_{\text{NN}}^{(i)}|^2 \tag{3.32}$$

In an ideal case where the target values are normally distributed and the training examples are i.i.d. using the (root) mean squared error is equivalent to maximizing the log-likelihood of the model if it can be expressed as a linear regression model. Making use of the kernel trick, this applies to NNs, at least if they have exactly one hidden layer, see sections 3.3.3 and 3.3.4.

However, even if these conditions are not met by the training set, the mean square error and root mean square error are commonly used loss functions if the avoidance of large errors for individual training examples is more important than reducing the average error fast. The high sensitivity with respect to large errors makes training difficult if there is a small number of training examples for which the NN predictions have large errors. This is due to the fact that the large errors obtained for these training examples dominates the loss function. Thus, improving predictions for the majority of training examples for which the prediction error is comparatively low is deferred significantly.

A disadvantage of employing the mean square error and the root mean square error as a loss function is that their interpretation is non trivial. As demonstrated by Willmott and Matsuura [48] the root mean square error is not a good measure for the average model error. They argue that, since the root mean square error does not only depend on the mean average error but also on the variability in the error magnitudes as well as the square root of the number of training examples $\sqrt{N}$ considered, it is without further information, like knowledge on the mean absolute error, hard to tell from the RMSE value alone how well the error describes average model behavior. From the findings in this article it can be concluded that the root

mean square error is particularly bad for the comparison of two different models, even if they were trained and tested with the same number of training examples as the variability of the magnitudes of errors will in general differ from model to model. Similar arguments are valid for the mean square error.

If models with small maximum errors are to be trained by NNs, it is recommended to train the NN using $\mathcal{L}_{\mathrm{RMSE}}$ or $\mathcal{L}_{\mathrm{MSE}}$ as large errors are penalized strongly. Nevertheless, it is advisable to also compute $\mathcal{L}_{\mathrm{MAE}}$, which is more appropriate for comparing different models and measuring a model's average performance.

3.7.4. Defining Depth of the Network and the Width of Hidden Layers

The architecture of a NN directly influences its flexibility to adjust to a task. In regression, the introduction of additional neurons and layers allows for a more flexible fitting of the training examples and thus, can reduce the loss of the NN. However, introducing too many weights, i.e. fitting parameters, leads to overfitting and compromises the transferability of the NN's predictions to unseen examples. Therefore, it is necessary to identify a NN architecture that is well-suited for the task at hand. One way to identify an appropriate NN architecture is to perform an empirical parameter study. Thereby the loss of a NN is computed after a fixed small number of training epochs for a given training set $\mathbb{X}_{\mathrm{train}}$ using NNs with varying width of the hidden layers and with varying depth. If the weight parameters are initialized randomly, it is necessary to train several NNs with the same architecture in order to sample the statistics of the resulting losses obtained for this network architecture. Subsequently the network architecture which yields on average the lowest loss is used for constructing models required for the actual application. This, of course, is a rather non-formal way of optimizing a network's architecture, however, it is often sufficient to obtain reasonable machine learned surrogate models.

Instead of performing the aforementioned manual parameter study it is also possible to determine a suitable network architecture by learning the NN architecture with the help of recurrent NNs [49] or by heuristically searching for architectures starting from a simple network architecture [50]. Further, the application of genetic algorithms for the search of neural network architectures was suggested [51–53].

However, the application of these methods is computationally very demanding and, thus, often not efficient enough to be employed in practical applications. Therefore, to keep the computational demand at bay, the architecture of all NNs studied in this thesis was determined by an empirical parameter study.

3.7.5. Pre-Conditioning Techniques

In order to accelerate the convergence of the training process it is beneficial to pre-condition the feature vectors used as input to the NNs as well as the outputs.

It has been shown that centering the output values of the neurons in each layer around zero facilitates the training process as a non-zero average introduces a bias to the direction of weight updates [54].

To ensure that all features defined by the feature vector are learned by the NN at a similar rate, all feature vectors of the neural network can be scaled such that all elements have the same covariance for a given reference data set. In order to ensure a non-vanishing gradient of the loss function with respect to the weights in the input layer, the covariance of the input values should be chosen such that the values of the input neurons are in the range of activations for which the activation functions of the neurons in the first hidden layer is not constant. For example the covariance should be 1 for the hyperbolic tangent to allow for optimal training.

In all NNs discussed in this thesis the identity was used as an activation function in the output layer to allow for the prediction of arbitrary energy values. Therefore, it does not seem to be necessary to scale the output values. However, in order to prevent a large difference in scale of the weights pointing to the output neuron in comparison to the other weights, which can cause numerical instabilities during training, it is common to shift and scale the output values in the same way as the input values such that they are zero-centered and have the same covariance as the inputs.

Another set of parameters whose initial values influence the efficiency and success of the training procedure as well as the generalization behavior of NNs strongly are the weights and biases. Therefore, techniques to choose initial weights and biases such that training the NN is efficient and the resulting NN generalizes well to unknown data are of high interest.

However, even though these techniques ensure an advantageous initialization of the weights, it is in most cases unclear under which circumstances these beneficial properties of the weights are preserved during the training process. On the other hand the initial choice of weights has to be optimized with respect to two different properties, namely the efficiency of the training process and the generalization behavior of the resulting NNs. This is because initial parameters that lead to NNs that generalize well do not necessarily imply that for the construction of the surrogate a small number of training epochs necessary and vice versa. According to Goodfellow *et al.* [34] there is practically no knowledge about how the choice of initial weights affects the generalization of NNs. They further claim that the only property of initial weights that is known for sure to be beneficial for training and generalization is that the weights and biases need to *break the symmetry* between different neurons. They argue that if two neurons with the same activation function are connected to the same inputs, the weights pointing from these two neurons to the respective next layer should not be equal. This is due to the fact that if these weights would be identical, i.e. symmetry is not broken, using a deterministic optimization algorithm for the weights as well as a deterministic loss function implies that the activations of these neurons is always identical and thus, one of these neurons is redundant.

In order to allow for an initialization of the weights and biases such that it requires minimal computational cost and to ensure that symmetry is broken between different neurons, it is common to initialize weights and biases with random numbers. Commonly the initial weights and biases are drawn from either a Gaussian or a uniform distribution and the choice of the type of random distribution used for the initialization does not seem to influence the resulting NN significantly [34]. However, the scale of the distribution and thus of the initial values of the weights influences the training success and generalization of NNs strongly. Therefore, the scale is a meta-parameter which has to be chosen to fit the NN framework.

Following the line of reason outlined for the rescaling of the feature vectors, it can be said that it should be avoided to initialize the weights with too large or too small values to ensure that the activation functions do not saturate and there's a non-vanishing gradient of the activation function. Therefore, the most simple approach to initialize the weights is to draw them from a uniform distribution which

matches the respective activation function's shape. For the hyperbolic tangent as activation function it is therefore beneficial to employ a zero-centered uniform distribution with a standard deviation of 1. However, this approach does not take into account that there are in general several inputs to each neuron and thus, in order to ensure in a more precise way that the activation function is not saturating, it can be beneficial to normalize the distribution with the number of neurons.

Assume that the NN is fully connected and that the number of neurons in the previous layer, i.e. the number of inputs to each neuron in the current layer, is m. LeCun *et al.* suggest in [54] to obtain a standard deviation of the output of each neuron of about 1 to initialize the weights by drawing them randomly from a zero-centered distribution with a standard deviation of $\sigma_w = 1/\sqrt{m}$, given that the standard deviation of the activation function is 1 as well.

There are various other methods suggesting more sophisticated standard deviations for the random distributions to ensure a beneficial initialization of the weights [55–57]. Unfortunately, most of these expressions are derived assuming that the NN defines a linear model, which implies that for a general non-linear model these choice of initial weights might not be optimal. Nevertheless, according to Goodfellow *et al.* [34] it is often the case that strategies designed for linear models perform also well for non-linear models. Another drawback of these initialization rules in which the standard deviation of the random distribution is scaled by a factor that depends on the inverse of the width of a layer is that for wide layers the actual values of the weights are initially close to zero, which is, as discussed before, unfavorable and can cause numerical instabilities during training.

It is common to make use of prior knowledge on properties of the reference data in order to define initial values of biases, which mainly influence the overall scale of a neuron's preactivation. For example, in the study employing SNNs to compute reaction rate constants presented in this thesis, the bias on the output neuron was initialized with the average energy value of the training examples, to ensure that the scale of the output value is close to the target values.

Part III.

Novel Methods and Algorithms

4. Direct Force and Hessian Training

In order to ensure a reliable prediction of atomic forces or Hessians without sampling configurational space very densely, it is required to include the force- or Hessian information in the training process. The straight forward way to include this information is to introduce a term depending on the atomic forces or Hessians in the loss function $\mathcal{L}(\boldsymbol{W})$.

In the following it is discussed how direct force training can be performed for SNNs and ANNs. For SNNs it is further shown how Hessian information can be included into the training process.

4.1. Direct Force and Hessian Training for Structure Neural Networks

The program that was used to train the SNNs studied in this thesis was implemented by Philipp Hallmen as a part of his master's thesis. For the work presented in this thesis the training algorithm was accelerated and some minor corrections were made by the author and J. Kästner.

The program can train SNNs with two hidden layers. The training was performed on the basis of the loss function

$$\begin{aligned}\mathcal{L}(\mathcal{W}) =& \frac{1}{n_E + n_G + n_H} \Bigg[A_E \sum_{i=1}^{n_E} ({}^{i}E^{\mathrm{NN}}(\boldsymbol{W}) - {}^{i}E^{\mathrm{REF}})^2 \\ &+ A_G \sum_{j=1}^{n_G} |\, {}^{j}\boldsymbol{g}^{\mathrm{NN}}(\boldsymbol{W}) - {}^{j}\boldsymbol{g}_j^{\mathrm{REF}}|^2 + A_H \sum_{k=1}^{n_H} \sum_{a,b}^{D^{(0)}} |\, {}^{k}H_{ab}^{\mathrm{NN}}(\boldsymbol{W}) - {}^{k}H_{ab}^{\mathrm{REF}}|^2 \Bigg] ,\end{aligned} \tag{4.1}$$

where n_E, n_G and n_H are the number of training examples for which information on the energy, gradient or Hessian is available. A_E, A_G and A_H are scaling factors, that can be used to weight the relative importance of the errors.

The SNNs were trained as described in section 3.4. The detailed expressions for the derivative of the loss function with respect to the weight and bias parameters, which are required for the backpropagation step are given in the appendix of this thesis.

Since the direct force training approach ensures reliable Hessian- and force predictions by the NN it is excellently suited to approximate PESs that shall be used for the computation of reaction rate constants, which requires highly accurate Hessians which are a smooth function of all spatial coordinates.

In order to accelerate convergence it can be exploited, that the energy as well as its gradients and Hessians with respect to the spatial coordinates is linear in the bias and the weight parameters acting on the output layer. Therefore, it is possible to first perform a linear optimization of these parameters while keeping the remaining weight and bias parameters fixed and to subsequently optimize the remaining weight and bias parameters with a non-linear optimization algorithm like the L-BFGS algorithm [37–39] while keeping the weights and biases acting on the output layer fixed. This acceleration method was employed for training all SNNs presented in this thesis.

The direct force and Hessian training method for SNNs was employed to construct surrogate models that allow for the efficient computation of reaction rate constants for the reaction: $CH_3OH+H \rightarrow CH_2OH + H_2$ on coupled cluster level of theory. Details on this application are given in chapter 6.

4.2. Direct Force Training for Atomic Neural Networks

The direct force training algorithm for ANNs, that will be presented in the following, was developed and implemented by the author. The algorithm is part of the open-source package ænet[58]. To allow for a maximal efficient training process, all relevant computations are, if possible, implemented as matrix-vector or matrix-

matrix operations, which can be executed highly efficiently. Further, the force training algorithm is, as the whole ænet[58] program package, parallelized with MPI, which allows for an even more efficient training procedure. Due to the local descriptors used for ANNs, which only represent the interactions of a central atom with atoms that lie within a cutoff sphere, the formal scaling of this algorithm is for large chemical systems linear in the total number of atoms. However, the prefactor of the scaling is rather large as it is given by the average number of atoms within a sphere with a radius twice as big as the cutoff radius R_c. For chemical systems with just a few atoms, where the local description still contains nearly all or all atoms, the scaling is quadratic with the number of atoms in the system.

In the following sections first all relevant notation will be introduced, then the expressions for the ANN energy, the corresponding gradients and the loss function are given. Finally the derivatives of the loss function with respect to the weight parameters are discussed in detail and the novel algorithm for computing the derivative of the ANN forces with respect to the weight parameters is outlined in detail.

4.3. Notation

This section introduces all additional notation required for the upcoming derivations. All vectorial quantities are given as bold symbols. $\equiv$ indicates that both symbols are used synonymously. In order to facilitate the lookup of symbols, the symbols are presented in an alphabetic list.

- $\mathbb{0}$ a null matrix, all entries of this matrix are 0.
- $\mathbb{1}$ identity matrix.
- $\alpha(l)$ index enumerating all neurons in layer l, where $l = 0$ denotes the input layer
- $c \in \{x, y, z\}$ denotes a Cartesian coordinate direction.
- c_φ denotes the Cartesian coordinate c of atom φ
- $D_t^{(l)} \equiv D^{(l)}$ width of layer l for ANN of species t

- E_s^{REF} is the energy of structure s obtained from quantum chemical simulations.
- E_s^{NN} is the energy predicted by the ANN for structure s.
- ${}_t^s\mathcal{E}_\varphi^{\mathrm{NN}} \equiv {}^s\mathcal{E}_\varphi^{\mathrm{NN}}$ is the atomic energy of atom φ in structure s predicted by the ANN for species t.
- ${}_\varphi^l f_{\alpha(l)} \equiv {}^l f_{\alpha(l)}$ is the value of the transfer function of node α in layer l if the corresponding ANN is evaluated for atom φ.
- $\boldsymbol{F}_\varphi^{\mathrm{REF}}$ is the *ab initio* force acting on atom φ. $F_{c,\varphi}^{\mathrm{REF}}$ is the force component acting along c.
- $\boldsymbol{F}_\varphi^{\mathrm{NN}}$ is the prediction of the force acting on atom φ by the ANN. $F_{c,\varphi}^{\mathrm{NN}}$ is the force component along c
- ${}^s\boldsymbol{g}^{\mathrm{NN}}$ is the gradient of the energy for structure s, where ${}^s g_c^{\mathrm{NN}}$ is the derivative of the NN-PES with respect to c for structure s .
- nstrucs is the number of structures in the training set.
- natoms is the number of atoms in a given structure s.
- ${}_\varphi^l n_{\alpha(l)} \equiv {}^l n_{\alpha(l)}$: values of neuron $\alpha(l)$ in layer l if the corresponding atomic NN is evaluated for atom φ. ${}_\varphi^{l=0} n_{\alpha(0)}$ denotes the $\alpha(0)$-th element of the feature vector describing atom φ.
- ${}_t^l w_b^a$: weight from neuron b in layer $l-1$ to neuron a in layer l for the ANN for species t.
- ${}_t^l\boldsymbol{W} \in \mathbb{R}^{l\times(l-1)}$ is the matrix of all weights ${}_t^l w_b^a$ in layer l

4.4. Expression for the Energy and its Gradient obtained from an Atomic Neural Network

To employ ANNs it has to be assumed that the total energy of a structure s is given by the sum of atomic energy contributions

$$ {}^{s}E^{\mathrm{NN}} = \sum_{\varphi=1}^{\mathrm{natoms}} {}^{s}_{t}\mathcal{E}^{NN}_{\varphi} $$

. For an atomic NN with L hidden layers ${}^{s}_{t}\mathcal{E}^{NN}_{\varphi}$ is defined recursively:

$$ {}^{s}_{t}\mathcal{E}^{NN}_{\varphi} = {}^{L+1}_{\varphi}n_{\alpha(L+1)} \tag{4.2} $$

with

$$ \begin{aligned} {}^{0}_{\varphi}n_{\alpha(0)} &= \sigma^{\varphi}_{\alpha(0)} \\ {}^{l}_{\varphi}n_{\alpha(l)} &= {}^{l}_{\varphi}f_{\alpha(l)} \left(\sum_{\alpha(l-1)=1}^{D^{(l-1)}} {}^{l}_{t}w^{\alpha(l)}_{\alpha(l-1)} \, {}^{l-1}_{\varphi}n_{\alpha(l-1)} \right), \; 1 \leq l \leq L+1. \end{aligned} \tag{4.3} $$

Thereby $\sigma^{\varphi}_{\alpha(0)}$ is the $\alpha(0)$-th element of the feature vector. Note that in equation (4.3) the biases ${}^{l}_{t}w^{\alpha(l)}_{D^{(l-1)}}$ are included and by definition ${}^{l-1}_{\varphi}n_{D^{(l-1)}} = 1$ holds for any bias neuron.

In all applications discussed in this thesis ANNs with two hidden layers were employed. Therefore, the expressions for the energy and atomic forces are given in detail for an ANN with two hidden layers.

For an ANN with two hidden layers the atomic energies ${}_t\mathcal{E}_\varphi^{NN}$ are

$$
\begin{aligned}
{}_t\mathcal{E}_\varphi^{NN} &= {}_\varphi^3 f_1 \left(\sum_{\alpha(2)=1}^{D^{(2)}} {}_t^3 w_{\alpha(2)}^1 \, {}_\varphi^2 n_{\alpha(2)} \right), \\
{}_\varphi^2 n_{\alpha(2)} &= {}_\varphi^2 f_{\alpha(2)} \left(\sum_{\alpha(1)=1}^{D^{(1)}} {}_t^2 w_{\alpha(1)}^{\alpha(2)} \, {}_\varphi^1 n_{\alpha(1)} \right), \\
{}_\varphi^1 n_{\alpha(1)} &= {}_\varphi^1 f_{\alpha(1)} \left(\sum_{\alpha(0)=1}^{D^{(0)}} {}_t^1 w_{\alpha(0)}^{\alpha(1)} \, {}_\varphi^0 n_{\alpha(0)} \right).
\end{aligned}
$$

The corresponding derivative ${}^s g_c^{\mathrm{NN}}$ of the NN-PES for structure s with respect to c is given by

$$
\begin{aligned}
{}^s g_c^{\mathrm{NN}} &= \frac{\partial \, {}^s E^{NN}}{\partial c} \\
&= \sum_{\varphi=1}^{\mathrm{natoms}} \frac{\partial \, {}_t^s \mathcal{E}_\varphi^{NN}}{\partial c} \\
&= \sum_{\varphi=1}^{\mathrm{natoms}} \sum_{\alpha(0)=1}^{D^{(0)}} \frac{\partial \, {}_\varphi^0 n_{\alpha(0)}}{\partial c} \sum_{\alpha(1)=1}^{D^{(1)}} \sum_{\alpha(2)=1}^{D^{(2)}} {}_\varphi^3 f_1' \, {}_t^3 w_{\alpha(2)}^1 \, {}_\varphi^2 f_{\alpha(2)}' \, {}_t^2 w_{\alpha(1)}^{\alpha(2)} \, {}_\varphi^1 f_{\alpha(1)}' \, {}_t^1 w_{\alpha(0)}^{\alpha(1)}.
\end{aligned}
$$

In order to simplify the equations and improve their readability the index φ on the node values ${}_\varphi^l n_{\alpha(l)}$ and the values of the transfer functions ${}_\varphi^l f_{\alpha(l)}$ will be omitted in the following.

4.5. The Loss Function and its Derivative with Respect to the Weight Parameters

The loss function used for direct force training with ANNs is:

$$
\mathcal{L}(\boldsymbol{w}) = \sum_{s=1}^{\mathrm{nstrucs}} N_E A_E \, |\, {}^s E^{\mathrm{REF}} - {}^s E^{\mathrm{NN}}(\boldsymbol{w})|^2 + N_G A_G \underbrace{|\, {}^s g^{\mathrm{REF}} - {}^s g^{\mathrm{NN}}(\boldsymbol{w})|^2}_{=:\Gamma}, \quad (4.4)
$$

where

$$\Gamma = \sum_{\varphi=1}^{\text{natoms}} \left({}^{s}F_{x,\varphi}^{\text{REF}} - {}^{s}F_{x,\varphi}^{\text{NN}}\right)^2 + \left({}^{s}F_{y,\varphi}^{\text{REF}} - {}^{s}F_{y,\varphi}^{\text{NN}}\right)^2 + \left({}^{s}F_{z,\varphi}^{\text{REF}} - {}^{s}F_{z,\varphi}^{\text{NN}}\right)^2 . \tag{4.5}$$

The normalization factors are chosen as follows: $N_E \coloneqq 1/(2\,\text{natoms})$ and $N_G \coloneqq 1/(2(3\,\text{natoms})^2)$. With the expression for Γ we get:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}) = \sum_{s=1}^{\text{nstrucs}} \Bigg[& N_E A_E \left| {}^{s}E^{\text{REF}} - {}^{s}E^{\text{NN}}(\boldsymbol{w}) \right|^2 \\ & + N_G A_G \sum_{\varphi=1}^{\text{natoms}} \sum_{c} \left({}^{s}F_{c,\varphi}^{\text{REF}} - {}^{s}F_{c,\varphi}^{\text{NN}}(\boldsymbol{w}) \right)^2 \Bigg] .\end{aligned} \tag{4.6}$$

Deriving the loss function with respect to the weight ${}_{t}^{l}w_{b}^{a}$ we get:

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial\, {}_{t}^{l}w_{b}^{a}} = \sum_{s=1}^{\text{nstrucs}} \Bigg[& 2N_E A_E \left({}^{s}E^{\text{REF}} - {}^{s}E^{\text{NN}}(\boldsymbol{w}) \right) \left(-\frac{\partial\, {}^{s}E^{\text{NN}}(\boldsymbol{w})}{\partial\, {}_{t}^{l}w_{b}^{a}} \right) \\ & + 2N_G A_G \sum_{\varphi=1}^{\text{natoms}} \sum_{c} \left({}^{s}F_{c,\varphi}^{\text{REF}} - {}^{s}F_{c,\varphi}^{\text{NN}}(\boldsymbol{w}) \right) \underbrace{\left(-\frac{\partial\, {}^{s}F_{c,\varphi}^{\text{NN}}(\boldsymbol{w})}{\partial\, {}_{t}^{l}w_{b}^{a}} \right)}_{=:\Phi} \Bigg] .\end{aligned} \tag{4.7}$$

In order to simplify notation further, the superscript s specifying the structure will be omitted. Further, the dependence of the energy and forces predicted by the ANN on the weights and biases will be implied and not stated explicitly.

From the expression for the force component of atom φ along c

$$\begin{aligned}F_{c,\varphi}^{\text{NN}} &= -\sum_{\psi=1}^{\text{natoms}} \frac{\partial E_{\psi}^{\text{NN}}}{\partial c_{\varphi}} \\ &= -\sum_{\psi=1}^{\text{natoms}} \sum_{\alpha(0)=1}^{D^{(0)}} \frac{\partial E_{\psi}^{\text{NN}}}{\partial \sigma_{\alpha(0)}^{\psi}} \frac{\partial \sigma_{\alpha(0)}^{\psi}}{\partial c_{\varphi}}\end{aligned}$$

we can calculate Φ as follows

$$\Phi = -\frac{\partial F^{\mathrm{NN}}_{c,\varphi}}{\partial\, {}^{l}_{t}w^{a}_{b}} = \sum_{\psi=1}^{\text{natoms}} \sum_{\alpha(0)=1}^{D^{(0)}} \frac{\partial^2 E^{\mathrm{NN}}_{\psi}}{\partial \sigma^{\psi}_{\alpha(0)} \partial\, {}^{l}_{t}w^{a}_{b}} \frac{\partial \sigma^{\psi}_{\alpha(0)}}{\partial c_{\varphi}}.$$

4.6. Computing $\partial F^{\mathrm{NN}}_{c,\varphi}/\partial\, {}^{l}_{t}w^{a}_{b}$ for Atomic Neural Networks

The derivative $\partial F^{\mathrm{NN}}_{c,\varphi}/\partial\, {}^{l}_{t}w^{a}_{b}$ can be computed during a training epoch with the generalized chain rule during a backpropagation step. Let ${}^{0}n_i \coloneqq \sigma^{\varphi}_{i}$ for atom φ. For an ANN with L hidden layers the derivative is

$$\begin{aligned}
-\frac{\partial F^{\mathrm{NN}}_{c,\varphi}}{\partial\, {}^{l}_{t}w^{a}_{b}} = \sum_{i} \frac{\partial\, {}^{0}n_i}{\partial c_{\varphi}} \Bigg[\sum_{\alpha(1)\ldots\alpha(L)} \Bigg[& \underbrace{\prod_{j=1}^{L} \frac{\partial \left({}^{L+1}f'_{1}\, {}^{L+1}_{t}w^{\alpha(L+1)}_{\alpha(L)} \right)}{\partial\, {}^{l}_{t}w^{a}_{b}}\, {}^{j}f'_{\alpha(j)}\, {}^{j}_{t}w^{\alpha(j)}_{\alpha(j-1)}}_{=:\mathbf{\Omega}} \\
& + \sum_{j=l}^{L} \left(\prod_{k=j+1}^{L+1} \prod_{m=1}^{j-1} {}^{j}f'_{\alpha(k)}\, {}^{k}_{t}w^{\alpha(k)}_{\alpha(k-1)} \frac{\partial\, {}^{j}f'_{\alpha(j)}}{\partial\, {}^{l}_{t}w^{a}_{b}}\, {}^{j}_{t}w^{\alpha(j)}_{\alpha(j-1)}\, {}^{m}f'_{\alpha(m)}\, {}^{m}_{t}w^{\alpha(m)}_{\alpha(m-1)} \right) \\
& + \prod_{j=l+1}^{L+1} \prod_{k=1}^{l-1} {}^{j}f'_{\alpha(j)}\, {}^{j}_{t}w^{\alpha(j)}_{\alpha(j-1)}\, {}^{l}f'_{\alpha(l)} \frac{\partial\, {}^{l}_{t}w^{\alpha(l)}_{\alpha(l-1)}}{\partial\, {}^{l}_{t}w^{a}_{b}}\, {}^{k}f'_{\alpha(k)}\, {}^{k}_{t}w^{\alpha(k)}_{\alpha(k-1)} \Bigg] \Bigg],
\end{aligned} \tag{4.8}$$

where $\alpha(L+1) = 1$ as there is exactly one neuron in the output layer. Further, it has been used that for any sequence of numbers $(a_n)_{n\in\mathbb{N}}$ we have for $n > m$ $\prod_n^m a_n = 1$ and $\sum_n^m a_n = 0$. In this case the sum is the *empty sum* and the product is the *empty product* as the set of indices n, for which the respective arithmetic operation is performed, is empty.

The first summand on the right hand side, $\mathbf{\Omega}$, vanishes trivially for all NNs studied in this thesis since the activation function acting on the output layer is the identity ${}^{L+1}f_1(x) = x$, which has a vanishing second derivative.

The algorithm for computing $\partial F^{\mathrm{NN}}_{c,\varphi}/\partial\, {}^{l}_{t}w^{a}_{b}$ that was implemented into the software

package ænet[58] is capable of computing these derivatives for an arbitrary number of hidden layers. A sketch of a naive algorithm to compute the derivatives of the atomic forces with respect to the weight parameters is given in algorithm 1. This

Algorithm 1 Naive computation of $\partial F^{\mathrm{NN}}_{c,\varphi}/\partial {}^{l}_{t}w^{a}_{b}$

```
for int i = L; i > 0; i++ do
    for int j = L; j >= i; j++ do
        for each a ∈ {1, 2, ..., D^(i) − 1} do
            for each b ∈ {1, 2, ..., D^(i−1)} do
                compute ∂^j f′/∂^i_t w^a_b
                if j == i then
                    compute ∂^j_t w^{α_j}_{α_{j−1}}/∂^i_t w^a_b
                end if
            end for
        end for
    end for
end for
```

naive algorithm computes each term contributing to the derivative consecutively. In order to accelerate the training process, it is desirable not to apply this naive algorithm but to compute the derivatives $\partial F^{\mathrm{NN}}_{c,\varphi}/\partial {}^{l}_{t}w^{a}_{b}$ for all possible combinations of a and b at once. Further, it is more time efficient to compute the derivatives of all three force components $(F_{x,\varphi}, F_{y,\varphi}, F_{z,\varphi})$ simultaneously. These methods to accelerate the naive direct force training algorithm can be applied by rewriting equation (4.8) in matrix form. A sketch of the resulting algorithm is given in 2.

Algorithm 2 Computation of $\partial \boldsymbol{F}^{\mathrm{NN}}_{\varphi}/\partial {}^{l}_{t}\boldsymbol{W}$ by matrix equations

```
for int i = L; i > 0; i++ do
    for int j = L; j >= i; j++ do
        compute ∂F^NN_φ/∂^l_t W
    end for
end for
```

4.7. Computing $\partial F^{\mathrm{NN}}_{c,\varphi} / \partial\, {}^{l}_{t}w^{a}_{b}$ for an Atomic Neural Network with Two Hidden Layers

In the following it will be demonstrated how $\partial F^{\mathrm{NN}}_{c,\varphi} / \partial\, {}^{l}_{t}w^{a}_{b}$ can be computed for an ANN with two hidden layers. The overall approach, however, is exemplary for an ANN with an arbitrary depth.

Let $\alpha(0) =: i$, $\alpha(1) =: j$, $\alpha(2) =: k$ be the indices enumerating the neurons in the layer $l \in \{0, 1, 2, 3\}$, where $\alpha(3) = 1$. Further, we define the width of the layers accordingly by $D^{(0)} =: I$, $D^{(1)} =: J$, $D^{(2)} =: K$ and note that $D^{(3)} = 1$. In the following expression each line represents one specific partial derivative, which is indicated by a comment of the form $\left| \frac{\partial A(\boldsymbol{w})}{\partial\, {}^{l}_{t}w^{a}_{b}} \right.$.

$$
\begin{aligned}
-\frac{\partial F^{\mathrm{NN}}_{c,\varphi}}{\partial\, {}^{3}_{t}w^{1}_{b}} = \underbrace{\sum_{i=1}^{I} \frac{\partial\, {}^{0}n_i}{\partial c_\varphi}}_{\boldsymbol{C}} \Bigg[& \sum_{j=1}^{J} \sum_{k=1}^{K} \underbrace{{}^{3}f''_{1}\, {}^{3}_{t}w^{1}_{k}}_{\mathcal{O}^*}\, \underbrace{{}^{2}n_b}_{\boldsymbol{I}_b}\, \underbrace{{}^{2}f'_{k}\, {}^{2}_{t}w^{k}_{j}\, {}^{1}f^{1}_{j}\, {}^{1}_{t}w^{j}_{i}}_{\boldsymbol{S}_-} && \left| \frac{\partial\, {}^{3}f'_{1}}{\partial\, {}^{3}_{t}w^{1}_{b}} \right. \\
& + \sum_{j=1}^{J} \underbrace{{}^{3}f'_{1}}_{\Omega}\, \underbrace{{}^{2}f'_{b}\, {}^{2}_{t}w^{b1}_{j}}_{\boldsymbol{B}^{**}_-}\, \underbrace{{}^{1}f'_{j}\, {}^{1}_{t}w^{j}_{i}}_{\boldsymbol{S}_-} \Bigg] && \left| \frac{\partial\, {}^{3}_{t}w^{1}_{k}}{\partial\, {}^{3}_{t}w^{1}_{b}} \right.
\end{aligned}
$$

$$
\begin{aligned}
-\frac{\partial F^{\mathrm{NN}}_{c,\varphi}}{\partial\, {}^{2}_{t}w^{a}_{b}} = \underbrace{\sum_{i=1}^{I} \frac{\partial\, {}^{0}n_i}{\partial c_\varphi}}_{\boldsymbol{C}} \Bigg[& \sum_{j=1}^{J} \sum_{k=1}^{K} \underbrace{{}^{3}f''_{1}\, {}^{3}_{t}w^{1}_{k}}_{\mathcal{O}^*}\, \underbrace{{}^{3}w^{1}_{a}\, {}^{2}f'_{a}}_{\boldsymbol{I}_a}\, \underbrace{{}^{1}n_b}_{\boldsymbol{I}_b}\, \underbrace{{}^{2}f'_{k}\, {}^{2}_{t}w^{k}_{j}\, {}^{1}f'_{j}\, {}^{1}_{t}w^{j}_{i}}_{\boldsymbol{S}_-} && \left| \frac{\partial\, {}^{3}f'_{1}}{\partial\, {}^{2}_{t}w^{a}_{b}} \right. \\
& + \sum_{j=1}^{J} \underbrace{{}^{3}f'_{1}\, {}^{3}_{t}w^{1}_{a}}_{\boldsymbol{B}_+}\, \underbrace{{}^{2}f''_{a}\, {}^{2}_{t}w^{a}_{j}}_{\mathcal{O}}\, \underbrace{{}^{1}n_b}_{\boldsymbol{I}_b}\, \underbrace{{}^{1}f'_{j}\, {}^{1}_{t}w^{j}_{i}}_{\boldsymbol{S}_-} && \left| \frac{\partial\, {}^{2}f'_{k}}{\partial\, {}^{2}_{t}w^{a}_{b}} \right. \\
& + \underbrace{{}^{3}f'_{1}\, {}^{3}_{t}w^{1}_{a}}_{\boldsymbol{B}_+}\, \underbrace{{}^{2}f'_{a}}_{\Omega}\, \underbrace{{}^{1}f'_{b}\, {}^{1}_{t}w^{b}_{i}}_{\boldsymbol{B}^{**}_-} \Bigg] && \left| \frac{\partial\, {}^{2}_{t}w^{k}_{j}}{\partial\, {}^{2}_{t}w^{a}_{b}} \right.
\end{aligned}
$$

$$
\begin{aligned}
-\frac{\partial F_{c,\varphi}^{\mathrm{NN}}}{\partial\,{}_t^1 w_b^a} = \underbrace{\sum_{i=1}^{I} \frac{\partial\,{}^0 n_i}{\partial c_\varphi}}_{\boldsymbol{C}} \Bigg[& \\
& \sum_{j=1}^{J}\sum_{k=1}^{K} \underbrace{{}^3 f_1^{''}\,{}_t^3 w_k^1}_{\boldsymbol{\mathcal{O}^*}} \underbrace{\sum_{\tau=1}^{K} {}_t^3 w_\tau^1\, {}^2 f_\tau^{'}\, {}_t^2 w_a^\tau}_{\boldsymbol{I}_a}\, {}^1 f_a^{'}\, \underbrace{{}^0 n_b}_{\boldsymbol{I}_b}\, \underbrace{{}^2 f_k^{'}\, {}_t^2 w_j^k\, {}^1 f_j^{'}\, {}_t^1 w_i^j}_{\boldsymbol{S}_-} & \left|\frac{\partial^3 f_k^{'}}{\partial\,{}_t^1 w_b^a}\right. \\
& + \sum_{j=1}^{J}\sum_{k=1}^{K} \underbrace{{}^3 f_1^{'}\,{}_t^3 w_k^1}_{\boldsymbol{S}_+}\, {}^2 f_k^{''}\, \underbrace{{}_t^2 w_j^k}_{\boldsymbol{\mathcal{O}}}\, \underbrace{{}_t^2 w_a^k}_{\boldsymbol{I}_a}\, {}^1 f_a^{'}\, \underbrace{{}^0 n_b}_{\boldsymbol{I}_b}\, \underbrace{{}^1 f_j^{'}\, {}_t^1 w_i^j}_{\boldsymbol{S}_-} & \left|\frac{\partial^2 f_k^{'}}{\partial\,{}_t^1 w_b^a}\right. \\
& + \sum_{k=1}^{K} \underbrace{{}^3 f_1^{'}\,{}_t^3 w_k^1}_{\boldsymbol{S}_+}\, \underbrace{{}^2 f_k^{'}\, {}_t^2 w_a^k}_{\boldsymbol{B}_+}\, \underbrace{{}^1 f_a^{''}\, {}_t^1 w_i^a}_{\boldsymbol{\mathcal{O}}}\, \underbrace{{}^0 n_b}_{\boldsymbol{I}_b} & \left|\frac{\partial^1 f_j^{'}}{\partial\,{}_t^1 w_b^a}\right. \\
& + \sum_{k=1}^{K} \underbrace{{}^3 f_1^{'}\,{}_t^3 w_k^1}_{\boldsymbol{S}_+}\, \underbrace{{}^2 f_k^{'}\, {}_t^2 w_a^k}_{\boldsymbol{B}_+}\, \underbrace{{}^1 f_a^{'}}_{\boldsymbol{\Omega}}\, \underbrace{\delta_{b,i}}_{\mathcal{D}_b^{**}} \Bigg] & \left|\frac{\partial\,{}_t^1 w_i^j}{\partial\,{}_t^1 w_b^a}\right.
\end{aligned}
$$

Groups of factors that are be represented by the same matrix are marked in the same color and the corresponding matrix name is given below the respective terms. Matrices with different names but the same color have the same basic structure, but differ in the index ranges of the quantities defining the matrix elements.

If a is the index of a bias neuron, any derivative $\partial F_{c,\varphi}^{\mathrm{NN}}/\partial\,{}_t^l w_b^a$ will vanish as there are no weight parameters that point to bias neurons. For the same reason matrices marked with the superscript **, e.g. $\boldsymbol{B}_-^{**}$, are zero if b is the index of a bias neuron. Matrices with the superscript *, e.g. $\boldsymbol{\mathcal{O}}^*$, vanish trivially as the transfer function used in the innermost hidden layer, i.e. the second hidden layer in this example, is always chosen to be $f(x) = x$, which has a vanishing second derivative.

4.7.1. Computing $\partial F_{c,\varphi}^{\mathrm{NN}}/\partial\,{}_t^l w_b^a$ Based on Matrix Equations

In this section the matrices required for the calculation of all terms $\partial F_{c,\varphi}^{\mathrm{NN}}/\partial\,{}_t^l w_b^a$ will be given. To increase efficiency the partial derivatives will be calculated for all neurons a in layer $l-1$ and all neurons b in layer l as well as for all 3 Cartesian

coordinates at once. In order to do so all terms depending on index a are collected in a matrix $\boldsymbol{\mathcal{A}}$ and all terms depending on index b are collected in $\boldsymbol{\mathcal{B}}$. The derivatives of interest are obtained by multiplying matrix $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ employing either the standard matrix product($\times$) or the Kronecker product($\otimes$). If the product to be used is not explicitly stated the standard matrix product is implied.

Since the number of terms to be calculated increases with decreasing layer index l, the derivatives are given in descending order, starting at the output layer($l = 3$) to facilitate comprehensibility.

Computing $\partial F^{\mathsf{NN}}_{c,\varphi} / \partial\, {}^{3}_{t}w^{1}_{b}$

All partial derivatives $\partial\, {}^{3}f'_{1} / \partial\, {}^{3}_{t}w^{1}_{b}$ vanish trivially as $\boldsymbol{\mathcal{O}}^{*}$ is the null matrix $\mathbb{0}$.

1) Computing $\partial\, {}^{3}_{t}w^{1}_{k} / \partial\, {}^{3}_{t}w^{1}_{b}$

Calculation of $\boldsymbol{\mathcal{A}}$:

Since the output layer has only a single neuron $a = 1$, $\boldsymbol{\mathcal{A}}$ is a real number. Let $\boldsymbol{\mathcal{A}} \coloneqq \boldsymbol{S}_{+}\boldsymbol{B}_{+}\boldsymbol{\Omega}\boldsymbol{I}_{a}$, where

$$\begin{aligned} \boldsymbol{S}_{+} &= \mathbb{1} \in \mathbb{R}, \\ \boldsymbol{B}_{+} &= \mathbb{1} \in \mathbb{R}, \\ \boldsymbol{I}_{a} &= \mathbb{1} \in \mathbb{R}, \\ \boldsymbol{\Omega} &= {}^{3}f'_{1} \in \mathbb{R}. \end{aligned}$$

Thus, we get

$$\boldsymbol{\mathcal{A}} = {}^{3}f'_{1}.$$

Calculation of $\boldsymbol{\mathcal{B}}$

Define $\boldsymbol{\mathcal{B}} \coloneqq \boldsymbol{I}_b \boldsymbol{B}_- \boldsymbol{S}_- C$, where

$$\boldsymbol{I}_b = \mathbb{1} \in \mathbb{R}^{K\times K},$$

$$\boldsymbol{B}_- = \begin{pmatrix} {}^{2}f_1'\, {}^{2}_{t}w_1^1 & \dots & {}^{2}f_1'\, {}^{2}_{t}w_J^1 \\ \vdots & & \vdots \\ {}^{2}f_K'\, {}^{2}_{t}w_1^K & \dots & {}^{2}f_K'\, {}^{2}_{t}w_J^K \end{pmatrix} \in \mathbb{R}^{K\times J},$$

$$\boldsymbol{S}_- = \begin{pmatrix} {}^{1}f_1'\, {}^{1}_{t}w_1^1 & \dots & {}^{1}f_1'\, {}^{1}_{t}w_I^1 \\ \vdots & & \vdots \\ {}^{1}f_J'\, {}^{1}_{t}w_1^J & \dots & {}^{1}f_J'\, {}^{1}_{t}w_I^J \end{pmatrix} \in \mathbb{R}^{J\times I},$$

$$\boldsymbol{C} = \begin{pmatrix} \frac{\partial\, {}^{0}n_1}{\partial x_c} & \dots & \frac{\partial\, {}^{0}n_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial\, {}^{0}n_I}{\partial x_c} & \dots & \frac{\partial\, {}^{0}n_I}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{I\times 3},$$

With this information $\boldsymbol{\mathcal{B}}$ is obtained:

$$\boldsymbol{\mathcal{B}} = \begin{pmatrix} \sum_{\mu=1}^{I}\sum_{\nu=1}^{J} {}^{2}f_1'\, {}^{2}_{t}w_\nu^1\, {}^{1}f_\nu'\, {}^{1}_{t}w_\mu^\nu \frac{\partial\, {}^{0}n_\mu}{\partial x_c} & \dots & \sum_{\mu=1}^{I}\sum_{\nu=1}^{J} {}^{2}f_1'\, {}^{2}_{t}w_\nu^1\, {}^{1}f_\nu'\, {}^{1}_{t}w_\mu^\nu \frac{\partial\, {}^{0}n_\mu}{\partial z_c} \\ \vdots & & \vdots \\ \sum_{\mu=1}^{I}\sum_{\nu=1}^{J} {}^{2}f_K'\, {}^{2}_{t}w_\nu^K\, {}^{1}f_\nu'\, {}^{1}_{t}w_\mu^\nu \frac{\partial\, {}^{0}n_\mu}{\partial x_c} & \dots & \sum_{\mu=1}^{I}\sum_{\nu=1}^{J} {}^{2}f_K'\, {}^{2}_{t}w_\nu^K\, {}^{1}f_\nu'\, {}^{1}_{t}w_\mu^\nu \frac{\partial\, {}^{0}n_\mu}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{K\times 3}.$$

$\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}}$ yields all terms $\partial\, {}^{3}_{t}w_k^1 / \partial\, {}^{3}_{t}w_b^1$.

Computing $\partial F_{c,\varphi}^{\mathsf{NN}}/\partial\, {}^{2}_{t}w_b^a$

All partial derivatives $\partial\, {}^{3}f_k'/\partial\, {}^{2}_{t}w_b^a$ vanish as $\boldsymbol{\mathcal{O}}^* = \mathbb{0}$.

1) Computing $\partial\, {}^2_t w^k_j / \partial\, {}^2_t w^a_b$

Calculation of $\boldsymbol{\mathcal{A}}$:

Let $\mathrm{diag}(\boldsymbol{v}) =: M$ denote a diagonal matrix with the non-zero elements being defined by the vector $\boldsymbol{v}$, where the i-th element of $\boldsymbol{v}$ defines the diagonal element in the i-th row and column M_{ii}. Using this notation we define $\boldsymbol{\mathcal{A}} := \boldsymbol{S}_+ \boldsymbol{B}_+ \boldsymbol{\Omega} \boldsymbol{I}_a$ by the matrices

$$
\begin{aligned}
\boldsymbol{S}_+ &= \mathbb{1} \in \mathbb{R} \\
\boldsymbol{B}_+ &= \left({}^3f'_1\, {}^3_t w^1_1, \quad \dots \quad , {}^3f'\, {}^3_t w^1_K \right) \in \mathbb{R}^{1\times K}, \\
\boldsymbol{\Omega} &= \mathrm{diag}\left({}^2f'_1, \quad \dots \quad , {}^2f'_K \right) \in \mathbb{R}^{K\times K}, \\
\boldsymbol{I}_a &= \mathbb{1} \in \mathbb{R}^{K\times K}.
\end{aligned}
$$

Therefore, we obtain

$$
\boldsymbol{\mathcal{A}} = \left({}^3f'_1\, {}^3_t w^1_1\, {}^2f'_1, \quad \dots \quad , {}^3f'_1\, {}^3_t w^1_K\, {}^2f'_K \right) \in \mathbb{R}^{1\times K}.
$$

Calculation of $\boldsymbol{\mathcal{B}}$:

Let $\boldsymbol{\mathcal{B}} := \boldsymbol{I}_b \boldsymbol{B}_- \boldsymbol{S}_- \boldsymbol{C}$ with

$$
\boldsymbol{I}_b = \mathbb{1} \in \mathbb{R}^{J\times J}
$$

$$
\boldsymbol{B}_- = \begin{pmatrix} {}^1f'_1\, {}^1_t w^1_1 & \dots & {}^1f'_1\, {}^1_t w^1_I \\ \vdots & & \vdots \\ {}^1f'_J\, {}^1_t w^J_1 & \dots & {}^1f'_J\, {}^1_t w^J_I \end{pmatrix} \in \mathbb{R}^{J\times I},
$$

$$\boldsymbol{S}_{-} = \mathbb{1} \in \mathbb{R}^{I\times I},$$

$$\boldsymbol{C} = \begin{pmatrix} \frac{\partial\, {}^{0}n_1}{\partial x_c} & \cdots & \frac{\partial\, {}^{0}n_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial\, {}^{0}n_I}{\partial x_c} & \cdots & \frac{\partial\, {}^{0}n_I}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{I\times 3},$$

$$\boldsymbol{\mathcal{B}} = \begin{pmatrix} \sum_{\mu=1}^{I} {}^{1}f'_{1}\, {}^{1}_{t}w_{\mu}^{1} \frac{\partial\, {}^{0}n_\mu}{\partial x_c} & \cdots & \sum_{\mu=1}^{I} {}^{1}f'_{1}\, {}^{1}_{t}w_{\mu}^{1} \frac{\partial\, {}^{0}n_\mu}{\partial z_c} \\ \vdots & & \vdots \\ \sum_{\mu=1}^{I} {}^{1}f'_{J}\, {}^{1}_{t}w_{\mu}^{J} \frac{\partial\, {}^{0}n_\mu}{\partial x_c} & \cdots & \sum_{\mu=1}^{I} {}^{1}f'_{J}\, {}^{1}_{t}w_{\mu}^{J} \frac{\partial\, {}^{0}n_\mu}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{J\times 3}.$$

$\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}}$ yields all partial derivatives of the form $\frac{\partial\, {}^{2}_{t}w_{j}^{k}}{\partial\, {}^{2}_{t}w_{b}^{a}}$.

2) Computing $\partial\, {}^{2}f'_{k}/\partial\, {}^{2}_{t}w_{b}^{a}$

Calculation of $\boldsymbol{\mathcal{A}}$

With

$$S+ = \mathbb{1} \in \mathbb{R},$$

$$\boldsymbol{B}_{+} = \begin{pmatrix} {}^{3}f'_{1}\, {}^{3}_{t}w_{1}^{1}, & \ldots & , {}^{3}f'_{1}\, {}^{3}_{t}w_{K}^{1} \end{pmatrix} \in \mathbb{R}^{1\times K},$$

$$\boldsymbol{\mathcal{O}} = \begin{pmatrix} {}^{2}f''_{1}\, {}^{2}_{t}w_{1}^{1} & \cdots & {}^{2}f''_{1}\, {}^{2}_{t}w_{J}^{1} \\ \vdots & & \vdots \\ {}^{2}f''_{K}\, {}^{2}_{t}w_{1}^{K} & \cdots & {}^{2}f''_{K}\, {}^{2}_{t}w_{J}^{K} \end{pmatrix} \in \mathbb{R}^{K\times J},$$

$$\boldsymbol{I}_{a} = \mathbb{1} \in \mathbb{R}^{J\times J},$$

computing $\boldsymbol{\mathcal{A}} \coloneqq \mathrm{diag}(\boldsymbol{S}_{+}\boldsymbol{B}_{+})\boldsymbol{\mathcal{O}}\boldsymbol{I}_{a}$ gives

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} {}^{3}f'_{1}\,{}^{3}_{t}w^{1}_{1}\,{}^{2}f''_{1}\,{}^{2}_{t}w^{1}_{1} & \dots & {}^{3}f'_{1}\,{}^{3}_{t}w^{1}_{1}\,{}^{2}f''_{1}\,{}^{2}_{t}w^{1}_{J} \\ \vdots & & \vdots \\ {}^{3}f'_{1}\,{}^{3}_{t}w^{1}_{K}\,{}^{2}f''_{K}\,{}^{2}_{t}w^{K}_{1} & \dots & {}^{3}f'_{1}\,{}^{3}_{t}w^{1}_{K}\,{}^{2}f''_{K}\,{}^{2}_{t}w^{K}_{J} \end{pmatrix} \in \mathbb{R}^{K\times J}.$$

Calculation of $\boldsymbol{\mathcal{B}}$

Define $\boldsymbol{\mathcal{B}} \coloneqq \boldsymbol{I}_{b} \otimes (\boldsymbol{B}\_\boldsymbol{S}\_\boldsymbol{C})$ with

$$\boldsymbol{I}_{b} = \begin{pmatrix} {}^{1}n_{1}, & \dots & , {}^{1}n_{J+1} \end{pmatrix} \in \mathbb{R}^{1\times(J+1)},$$

$$\boldsymbol{B}_{-} = \mathbb{1} \in \mathbb{R}^{J\times J},$$

$$\boldsymbol{S}_{-} = \begin{pmatrix} {}^{1}f'_{1}\,{}^{1}_{t}w^{1}_{1} & \dots & {}^{1}f'_{1}\,{}^{1}_{t}w^{1}_{I} \\ \vdots & & \vdots \\ {}^{1}f'_{J}\,{}^{1}_{t}w^{J}_{1} & \dots & {}^{1}f'_{J}\,{}^{1}_{t}w^{J}_{I} \end{pmatrix} \in \mathbb{R}^{J\times I},$$

$$C = \begin{pmatrix} \frac{\partial\,{}^{0}n_{1}}{\partial x_{c}} & \dots & \frac{\partial\,{}^{0}n_{1}}{\partial z_{c}} \\ \vdots & & \vdots \\ \frac{\partial\,{}^{0}n_{I}}{\partial x_{c}} & \dots & \frac{\partial\,{}^{0}n_{I}}{\partial z_{c}} \end{pmatrix} \in \mathbb{R}^{I\times 3}.$$

Thus, we obtain

$$\boldsymbol{\mathcal{B}} = \begin{pmatrix} \left({}^{1}n_{1}\boldsymbol{B}\_\boldsymbol{S}\_C\right) & \dots & \left({}^{1}n_{J+1}\boldsymbol{B}\_\boldsymbol{S}\_C\right) \end{pmatrix} \in \mathbb{R}^{J\times 3(J+1)}.$$

$\boldsymbol{\mathcal{B}}$ is a block matrix with each block ${}^{1}n_i\boldsymbol{B}\_\boldsymbol{S}\_C$ being defined as follows:

$$ {}^{1}n_i\boldsymbol{B}\_\boldsymbol{S}\_C = \begin{pmatrix} {}^{1}n_i \sum_{\mu=1}^{I} {}^{1}f_1^{'}\, {}^{1}_{t}w_{\mu}^{1} \frac{\partial^0 n_\mu}{\partial x_c} & \dots & {}^{1}n_i \sum_{\mu=1}^{I} {}^{1}f_1^{'}\, {}^{1}_{t}w_{\mu}^{1} \frac{\partial^0 n_\mu}{\partial z_c} \\ \vdots & & \vdots \\ {}^{1}n_i \sum_{\mu=1}^{I} {}^{1}f_J^{'}\, {}^{1}_{t}w_{\mu}^{J} \frac{\partial^0 n_\mu}{\partial x_c} & \dots & {}^{1}n_i \sum_{\mu=1}^{I} {}^{1}f_J^{'}\, {}^{1}_{t}w_{\mu}^{J} \frac{\partial^0 n_\mu}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{J\times 3}. $$

Matrix multiplication $\boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{B}}$ yields all partial derivatives $\frac{\partial^2 f_k^{'}}{\partial\, {}^{2}_{t}w_{b}^{a}}$.

4.7.2. Computing $\partial F_{c,\varphi}^{\mathsf{NN}}/\partial\, {}^{1}_{t}w_{b}^{a}$

The partial derivatives $\partial\, {}^{3}f_k^{'}/\partial\, {}^{2}_{t}w_{b}^{a}$ vanish as $\boldsymbol{\mathcal{O}}^* = \mathbb{0}$.

1) Computation of $\partial\, {}^{1}_{t}w_{i}^{j}/\partial\, {}^{1}_{t}w_{b}^{a}$

Calculation of $\boldsymbol{\mathcal{A}}$

Defining $\boldsymbol{\mathcal{A}} \coloneqq \boldsymbol{S}_{+}\boldsymbol{B}_{+}\boldsymbol{\Omega}\boldsymbol{I}_a$, where

$$ \boldsymbol{S}_{+} = \begin{pmatrix} {}^{3}f_1^{'}\, {}^{3}_{t}w_1^1, & \dots & , {}^{3}f_1^{'}\, {}^{3}_{t}w_K^1 \end{pmatrix} \in \mathbb{R}^{1\times K}, $$

$$ \boldsymbol{B}_{+} = \begin{pmatrix} {}^{2}f_1^{'}\, {}^{1}_{t}w_1^1 & \dots & {}^{2}f_1^{'}\, {}^{2}_{t}w_J^1 \\ \vdots & & \vdots \\ {}^{2}f_K^{'}\, {}^{1}_{t}w_1^K & \dots & {}^{2}f_K^{'}\, {}^{2}_{t}w_J^K \end{pmatrix} \in \mathbb{R}^{K\times J}, $$

$$ \boldsymbol{\Omega} = \operatorname{diag}\begin{pmatrix} {}^{1}f_1^{'}, & \dots & , {}^{1}f_J^{'} \end{pmatrix} \in \mathbb{R}^{J\times J}, $$

$$ \boldsymbol{I}_a = \mathbb{1} \in \mathbb{R}^{J\times J}, $$

it follows

$$ \boldsymbol{\mathcal{A}} = \begin{pmatrix} \sum_{\tau=1}^{K} {}^{3}f_1^{'}\, {}^{3}_{t}w_{\tau}^{1}\, {}^{2}f_{\tau}^{'}\, {}^{2}_{t}w_1^{\tau}\, {}^{1}f_1^{'}, & \dots & , \sum_{\tau=1}^{K} {}^{3}f_1^{'}\, {}^{3}_{t}w_{\tau}^{1}\, {}^{2}f_{\tau}^{'}\, {}^{2}_{t}w_J^{\tau}\, {}^{1}f_J^{'} \end{pmatrix} \in \mathbb{R}^{1\times J}. $$

Calculation of $\boldsymbol{\mathcal{B}}$

Define $\boldsymbol{\mathcal{B}} \coloneqq \boldsymbol{I}_b \boldsymbol{B}_- \boldsymbol{S}_- \boldsymbol{C}$ with

$$\begin{aligned} \boldsymbol{I}_b &= \mathbb{1} \in \mathbb{R}^{I \times I}, \\ \boldsymbol{B}_- &= \mathbb{1} \in \mathbb{R}^{I \times I}, \\ \boldsymbol{S}_- &= \mathbb{1} \in \mathbb{R}^{I \times I}, \end{aligned}$$

$$\boldsymbol{C} = \begin{pmatrix} \frac{\partial\, {}^0 n_1}{\partial x_c} & \cdots & \frac{\partial\, {}^0 n_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial\, {}^0 n_I}{\partial x_c} & \cdots & \frac{\partial\, {}^0 n_I}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{I \times 3}.$$

Thus, it follows

$$\boldsymbol{\mathcal{B}} = \begin{pmatrix} \frac{\partial\, {}^0 n_1}{\partial x_c} & \cdots & \frac{\partial\, {}^0 n_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial\, {}^0 n_I}{\partial x_c} & \cdots & \frac{\partial\, {}^0 n_I}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{I \times 3}.$$

Calculating the Kronecker product $\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}}$ yields all partial derivatives $\partial\, {}^1_t w_i^j / \partial\, {}^1_t w_b^a$.

2) Computation of $\partial\, {}^1 f_j' / \partial\, {}^1_t w_b^a$

Calculation of $\boldsymbol{\mathcal{A}}$

Define $\boldsymbol{\mathcal{A}} \coloneqq \operatorname{diag}(\boldsymbol{S}_+ \boldsymbol{B}_+) \boldsymbol{\mathcal{O}} \boldsymbol{I}_a$, where

$$\boldsymbol{S}_+ = \begin{pmatrix} {}^3 f_1' \, {}^3_t w_1^1, & \ldots & , {}^3 f_1' \, {}^3_t w_K^1 \end{pmatrix} \in \mathbb{R}^{1 \times K},$$

$$\boldsymbol{B}_+ = \begin{pmatrix} {}^2 f_1' \, {}^2_t w_1^1 & \cdots & {}^2 f_1' \, {}^2_t w_J^1 \\ \vdots & & \vdots \\ {}^2 f_K' \, {}^2_t w_1^K & \cdots & {}^2 f_K' \, {}^2_t w_J^K \end{pmatrix} \in \mathbb{R}^{K \times J},$$

$$\boldsymbol{\mathcal{O}} = \begin{pmatrix} {}^{1}f''_{1}\,{}^{1}_{t}w_{1}^{1} & \dots & {}^{1}f''_{1}\,{}^{1}_{t}w_{I}^{1} \\ \vdots & & \vdots \\ {}^{1}f''_{J}\,{}^{1}_{t}w_{1}^{J} & \dots & {}^{1}f''_{J}\,{}^{1}_{t}w_{I}^{J} \end{pmatrix} \in \mathbb{R}^{J \times I},$$

$$\boldsymbol{I}_{a} = \mathbb{1} \in \mathbb{R}^{I \times I}.$$

Therefore, we obtain

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} \sum_{\tau=1}^{K} {}^{3}f'_{1}\,{}^{3}_{t}w_{\tau}^{1}\,{}^{2}f'_{\tau}\,{}^{2}_{t}w_{1}^{\tau}\,{}^{1}f''_{1}\,{}^{1}_{t}w_{1}^{1} & \dots & \sum_{\tau=1}^{K} {}^{3}f'_{1}\,{}^{3}_{t}w_{\tau}^{1}\,{}^{2}f'_{\tau}\,{}^{2}_{t}w_{1}^{\tau}\,{}^{1}f''_{1}\,{}^{1}_{t}w_{I}^{1} \\ \vdots & & \vdots \\ \sum_{\tau=1}^{K} {}^{3}f'_{1}\,{}^{3}_{t}w_{\tau}^{1}\,{}^{2}f'_{\tau}\,{}^{2}_{t}w_{J}^{\tau}\,{}^{1}f''_{J}\,{}^{1}_{t}w_{1}^{J} & \dots & \sum_{\tau=1}^{K} {}^{3}f'_{1}\,{}^{3}_{t}w_{\tau}^{1}\,{}^{2}f'_{\tau}\,{}^{2}_{t}w_{J}^{\tau}\,{}^{1}f''_{J}\,{}^{1}_{t}w_{I}^{J} \end{pmatrix} \in \mathbb{R}^{J \times I}.$$

Calculation of $\boldsymbol{\mathcal{B}}$

Define $\boldsymbol{\mathcal{B}} \coloneqq \boldsymbol{I}_{b} \otimes (\boldsymbol{B}\_\boldsymbol{S}\_\boldsymbol{C})$, with

$$\boldsymbol{I}_{b} = \begin{pmatrix} {}^{0}n_{1}, & \dots & , {}^{0}n_{I+1} \end{pmatrix} \in \mathbb{R}^{1 \times (I+1)},$$

$$\boldsymbol{B}_{-} = \mathbb{1} \in \mathbb{R}^{I \times I},$$

$$\boldsymbol{S}_{-} = \mathbb{1} \in \mathbb{R}^{I \times I},$$

$$\boldsymbol{C} = \begin{pmatrix} \frac{\partial\, {}^{0}n_{1}}{\partial x_{c}} & \dots & \frac{\partial\, {}^{0}n_{1}}{\partial z_{c}} \\ \vdots & & \vdots \\ \frac{\partial\, {}^{0}n_{I}}{\partial x_{c}} & \dots & \frac{\partial\, {}^{0}n_{I}}{\partial z_{c}} \end{pmatrix} \in \mathbb{R}^{I \times 3}.$$

This yields

$$\boldsymbol{\mathcal{B}} = \begin{pmatrix} \left({}^{0}n_{1}\boldsymbol{B}\_\boldsymbol{S}\_\boldsymbol{C} \right) & \dots & \left({}^{0}n_{I+1}\boldsymbol{B}\_\boldsymbol{S}\_\boldsymbol{C} \right) \end{pmatrix} \in \mathbb{R}^{I \times 3(I+1)},$$

where $\boldsymbol{\mathcal{B}}$ is a block matrix with the individual blocks being defined as follows:

$$ {}^{0}n_i \boldsymbol{B}_{-} \boldsymbol{S}_{-} \boldsymbol{C} = \begin{pmatrix} {}^{0}n_i \frac{\partial\, {}^{0}n_1}{\partial x_c} & \dots & {}^{0}n_i \frac{\partial\, {}^{0}n_1}{\partial z_c} \\ \vdots & & \vdots \\ {}^{0}n_i \frac{\partial\, {}^{0}n_I}{\partial x_c} & \dots & {}^{0}n_i \frac{\partial\, {}^{0}n_I}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{I \times 3}. $$

Matrix multiplication $\boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{B}}$ yields all partial derivatives $\partial\, {}^{1}f'_j / \partial\, {}^{1}_{t}w^{a}_{b}$.

3) Computation of $\partial\, {}^{2}f'_k / \partial\, {}^{1}_{t}w^{a}_{b}$

Since we derive the activation function of node k in layer 2 with respect to the weights in layer 1, the calculation of $\boldsymbol{B}_{+}$ and $\boldsymbol{B}_{-}$ can be omitted.

Calculation of $\boldsymbol{\mathcal{A}}$

Define

$$ \boldsymbol{S}_{+} = \begin{pmatrix} {}^{3}f'_1\, {}^{3}_{t}w^{1}_{1}, & \dots & , {}^{3}f'_1\, {}^{3}_{t}w^{1}_{K} \end{pmatrix} \in \mathbb{R}^{1 \times K}, $$

$$ \boldsymbol{\mathcal{O}} = \begin{pmatrix} {}^{2}f''_1\, {}^{2}_{t}w^{1}_{1} & \dots & {}^{2}f''_1\, {}^{2}_{t}w^{1}_{J} \\ \vdots & & \vdots \\ {}^{2}f''_K\, {}^{2}_{t}w^{K}_{1} & \dots & {}^{2}f''_K\, {}^{2}_{t}w^{K}_{J} \end{pmatrix} \in \mathbb{R}^{K \times J}, $$

$$ \boldsymbol{I}_a = \begin{pmatrix} {}^{2}_{t}w^{1}_{1}\, {}^{1}f'_1 & \dots & {}^{2}_{t}w^{1}_{J}\, {}^{1}f'_J \\ \vdots & & \vdots \\ {}^{2}_{t}w^{K}_{1}\, {}^{1}f'_1 & \dots & {}^{2}_{t}w^{K}_{J}\, {}^{1}f'_J \end{pmatrix} \in \mathbb{R}^{K \times J}. $$

Writing $\boldsymbol{I}_a$ column wise into diagonal matrices and arranging these matrices in the order in which they occur in $\boldsymbol{I}_a$ as blocks of $\tilde{\boldsymbol{I}}_a$ we get:

$$ \tilde{\boldsymbol{I}}_a = \begin{pmatrix} \left(\operatorname{diag}\left({}^{2}_{t}w^{1}_{1}\, {}^{1}f'_1, & \dots & , {}^{2}_{t}w^{K}_{1}\, {}^{1}f'_1 \right) \right) \\ \vdots \\ \left(\operatorname{diag}\left({}^{2}_{t}w^{1}_{J}\, {}^{1}f'_J, & \dots & , {}^{2}_{t}w^{K}_{J}\, {}^{1}f'_J \right) \right) \end{pmatrix} \in \mathbb{R}^{JK \times K}. $$

Thus, we obtain a block matrix for $\tilde{\boldsymbol{I}}_a \boldsymbol{\mathcal{O}}$

$$\tilde{\boldsymbol{I}}_a \boldsymbol{\mathcal{O}} = \begin{pmatrix} \mathcal{I}_1 \\ \vdots \\ \mathcal{I}_J \end{pmatrix} \in \mathbb{R}^{(JK)\times J},$$

where the blocks $\mathcal{I}_j$ are defined as follows:

$$\mathcal{I}_j := \begin{pmatrix} {}_t^2 w_j^1 \, {}^1 f_j' \, {}^2 f_1'' \, {}_t^2 w_1^1 & \dots & {}_t^2 w_j^1 \, {}^1 f_j' \, {}^2 f_1'' \, {}_t^2 w_J^1 \\ \vdots & & \vdots \\ {}_t^2 w_j^K \, {}^1 f_j' \, {}^2 f_K'' \, {}_t^2 w_1^K & \dots & {}_t^2 w_j^K \, {}^1 f_j' \, {}^2 f_K'' \, {}_t^2 w_J^K \end{pmatrix} \in \mathbb{R}^{K\times J}.$$

Let $\top_{\text{block}}$ be an operator for transposing a block matrix block wise without changing the blocks' internal structures:

$$\tilde{I}_a \boldsymbol{\mathcal{O}} = \begin{pmatrix} \mathcal{I}_1 \\ \vdots \\ \mathcal{I}_J \end{pmatrix} \in \mathbb{R}^{(JK)\times J},$$
$$\left(\tilde{I}_a \boldsymbol{\mathcal{O}}\right)^{\top_{\text{block}}} := \begin{pmatrix} \mathcal{I}_1, & \dots & , \mathcal{I}_J \end{pmatrix} \in \mathbb{R}^{K\times J^2},$$

with this definition of the $\top_{\text{block}}$ operator we get

$$\boldsymbol{S}_+(\tilde{I}_a \boldsymbol{\mathcal{O}})^{\top_{\text{block}}} = \left(\begin{pmatrix} {}^3 f_1' \, {}_t^3 w_1^1, & \dots & , {}^3 f_1' \, {}_t^3 w_K^1 \end{pmatrix} \times \begin{pmatrix} \mathcal{I}_1, & \dots & , \mathcal{I}_J \end{pmatrix} \right) \in \mathbb{R}^{1\times J^2}.$$

Defining $\boldsymbol{\mathcal{A}}$ by $\boldsymbol{\mathcal{A}} := (\boldsymbol{S}_+(\tilde{I}_a \boldsymbol{\mathcal{O}})^{\top_{\text{block}}})^{\top_{\text{block}}} \in \mathbb{R}^{J\times J}$ results in:

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} \left(\sum_{\tau=1}^K {}^3 f_1' \, {}_t^3 w_\tau^1 \, {}_t^2 w_1^\tau \, {}^1 f_1' \, {}^2 f_\tau'' \, {}_t^2 w_1^\tau, & \dots & , \sum_{\tau=1}^K {}^3 f_1' \, {}_t^3 w_\tau^1 \, {}_t^2 w_1^\tau \, {}^1 f_1' \, {}^2 f_\tau'' \, {}_t^2 w_J^\tau \right) \\ \vdots \\ \left(\sum_{\tau=1}^K {}^3 f_1' \, {}_t^3 w_\tau^1 \, {}_t^2 w_J^\tau \, {}^1 f_J' \, {}^2 f_\tau'' \, {}_t^2 w_1^\tau, & \dots & , \sum_{\tau=1}^K {}^3 f_1' \, {}_t^3 w_\tau^1 \, {}_t^2 w_J^\tau \, {}^1 f_J' \, {}^2 f_\tau'' \, {}_t^2 w_J^\tau \right) \end{pmatrix}.$$

Calculation of $\boldsymbol{\mathcal{B}}$

Defining $\boldsymbol{\mathcal{B}} \coloneqq \boldsymbol{I}_b \otimes \boldsymbol{S}\_C$ with

$$\boldsymbol{I}_b = \begin{pmatrix} {}^0n_1, & \dots & , {}^0n_{I+1} \end{pmatrix} \in \mathbb{R}^{1\times(I+1)},$$

$$\boldsymbol{S}_- = \begin{pmatrix} {}^1f'_1\, {}^1_t w^1_1 & \dots & {}^1f'_1\, {}^1_t w^1_I \\ \vdots & & \vdots \\ {}^1f'_J\, {}^1_t w^J_1 & \dots & {}^1f'_J\, {}^1_t w^J_I \end{pmatrix} \in \mathbb{R}^{J\times I},$$

$$\boldsymbol{C} = \begin{pmatrix} \frac{\partial\, {}^0n_1}{\partial x_c} & \dots & \frac{\partial\, {}^0n_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial\, {}^0n_I}{\partial x_c} & \dots & \frac{\partial\, {}^0n_I}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{I\times 3},$$

we obtain

$$\boldsymbol{\mathcal{B}} = \begin{pmatrix} \begin{pmatrix} {}^0n_1 \boldsymbol{S}\_C \end{pmatrix} & \dots & \begin{pmatrix} {}^0n_{I+1} \boldsymbol{S}\_C \end{pmatrix} \end{pmatrix} \in \mathbb{R}^{J\times 3(I+1)},$$

where the individual blocks are:

$${}^0n_i \boldsymbol{S}\_C = \begin{pmatrix} {}^0n_i \sum_{\mu=1}^{I} {}^1f'_1\, {}^1_t w^1_\mu \frac{\partial\, {}^0n_\mu}{\partial x_c} & \dots & {}^0n_i \sum_{\mu=1}^{I} {}^1f'_1\, {}^1_t w^1_\mu \frac{\partial\, {}^0n_\mu}{\partial z_c} \\ \vdots & & \vdots \\ {}^0n_i \sum_{\mu=1}^{I} {}^1f'_J\, {}^1_t w^J_\mu \frac{\partial\, {}^0n_\mu}{\partial x_c} & \dots & {}^0n_i \sum_{\mu=1}^{I} {}^1f'_J\, {}^1_t w^J_\mu \frac{\partial\, {}^0n_\mu}{\partial z_c} \end{pmatrix} \in \mathbb{R}^{J\times 3}.$$

Matrix multiplication $\boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{B}}$ yields all partial derivatives $\frac{\partial\, {}^2f'_k}{\partial\, {}^1_t w^a_b}$.

5. Taylor Expansion Based Force Training Approach

The novel Taylor expansion based force training method presented in this chapter was developed and implemented into the software package ænet by the author under the supervision of A. Urban and N. Artrith (Columbia University, NY). This chapter's description of the algorithm is based on the article [59]. This novel training approach will in the following be referred to as *the Taylor expansion approach*.

As discussed before, direct force training is an effective training approach to ensure that a NN-PES represents atomic forces accurately. However, due to its high computational effort, this approach can only be employed for the construction of NN-potentials for chemical systems with a small number of atoms. Thus, for larger chemical systems an alternative force training approach is required.

The Taylor expansion based force training approach presented here tries to make use of the advantages of both the energy- and the direct force training approach: By employing a loss function that solely depends on the error of the energy predictions (see equation (3.31)) it is computationally almost as efficient as the conventional energy training method. However, instead of requiring a vast number of electronic structure calculations this novel training approach includes force information into the training process indirectly by transforming information on atomic forces into energy information, which improves the accuracy of force predictions obtained from the NN-PES even for comparatively small training sets.

This idea of translating force information to energy information was inspired by an article of Vlcek *et al.* [19], who optimized conventional molecular force-fields by a similar approach.

In the novel Taylor expansion approach additional structures $\sigma' = \{\boldsymbol{R}_i'\}$ are created from structures $\sigma = \{\boldsymbol{R}_i\}$ of the training set by slightly displacing atoms. The energy $E(\sigma')$ of the additional structure is approximated by a Taylor expansion around $E(\sigma)$:

$$E(\sigma') = E(\sigma) + \sum_i (\boldsymbol{R}_i' - \boldsymbol{R}_i)\boldsymbol{\nabla}_i E(\sigma) + \dots.$$

Truncating the Taylor series after the first order term results in:

$$E(\sigma') \approx E(\sigma) - \sum_i \boldsymbol{\delta}_i \boldsymbol{F}_i(\sigma), \tag{5.1}$$

where $\boldsymbol{\delta}_i = \boldsymbol{R}_i' - \boldsymbol{R}_i$ and $\boldsymbol{\nabla}_i E(\sigma) = -\boldsymbol{F}_i(\sigma)$.

$E(\sigma)$ and $F(\sigma)$ are known *a priori* since σ is a structure from the training set. Thus, no further electronic structure calculations, are required for extending the training set. The additionally created structures aid the prediction of forces by providing a better description of configurational space.

In this thesis two different strategies to displace atoms were studied. Both displacement strategies will be discussed in detail in the following.

5.1. Displacements along Cartesian Coordinate Axes

Vlcek *et al.* suggested to generate additional structures by selecting one atom of a given training set structures and create six additional structures by displacing only this atom by a small displacement $\pm\delta$ along all 3 Cartesian spatial directions.

Assume that atom i of structure $\sigma = \{\boldsymbol{R}_1, \dots, \boldsymbol{R}_N\}$ was chosen. Displacing atom i for example along the unit vector $\boldsymbol{e}_y$ in the Cartesian y direction, yields an additional structure σ' and its corresponding energy $E(\sigma')$ is computed as follows:

$$\sigma' = \{\boldsymbol{R}_1, \dots, \boldsymbol{R}_i - \delta\boldsymbol{e}_y, \dots, \boldsymbol{R}_N\}. \tag{5.2}$$

$$E(\sigma') = E(\sigma) - \delta F_i^y(\sigma), \tag{5.3}$$

where F_i^y is the projection of the force acting on atom i on $\boldsymbol{e}_y$. The other structure-energy pairs obtained for displacements in the remaining 5 coordinate directions are computed accordingly. In the following this displacement strategy will be referred

to as *displacement strategy (C)*.

5.2. Random Displacements

This alternative displacement approach was developed and implemented into ænet by N. Artrith and A. Urban. The author tested this displacement strategy extensively and compared the accuracy of atomic forces obtained by both displacement strategies, see chapter 7.

The motivation for introducing another displacement strategy was to allow for a more flexible sampling of configurational space by the additional structures as displacing one atom at a time along the directions of the Cartesian coordinate axes samples many very similar structures. Therefore, displacing each atom i by a small displacement $\boldsymbol{\delta}_i$, defined by a random vector, was suggested as an alternative displacement strategy. Thereby the displacement vectors where chosen such that $|\boldsymbol{\delta}_i| < \delta_{\max}$ for all atoms i. A translation of the entire structure does not change the structure's potential energy. In order to account for this, the center-of-mass displacement caused by $\boldsymbol{\delta}_i$ is subtracted from the total displacement. The energy corresponding to the displaced structure is computed by equation (5.1).

This displacement strategy will be referred to as *displacement strategy (R)* in the following.

5.3. Training Neural Networks with the Taylor Expansion Approach

In order to train NNs with the Taylor expansion approach, the standard workflow of direct force training and energy training has to be altered by adding an additional step in which the displaced structures are generated and their corresponding energies are computed. A flowchart depicting the work flow of Taylor force training is given in figure 5.1.

The performance of this force training method will depend on the choice of the displacement δ in displacement strategy (C) or the maximum displacement $\delta_{\max}$ in displacement strategy (R). The displacement is a parameter that has to be

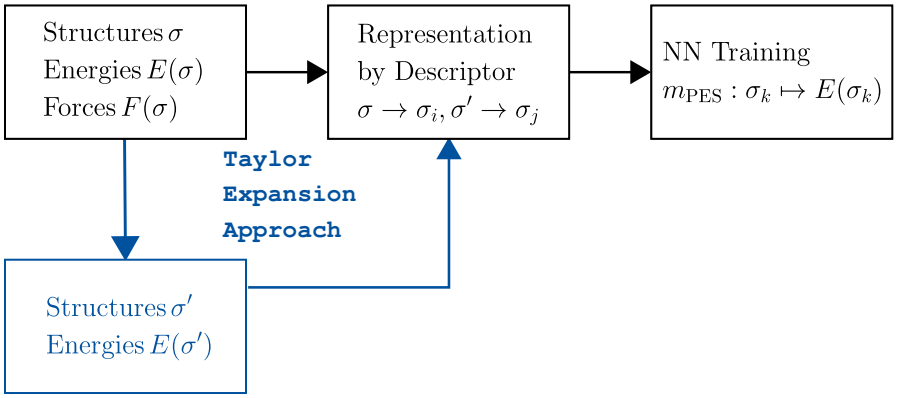


Figure 5.1.: **Workflow for force training with the Taylor expansion approach.** The training set is extended by generating additional training structures from known training structures by displacing atoms. The corresponding energy is approximated by a Taylor expansion of first order. After generating additional structures the workflow is identical to the workflow of force training: first structures are represented by descriptors and then a NN is trained to map the structure as described by the descriptor to its potential energy.

optimized to ensure accurate predictions of atomic forces.

The number of additional structures σ' which are generated by displacing atoms is defined by the user defined multiple a, where $a = X$ means that $n_{\sigma'} = X \cdot n_{\mathrm{ref}}$, where $n_{\sigma'}$ is the number of additional structures and n_{ref} is the number of structures in the original training data set. The total number of structures in the training data set is given by $n_{\mathrm{train}} = n_{\sigma'} + n_{\mathrm{ref}}$. Thus, a is a parameter that has to be optimized, because the approximation of the energy for σ' introduces additional noise to the training data set. Thus, including too many approximate data points can potentially diminish the predictive power of the NN. On the other hand, including too few approximate data points will not improve the prediction of atomic forces. Further, it is desirable to choose a as small as possible as the computational effort for training the NN scales linearly with the number of atoms in the training set. Thus, a value for a has to be determined that balances these three counteracting aspects.

The formal scaling of training ANNs employing the Taylor expansion approach to force training with the number of atoms in the reference data set is linear as is the scaling of conventional energy training with ANNs. However, generating additional

structures increases the computational effort for training. This additional effort, however, is negligible in comparison to additional electronic structure calculations required to extend a training set such that a NN-potential obtained by conventional energy training can be employed for the accurate prediction of atomic forces.

Part IV.

Applications and Results

6. Reaction Rate Constants for $CH_3OH + H \rightarrow CH_2OH + H_2$

In this chapter it is demonstrated how a NN-potential trained with energy, force and Hessian information can be used to compute highly accurate reaction rate constants applying instanton theory. The content presented in this chapter is based on published work of the author [60, 61].

6.1. Introduction

Methanol is observed very frequently in the interstellar medium (ISM). It was detected in various regions of the ISM, where it was found to be present in ices as well as in the gas phase [62]. Knowledge on relative abundances of differently deuterated species allows for the characterization of local physical quantities. Therefore, methanol has been studied extensively by experiments and simulations. Towards IRAS 16293-2422 CH_2DOH was found to be almost as abundant as CH_3OH with an abundance ratio of $[CH_2DOH]/[CH_3OH] = 0.9 \pm 0.3$. Especially the abundance of triply deuterated methanol CD_3OH being 1.4% of the abundance of CH_3OH is unexpectedly high in the proximity of these protostars [63]. Taking into account that the cosmic abundance of elemental D is about 1.5×10^{-5} the abundance of H [64], the relative abundance of triply deuterated methanol is 13 orders of magnitude higher than expected from purely statistical exchange of H and D.

Methanol was found to be heavily deuterated at the methyl group, whereas such a preference was not observed for the deuteration at the OH group [63, 65–68]. Several attempts to explain why methanol is unexpectedly strongly deuterated in various regions of the ISM have been made and a discussion of the reasons

for the preferential deuteration at the C atom is given in previous work [69, 70]. Since the strong deuteration was found solely for the methyl group and not at the oxygen atom, explanations for the strong deuteration based on high local [D]/[H] abundance ratios are insufficient.

Also the attempt to explain it by comparing the zero-point vibrational energy (ZPE) of different methanol isotopes falls short. It was shown that the ZPE of D_3OH is lower than the ZPE of CH_3OH, which could be a hint for deuterated species being favored over CH_3O. This argument, however, is not sufficient, since most deuterium is stored in the ISM in the form of HD, which has an even lower ZPE than D_3OH [69, 70].

A promising approach to shed light on the reason for the unexpectedly strong deuteration of methanol at the methyl group is the investigation of H-D exchange reactions in solid methanol [67, 69, 70]. Therefore, the investigation of rate constants of H-D exchange reactions for a variety of methanol isotopes was the focus of the work presented in this chapter. In these simulations a dust grain's ice mantle on which the reaction is supposed to take place was mimicked by an implicit surface description.

This approach yields an intuitive qualitative explanation for the D-enrichment of methanol since the rate constant for abstracting H is significantly higher than for abstracting D [67, 69, 70]. Further, since elemental D or H can be added to a hydroxymethyl radical barrierless, the rate at which methanol is formed is almost equally fast for D and H addition.

In a preceding article Goumans and Kästner studied hydrogen abstraction reactions from methanol at the CH_3 and OH group and computed reaction rate constants based on density functional theory (DFT) for temperatures between about 500 K and 30 K [67]. The work presented here aims at improving the results obtained in the aforementioned article by treating the problem with coupled cluster theory. Further, this study provides kinetic isotope effects (KIEs) and reaction rate constants for various deuteration patterns not studied in previous work, which can be incorporated in astrochemical models to enhance their predictive power. Since the article by Goumans and several other studies found that a hydrogen abstraction at the CH_3 group of methanol is significantly more likely than an abstaction at the OH group [65–68], abstractions at the OH group are not investigated.

Reaction rate constants were computed for all H/D isotope patterns on the C atom of methanol and the incoming hydrogen atom (H or D) explicitly. On top of that KIEs are given for all isotope patterns. Further, the heavy-atom $^{12}C/^{13}C$ and $^{16}O/^{18}O$ KIEs are discussed for an incoming H atom.

The different deuteration patterns under study determine the following hydrogen abstraction reactions that were investigated, where the incoming atom X can be H or D. In reactions R 1 to R 3 H is abstracted by the incoming atom X, whereas reactions R 4 to R 6 involve D abstractions. Reactions R 7 to R 8 were investigated to obtain the heavy-atom KIEs.

$$CH_3OH + X \rightarrow CH_2OH + XH \quad \text{(R 1)}$$

$$CH_2DOH + X \rightarrow CHDOH + XH. \quad \text{(R 2)}$$

$$CHD_2OH + X \rightarrow CD_2OH + XH. \quad \text{(R 3)}$$

$$CDH_2OH + X \rightarrow CH_2OH + XD. \quad \text{(R 4)}$$

$$CD_2HOH + X \rightarrow CDHOH + XD. \quad \text{(R 5)}$$

$$CD_3OH + X \rightarrow CD_2OH + XD. \quad \text{(R 6)}$$

$$^{13}CH_3\,OH + H \rightarrow {}^{13}CH_2\,OH + H_2 \quad \text{(R 7)}$$

$$CH_3\,{}^{18}OH + H \rightarrow CH_2\,{}^{18}OH + H_2 \quad \text{(R 8)}$$

Since H-D exchange reactions in solid methanol were to be simulated, unimolecular reaction rate constants are reported, as they describe how a pre-reactive complex between the incoming hydrogen atom and a methanol molecule on an ice surface decays in a Langmuir-Hinshelwood process [71]. The presence of a surface

potentially influences the reaction under study in the following ways: Firstly, it allows for the dissipation of excess energy from the reaction. Secondly, on a surface a higher concentration of reactive species than in the gas phase is to be expected, which facilitates chemical reactions. Thirdly, a surface can, in principle, influence reaction rate constants directly by increasing or decreasing reaction barriers. Finally, surfaces restrict rotations and translation of the molecules adsorbed to it, which has to be considered when computing reaction rate constants.

In the work presented here an implicit surface model was employed to represent the influence of the surface on the hydrogen abstraction reaction from methanol. This comparatively simple representation allows for the description of surface reactions by a gas-phase structural model. In this implicit surface model the restrictions of translations and rotations of a methanol molecule that is adsorbed to the surface is mimicked by keeping the rotational and translational partition functions of methanol constant between the reactant and the instanton. It was shown previously that such an implicit surface model can represent the restrictions of rotations and translations caused by a surface successfully [71].

The dissipation of the excess energy by the surface and the higher concentration of reactive species present on a surface are implicitly included in the reaction rate constants, since a thermalized ensemble is assumed in our approach and canonical reaction rate constants are computed, which are independent of the concentration of reactants.

Especially for the comparatively apolar ice surfaces on which methanol is expected to be found, like dirty CO ices, catalytic effects of the surface are assumed to be minimal. Even the catalytic effect of a water ice surface, that is in comparison to CO ices highly polar, was found to be small [71–75].

6.2. Computational Details

6.2.1. Neural Network Potentials

The NN-PESs employed in the two studies presented in this chapter were obtained by training SNNs with 2 hidden layers. The architecture of these NNs is 15-50-50-1. The number of neurons in the input layer is 15 since elongations along all 15 normal

modes of the system were used as a descriptor of the molecular structures. The number of neurons per hidden layer (50) was found by a parameter study. In this parameter study the width of both hidden layers was assumed to be equal to reduce the dimensionality of the parameter space in which the optimization has to be performed. In the parameter study widths of 1-100 neurons per hidden layer were tested to ensure a thorough sampling of the parameter space. For each choice of the width of the hidden layers 10 NNs were trained. For all NNs trained for the same width of the hidden layers the average of the RMSE of the energy predictions that were made by the NN for the test set was computed. The width that lead on average to the smallest error in the test set was considered to be optimal.

In all neurons in the hidden layers the hyperbolic tangent was used as activation function. For the output neuron the identity was chosen as activation function.

The loss function that was used for training is given in equation (4.1) and quantifies the RMSE of the energy, its gradient and its Hessian. Thereby the scaling parameters A_E, A_G and A_H were chosen as follows: $A_E = 1.0$, $A_G = 0.1$ and $A_H = 5.0$, where all parameter values are given in atomic units. Again, as for the optimization of the width of the hidden layers, a parameter study in which multiple NNs were trained for various values of the scaling parameters, was performed to find optimal values for the scaling parameters.

The initial weight and bias parameters were drawn from a uniform random distribution, with the initial values being restricted to the lie interval $[-0.5, 0.5]$. The bias on the output node ${}^3W_0^1$ was initialized with the average of the target values for the energies defined by training set. All NNs were trained for 5000 epochs to ensure a sufficient convergence of the loss function.

To accelerate the training process a linear optimization of the bias and weights acting on the output layer was performed and subsequently the remaining weight and bias parameters were optimized with the non-linear L-BFGS algorithm [37–39]. Details on this optimization approach are given in section 4.1.

Incorporating gradient and Hessian information into the training process increases the computational effort of the training process itself drastically. Apart from that, this training approach also introduces a significant computational overhead to the NN training procedure since the computational effort for the generation of the reference data is significantly higher. This is due to the fact that the

computation of gradients and Hessians in addition to the potential energy is time-consuming, especially if the training- and test set data is constructed by employing a highly accurate, but computationally expensive, electronic structure method. Nevertheless, obtaining a NN-PES, which was trained with energy gradient and Hessian information, that allows for the computation of reaction rate constants is often computationally feasible, since only a small section of the underlying PES has to be regressed by the NN-PES.

A local description of the PES is sufficient because, as explained in chapter 2, in instanton theory only harmonic deviations from the instanton, i.e. the most likely tunneling path, are considered. This implies that the computation of reaction rate constants on the basis of a NN-PES requires only an accurate description of the proximity of the transition state and the reactant state.

Since a local NN-PES is sufficient, comparatively few training data points are required to allow for an accurate regression of the PES in the area of interest. Thus, the number of *ab initio* electronic structure calculations required for generating the training- and test set is therefore manageable even for highly accurate quantum chemical methods.

The chosen level of theory at which energies as well as gradients and Hessians are to be computed is unrestricted explicitly correlated coupled-cluster theory including single and double excitations and considering triple excitations perturbatively, UCCSD(T)-F12/VTZ-F12, on a restricted Hartree-Fock basis [76–78].

Since the area of the PES which is of interest for reaction rate computations is defined by the proximity of instanton paths and the proximity of the reactant minimum, it seems to be a natural approach to first locate instantons at varying temperatures, as well as the reactant well on the PES and then choose structures in this area to construct the training- and test set.

Generation of the Training and Test Set

Due to the fact that computing instantons on UCCSD(T)-F12/VTZ-F12 level of theory for various temperatures is computationally infeasible, the training and test sets were generated in an iterative process, where first the reactant state as well as two instantons for the temperatures 285 K and 200 K were computed for

reaction H-R 1 employing density functional theory on BB1K/6-311+G** level of theory[79].

It was shown previously that the classical transition structure obtained at BB1K/6-311+G** level of theory is close to the one obtained from UCCSD(T) calculations that were extrapolated to a complete basis set [67]. All DFT calculations were performed in ChemShell [80, 81] using NWchem [82].

Along each of the first two instanton paths 20 geometries were chosen, where the structures along the instanton at 285 K were used as training examples and the remaining 20 structures along the instanton at 200 K were chosen as test examples. Further, the structure of the pre-reactive van der Waals minimum and the classical transition state were incorporated in the training set.

In order to obtain a NN-potential that describes the PES at UCCSD(T)-F12/VTZ-F12 level of theory, energies, gradients and Hessians were computed for each training- and test example at this level of theory in Molpro 2012.1 [83] via ChemShell [80, 81] with an energy threshold of 10^{-10} Hartree. Gradients and Hessians were computed in DL-FIND [84] via Chemshell [80, 81] by finite differences of energies.

Thus, the initial training set was made up of 22 training examples, which corresponds in total to $2992 = 22 \times (1 + 15 + 0.5 \times (15 \times 16))$ unique pieces of information as the structure was described by elongations along its 15 normal modes and the Hessian matrix is symmetric.

With this initial training set several NNs were trained, where each training started from different choices for the initial weight and bias parameters, but the architecture was always identical with the one given in section 6.2.1. In a second step instanton path optimizations were preformed on these NN-PESs for a large temperature range from 285 K down to $\leq$30 K.

Since for the definition of the initial training- and test set instantons at comparatively high temperature were used, it was necessary to add further data points to the training- and test set because the instanton path elongates with decreasing temperatures, which caused the instantons to extend into areas of the PES that were not yet well described by these training and test sets. These additional training- and test examples ensure that the NN-PES can describe tunneling correctly over the whole temperature range of interest and especially for low temperatures, where tunneling is expected to dominate the reaction rate constants. Thus, especially

structures along instantons computed for the medium to low temperature regime were added to the reference data.

By iterating this process of first training several NNs to approximate the PES and subsequently adding new examples to the training- and test set, that allow for a more accurate regression of the PES in the section of interest, the final training- and test set were constructed. However, in order to make accurate predictions for unimolecular reaction rate constants, in addition to reference data along instantons, also information on the immediate vicinity of the pre-reactive van der Waals complex is required as it is not sufficiently well described by structures along instantons. Therefore, training examples describing structures that were selected along paths obtained from energy minimizations starting from the end point of the instanton at 30 K, which is comparatively close to the pre-reactive complex, were added to the training- and test set.

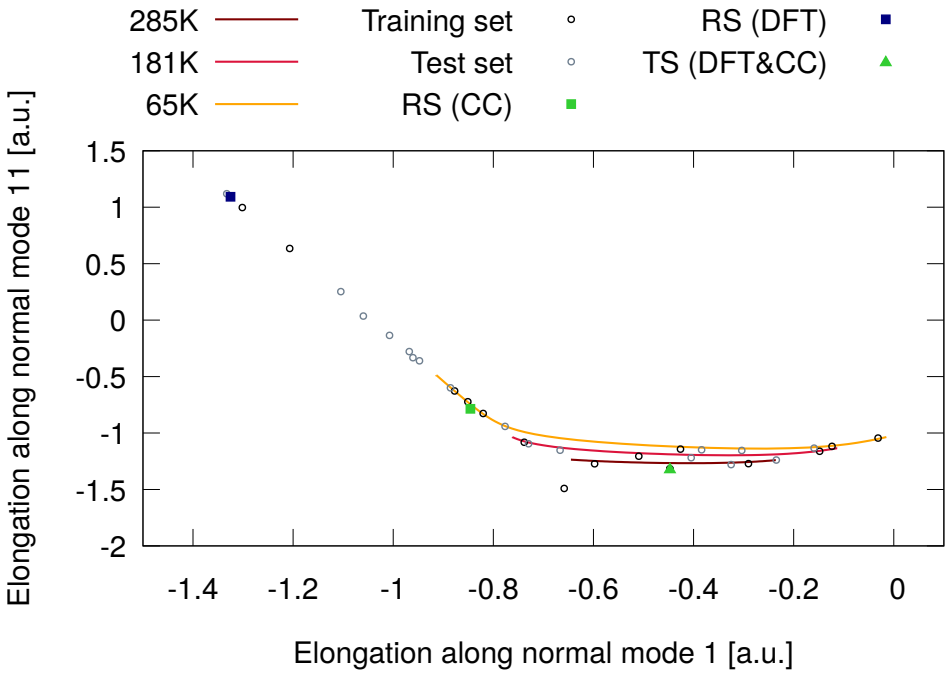


Figure 6.1.: **Distribution of training- and test set structures on the PES (open symbols).** Instanton paths are shown as lines, stationary points are given as filled symbols. All coordinates are given along normal modes 1 and 11.

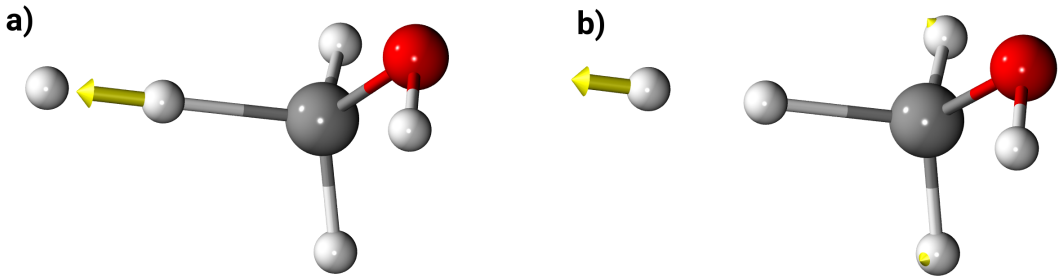


Figure 6.2.: **Normal modes 1 and 11**, on which the geometries are projected to result in figure 6.1. Taken from [60].

The final training- and test set differs slightly between the two articles [60] and [61]. In [60] the final training set was made up from 66 training examples defining 66 different structures and the corresponding total potential energy as well as all gradients and Hessians. This means that the training set contains 8976 unique pieces of information, that are available for training. The final test set, which consists of 18 test examples contains 2448 pieces of information. This final training set contains not only structures in the proximity of instantons and the reactant minimum, but also 5 structures in the proximity of the product minimum. This training- and test set was used in the work presented in section 6.3.1.

In [61] some slight adjustments have been made with respect to the training set used in [60]: firstly redundant structures, whose coordinates differed by less than 10^{-2} Bohr and the 5 structures in the proximity of the product minimum were deleted, since it is to be expected that these structures do not improve the regression of the PES by NN-potentials. On top of that, additional structures in the direct vicinity of the reactant were added to the training set to further improve the description of the reactant minimum. In total the final training set in [61] consists of 70 structures, i.e. 9520 unique pieces of information. The test set is identical in both articles. This revised training set was used to compute the rate constants and KIEs given in section 6.3.2.

The distribution of the final training- and test set structures on the PES is shown in figure 6.1 together with instantons for reaction H-R 1 at 285 K, 181 K and 65 K, that were obtained from a NN-potential trained with the final training set. In order

to facilitate the visualization of the structures not the full 15 dimensional location of the training- and test examples on PES but the projection of all coordinates of the training- and test examples as well as the three instantons on the two normal modes mode 1 and mode 11 that correspond to the movement of H_2 in the hydrogen abstraction reaction, which are depicted in figure 6.2, is shown.

Averaging Neural Network Potentials

Since the initial weight and bias parameters are chosen randomly, two NN-PES will differ from another even if all other parameters influencing the training process like the network architecture or the loss function are the same. This effect and the fact that during training weights are optimized not globally but with an optimization algorithm that searches local optima, the resulting NN-potentials will differ from each other even if tight convergence criteria and a sufficiently large training set is used.

Unfortunately it is *a priori* not obvious which NN-PES is the best approximation of the underlying PES at the target level of theory, especially since slightly different NN-PESs will often only differ slightly in the loss computed for either the test or the training set, which makes a decision for a specific NN-PES on the basis of the loss rather difficult.

However, since reaction rate constants are very sensitive with respect to changes in the curvature of the PES, small differences in the approximation of the PES can lead to comparatively large differences in the predictions of reaction rate constants. To investigate the local accuracy of the machine learned model and to obtain the best possible approximation of the PES, an average NN-PES was computed by calculating the arithmetic average of N NN-potentials. This means that the average NN-PES represents the mean energy predicted for each structure, but also its corresponding mean gradients and Hessians.

This mean NN-PES was employed for reaction rate constant calculations with the instanton method. In order to estimate how well different NN-potentials agree with each other, the standard error in the energy was computed for each structure $\boldsymbol{\sigma}$ along the instanton for reaction H-R 1 at 65 K, see figure 6.3. The standard error

is defined as follows:

$$s_{\bar{E}}(\boldsymbol{\sigma}) = \sqrt{\frac{\sum_{n+1}^{N} \left(E_n^{\mathrm{NN}}(\boldsymbol{\sigma}) - \bar{E}_n^{\mathrm{NN}}(\boldsymbol{\sigma})\right)^2}{N(N-1)}}, \tag{6.1}$$

where $E_n^{\mathrm{NN}}(\boldsymbol{\sigma})$ is the energy predicted by the n-th individual NN-PES and $\bar{E}_n^{\mathrm{NN}}(\boldsymbol{\sigma})$ is the arithmetic average of the energies predicted by all N NN-PESs for structure $\boldsymbol{\sigma}$.

If there are regions of the PES with particularly large values of $s_{\bar{E}}$, additional training points have to be added and a new set of NN-potentials has to be constructed, since a strong disagreement of different NN-potentials indicate that the NN-potentials are extrapolating energies for structures in this area of the PES due to a lack of training data.

Given that the standard error is small, averaging NN-PESs can also serve as a way to regularize NN-potentials. Averaging can regularize NN-PESs since local errors of a single NN-PES, are averaged out if the majority of the other NN-PESs entering the average make more accurate predictions in that region. In section 6.3.1 it is demonstrated that averaging several NN-PESs indeed improves the prediction of reaction rate constants.

6.2.2. Reaction Rate Constants

In order to take the quantum mechanical tunnel effect into account, reaction rate constants were computed using instanton theory [20–27]. Details on instanton theory are given in chapter 2 of this thesis.

In the work presented in this chapter instantons were, if not stated differently, discretized by 200 images. The instanton was located by employing a convergence criterion ensuring a gradient of the Euclidean action S_{E} with respect to mass-weighted coordinates of less than 5.0×10^{-11} atomic units. The applicability of such a tight convergence criterion ensures that the noise of the NN-PES and its derivatives is negligible, since such a small threshold is generally only achievable for extraordinarily smooth potentials.

Unimolecular rate constants were calculated for all isotope patterns under study using an implicit surface model in order to model Langmuir-Hinshelwood pro-

cesses [71]. Further, bimolecular gas phase reaction rate constants were computed for H-R 1. The bimolecular thermal rate constants were obtained from rate constants computed by employing a microcanonical formulation of instanton theory [21, 22, 24, 25], which allows for the description of a system in the low-pressure limit where the assumption of the pre-reactive complex being in thermal equilibrium is not valid. The microcanonical rate constants were obtained by solving the stability matrix differential equation [85].

6.3. Results and Discussion

In this section first a proof of concept is outlined in which it is demonstrated that obtaining reaction rate constants from NN-potentials is highly efficient and allows for predictions of rate constants with outstanding accuracy, see section 6.3.1. Subsequently in section 6.3.2 the work on the actual research question of this project aiming at an explanation for the unexpectedly high deuteration of methanol in various regions of the ISM is discussed in detail.

6.3.1. Bimolecular Canonical Reaction Rate Constants from Average Neural Network Potentials

The results presented in this section are published in [60]. In order to demonstrate how averaging NN-potentials can enhance the predictive power of NN-PESs, reaction rate constants for reaction R 1 were computed on several average NN-PESs.

Averaging several NN-potentials only improves predictions if this process regularizes, i.e. smooths, the resulting NN-PES. Since it is *a priori* not clear if the potential hyper surfaces are reasonably similar, the standard error of all NN-PESs entering the average NN-potential was studied. In total 103 NN-PESs were trained. In figure 6.3 the average potential energy and its corresponding standard error as defined in equation (6.1) is given along the instanton at 65 K.

The error was computed at the 200 images used to discretize the path. In figure 6.3 the standard error was multiplied by 20 to make differences in the very small standard errors visible. The first observation that can be made is that all PESs are very similar since the standard error is very small along the whole instanton.

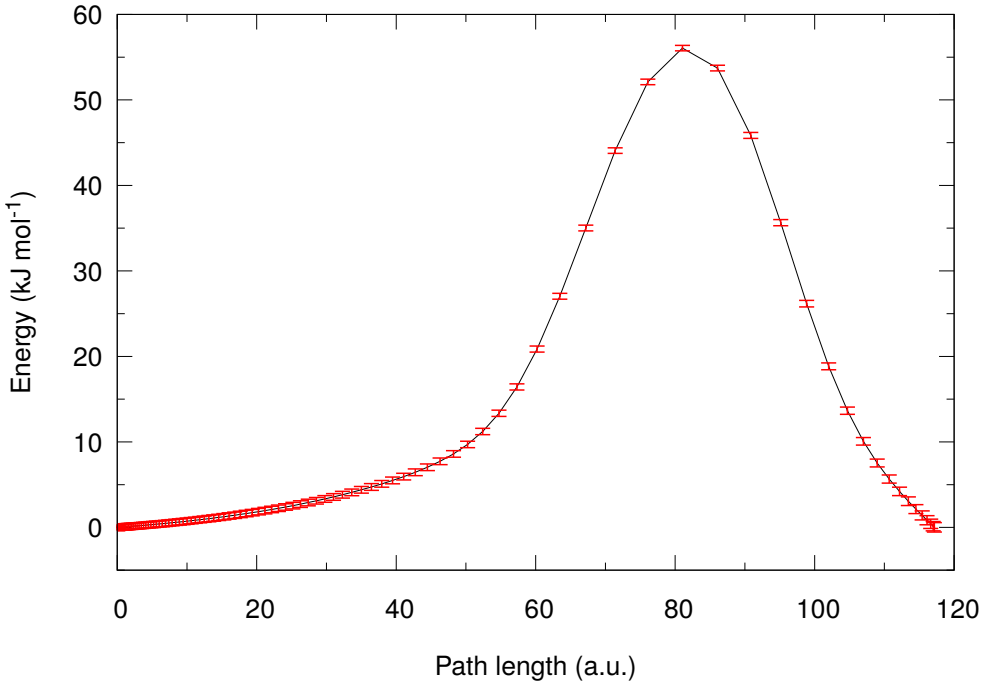


Figure 6.3.: **Average potential energy (black) ± 20 standard errors (red)** along the instanton at $T = 65\,\mathrm{K}$. The energy is given relative to the energy of the first image. Energy predictions from all 103 NN-PESs entered the average potential as well as the standard error. Taken from [60].

The maximum standard error is only about 0.027 kJ/mol.

It can further be seen that all NN-potentials agree best in the proximity of the transition structure, i.e. at path lengths of about 80 a.u., where the energy is maximal. The standard error at the transition structure is 0.016 kJ/mol and increases towards both ends of the instanton. Since there are only five training examples in the vicinity of the product minimum, the standard error is, as expected, greatest towards that end of the instanton, i.e. the right end of the instanton depicted in figure 6.3.

A similar behavior is observed towards the left end of the instanton, which is closest to the reactant minimum. However, since a comparatively large number of training examples represent the proximity of the reactant minimum, the standard error at this end of the reaction path is with about 0.018 kJ/mol smaller than the

standard error at the product end of the instanton.

It is expected that predictions of reaction rate constants will be highly accurate on the average NN-potential, since the standard error is very small along the whole instanton.

So far only differences of the NN-PESs in the predicted potential energy was studied but neither the differences in the predicted gradients nor the Hessians was discussed. Unfortunately, it is not guaranteed that a good agreement of the NN-potentials in the predicted energies implies a good agreement in the respective gradients and Hessians along the instanton as the error is studied at discrete points along the instanton and not continuously along the path. Thus, it is *a priori* unclear if the gradient and Hessian of the average NN-potential is a smooth function of the atom coordinates.

Instead of computing standard errors for all Hessian and gradient components an indirect evaluation of the deviations of the Hessians and gradients between individual NN-potentials was done by investigating how strong reaction rate constants obtained from individual NN-PESs contributing to the average potential differ from the rate constants obtained from the average NN-potential.

A graph comparing bimolecular reaction rate constants calculated on individual NN-potentials to reaction rate constants computed on the average NN-PES, constructed from all 103 individual NN-potentials is given in figure 6.4.

From this graph it can be seen that in general the reaction rate constants obtained from individual NN-potentials agree well for high temperatures , whereas with decreasing temperature the deviations in the reaction rate constants increases to differences of about a factor 5. Further, it can be seen that for some individual NN-PESs the temperature dependence of the reaction rate constant is unexpectedly rather uneven, and therefore somewhat unphysical. On the other hand it can be seen that averaging all 103 NN-potentials leads to a smooth and physically plausible description of the reaction rate constant's temperature dependence.

It is *a priori* not guaranteed that the predictive power of the average NN-potential is independent of the selection of individual NN-PESs that enter the average. To investigate the influence of that choice on the prediction of reaction rate constants on the average NN-PES, three different subsets of NN-potentials were chosen to construct average NN-PESs:

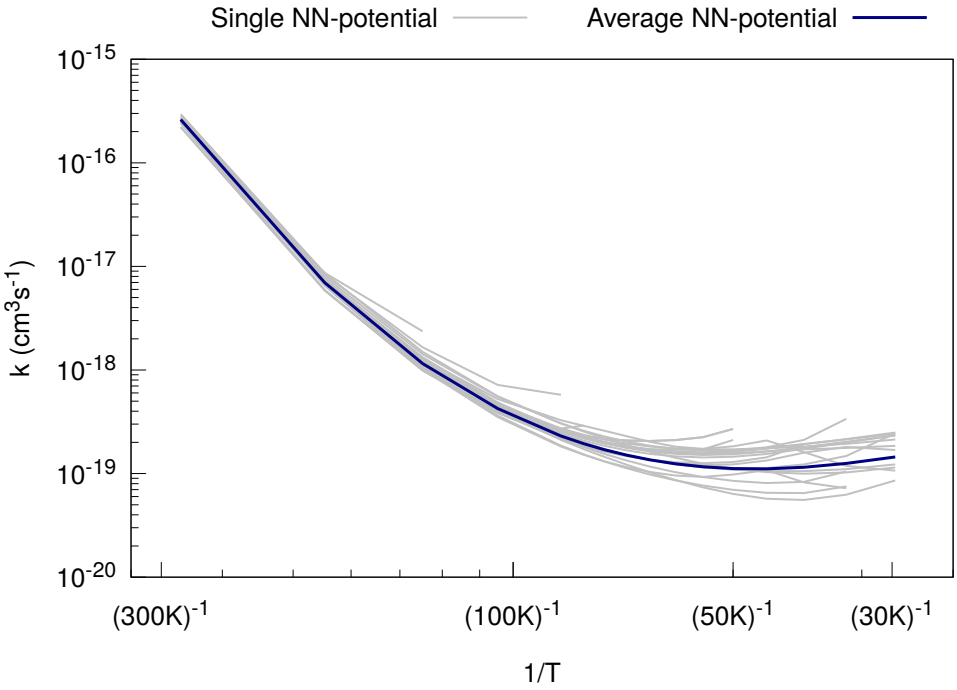


Figure 6.4.: **Comparison of bimolecular reaction rate constants** for a representative selection of individual NN-potentials (gray) to the rate constants obtained for the average NN-PES obtained by averaging over all 103 NN-PESs available.

Set 1 All NN-potentials for which at least one instanton optimization converged (103 NN-PESs).

Set 2 All NN-potentials for which the instanton optimization converged for all temperatures T with $30 \leq T \leq 285\,\mathrm{K}$ (63 NN-PESs).

Set 3 All NN-potentials for which the instanton optimization converged for all temperatures tested and for which a product state geometry could be obtained by geometry optimization (33 NN-PESs).

The largest set of NN-potentials, set 1, serves as a point of reference, since no preselection of individual NN-PESs is required, which ensures that this set represents an unbiased selection of NN-potentials. The selection of NN-potentials in set 2 ensures that only those NN-PESs which describe the shape of the barrier and the vicinity of the pre-reactive complex well are considered in the average. The

selection criterion for set 3 ensures all properties of the potentials chosen for set 2 and further ensures that the vicinity of the product state is described well by the potentials entering the average. In set 1 and set 2 are several potential hyper surfaces contained for which the product channel leads to a deep energy valley instead of a shallow van der Waals minimum. The reason for this is probably the scarcity of training data in the proximity of the product minimum. However, since for the computation of reaction rate constants no information on the PES in the proximity of the product minimum is required, the third set should in principle not improve the predictions of reaction rate constants obtained from an average NN-PES.

For the three sets of NN-potentials described above, reaction rate constants were computed temperatures between 30 K and 285 K. The reaction rate constants obtained from the three average NN-PESs constructed from set 1-3 were found to be nearly identical for the whole temperature range. A comparison of the reaction rate constants at 65 K for all 3 sets is given in table 6.1. Further, this

Table 6.1.: Reaction rate constants for different representations of the PES and deviations from the CC reference. All values at $T = 65\,\mathrm{K}$ and with 60 images.

Representation of the PES	Rate constant $[10^{-19}\ \mathrm{cm}^3/\mathrm{s}]$	Deviation from the CC reference[%]
CC reference	2.00	—
NN-PES set 1	2.03	1.50
NN-PES set 2	1.95	−2.50
NN-PES set 3	2.11	5.50

table also shows how the reaction rates obtained for the average NN-PESs compare to rate constants obtained from an instanton, which was computed directly with UCCSD(T)-F12/VTZ-F12, i.e. without a precomputed PES by calculating all energies, gradients and Hessians at this level of theory on-the-fly at $T = 65\,\mathrm{K}$. In the following this reaction rate constant will be referred to as *the CC reference*. However, since the computational effort for such an instanton optimization and the subsequent reaction rate computation is immense, the discretization of the instanton was restricted to 60 images. To allow for a direct comparison of the reaction rate

constants, the ones obtained from the three average NN-PESs constructed from set 1-3 at 65 K given in table 6.1 were recomputed for a discretization of the instanton by 60 images.

From the results given in table 6.1 it can be seen that all reaction rate constants are very similar to the CC reference, with the deviations to the CC reference being just a few percent. All errors reported are much smaller than the expected intrinsic error of the semiclassical approximation in instanton theory or the error caused by the remaining inaccuracies of the UCCSD(T)-F12 approach. Since all average NN-PESs yield similar results, in the following only the results for the average NN-PES constructed from all 103 NN-potentials available, i.e. set 1, will be discussed.

The fact that a NN-PES trained with energy, gradient and Hessian information allows for highly accurate reaction rate constant computations is very desirable, since the time effort for rate constant calculations is drastically reduced when they are obtained from NN-potentials. In comparison to performing conventional instanton computations the requirement of computational time for a rate constant calculation is reduced by about 5 orders of magnitude, if the NN-PES is readily available. For the construction of the training- and test set 84 Hessians were computed (66 for the training set and 18 for the test set). The computational effort for computing these 84 Hessians is comparable to the computational effort required for the calculation of one instanton reaction rate constant at a single fixed temperature T, where all required information on the PES is computed on-the-fly for an instanton that is discretized by 60 images if 20 optimization steps for locating the instanton are required. Under these assumptions 30 Hessian- and $20 \times 30 = 600$ gradient calculations are required. Thus, employing a NN-PES for the calculation of reaction rate constants is highly time efficient. Once a NN-potential is obtained, it allows for the discretization of the instanton to many images, like the 200 images used in the work presented here. Moreover, the optimization of instantons at more than a single temperature involves minimal additional computational cost in comparison to computing a single rate constant with the conventional approach. This approach for obtaining rate constants also enables the efficient computation of reaction rate constants for different isotopologues, because the PES is invariant with respect to the masses of the atoms and thus the same NN-PES can be utilized

to compute rate constants for all isotopologues of interest.

6.3.2. Reaction Rate Constants and Kinetic Isotope Effects

So far it was shown that obtaining reaction rate constants from NN-potentials that were trained with energy, gradient and Hessian information is in comparison to conventional instanton rate computations highly efficient and enables excellent predictions of reaction rate constants at coupled cluster level.

Unimolecular Reaction Rate Constants and Kinetic Isotope Effects

In order to investigate if H-D exchanges in solid methanol could cause the unexpectedly high deuteration of methanol in various regions of the ISM, temperature-dependent unimolecular reaction rate constants for the 12 H/D combination defined by reactions R 1-R 6 were computed. In contrast to the results shown so far, no average NN-PES but a single NN-PES which was found to be well-suited for the prediction of unimolecular rate constants was used.

Graphs of the rate constants are given in figure 6.5 and 6.6. Numbers for rate constants and KIEs at some temperatures are given in tables 6.4 and 6.5. Rate constant values for various temperatures are given in tables 6.2 and 6.3.

Firstly, it can be observed that the unimolecular rate constants are nearly temperature-independent below 40 K for all reactions under study. This temperature-independence is due to the fact that below 40 K all reactions are dominated by tunneling from the ground state of the reactant-state complex. Secondly, it was found that primary KIEs are substantial: Replacing the abstracted H by D reduces the rate constant significantly by a factor of about 3000 to 4000 at 30 K, depending on the H/D pattern of the other atoms. On the other hand, changing the abstracting atom from H to D also decreases the rate constant by a significantly smaller factor of about 6 to 9. Thus, the rate constant depends on the mass of the abstracting atom but to a much lesser extent than on the mass of the abstracted atom. This comparatively small influence can be reasoned as follows: on one hand tunneling decreases the rate constant for the hydrogen abstraction from the methyl group by an incoming D atom due to its higher mass, on the other hand, however, the vibrational zero-point energy increases the rate constant because the activation

Table 6.2.: Unimolecular rate constants (s^{-1}) for incoming H

T (K)	R1	R2	R3	R4	R5	R6	R7	R8
260	$1.908 \cdot 10^{7}$	$1.078 \cdot 10^{7}$	$4.668 \cdot 10^{6}$	$1.559 \cdot 10^{5}$	$2.693 \cdot 10^{5}$	$3.558 \cdot 10^{5}$	$1.837 \cdot 10^{7}$	$1.904 \cdot 10^{7}$
150	$4.081 \cdot 10^{5}$	$2.082 \cdot 10^{5}$	$8.194 \cdot 10^{4}$	$5.060 \cdot 10^{2}$	$7.632 \cdot 10^{2}$	$8.921 \cdot 10^{2}$	$3.865 \cdot 10^{5}$	$4.067 \cdot 10^{5}$
105	$7.516 \cdot 10^{4}$	$3.571 \cdot 10^{4}$	$1.367 \cdot 10^{4}$	$4.055 \cdot 10^{1}$	$5.736 \cdot 10^{1}$	$6.392 \cdot 10^{1}$	$7.061 \cdot 10^{4}$	$7.495 \cdot 10^{4}$
75	$2.583 \cdot 10^{4}$	$1.124 \cdot 10^{4}$	$4.411 \cdot 10^{3}$	$9.020 \cdot 10^{0}$	$1.192 \cdot 10^{1}$	$1.314 \cdot 10^{1}$	$2.416 \cdot 10^{4}$	$2.581 \cdot 10^{4}$
60	$1.489 \cdot 10^{4}$	$6.396 \cdot 10^{3}$	$2.549 \cdot 10^{3}$	$4.642 \cdot 10^{0}$	$5.860 \cdot 10^{0}$	$6.568 \cdot 10^{0}$	$1.390 \cdot 10^{4}$	$1.490 \cdot 10^{4}$
50	$1.055 \cdot 10^{4}$	$4.595 \cdot 10^{3}$	$1.808 \cdot 10^{3}$	$3.014 \cdot 10^{0}$	$3.782 \cdot 10^{0}$	$4.265 \cdot 10^{0}$	$9.822 \cdot 10^{3}$	$1.056 \cdot 10^{4}$
45	$9.131 \cdot 10^{3}$	$4.011 \cdot 10^{3}$	$1.561 \cdot 10^{3}$	$2.472 \cdot 10^{0}$	$3.120 \cdot 10^{0}$	$3.499 \cdot 10^{0}$	$8.491 \cdot 10^{3}$	$9.146 \cdot 10^{3}$
40	$8.133 \cdot 10^{3}$	$3.601 \cdot 10^{3}$	$1.384 \cdot 10^{3}$	$2.089 \cdot 10^{0}$	$2.657 \cdot 10^{0}$	$2.949 \cdot 10^{0}$	$7.560 \cdot 10^{3}$	$8.151 \cdot 10^{3}$
35	$7.508 \cdot 10^{3}$	$3.347 \cdot 10^{3}$	$1.269 \cdot 10^{3}$	$1.846 \cdot 10^{0}$	$2.366 \cdot 10^{0}$	$2.591 \cdot 10^{0}$	$6.975 \cdot 10^{3}$	$7.525 \cdot 10^{3}$
30	$7.217 \cdot 10^{3}$	$3.227 \cdot 10^{3}$	$1.210 \cdot 10^{3}$	$1.730 \cdot 10^{0}$	$2.231 \cdot 10^{0}$	$2.407 \cdot 10^{0}$	$6.691 \cdot 10^{3}$	$7.235 \cdot 10^{3}$

Table 6.3.: Unimolecular rate constants (s^{-1}) for incoming D

T (K)	R1	R2	R3	R4	R5	R6
260	$3.871 \cdot 10^{7}$	$2.216 \cdot 10^{7}$	$9.690 \cdot 10^{6}$	$3.784 \cdot 10^{5}$	$6.546 \cdot 10^{5}$	$8.654 \cdot 10^{5}$
150	$4.430 \cdot 10^{5}$	$2.317 \cdot 10^{5}$	$9.249 \cdot 10^{4}$	$9.846 \cdot 10^{2}$	$1.512 \cdot 10^{3}$	$1.793 \cdot 10^{3}$
105	$4.502 \cdot 10^{4}$	$2.170 \cdot 10^{4}$	$8.284 \cdot 10^{3}$	$3.982 \cdot 10^{1}$	$5.694 \cdot 10^{1}$	$6.348 \cdot 10^{1}$
75	$9.051 \cdot 10^{3}$	$3.893 \cdot 10^{3}$	$1.513 \cdot 10^{3}$	$4.611 \cdot 10^{0}$	$6.069 \cdot 10^{0}$	$6.551 \cdot 10^{0}$
60	$3.769 \cdot 10^{3}$	$1.576 \cdot 10^{3}$	$6.217 \cdot 10^{2}$	$1.643 \cdot 10^{0}$	$2.023 \cdot 10^{0}$	$2.209 \cdot 10^{0}$
50	$2.102 \cdot 10^{3}$	$8.886 \cdot 10^{2}$	$3.435 \cdot 10^{2}$	$8.187 \cdot 10^{-1}$	$9.850 \cdot 10^{-1}$	$1.084 \cdot 10^{0}$
45	$1.611 \cdot 10^{3}$	$6.862 \cdot 10^{2}$	$2.606 \cdot 10^{2}$	$5.823 \cdot 10^{-1}$	$7.022 \cdot 10^{-1}$	$7.669 \cdot 10^{-1}$
40	$1.270 \cdot 10^{3}$	$5.455 \cdot 10^{2}$	$2.032 \cdot 10^{2}$	$4.253 \cdot 10^{-1}$	$5.167 \cdot 10^{-1}$	$5.552 \cdot 10^{-1}$
35	$1.041 \cdot 10^{3}$	$4.503 \cdot 10^{2}$	$1.645 \cdot 10^{2}$	$3.255 \cdot 10^{-1}$	$3.984 \cdot 10^{-1}$	$4.191 \cdot 10^{-1}$
30	$8.966 \cdot 10^{2}$	$3.903 \cdot 10^{2}$	$1.400 \cdot 10^{2}$	$2.655 \cdot 10^{-1}$	$3.272 \cdot 10^{-1}$	$3.365 \cdot 10^{-1}$

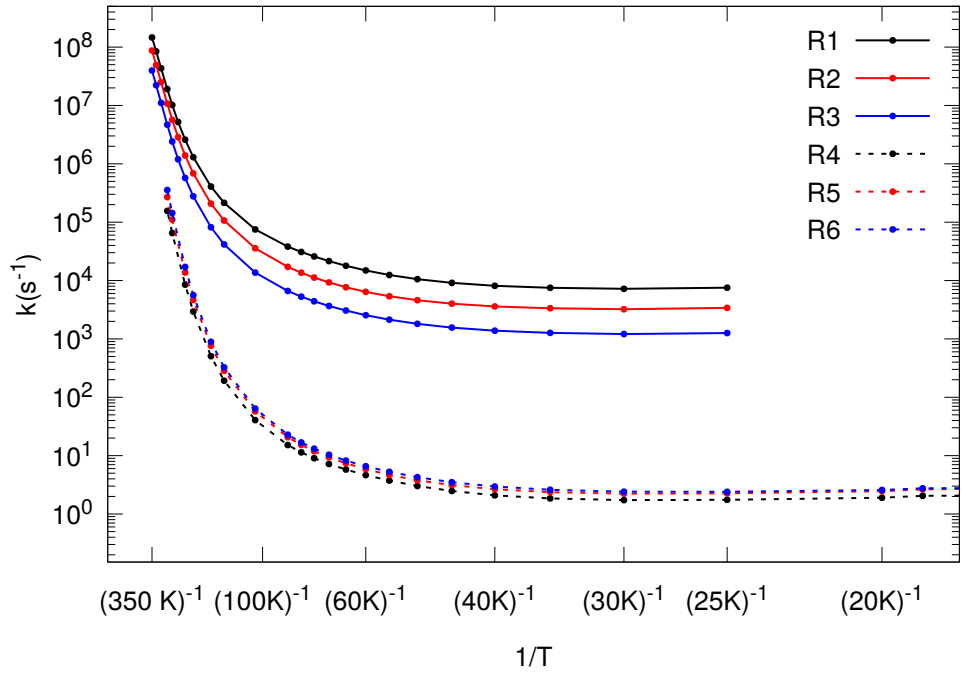


Figure 6.5.: **Rate constants** for reactions R 1-R 6 for an incoming H atom. Taken from [61].

Table 6.4.: Data for the reactions R 1 to R 8 for an incoming H atom. $E_{\mathrm{uni,act}}$ refers to the unimolecular activation energy including ZPE, T_{c} is the crossover temperature. The KIE is given with respect to H-R 1, values in parentheses refer to powers of 10.

Reactions	$E_{\mathrm{uni,act}}$ (kJ/mol)	T_{c} (K)	KIE w.r.t. H-R1 105 K	KIE w.r.t. H-R1 30 K	k at 30 K (s^{-1})
R 1: $CH_3OH + H$	33.1	357			7.22(3)
R 2: $CH_2DOH + H$	33.2	356	2.11	2.24	3.23(3)
R 3: $CHD_2OH + H$	33.4	355	5.50	5.96	1.21(3)
R 4: $CDH_2OH + H$	37.9	269	1850	4170	1.73(0)
R 5: $CD_2HOH + H$	38.1	269	1310	3230	2.23(0)
R 6: $CD_3OH + H$	38.3	268	1180	3000	2.41(0)
R 7: $^{13}CH_3OH + H$	33.1	356	1.06	1.08	6.69(3)
R 8: $CH_3{}^{18}OH + H$	33.1	357	1.00	0.998	7.24(3)

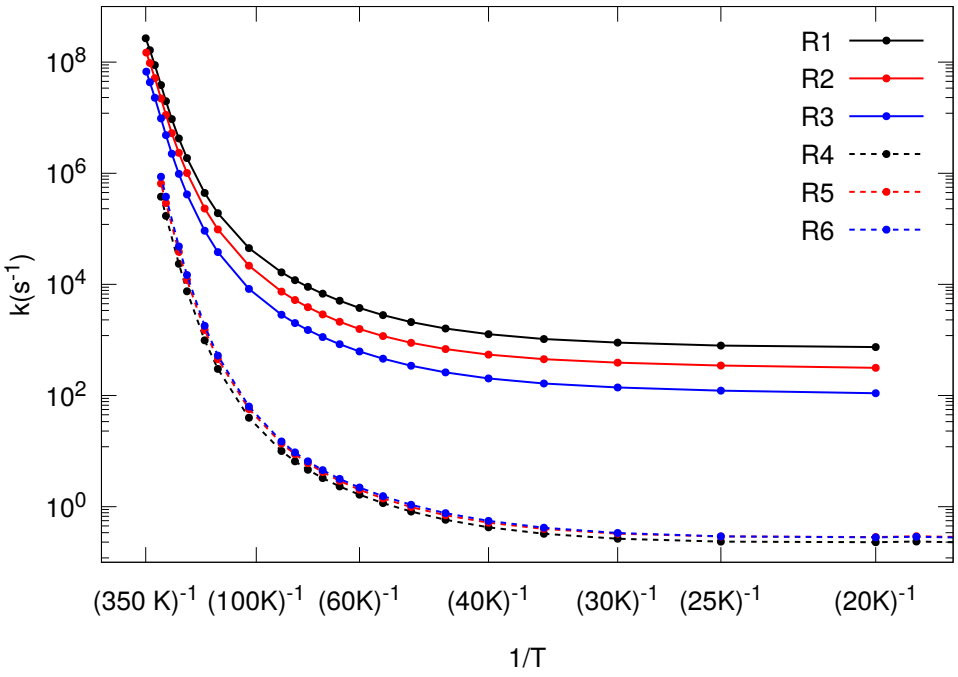


Figure 6.6.: **Rate constants** for reactions R 1-R 6 for an incoming D atom. Taken from [61].

Table 6.5.: Data for the reactions R 1 to R 8 for an incoming D atom. $E_{\mathrm{uni,act}}$ refers to the unimolecular activation energy including ZPE, T_{c} is the crossover temperature. The KIE is given with respect to D-R 1, values in parentheses refer to powers of 10.

Reactions	$E_{\mathrm{uni,act}}$ (kJ/mol)	T_{c} (K)	KIE w.r.t. D-R1 105 K	KIE w.r.t. D-R1 30 K	k at 30 K (s^{-1})
R 1: $CH_3OH + D$	30.3	353			8.97(2)
R 2: $CH_2DOH + D$	30.4	351	2.08	2.30	3.90(2)
R 3: $CHD_2OH + D$	30.5	350	5.44	6.41	1.40(2)
R 4: $CDH_2OH + D$	35.0	265	1130	3380	2.66(−1)
R 5: $CD_2HOH + D$	35.2	264	791	2740	3.27(−1)
R 6: $CD_3OH + D$	35.4	263	709	2660	3.37(−1)

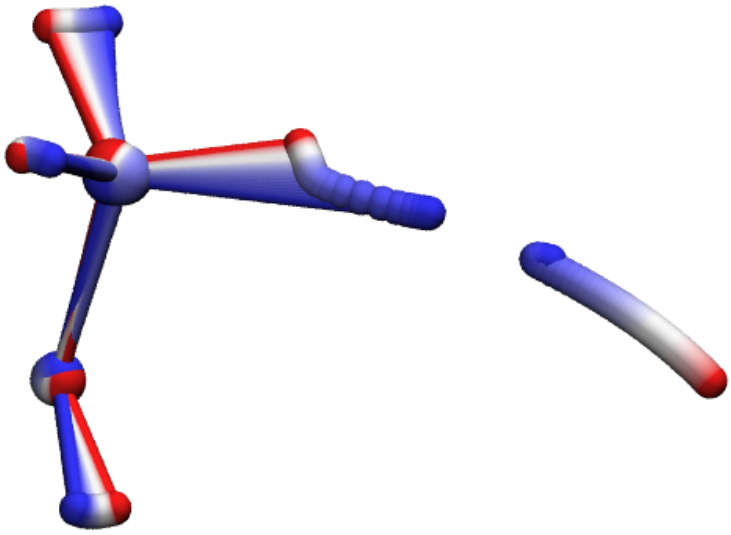


Figure 6.7.: **Instanton path for H-R 1 at** 30 K**.** The red atom positions refer to the reactant state CH_3OH + H, whereas the blue atom positions refer to the turning point of the instanton path closest to the product, CH_2OH+H_2. The molecule is displayed such that the OH group is located at the bottom of the image.

energy for reaction H-R 1 is with 33.1 kJ/mol higher than the one of reaction D-R 1, which is 30.3 kJ/mol. Thereby the increased mass of the abstracting atom has nearly no influence on the ZPE of the reactant, where this atom is bound only weakly, but it considerably reduces the ZPE of the transition state. In contrast to the significance of primary KIEs, secondary KIEs are notably less substantial. For example the rate constant of abstracting a H atom from CH_3OH by a H atom at 30 K is only about 6 times higher than the rate constant for abstracting H from CHD_2OH at the same temperature. From this difference in the rate constants, a factor of 3 originates solely from the rotational symmetry factor, which accounts for the fact that there are three H atoms that could abstracted by the incoming H atom in CH_3OH, but there is only one H atom in CHD_2OH. Thus, the effect of the different masses causes a difference in the rate constants of a factor of about 2.

In order to explain why exchanging the abstracted H atom by D leads to a substantial KIE, while exchanging the abstracting H atom by D does not change the rate constants nearly as much, one can investigate the instanton path length of the individual atoms involved in the reaction. Making use of the fact that the

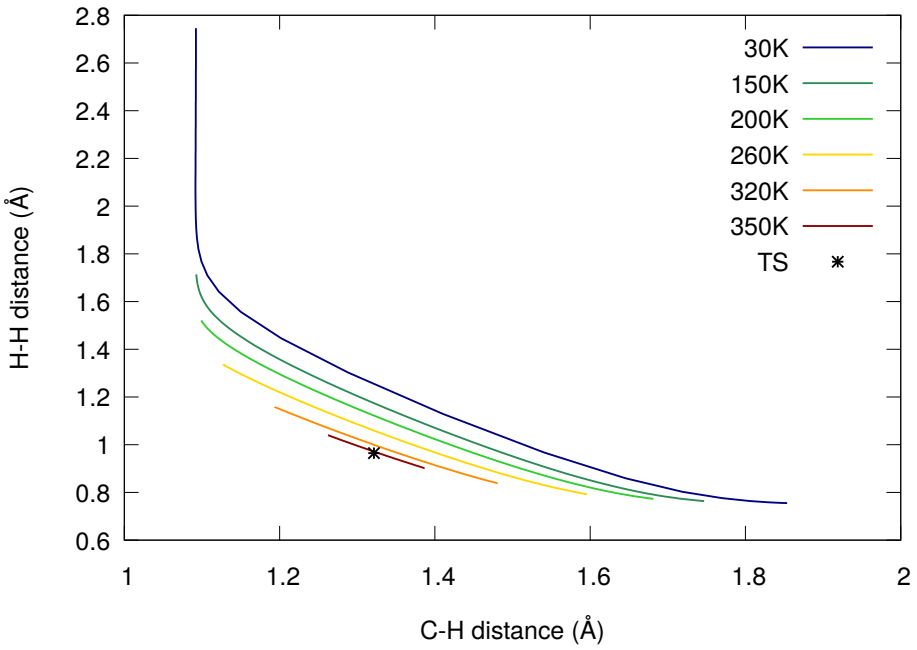


Figure 6.8.: **Instanton paths for H-R 1 at Different Temperatures.** The changes in the C–H and H–H distances of the abstraction are displaced.

instanton path is a closed Feynman path, that retraces itself between its two turning points, it is possible to study the movement of atoms during a reaction dominated by tunneling by studying the change of the molecular structures corresponding to the individual images used to discretize the instanton along the instanton path. A picture visualizing the motion of the atoms along the instanton is given in figure 6.7. In this figure the red atom positions refer to the turning point of the instanton closest the reactant, while the blue atom positions correspond to the instanton's turning point close the product valley. The energy of both turning points of the instanton is equal. Figure 6.7 shows clearly that all atoms contribute to the tunneling path for reaction H-R 1 at 30 K.

It is expected that the length of the instanton path changes with temperature. While at low temperatures the path length of the instanton becomes large such that one of its turning points gets close to the reactant minimum, the turning point will be significantly further away from the reactant minimum for large temperatures. This is due to the fact that for large temperatures the path length

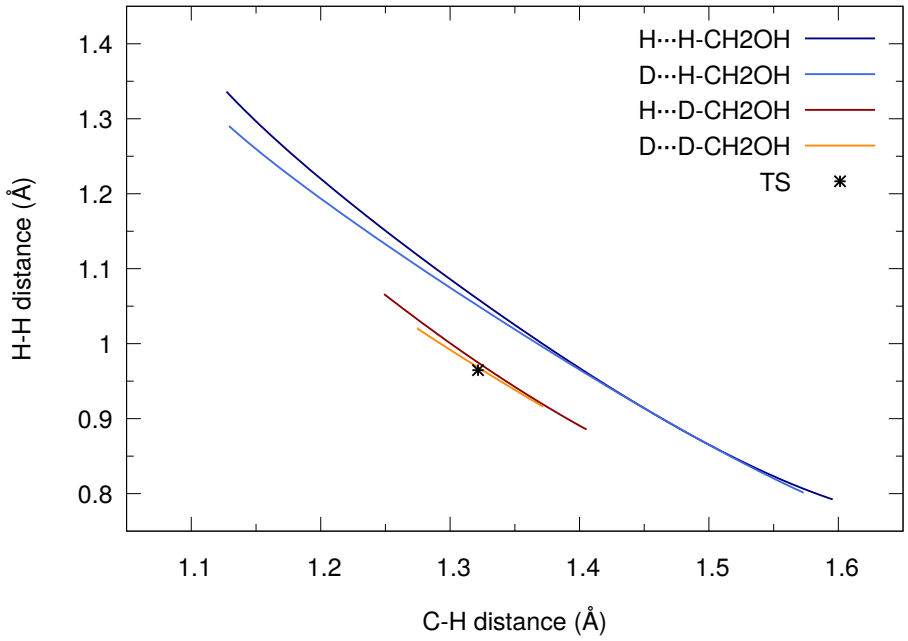


Figure 6.9.: **Instanton paths for different mass combinations at** 260 K**.**

of the instanton gets short and the instanton is located in the direct vicinity of the classical transition structure. This temperature dependence of the instanton path is depicted in figure 6.8. At 30 K the path length of the abstracted H atom is 0.92 Å, while the path length of the incoming H atom is 1.31 Å. The path lengths of the secondary hydrogen atoms are considerably smaller being 0.18 Å, 0.2 Å and 0.32 Å respectively. This difference in the path lengths of the hydrogen atoms can also be observed in figure 6.7. If an atom is replaced by a heavier isotope its contribution to the instanton path is reduced and its respective path length is effectively shortened.

In figure 6.9 it is shown how deuterium substitution influences the C-H and H-H distances along the instanton. This figure confirms that increasing the mass indeed shortens the path. Especially the H-H distance in the proximity of the reactant state (top left) is significantly reduced by increasing the mass.

This effective shortening of the path does not raise the energy a lot in case of the incoming hydrogen atom because the PES is rather flat in this area. Changes to the path of the abstracted atom, however, imply a significant influence on the energy. This is the reason for the KIE with respect to exchanging the abstracted H atom

by D being huge, as the rate constant changes by about 3.5 orders of magnitude, but the KIE for exchanging the incoming H atom by D being comparatively small, due a change in the rate constant of about 1 order of magnitude.

Heavy-Atom Kinetic Isotope Effects

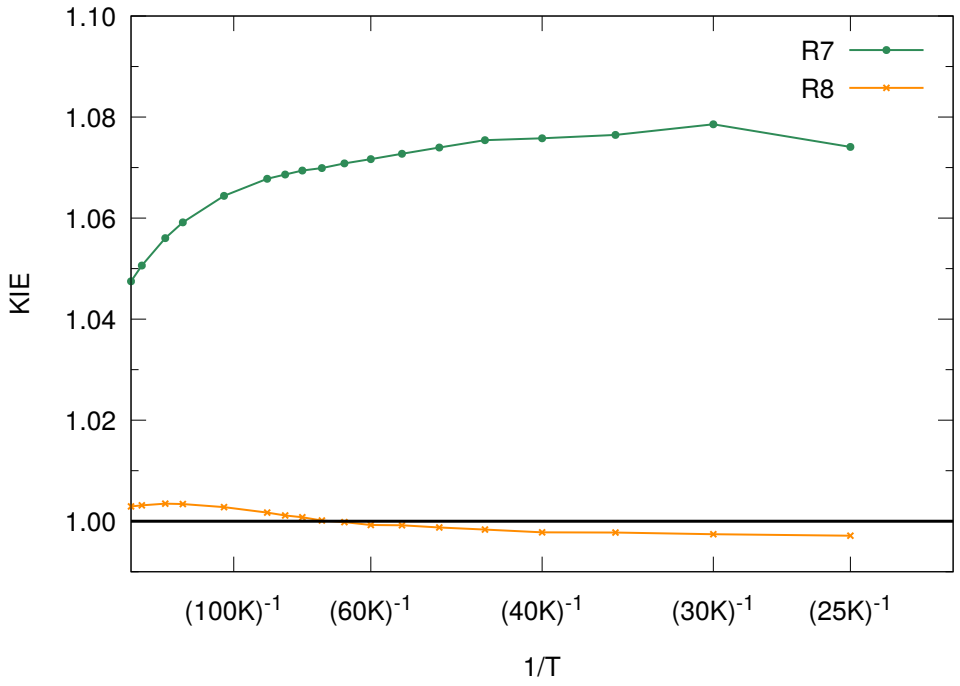


Figure 6.10.: **Heavy-atom KIEs** for the title reaction, $^{12}C/^{13}C$ in green, $^{16}O/^{18}O$ in orange. Taken from [61].

On top of KIEs for H-D exchanges also heavy-atom KIEs were computed. Of course these are, as expected, much smaller than the KIEs for H-D exchanges. In figure 6.10 the $^{12}C/^{13}C$ and $^{16}O/^{18}O$ KIEs are given for temperatures between 25 K and 200 K. From table 6.4 it can be seen that exchanging ^{16}O by ^{18}O has nearly no effect even at temperatures as low as 30 K. At 30 K even an inverse KIE of 0.9974 is predicted by our model. On top of that it was found that the oxygen atom barely moves during the reaction, having an instanton path length of merely 0.03 Å. Exchanging ^{12}C with ^{13}C, however, leads to a KIE of 1.0786 at 30 K. This KIE is, even though the masses of the carbon isotopes are very similar, larger than the one

for exchanging the oxygen atom by another isotope. This is due to the fact that the carbon atom is involved in the tunneling process enabling the hydrogen abstraction reaction since it moves during the abstraction towards the abstracted hydrogen atom. After the reaction it returns to its original position. In reaction H-R 1 at 30 K the path length of the carbon atom is with 0.15 Å comparatively long.

Bimolecular Reaction Rate Constants from a Microcanonical Formulation

So far unimolecular rate constants, which describe a thermalized surface process, were presented. These unimolecular rate constants correspond to a Langmuir-Hinshelwood mechanism [71] on the surface. However, in principle the abstraction reaction might also take place in the gas phase. In canonical instanton theory reactions with a pre-reactive minimum lead to technical difficulties, as it would be assumed implicitly that the corresponding pre-reactive complex is thermalized [85]. This assumption, however, is not valid at low pressure as under such conditions a thermalization of the pre-reactive complex is unlikely. In order to describe low-pressure bimolecular processes appropriately, microcanonical rate constants, i.e. cumulative reaction probabilities, have to be computed. These microcanonical rate constants can subsequently be used to compute thermal rate constants by using a thermal ensemble of the separated reactants. Bimolecular reaction rate constants computed with this approach are given in table 6.6 and the corresponding graphs are shown in figure 6.11. Thereby the bimolecular reaction rate constants were computed on the same NN-PES as the unimolecular rate constants.

From the graph in figure 6.11 it can be seen that the rate constants steeply decrease with decreasing temperature until the crossover temperature of 357 K is reached. At about this temperature tunneling sets in, which causes the reaction rate constant to become nearly constant for temperatures below 60 K. At very low temperature it can be seen, that the rate constants slightly increase for decreasing temperatures.

This unintuitive increase of the rate constant was observed for many bimolecular processes and is caused by a delicate balance between the additional vibrational degrees of freedom that arise when the two reactants form a single transition state and the loss of rotational and translational degrees of freedom during the

transition state formation. At high temperature the bimolecular instanton rate constants shown by the solid line in figure 6.11 are in good agreement with the experimental data by Meagher *et al.* [86] and the data by Baulch *et al.* [87], which are experiment-based.

Table 6.6.: Bimolecular rate constants for R1 with incoming H, obtained from a microcanonical formalism.

T (K)	Rate Constant (cm^3s^{-1})
343.1	$2.604 \cdot 10^{-15}$
72.4	$1.561 \cdot 10^{-18}$
41.3	$1.325 \cdot 10^{-18}$
28.4	$1.467 \cdot 10^{-18}$
21.7	$1.759 \cdot 10^{-18}$
17.6	$2.174 \cdot 10^{-18}$
14.7	$2.694 \cdot 10^{-18}$
12.8	$3.268 \cdot 10^{-18}$
11.2	$3.936 \cdot 10^{-18}$
10.0	$4.660 \cdot 10^{-18}$

Apart from experimental data, it is also possible to compare the instanton rate constants to simulation data from literature, which is only available at the high-temperature regime. To our knowledge prior to this work, no data on reaction rate constants below 180 K was available. Comparing the reaction rate constants presented in this work to other published rate constants obtained from simulations like the VTST/ZCT data by Carvalho *et al.* [88], data from quantum dynamics simulations by Kerkeni and Clary [66], as well as the expression by Meana-Pañeda *et al.* [68] and DFT-based instanton data [67], shows that all rate constants obtained by simulation considered are in general in rather good agreement with each other.

In order to estimate the influence of the H-D exchange in methanol on its deuterium fractionation, in [60] a small chemical network was studied in which the rate constants for reactions R 1-R 6 as well as secondary KIEs and surface diffusion were taken into account. A scheme of the proposed network is shown in figure 6.12. Unfortunately, relating the deuterium fractionations of methanol obtained from this rather simplistic model to the availability of H in comparison to the availability of D on the surface can't explain experimental observations

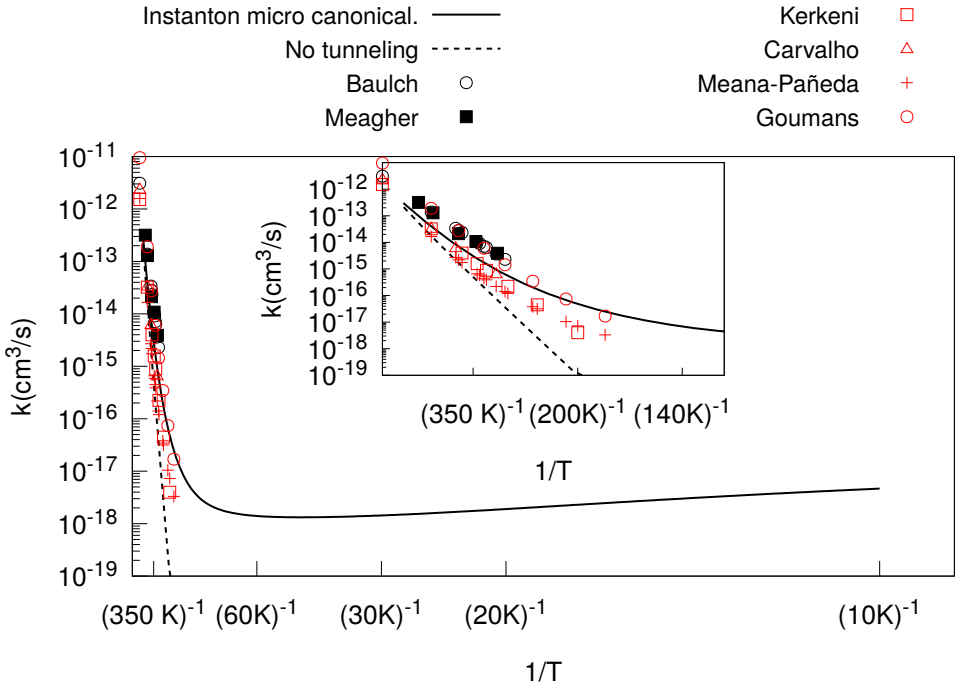


Figure 6.11.: **Bimolecular rate constants** for H-R 1 obtained from a microcanonical formulation (solid line). Experimental data is shown as black point, simulation data from literature is given as red points.

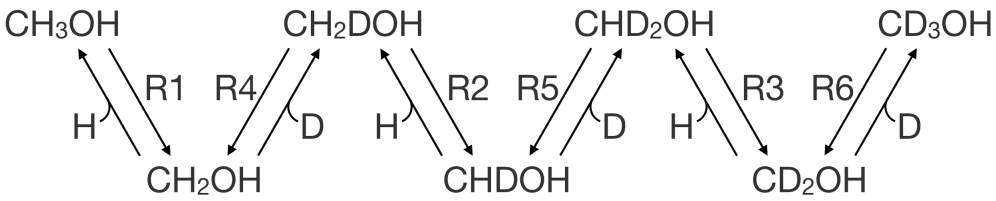


Figure 6.12.: **Kinetic model** to explain the contribution of the hydrogen abstraction reaction from the C atom of methanol to the deuteration of methanol in the ISM. Taken from [61].

towards IRAS 16293-2422 or towards the prototypical pre-stellar core L1544, as they would correspond to unexpectedly high D/H ratios. Nevertheless, the model predicts trends in the abundances of deuterated species of methanol correctly. A detailed description and discussion of the kinetic model can be found in [60].

6.3.3. Summary

In the work presented in this chapter the kinetics of $H+CH_3OH \rightarrow H_2+CH_2OH$ and all H/D isotope patterns on the CH_3 group were studied. In order to allow for an efficient computation of reaction rate constants and KIEs it was shown that incorporating information on the energy, its gradient and Hessian into the training of NN-potentials enables the computation of extraordinarily accurate reaction rate constants on the basis of NN-PESs that were trained to highly accurate coupled cluster reference data. Employing NN-PESs trained with UCCSD(T)-F12/VTZ-F12 data, unimolecular rate constants, which describe reactions on a surface by means of a Langmuir-Hinshelwood process, were computed down to temperatures as low as 25 K. Further, KIEs obtained from unimolecular rate constants are given for all studied isotope pattern at 105 K and 30 K. It has been shown that primary KIEs are substantial for all isotope patterns. The most significant KIEs were found for exchanging the abstracted H atom at the CH_3 group by D, which reduces the reaction rate constants at 30 K by as much as a factor of 3000 to 4000, depending on the isotope pattern of the other atoms. On the other hand it was shown that exchanging the incoming H atom by D has a significantly smaller influence on the reaction rate constant, with KIEs at 30 K being between 6 and 9. Further, heavy-atom KIEs were computed. It was found that exchanging ^{16}O by ^{18}O has a negligible influence on the reaction rate constant, however, exchanging ^{12}C by ^{13}C leads to a KIE of 1.0786 at 30 K. Apart from that secondary KIEs as well as bimolecular rate constants, which are relevant for the description of reactions in the gas phase in the low pressure limit, are given.

7. Predicting Correct Forces for Small Water Clusters

In this chapter it is discussed how the predictive power of NN-potentials for atomic forces is influenced by the different training approaches introduced so far. For the three training approaches discussed the accuracy of energy- and force predictions is compared. Further, this chapter aims at providing insight into which training approach should be employed given a certain well-defined problem of interest. This chapter is based on published work of the author [59].

7.1. Introduction

As mentioned previously, many quantum chemical simulations require accurate information on the PES. An important property of a surrogate model of a PES is that it describes the underlying PES such that energy conservation is ensured during simulations if it applies. Often energy conservation is not guaranteed by conventional NN-potentials obtained by energy training approaches. This is due to the fact that their force predictions are not necessarily smooth functions of the atom coordinates. Discontinuities in the predicted atomic forces, however, cause sudden, non-physical jumps in the atomic forces, which violates energy conservation.

Employing the direct force training approach can, as demonstrated in the previous chapter, ensure smooth force and Hessian surfaces and thus energy conservation. However, the computational demand for direct force training is significantly higher than the demand for standard energy training approaches, which limits its applicability to rather small molecular systems.

In order to overcome this limitation the author developed in cooperation with N. Artrith and A. Urban, who are the co-authors of the article [59] on which this

chapter is based on, an alternative force training approach. It includes information on the atomic forces indirectly by generating additional structures, which are added to the training data set. These structures are generated by displacing atoms in reference structures and extrapolating the energy of the displaced structure from the energy and forces of the original structure by employing a first order Taylor expansion of the energy. Details on this force training method are given in chapter 5.

The aim of this chapter is to provide an in-depth insight into force training by a comparison of the accuracy of force predictions obtained from NNs trained with the conventional energy training approach as well as the Taylor expansion based and direct force training approach. To provide a thorough review of each training approach, the predictive power of NNs is studied in detail for each training method individually. Further, the energy- and force predictions obtained by employing the different training approaches are compared and subsequently the most suitable fields of application for each of the three training approaches under study is discussed.

In order to allow for a thorough study and comparison of all three training approaches, a cluster of six water molecules was studied in detail. This system was chosen since it is a comparatively small chemical system for which all three training approaches can be employed with reasonable computational and time effort. A picture of a representative water cluster is shown in figure 7.1.

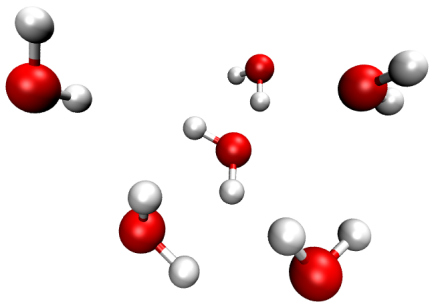


Figure 7.1.: **Exemplaric water cluster**

The article [59] covers in addition to the author's results presented in this chapter work by N. Artrith and A. Urban, who employed the Taylor expansion force training approach for training NN-potentials for more complex chemical systems as well as an analytical Lennard-Jones test system. This chapter, however, focuses on the author's work alone.

In the following fist the computational details on the NN training process, the training-, test- and validation set generation as well as the error measure employed to quantify the error in the atomic forces predicted by the NN-potentials under

study are provided. Subsequently the results obtained from all training approaches are discussed in detail and compared.

7.2. Computational Details

7.2.1. Error Measures for Atomic Forces Predicted by Neural Networks

A vector is uniquely defined by its absolute value and its direction. Making use of this property two different error measures were used to quantify the accuracy of forces $\boldsymbol{F}_\varphi^{\mathrm{NN}}$ predicted by NN-potentials.

The first error measure quantifies the absolute error in the predicted absolute value of the predicted force: $\Delta F_{\mathrm{abs}} = ||\boldsymbol{F}_\varphi^{\mathrm{REF}}| - |\boldsymbol{F}_\varphi^{\mathrm{NN}}||$, where $\boldsymbol{F}_\varphi^{\mathrm{REF}}$ is the reference obtained from *ab initio* electronic structure calculations.

In order to measure the error in the prediction of the direction of the force vector the angle enclosed by $\boldsymbol{F}_\varphi^{\mathrm{REF}}$ and $\boldsymbol{F}_\varphi^{\mathrm{NN}}$ was considered, see figure 7.2.

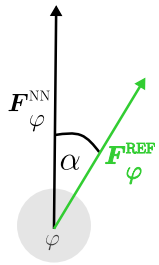


Figure 7.2.: **Error in direction of predicted force vector acting on atom φ.** The angle α enclosed by the reference force vector $\boldsymbol{F}_\varphi^{\mathrm{REF}}$ (black) and the force vector predicted by the NN $\boldsymbol{F}_\varphi^{\mathrm{NN}}$ (green) is used as one error measure for force predictions.

7.2.2. Neural Network Potentials

In the work presented ANNs with 2 hidden layers were trained to define NN-potentials. Since water clusters were studied, ANNs for two atomic species, H and O, were trained.

As a descriptor for the molecular structures symmetry functions by Behler and Parinello [3, 6] were used. The parameters defining the descriptor for water were taken from an article by T. Morawietz *et al.* [89]. The architecture of the ANNs is given by N_{symm}-10-10-1, where N_{symm} is the dimension of the feature vector. According to the parameters by T. Morawietz *et al.* N_{symm} was defined as follows: N_{symm}=30 for hydrogen and N_{symm}=27 for oxygen atoms. The width of the hidden layers was found by a parameter study, which was performed analogously to the parameter study preformed for defining the width of the hidden layers of SNNs in section 6.2.1.

The weight and bias parameters were initialized by assigning random values drawn from a uniform random distribution such that the output values of each network layer follow a zero-centered distribution with a standard deviation of 1.

For all neurons in the two hidden layers the hyperbolic tangent was used as activation function and for the output neuron the activation function was chosen to be the identity. In order to investigate the influence of the size of the reference data set on the predictive power of NN-potentials three reference data sets `train_0500`, `train_1000` and `train_2000` of different size were generated. Details on the reference data sets are discussed in the following section.

The loss functions were chosen as follows: For the direct force training approach the loss function given in equation (4.6) was used. The parameters A_E and A_G were optimized for each reference data set independently. The parameter optimization was performed analogously to the parameter optimization which was performed to define the width of the hidden layers. The optimal values for all three reference data sets are $A_E = 100$ and $A_G = 10$.

In order to obtain statistics on the accuracy of the prediction of atomic forces for the three different training approaches, the potential training was repeated for each training approach ten times starting from different random initial weight and bias parameters.

For training the respective reference data set was divided randomly into a training- and test set, where 90% of the reference structures were used as training set and the remaining 10% of the structures as test set. All NNs were trained for 5000 epochs, however, a early stopping criterion was employed which stopped the training process whenever the loss function converged. Thereby conversion is

assumed if the loss of the training set is not changing for three consecutive epochs. All weight and bias parameters were optimized with the L-BFGS algorithm [37–39].

The computation of the feature vectors, the NN training and the evaluation of the resulting NN-potentials was done with the program package ænet [58].

Generation of the Training, Test and Validation Set

The reference data set was generated in an iterative process. In a first step three *ab initio* molecular dynamics simulations (AIMD) were performed in DL_POLY [90] via Chemshell [80, 81]. As *ab initio* method the semiempirical GFN-xTB [91, 92] method was used. In total a time of 30 ps was simulated employing a time step of 0.5 fs. The simulations were performed in the NVT ensemble. In two of these simulations a temperature of 300 K was simulated, whereas the temperature simulated in the third AIMD simulation was 800 K. For all MD simulations performed for this work the temperature was controlled by a Nosé-Hoover thermostat. In order to retain a cluster of water molecules, a harmonic restraint was applied on all atoms to keep the molecules confined to a sphere with a radius of about 5 Å. For the restraint a harmonic force was applied to every atom that is more than 0.0005 Å away from the central atom, which was defined to be the first atom in the structure file. The force constant of the harmonic force was chosen as 190.5 eV. For each reference structure obtained from the MD trajectories a single point calculation of the energy and atomic forces at BLYP-D3/def2-TZVP [93–95] level of theory was performed to obtain a more precise description of the energy and forces. The single point computations were done in Turbomole [96] via Chemshell [80, 81]. In order to ensure maximum decorrelation of the reference data, the training examples were chosen from the trajectory at regular time intervals. Due to the small number of reference structures available, additional structures of a single water molecule with varying bond angle and bond lengths were added to the preliminary data set to ensure that the NN-PES predicts physically reasonable structures of the water molecules. The 170 artificial structures describing the energy- and force change induced by varying the bond angle of a single water molecule was generated by varying the bond angle in constant steps of 10° between 10° and 180°. The 15 artificial structures describing the energy difference for varying the O-H bond

length for a water molecule were generated by fixing the length of both O-H bonds to the same value and varying the bond length between 0.7 Å and 2.1 Å in constant steps of 0.1 Å. To match the reference data obtained from MD-simulations for each of these artificial, i.e. non-physical, structures the energies and atomic forces were computed in Turbomole [96] via Chemshell [80, 81] at BLYP-D3/def2-TZVP level of theory. By this approach a first reference data set consisting of 3335 structures and the respective target energies and forces was generated.

In a second step this reference data was used for training two NNs, to obtain a first estimate of the NN-potential for a cluster of six water molecules at temperatures of about 300 K. Employing these NN-potentials to describe the energetics and forces further MD simulations were run at 300 K for 75 ps to obtain additional structures for the reference data set. Apart from that all other parameters for the MD simulation were chosen as described before. The energies and forces of the newly obtained structures along the MD trajectories were, again, re-computed at BLYP-D3/def2-TZVP level of theory. By iterating the process of obtaining a preliminary, incrementally improved NN-PES and obtaining additional reference data by running MD simulations employing the NN-potential several times, a reference data set consisting of 7755 structures was constructed. Due to the size of this reference data set it was not necessary anymore to include reference data on single water molecules in order to ensure a realistic representation of individual water molecules in the cluster. Therefore, these artificial structures were removed from the reference data set. Thus, a final reference data set consisting of 7570 structures and their respective energies and atomic forces was obtained. From this pool of data three reference data sets were assembled: `train_0500` consisting of 471, `train_1000` consisting of 943 and `train_2000` consisting of 1886 training examples. Thereby the training sets were constructed such that `train_0500` $\subsetneq$ `train_1000` $\subsetneq$ `train_2000`.

Further, an independent validation set consisting of 2000 reference structures not included in any of the three training sets `train_0500` - `train_2000` was constructed.

7.3. Results

7.3.1. Force Predictions from Neural Networks Trained with the Energy Training Approach

If the energy training approach is employed, the common approach to improve the reliability of force predictions by NN-PESs is to increase the size of the reference data set. The influence of the reference data set's size on the predictive power of NN-PESs is investigated by training NN-potentials with the energy training approach using one of the three training sets and quantifying the error in the force predictions by employing the error measures introduced in section 7.2. If not stated differently, all errors are reported for the independent validation data set.

In figure 7.3 the distribution of the error in the absolute value of the energies of the structures in the validation data set is shown. This figure displays the error distribution as a histogram. To allow for a straight forward comparison of the three histograms shown, the histograms were computed employing the same bins and are scaled such that the overall maximum frequency of occurrence is 1. In this chapter all graphs of error distributions of the absolute value of the energy or of the absolute force are scaled in the same way.

As expected figure 7.3 shows that the absolute value of the energies of the structures in the validation data set improves systematically with increasing training set size. However, all errors in the energy reported in this graph are very small and well below chemical accuracy, which is $1.0\,\mathrm{kcal/mol} \approx 0.04\,\mathrm{eV}$.

The error in the force predictions are shown in figure 7.4 and figure 7.5.

From figure 7.4 it can be deduced that employing the energy training approach with the smallest training set `train_0500` leads to a wide distribution of errors in the predicted absolute values of atomic forces. The pronounced tail of this error distribution is particularly unfavorable as it implies that the error in the predicted absolute value of the force is for a large number of atoms more than $1.0\,\mathrm{eV/Å}$. Fortunately, it can be seen that the tail of the error distribution is reduced significantly if the training set size is increased and that the accuracy of the predicted absolute value of the force can be systematically improved by increasing the size of the training set.

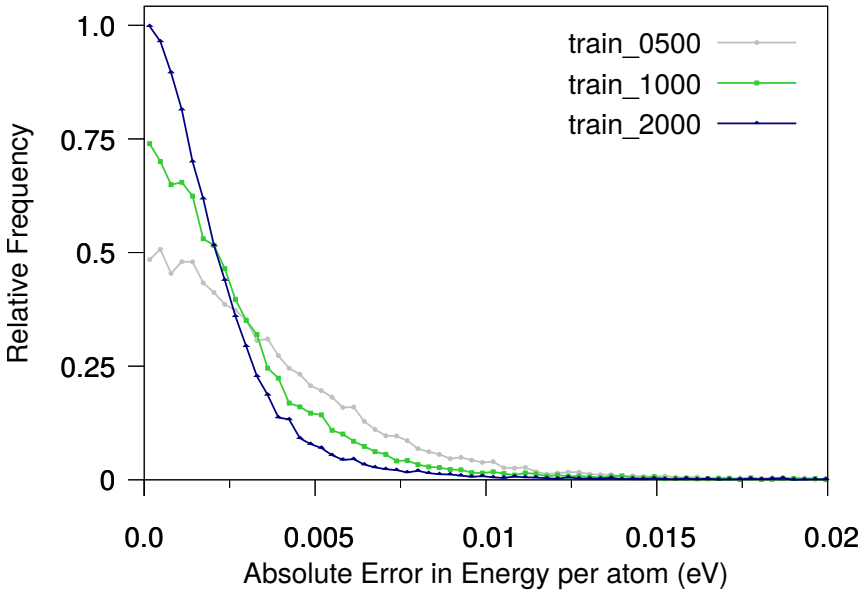


Figure 7.3.: **Distribution of the error in the absolute value of the energy** for training on all three training sets applying the energy training approach.

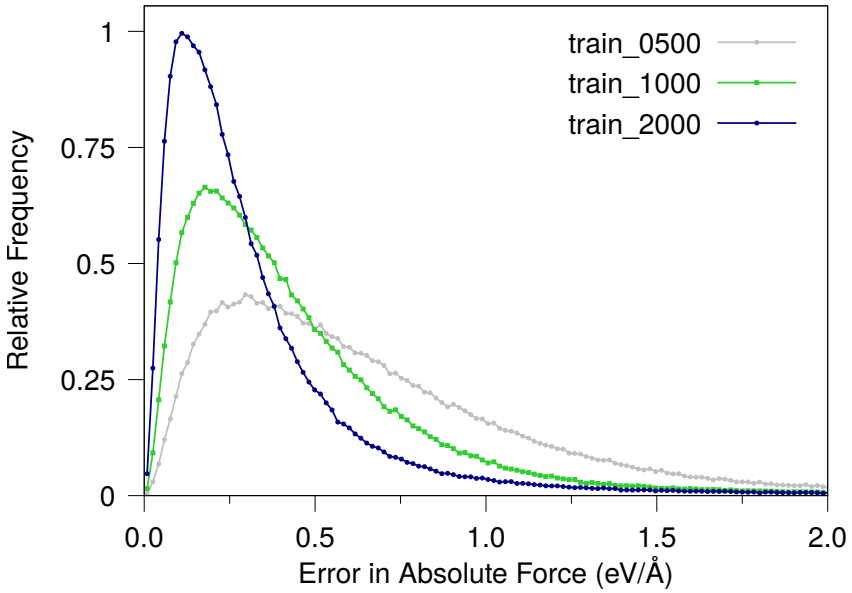


Figure 7.4.: **Distribution of the error in the absolute force** for training on all three training sets applying the energy training approach.

In figure 7.5 a heat map of the error in the direction of the predicted atomic forces is shown. In this heat map the relative frequency of the angle α enclosed by the reference force vector obtained from *ab initio* electronic structure calculations and the force vector predicted by the NN-PES is given as a function of the absolute value of the reference force vector. This description of the error has been chosen, since for forces with small absolute values even tiny fitting errors can lead to large angles $\alpha \approx 180°$ while the same fitting error will lead to small $\alpha \approx 0$ if the force's absolute value is sufficiently large. Thus, this depiction helps to estimate how drastic the fitting errors in the predicted direction are. In the heat maps high relative frequencies are shown in red and yellow, whereas low relative frequencies are colored in shades of gray, with white indicating relative frequencies of about 0. In order to allow for a more detailed investigation of the force error also the mean absolute error (MAE) of the atomic forces is reported to facilitate the discussion of the findings since the differences in the heat maps are often subtle.

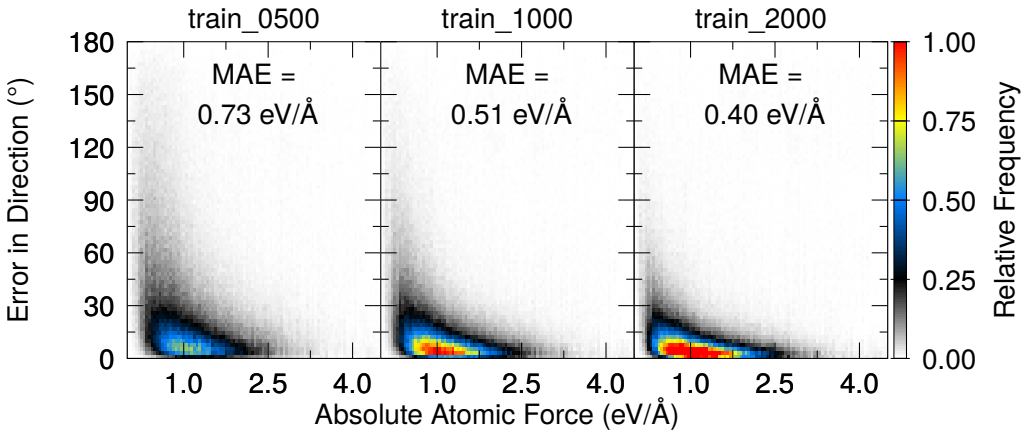


Figure 7.5.: **Distribution of the error in the force direction** for training on all three training sets applying the energy training approach. Taken from [59].

Figure 7.5 shows that the predictive power of NN-PESs concerning the force vector's direction increases with increasing absolute value of the force vector. This is reflected in the fact that the relative frequencies for large values of α, i.e. large errors, are drastically decreasing with increasing absolute values of the force vectors. Assuming that all force predictions are made with similar errors this is expected. For

train_0500 the error distribution exhibits a shallow maximum between 0° and 25° depending on the absolute value of the force vector. For forces with small absolute values of less than 1 eV/Å, however, the errors in the direction scatter strongly and are in some cases almost 180°. The heat maps for the two larger training data sets train_1000 and train_2000 show that the scattering in the predicted directions of the force vectors, especially for forces with small absolute values, is notably reduced. This is also mirrored in the fact that the relative frequency of $\alpha \leq 15°$ is significantly increased, especially for forces with absolute values between about 0.5 eV/Å and 1.75 eV/Å. The relative improvement of the force predictions can also be observed when comparing the MAE of the atomic forces. The MAE is maximal for the smallest training set with a value of 0.73 eV/Å and minimal for the largest training data set, with the MAE being 0.4 eV/Å, which is 45% smaller than the MAE obtained for train_0500.

All results presented so far show that the conventional approach of increasing the size of the reference data set indeed improves the accuracy of energy- and force predictions from NN-potentials for the water cluster under study.

7.3.2. Force Predictions from Neural Networks Trained with the Taylor Expansion Based Approach

Meta-Parameters for the Taylor expansion Approach

In order to apply the Taylor expansion force training approach its optimal meta-parameters defining the number of additionally created structures as well as the displacement δ or δ_{max} have to be determined. The number of additional structures is defined by the multiple a, which was defined in section 5.3. For example, if the train_0500 training set consisting of 471 structures is used then $a = 10$ means that $10 \cdot 471 = 4710$ structures will be generated by displacing structures of the train_0500 training set.

For the parameter optimization of the multiple a values $a \in \{1, 5, 11, 22, 32, 54\}$ were considered. For the displacement strategy (R) δ_{max} was varied between 0.0005 Å and 0.08 Å and for displacement strategy (C) displacements δ between 0.005 Å and 0.08 Å were studied.

For the three training data sets all possible choices of the parameter pair (a,δ) and (a,$\delta_{\max}$) were studied. Ten NNs were trained for each parameter pair to obtain insight into the statistics of the resulting NN-potentials' performance. In figure 7.6 the MAE of the atomic forces obtained from 10 NN-PESs trained with the Taylor expansion approach employing displacement strategy (R) and training on the `train_0500` reference data set is shown. The MAE reported in the figure is given relative to the MAE obtained from 10 NN-potentials trained with the energy training approach.

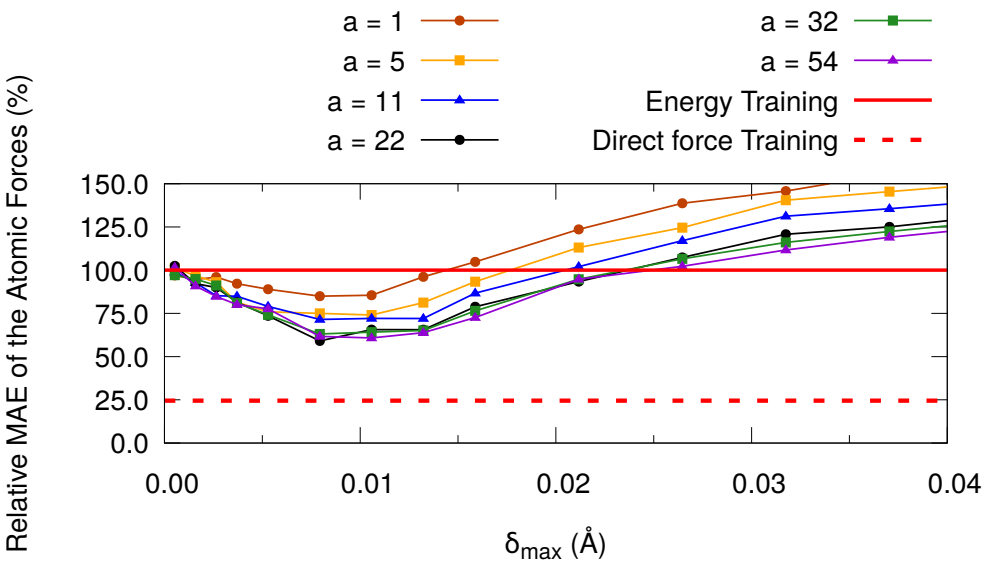


Figure 7.6.: **Relative mean absolute error (MAE) of the atomic forces for different force training parameters.** The MAE is reported relative to the MAE obtained by energy training (solid red line). Results are shown for displacement strategy (R) applied to reference data set `train_0500`. The dashed red line indicates the MAE obtained by direct force training. Taken from [59].

Figure 7.6 shows that the displacements $\delta_{\max}$ which lead to the greatest reduction of the MAE are independent of a and about $\delta_{\max} = 0.01\,\text{Å}$. The increase of the MAE for large displacements is caused by the fact that the first-order Taylor expansion breaks down under these conditions. It can be observed that the MAE decreases for increasing a if $a \leq 22$. For multiples a bigger than 22 no notable

Table 7.1.: Optimal meta-parameters for applying the Taylor expansion approach

Reference data set	Displacement strategy (C)		Displacement strategy (R)	
	δ [Å]	multiple a	$\delta_{\max}$ [Å]	multiple a
train_0500	0.03	11	0.008	22
train_1000	0.04	32	0.010	32
train_2000	0.04	32	0.010	32

further improvement could be detected. The MAE probably does not decrease further since a further increase of a increases the number of reference structures. This should in principle improve the predictions made by NNs but, since the Taylor expansion approach introduces noise to the reference data, also the amount of noisy data in the reference data set is increased, which inhibits a further reduction of the MAE. Analogous analyses have been performed for the two larger training sets `train_1000` and `train_2000` as well as for all three reference data sets if NN-potentials are trained by employing displacement strategy (C). The optimal choices for the displacement and the multiple a are summarized in table 7.1.

For the water cluster studied here, the optimal multiple a for displacement strategy (C) is smaller or equal to the optimal a for the random displacement strategy (R). This trend, however, does not apply in general. In [59] it is shown that for increasing structure size the optimal multiple a increases more rapidly for displacement strategy (C) than for displacement strategy (R).

Force Predictions Obtained from the Taylor expansion Approach

This section deals with a detailed analysis of the force predictions that can be obtained from NN-PESs trained with the Taylor expansion force training approach. First the force predictions obtained from NN-potentials trained with the Taylor expansion approach are compared for different displacement strategies. For this comparison ten NN-PESs were trained for each reference data set and each displacement strategy using the respective optimized meta-parameters for the displacement and multiple a. From these NN-potentials atomic force predictions were obtained and subsequently the improvement of the force predictions was quantified by comparing these forces to forces obtained from NN-potentials trained with the energy training

approach.

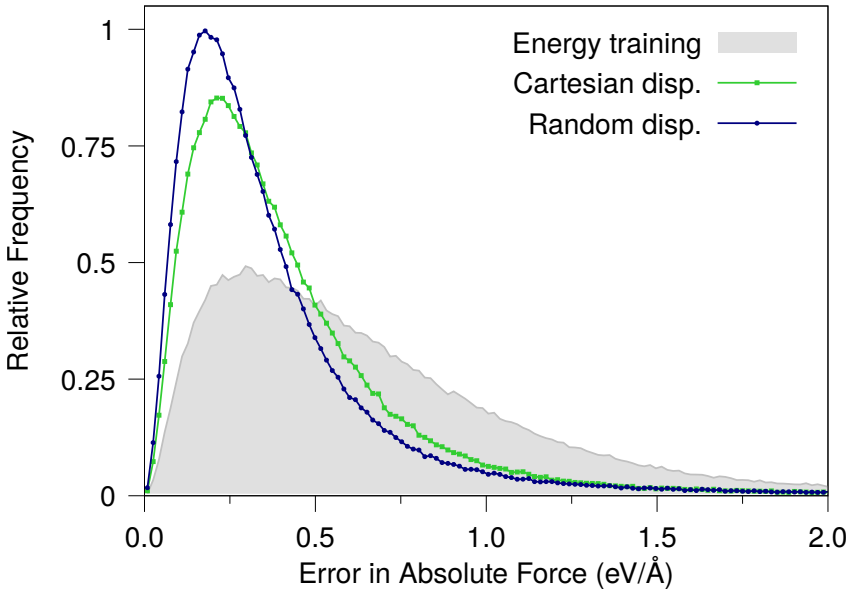


Figure 7.7.: **Distribution of the error in the absolute force** for training on the `train_0500` training set applying the energy training approach (gray) and approximate force training by the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy.

In figure 7.7 the distributions of the errors in the predicted absolute values of the atomic forces for the validation set are shown. NN-potentials that were used for the force predictions presented in this graph were trained with the `train_0500` reference data set. From this graph it can be seen that the distribution obtained by employing the energy training approach is significantly broader than the error distributions obtained from Taylor expansion based force training. Thus, the errors in the absolute force are drastically reduced by employing the Taylor expansion approach for both displacement strategies. This notable improvement of the force predictions is also mirrored in a reduction of the MAE of the atomic forces given for displacement strategy (R) in figure 7.8 with respect to the MAEs given in figure 7.5. Comparing the MAEs for each training set individually shows that the most significant relative improvement of the force predictions is found for the smallest training set `train_0500`. For this training set the MAE of the atomic

forces is reduced by training with the Taylor expansion approach from 0.53 eV/Å obtained by energy training to 0.42 eV/Å, which implies an error reduction of about 42%. For `train_1000` employing the Taylor expansion approach leads to a reduction of the MAE by 29% and for `train_2000` the mean absolute error is reduced by 20%. It can further be seen that for both displacement strategies there are hardly any atoms with absolute force errors of more than 1.0 eV/Å, which is mirrored in the fact that the tail of error distributions, which is very pronounced in case of energy training, has nearly disappeared. Further, it can be observed that the distribution of the error in the predicted absolute value of the force does not depend strongly on the displacement strategy. However, it can be seen that the random displacement strategy (R) leads, at least for the system under study, to slightly better force predictions. Therefore, in the following the discussion of the Taylor expansion approach will imply that displacement strategy (R) is used. On top of that it can be observed that the prediction of the direction of the force vectors is notably improved if the Taylor expansion approach is used by comparing the results obtained for energy training shown in figure 7.5 to the results obtained employing the Taylor expansion force training approach given in figure 7.8. Comparing these heat maps and the respective MAEs of the forces one can observe that training NN-PESs with the smallest training set `train_0500` employing the Taylor expansion approach results in force predictions which are of similar quality than the force predictions obtained from NN-potentials trained with the energy training approach with the four times larger `train_20000` reference data set. This finding can also be interpreted such that the Taylor expansion based force training approach reduces the number of reference data points required for a given target accuracy of the force predictions.

In order to quantify the influence of the noise in the energy information introduced into the reference data set by employing the Taylor expansion approach, the distribution of the error in the prediction of the absolute energy values obtained for structures in the training set shown in figure 7.9 a) is compared to the same error distribution obtained for the validation set given in figure 7.9 b). In figure 7.9 a) it is shown that the Taylor expansion based force training approach introduces, as expected, some noise into the reference data, which is mirrored in the fact that the distributions of the absolute error in the energy obtained by training

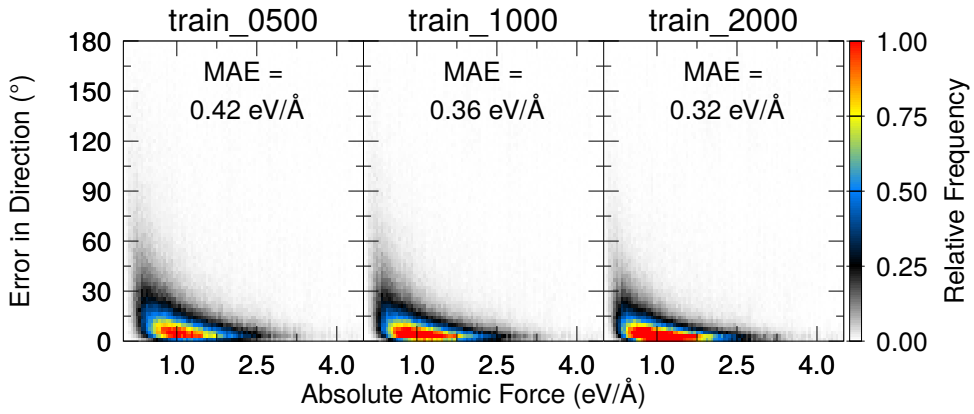


Figure 7.8.: **Distribution of the error in the force direction** for training on all three training sets applying the Taylor expansion training approach employing the random displacement strategy. Taken from [59].

NN-potentials with the Taylor expansion approach have maxima which are shifted slightly to the right in comparison to the error distribution obtained from energy training. This slight negative effect on the energy predictions, however, is limited to structures from the training set. The data shown in figure 7.9 b) demonstrates that there is almost no difference between the error distributions of the absolute error in the energy obtained from the two training approaches for the validation set. Thus, the noise introduced by this force training approach can be considered insignificant since it influences only structures in the training set and does not affect the NN-potentials predictive power for general chemical structures. Moreover, even the errors obtained for the training set are well below chemical accuracy.

7.3.3. Force Predictions from Neural Networks Trained with the Direct Force Training Approach

As a second point of reference the results obtained with the direct force training approach shall be discussed and compared to the results obtained from the Taylor expansion- and energy training approach.

In figure 7.10 the distribution of the error in the absolute force is given for all three reference data sets. This graph shows that the error distribution is nearly identical for all three training data sets. This trend was also found for the distribution of

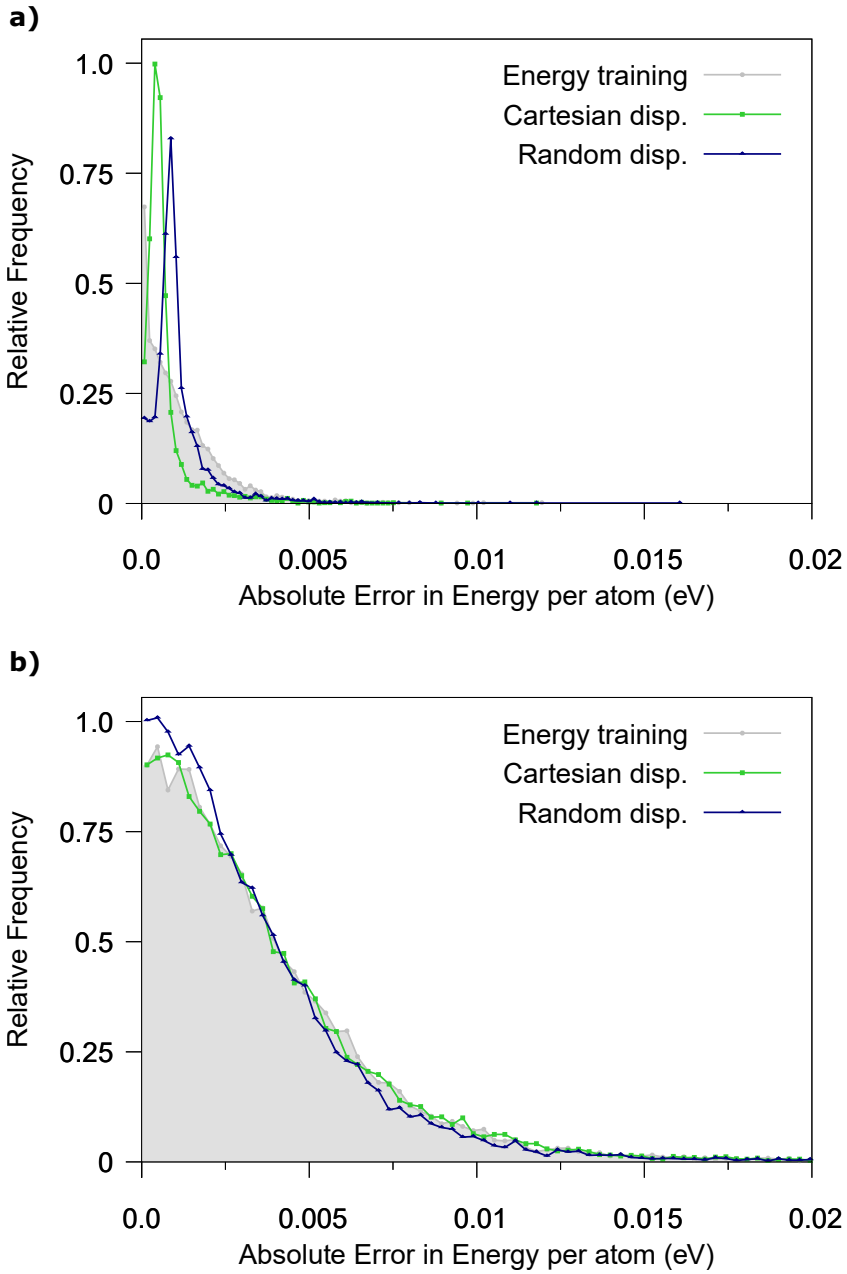


Figure 7.9.: **Distribution of the absolute error in the energy predictions** obtained by training on the `train_0500` training set applying the energy training approach (gray) and the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy. Panel **a)** shows the distribution for the training set, panel **b)** shows the distribution for the validation set.

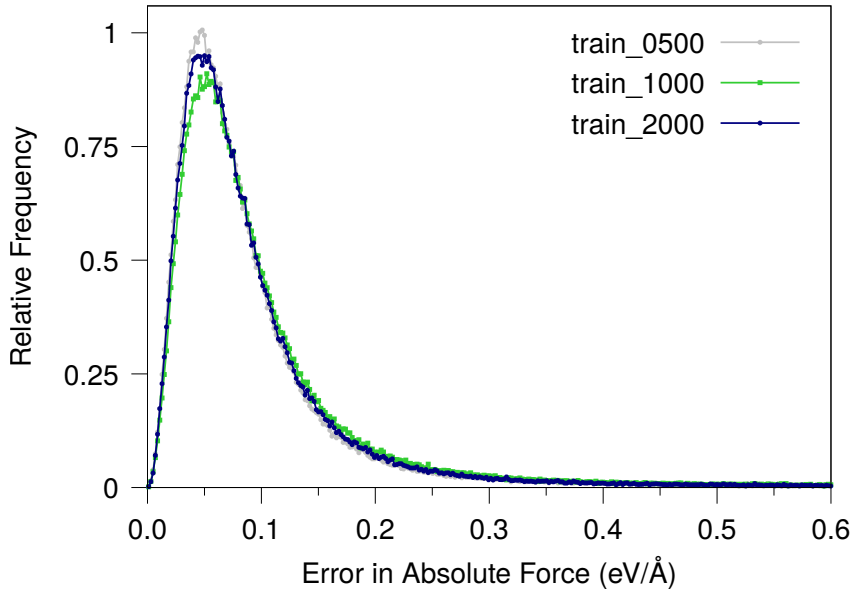


Figure 7.10.: **Distribution of the error in the absolute force** for training on all three training sets applying the direct force training approach.

the error in the direction of the predicted force vectors given in figure 7.11. The distributions of the error in the direction are practically identical for all reference data sets tested and even a zoomed in version of the graphs given in panel **b)**, which limits the errors in the direction to the range between $0° - 60°$, shows that there is no significant change in the distribution of the errors from one reference data sets to another. The fact that the MAE of the predicted atomic forces is nearly constant for all three training data sets further underlines that the size of the training set does not influence the predictive power of NN-potentials significantly. This indicates that for the system under study it is sufficient to use the `train_0500` reference data set for training in order to minimize the computational effort. In figure 7.12 the distribution of the absolute error in the energy is shown. It can be seen that the errors for the two largest reference data sets `train_1000` and `train_2000` are very similar. However, it can be observed that the peak of the error distribution of the `train_0500` reference data set is lower, indicating a slightly broader error distribution than the one obtained for the other training sets. This can also be seen from the MAEs of the energy which are about 1.05 meV/atom for training

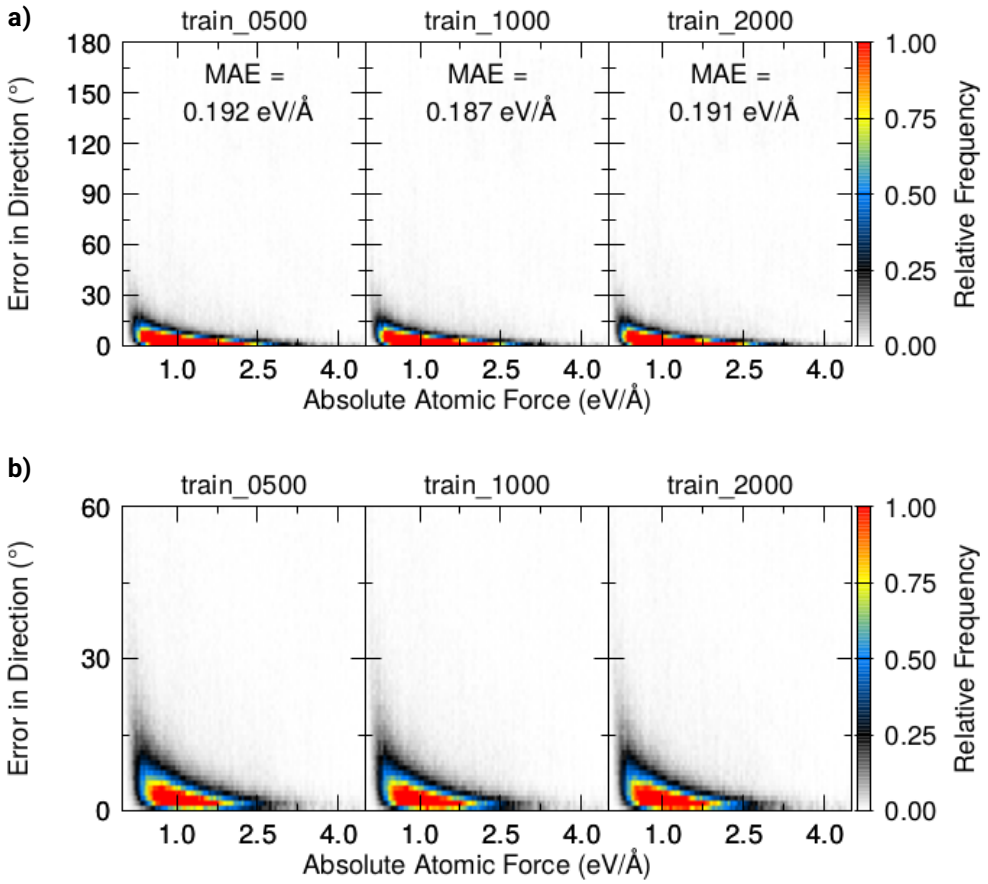


Figure 7.11.: **a) Distribution of the error in the force direction** for training on all three training sets applying the direct force training approach. Panel **b)** shows a zoomed in version of the graphs given in panel **a)** to show that the error distribution is really nearly independent of the reference data set. Panel **a)** taken from [59].

set `train_0500` and about 1.04 meV/atom for `train_1000` and `train_2000`. Since the errors in the energy are very small and well below chemical accuracy it is still well justified to make use of the NN-potentials trained with the smallest reference data set for simulation applications.

In figures 7.13 and 7.14 the results obtained by direct force training are compared to the results obtained by employing the Taylor expansion force training approach and the conventional energy training approach. In figure 7.13 the distribution of the error in the absolute force is given for NN-potentials trained with the `train_0500`

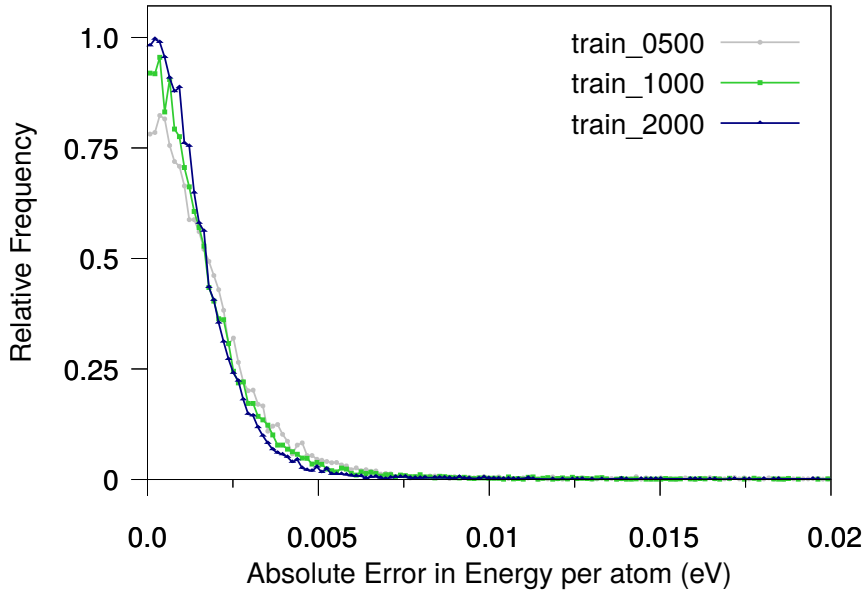


Figure 7.12.: **Distribution of the absolute error in the energy** for training on all three training sets applying the direct force training approach.

reference data set. From this graph it can be seen that the distribution obtained by the conventional energy training approach is, as expected, the broadest distribution. As discussed before, this distribution has a rather pronounced tail indicating a rather large number of atoms for which the errors in the force predictions are unfavorably large. Further, it can be seen that the direct force training approach performs best since the distribution obtained for this training approach is by far the narrowest distribution with a peak, which is furthest to the left at about 0.04 eV/Å. The error distribution obtained by direct force training indicates that for most atoms the predicted errors in the absolute force are between 0 eV/Å and 0.25 eV/Å. As discussed before, Taylor-force training leads to an improvement of the force prediction in comparison to the energy training approach, however, the respective error distribution shows that a significantly greater improvement of the force prediction can be obtained by employing the direct force training approach instead. This result was to be expected since direct force training directly includes force information into the training process. Therefore, it ensures an excellent force prediction, whereas force information is only included implicitly into the

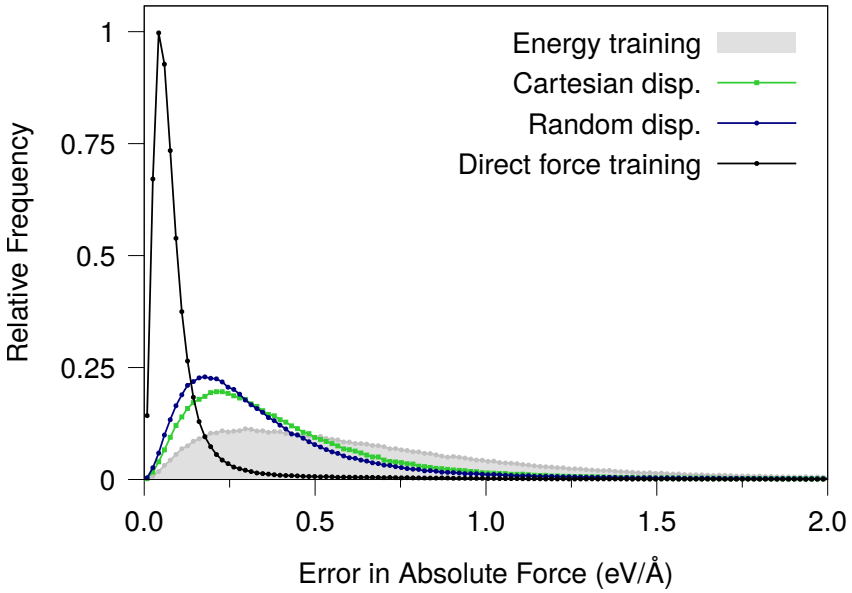


Figure 7.13.: **Distribution of the error in the absolute force** for training on the `train_0500` training set applying the energy training approach (gray), the direct force training approach (black) and approximate force training by the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy.

training procedure by the Taylor expansion force training approach in which force information is transferred to energy information.

In table 7.2 the MAEs of the atomic forces for all three validation sets are summarized for the energy training approach, the Taylor expansion based force training approach employing displacement strategy (R) and the direct force training approach. In general it can be observed that the improvement obtained by employing both force training approaches studied is greatest for the smallest reference data set `train_0500`. In comparison to the MAE of the atomic forces for the validation set obtained by energy training the Taylor expansion approach leads to a reduction of the MAE by 42.5%. Employing the direct force training approach reduces the MAE by 73%. For the largest training set `train_2000` the relative improvement of the MAE is smallest. The MAE of the forces is reduced by 20% by employing the Taylor expansion force training approach and for the direct force training approach the MAE is reduced by 52%. Again, the improvement is reported relative to the

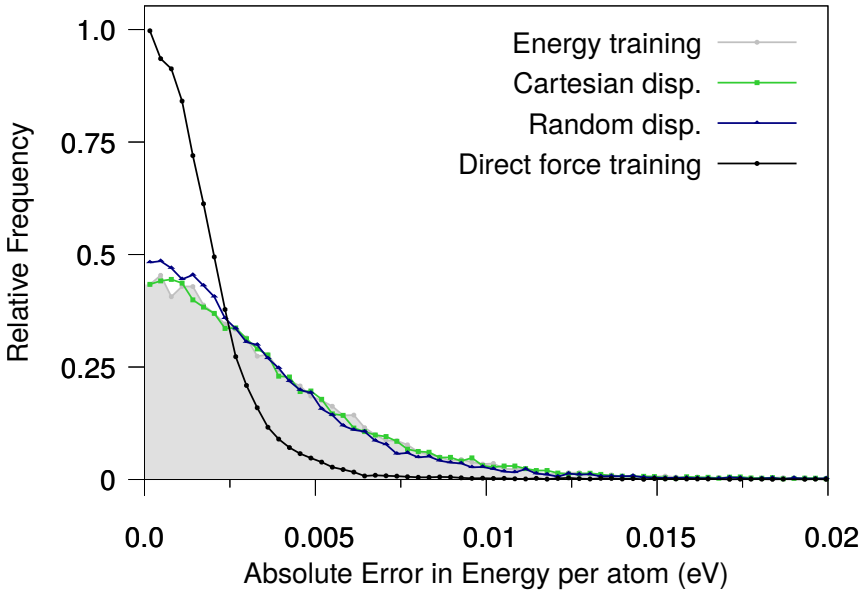


Figure 7.14.: **Distribution of the error in the energy predictions** for training on the `train_0500` training set applying the energy training approach (gray), the direct force training approach (black) and approximate force training by the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy.

Table 7.2.: MAE of the atomic forces for all training approaches and all reference data sets. For the Taylor expansion based approach displacement strategy (R) was employed.

Training approach	MAE of the atomic forces (eV/Å)		
	`train_0500`	`train_1000`	`train_2000`
Energy training	0.73	0.51	0.40
Taylor expansion based	0.42	0.36	0.32
Direct force training	0.19	0.19	0.19

MAE obtained by energy training.

In contrast to the distribution of the error of the absolute force, the distribution of the absolute error in the energy per atom given in figure 7.14 does not show an improvement of the energy prediction if the Taylor expansion approach is employed. The energies predicted by NN-PESs trained with the Taylor expansion approach

are of about the same accuracy as the energies obtained from NN-potentials trained with the conventional energy training approach. Direct force training, however, enhances the predictive power of NN-PESs with respect to the energy drastically. Figure 7.14 shows that the error distribution obtained by employing direct force training is significantly narrower, indicating overall smaller errors and especially a decrease of the number of atoms with comparatively large errors. However, as discussed before, all errors in the energy reported are well below chemical accuracy.

7.4. Summary and Discussion

Summarizing the work presented it can be stated that employing the Taylor expansion approach improves force predictions notably in comparison to predictions based on NN-PESs trained by employing the energy training approach. Moreover, this indirect force training approach fortunately does not reduce the accuracy of energy predictions obtained from NN-PESs trained with this force training approach even though it introduces some noise into the extended reference data set's energies. At the same time the improvements of the force predictions gained by employing the direct force training approach are significantly bigger than the ones obtained by employing the Taylor expansion approach. On top of that, it was seen that direct force training, in contrast to the Taylor expansion approach, also notably improves the accuracy of energy predictions obtained from NN-potentials. However, direct force training is computationally very expensive, since its computational effort scales quadratically for small systems and linearly with a large prefactor, given by the average number of atoms in two-times the volume of the cutoff ball, for larger systems. Further, even though its formal derivation is straight forward, an efficient implementation of the algorithm is rather tedious and error-prone. It was shown that for the smallest training set `train_0500`, for which the influence of both force training approaches on the predictive power of NN-PESs is greatest, the improvement relative to the force predictions obtained by energy training that were found by preforming Taylor expansion based force training is already about 58% of the improvement obtained by direct force training. This significant improvement is remarkable since the Taylor expansion approach is an approximate force training method. Its linear scaling of the computational effort with system size, where the

prefactor is the number of atoms within a single cutoff ball, is also notably more favorable than the scaling behavior of direct force training. Further, it should be noted that the implementation of the Taylor expansion based force training approach is straightforward, which facilitates an implementation of this method in other machine learning libraries for training ANNs.

In order to reduce the computational effort of direct force training, an alternative approach was suggested. In this approach direct force training is employed to a small subset of atoms in the training set and for all other atoms in the training set the energy training approach is employed. It was shown that this training approach yields significantly improved force predictions even if the direct force training approach is applied to as few as 10% or less of the atoms [44, 97–99].

In the following a first estimate of the relative computational effort of direct force training applied to all atoms, direct force training applied to 10% of the atoms and Taylor force training will be made. The computational effort is estimated for training NN-PESs with the reference data introduced before describing a cluster of six water molecules.

This cluster of water molecules is a rather small chemical system. This implies that for most atoms all atoms of the water cluster are within the cutoff sphere. Thus, the computational demand can be assumed to increase quadratically with the number of atoms (N_{atoms}) in the reference data with a prefactor of 1, i.e. it scales like $\mathcal{O}(N_{\text{atoms}}^2)$. The Taylor expansion based approach is accordingly assumed to scale like $\mathcal{O}(N_{\text{atoms}})$. Since the effects of force training were most prominent for the smallest reference data set `train_0500` consisting of 471 structures $\equiv$ 8478 atoms this reference data set is considered in the following. Under the aforementioned assumptions it is possible to estimate the number of operations required for a single training iteration: Since 10% of the structures in the reference data set were used as test set, the training set was made up of 424 structures $\equiv$ 7632 atoms. The computationally most demanding approach is the direct force training approach applied to all atoms. The number of operations required is $N_{\text{direct-all}} = (7632)^2 = 58247424 \approx 5.825 \cdot 10^7$. If direct force training would be employed for 10% of the atoms the number of operations required is $N_{\text{direct-10\%}} = (763)^2 + (7632 - 763) = 589038 \approx 5.890 \cdot 10^5$. Employing the Taylor expansion approach for $a = 22$ with displacement strategy (R) required $471 \cdot 22 = 10362$ additional structures, i.e. 186516 additional atoms, for optimal performance.

Thus, the number of operations required is $N_{\text{Taylor}} = 7632 + 186516 \approx 1.941 \cdot 10^5$. Under these assumptions the computational effort for Taylor force training is still 67% lower than the effort of direct force training applied to 10% of the atoms. However, Taylor force training requires about 25 times as many operations than energy training, since the number of operations required for energy training is $N_{\text{Energy}} = 7632$.

On top of the findings presented in this thesis it was shown by N. Artrith and A. Urban in the article discussed in this section [59] that employing the Taylor expansion force training approach can ensure energy conservation in cases where the energy is not conserved for NN-potentials trained for the same training set employing the energy training approach. Employing force training approaches can, if the cutoff function is chosen carefully, ensure energy conservation by reinforcing continuous and physically representative forces. Thus, force training in general enhances the applicability of NN-PESs to real world simulation problems.

Therefore, it can be concluded that if it is affordable the direct force training approach should be the method of choice, since it leads to the most reliable predictions of atomic forces and energies. However, due to its high computational cost, direct force training, at least if applied to all atoms, will be computationally not feasible for systems with more than a few atoms. If a larger system is to be studied or direct force training routines are not readily available, then employing the Taylor expansion based force training approach is recommended as this indirect force training approach still leads to a significant improvement of the atomic forces predicted by NN-PESs at minimal additional computational cost in comparison to training ANNs with the energy training approach.

8. Conclusion

Obtaining information on the PES and its derivatives from electronic structure calculations during a simulation is computationally very demanding. The evaluation of a NN-PES, however, allows for fast predictions of the energy and its derivatives at insignificant computational cost. Therefore, the computational demand of many chemical simulations can be significantly reduced by employing a NN-PES. In order to ensure accurate results, precise surrogate models of the actual PES have to be employed. Thus, the goal of the studies presented in this thesis was to investigate how potential energy surfaces can be approximated by a NN-potential that allows for accurate energy-, force- and Hessian predictions.

It was shown that directly including force- and Hessian information into the training process, i.e. direct force training, yields highly accurate surrogate models. Their precision even allows for the application of these models in computer simulations which are highly sensitive to errors in the derivatives of the PES, like instanton reaction rate constant calculations.

In the work presented the direct force training approach was employed to obtain reaction rate constants for a hydrogen abstraction from methanol by an incoming hydrogen atom. The reaction rate constants were computed for various deuteration patterns. On the basis of these rate constants it was demonstrated that this approach to obtain reaction rate constants yields an excellent qualitative explanation for the unexpectedly high deuteration of methanol molecules in various regions of the interstellar medium.

A quantitative comparison to experimental results on the basis of a simplistic kinetic model didn't lead to satisfactory results. However, this lack of agreement was to be apprehended due to the simplicity of the kinetic model, in which strong assumptions, like a steady-state approximation, were made and rather few chemical reactions were included. A promising approach to obtain a better quantitative

description in future work would be to employ a more sophisticated kinetic model incorporating a larger number of chemical reactions and making use of weaker assumptions in the kinetics. Nevertheless, the qualitative results are in excellent agreement with various simulation- and experimental results found in the literature.

In this context it was found that the computational effort of computing a rate constant at one single temperature employing the standard approach in which information on the PES is obtained from quantum chemical calculations is comparable to the effort required for the construction of the reference data set. The construction of the reference data set, however, is the most time consuming step required for training a NN-potential. Thus, the computation of reaction rate constants on NN-potentials is highly efficient, especially if rate constants are to be calculated at various temperatures and for different isotopologues.

The direct force training approach is highly effective, but also computationally very demanding. Therefore, in the course of the work presented in this thesis a novel, indirect force training approach based on generating additional training data and approximating the energy of these additional training examples by a first order Taylor expansion of the energy, was developed and tested. The proposed force training approach was studied and compared to conventional energy training as well as direct force training. For this comparison NN-PESs describing a cluster of six water molecules were trained with each of the training approaches.

The computational effort of this novel force training approach is significantly lower than the one of direct force training and comparable to the one of energy training.

The Taylor expansion based force training approach was found to yield a significant improvement of the force predictions in comparison to the predictions obtained by employing the conventional energy training approach. As expected, this approach is less accurate than direct force training due to its approximate nature. However, since its computational demand is considerably lower than the one of direct force training, it allows for the application of force training even to complex systems where direct force training can't be applied.

Alternatively, it has been suggested to employ the direct force training approach to a subset of atoms in the training set and to employ conventional energy training for the remaining atoms in order to reduce the computational effort for training.

This training approach was shown to improve force predictions even if as few as 10% or less of the atomic force vectors were included in the training process [44, 97–99]. Unfortunately this form of direct force training is not yet implemented into ænet. Therefore, a direct comparison with force training results obtained by employing direct force training to a subset of atoms is impossible. In future work it might also be interesting to investigate an alternative force training approach in which direct force training is employed to a subset of atoms and the Taylor expansion based force training approach is applied to the remaining atoms. This approach might further reduce the number of atoms that have to be treated with the direct force training approach to ensure accurate force predictions and thus, lower the computational effort of the training procedure. However, it has been discussed in this thesis that applying direct force training to a small fraction of atoms will often still be computationally more expensive than employing the Taylor expansion based approach.

In summary, it is possible to state that the Taylor expansion based approach is expected to be in most cases computationally less demanding than direct force training approaches, and thus allows for an efficient treatment of large systems by still yielding a significant improvement of the force predictions. As a consequence employing the Taylor expansion based approach can be considered to be a promising force training method, especially if a low computational effort of the training procedure is required. Nevertheless, due to its significantly higher accuracy, it is advisable to employ direct force training whenever possible.

Part V.

Appendix

A. Direct Force and Hessian Training for Structure Neural Networks

In the following all derivatives required for employing direct force and Hessian training for SNNs are reported for a SNN with two hidden layers. The notation used in the expressions reported is consistent with the notation introduced in section 4.1.

A.1. Gradient and Hessian of the Energy with Respect to the Input Coordinates

The gradient of the energy with respect to the i-th descriptor element of atom φ ${}^0n_i \coloneqq \sigma_i^\varphi$ is given by:

$$\begin{aligned} g_i^{\mathrm{NN}} \coloneqq \frac{\partial E^{\mathrm{NN}}}{\partial\, {}^0n_i} &= \sum_{k=1}^{D^{(2)}} \sum_{j=1}^{D^{(1)}} \frac{\partial E^{\mathrm{NN}}}{\partial\, {}^2n_k} \frac{\partial\, {}^2n_k}{\partial\, {}^1n_j} \frac{\partial\, {}^1n_j}{\partial\, {}^0n_i} \\ &= \sum_{k=1}^{K} \sum_{j=1}^{J} w_{1,k}^3 w_{k,j}^2 w_{j,i}^1 f_k^{2\prime} f_j^{1\prime}. \end{aligned}$$

The Hessian with respect to the input coordinates ${}^0n_i, {}^0n_a$ is

$$\frac{\partial^2 E^{\mathrm{NN}}}{\partial\, {}^0n_i \partial\, {}^0n_a} = \sum_{k=1}^{D^{(2)}} \sum_{j=1}^{D^{(1)}} \left[\left(\sum_{m=1}^{D^{(1)}} {}^3w_k^{1\ 2}w_j^{k\ 2}w_m^{k\ \ 1}w_i^{j\ 1}w_a^{m\ 2}f_k''\, {}^1f_m'\, {}^1f_j' \right) + {}^3w_k^{1\ 2}w_j^{k\ 1}w_i^{j\ 1}w_a^{j\ 2}f_k'\, {}^1f_j'' \right].$$

Thereby $f_c^{l\prime}$ denotes component c of the total differential of the activation function f^l of a neuron in layer l.

A.2. Derivatives of E, g, H with Respect to Weights and Biases

With ${}^lf_c'$ denoting component c of the total differential of the activation function ${}^lf$, the derivative of the loss function $\mathcal{L}(\boldsymbol{W})$ with respect to a weight or bias ω is given by:

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{W})}{\partial \omega} = \frac{1}{n_{\mathrm{E}} + n_{\mathrm{G}} + n_{\mathrm{H}}} \Bigg[& A_{\mathrm{E}} \sum_{i=1}^{n_{\mathrm{E}}} 2\left({}^iE^{\mathrm{NN}} - {}^iE^{\mathrm{REF}}\right) \frac{\partial\, {}^iE^{\mathrm{NN}}}{\partial \omega} \\ & + A_{\mathrm{G}} \sum_{j=1}^{n_{\mathrm{G}}} \left(\sum_{\sigma=1}^{D^{(0)}} 2\left({}^jg_\sigma^{\mathrm{NN}} - {}^jg_\sigma^{\mathrm{REF}}\right) \frac{\partial\, {}^jg_\sigma^{\mathrm{NN}}}{\partial \omega} \right) \\ & + A_{\mathrm{H}} \sum_{k=1}^{n_{\mathrm{H}}} \left(\sum_{\sigma,\gamma=1}^{D^{(0)}} 2\left({}^kH_{\sigma\gamma}^{\mathrm{NN}} - {}^kH_{\sigma\gamma}^{\mathrm{REF}}\right) \frac{\partial\, {}^kH_{\sigma\gamma}^{\mathrm{NN}}}{\partial \omega} \right) \Bigg].\end{aligned}$$

To calculate $\partial\mathcal{L}(\boldsymbol{W})/\partial\omega$ the following partial derivatives are needed:

$$
\begin{aligned}
\frac{\partial^{\alpha} E^{\mathrm{NN}}}{\partial {}^{3}w^{1}_{D^{(2)}}} &= 1, \\
\frac{\partial^{\alpha} E^{\mathrm{NN}}}{\partial {}^{2}w^{a}_{D^{(1)}}} &= {}^{3}w^{1}_{a}\,{}^{2}f'_{a}, \\
\frac{\partial^{\alpha} E^{\mathrm{NN}}}{\partial {}^{1}w^{a}_{D^{(0)}}} &= \sum_{k=1}^{D^{(2)}} {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{2}f'_{k}\,{}^{1}f'_{a}, \\
\frac{\partial^{\alpha} E^{\mathrm{NN}}}{\partial {}^{3}w^{1}_{a}} &= {}^{2}n_{a}, \\
\frac{\partial^{\alpha} E^{\mathrm{NN}}}{\partial {}^{2}w^{a}_{b}} &= {}^{3}w^{1}_{a}\,{}^{1}n_{b}\,{}^{2}f'_{a}, \\
\frac{\partial^{\alpha} E^{\mathrm{NN}}}{\partial {}^{1}w^{a}_{b}} &= \sum_{k=1}^{D^{(2)}} {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{2}f'_{k}\,{}^{1}f'_{a}\,{}^{0}n_{b}, \\
\frac{\partial^{\alpha} g^{\mathrm{NN}}_{\sigma}}{\partial {}^{3}w^{1}_{D^{(2)}}} &= 0, \\
\frac{\partial^{\alpha} g^{\mathrm{NN}}_{\sigma}}{\partial {}^{2}w^{a}_{D^{(1)}}} &= \sum_{j=1}^{D^{(1)}} {}^{3}w^{1}_{a}\,{}^{2}w^{a}_{j}\,{}^{1}w^{j}_{\sigma}\,{}^{2}f''_{a}\,{}^{1}f'_{j}, \\
\frac{\partial^{\alpha} g^{\mathrm{NN}}_{\sigma}}{\partial {}^{1}w^{a}_{D^{(0)}}} &= \sum_{k=1}^{D^{(2)}} \left[\left(\sum_{j=1}^{D^{(1)}} {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{j}\,{}^{2}w^{k}_{a}\,{}^{1}w^{j}_{\sigma}\,{}^{2}f''_{k}\,{}^{1}f'_{a}\,{}^{1}f'_{j} \right) \right. \\
&\qquad \left. + {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{1}w^{a}_{\sigma}\,{}^{2}f'_{k}\,{}^{1}f''_{a} \right], \\
\frac{\partial^{\alpha} g^{\mathrm{NN}}_{\sigma}}{\partial {}^{3}w^{1}_{a}} &= \sum_{j=1}^{D^{(1)}} {}^{2}w^{a}_{j}\,{}^{1}w^{j}_{\sigma}\,{}^{2}f'_{a}\,{}^{1}f'_{j}, \\
\frac{\partial^{\alpha} g^{\mathrm{NN}}_{\sigma}}{\partial {}^{2}w^{a}_{b}} &= {}^{3}w^{1}_{a}\,{}^{1}w^{b}_{\sigma}\,{}^{2}f'_{a}\,{}^{1}f'_{b} + {}^{1}n_{b} \sum_{j=1}^{D^{(1)}} {}^{3}w^{1}_{a}\,{}^{2}w^{a}_{j}\,{}^{1}w^{j}_{\sigma}\,{}^{2}f''_{a}\,{}^{1}f'_{j},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^{\,\alpha} g_{\sigma}^{\mathrm{NN}}}{\partial^{\,1} w_{b}^{a}} =& \sum_{k=1}^{D^{(2)}} \left[\left(\sum_{j=1}^{D^{(1)}} {}^{3}w_{k}^{1\ 2} w_{j}^{k\ 2} w_{a}^{k\ 1} w_{\sigma}^{j\ 2} f_{k}^{\prime\prime\ 1} f_{a}^{\prime\ 1} f_{j}^{\prime\ 0} n_{b} \right) \right. \\
& \left. + {}^{3}w_{k}^{1\ 2} w_{a}^{k\ 1} w_{\sigma}^{a\ 2} f_{k}^{\prime\ 1} f_{a}^{\prime\prime\ 0} n_{b} + \delta_{b,\sigma}\, {}^{3}w_{k}^{1\ 2} w_{a}^{k\ 2} f_{k}^{\prime\ 1} f_{a}^{\prime} \right], \\
\frac{\partial^{\,\alpha} H_{\sigma\gamma}^{\mathrm{NN}}}{\partial^{\,3} w_{D^{(2)}}^{1}} =& 0, \\
\frac{\partial^{\,\alpha} H_{\sigma\gamma}^{\mathrm{NN}}}{\partial^{\,2} w_{D^{(1)}}^{a}} =& \sum_{j=1}^{D^{(1)}} \left[\left(\sum_{m=1}^{D^{(1)}} {}^{3}w_{a}^{1\ 2} w_{j}^{a\ 2} w_{m}^{a\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{m\ 2} f_{a}^{\prime\prime\prime\ 1} f_{m}^{\prime\ 1} f_{j}^{\prime} \right) \right. \\
& \left. + {}^{3}w_{a}^{1\ 2} w_{j}^{a\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{j\ 2} f_{a}^{\prime\prime\ 1} f_{j}^{\prime\prime} \right], \\
\frac{\partial^{\,\alpha} H_{\sigma\gamma}^{\mathrm{NN}}}{\partial^{\,1} w_{D^{(0)}}^{a}} =& \sum_{k=1}^{D^{(2)}} \left[\sum_{j=1}^{D^{(1)}} \left(\sum_{m=1}^{D^{(1)}} \left({}^{3}w_{k}^{1\ 2} w_{j}^{k\ 2} w_{m}^{k\ 2} w_{a}^{k\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{m\ 2} f_{k}^{\prime\prime\prime\ 1} f_{a}^{\prime\ 1} f_{m}^{\prime\ 1} f_{j}^{\prime} \right) \right. \right. \\
& \left. \left. + {}^{3}w_{k}^{1\ 2} w_{j}^{k\ 2} w_{a}^{k\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{a\ 2} f_{k}^{\prime\prime\ 1} f_{a}^{\prime\prime\ 1} f_{j}^{\prime} \right) \right. \\
& \left. + \sum_{m=1}^{D^{(1)}} {}^{3}w_{k}^{1\ 2} w_{a}^{k\ 2} w_{m}^{k\ 1} w_{a,\sigma}^{1\ 1} w_{\gamma}^{m\ 2} f_{k}^{\prime\prime\ 1} f_{m}^{\prime\ 1} f_{a}^{\prime\prime} \right], \\
\frac{\partial^{\,\alpha} H_{\sigma\gamma}^{\mathrm{NN}}}{\partial^{\,3} w_{k}^{1}} =& \sum_{j=1}^{D^{(1)}} \left[\sum_{m=1}^{D^{(1)}} \left({}^{2}w_{j}^{k\ 2} w_{m}^{k\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{m\ 2} f_{k}^{\prime\prime\ 1} f_{m}^{\prime\ 1} f_{j}^{\prime} \right) + {}^{2}w_{j}^{k\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{j\ 2} f_{k}^{\prime\ 1} f_{j}^{\prime\prime} \right], \\
\frac{\partial^{\,\alpha} H_{\sigma\gamma}^{\mathrm{NN}}}{\partial^{\,2} w_{b}^{a}} =& \sum_{j=1}^{D^{(1)}} \left[\sum_{m=1}^{D^{(1)}} \left({}^{3}w_{a}^{1\ 2} w_{j}^{a\ 2} w_{m}^{a\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{m\ 2} f_{a}^{\prime\prime\prime\ 1} f_{m}^{\prime\ 1} f_{j}^{\prime\ 1} n_{b} \right) \right. \\
& \left. + {}^{3}w_{a}^{1\ 2} w_{j}^{a\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{b\ 2} f_{a}^{\prime\prime\ 1} f_{b}^{\prime\ 1} f_{j}^{\prime} + {}^{3}w_{a}^{1\ 2} w_{j}^{a\ 1} w_{\sigma}^{j\ 1} w_{\gamma}^{j\ 2} f_{a}^{\prime\prime\prime\ 1} f_{j}^{\prime\prime\ 1} n_{b} \right] \\
& + \sum_{m=1}^{D^{(1)}} {}^{3}w_{a}^{1\ 2} w_{m}^{a\ 1} w_{\sigma}^{b\ 1} w_{\gamma}^{m\ 2} f_{a}^{\prime\prime\ 1} f_{m}^{\prime\ 1} f_{b}^{\prime} \\
& + {}^{3}w_{a}^{1\ 1} w_{\sigma}^{b\ 1} w_{\gamma}^{b\ 2} f_{a}^{\prime\ 1} f_{b}^{\prime\prime},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^{\,\alpha} H^{\mathrm{NN}}_{\sigma\gamma}}{\partial\,{}^{1}w^{a}_{b}} = \sum_{k=1}^{D^{(2)}} \Bigg[\sum_{j=1}^{D^{(1)}} \Bigg(& {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{j}\,{}^{2}w^{k}_{a}\,{}^{1}w^{j}_{\sigma}\,{}^{2}f''_{k}\,{}^{1}f'_{a}\,{}^{1}f'_{j}\delta_{\gamma,b} \\
& + {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{j}\,{}^{2}w^{k}_{a}\,{}^{1}w^{j}_{\sigma}\,{}^{1}w^{a}_{\gamma}\,{}^{2}f''_{k}\,{}^{1}f''_{a}\,{}^{1}f'_{j}\,{}^{0}n_{b} \\
& + {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{j}\,{}^{2}w^{k}_{a}\,{}^{1}w^{j}_{\sigma}\,{}^{1}w^{j}_{\gamma}\,{}^{2}f''_{k}\,{}^{1}f'_{a}\,{}^{1}f''_{j}\,{}^{0}n_{b} \\
& + \sum_{m=1}^{D^{(1)}} {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{j}\,{}^{2}w^{k}_{m}\,{}^{2}w^{k}_{a}\,{}^{1}w^{j}_{\sigma}\,{}^{1}w^{m}_{\gamma}\,{}^{2}f'''_{k}\,{}^{1}f'_{a}\,{}^{1}f'_{m}\,{}^{1}f'_{j}\,{}^{0}n_{b} \Bigg) \\
& + \sum_{m=1}^{D^{(1)}} \Bigg[{}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{2}w^{k}_{m}\,{}^{1}w^{m}_{\gamma}\,{}^{2}f''_{k}\,{}^{1}f'_{m}\,{}^{1}f'_{a}\delta_{\sigma,b} \\
& + {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{2}w^{k}_{m}\,{}^{1}w^{a}_{\sigma}\,{}^{1}w^{m}_{\gamma}\,{}^{2}f''_{k}\,{}^{1}f'_{m}\,{}^{1}f''_{a}\,{}^{0}n_{b} \Bigg] \\
& + {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{1}w^{a}_{\gamma}\,{}^{2}f'_{k}\,{}^{1}f''_{a}\delta_{b,\sigma} \\
& + {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{1}w^{a}_{\sigma}\,{}^{2}f'_{k}\,{}^{1}f''_{a}\delta_{\gamma,b} \\
& + {}^{3}w^{1}_{k}\,{}^{2}w^{k}_{a}\,{}^{1}w^{a}_{\sigma}\,{}^{1}w^{a}_{\gamma}\,{}^{2}f'_{k}\,{}^{1}f'''_{a}\,{}^{0}n_{b} \Bigg] \,,
\end{aligned}
$$

where for ${}^{l}w^{a}_{b}$ $a \in \{1, \ldots, D^{(l)}\}$ and $b \in \{1, \ldots, D^{(l-1)} - 1\}$. The terms for the bias parameters (b=$D^{(l-1)}$) are given separately.

List of Figures

3.1. **Flowchart of the training process.** The forward propagation step is shown in green, the backpropagation step is shown in blue. 19

3.2. **Schematic representations of a feedforward neural network.** **a)** shows a detailed representation of a feedforward neural network with 2 hidden layers where each neuron is shown as one node in the graph. Bias neurons and their corresponding weights are shown in blue. **b)** shows a simplified representation of the same NN where each node represents a whole layer. 21

3.3. **Schematic representation of a neuron.** 22

3.4. **Example for overfitting.** Training data (black points) that behaves nearly linear is fitted by a linear model (green) and a high-order polynomial (blue). The polynomial is overfitting the data strongly by meeting all training points exactly but compromising the predictive quality of the model for points not included in the fitting process. . 27

3.5. **Diagnosing overfitting with learning curves.** Overfitting occurs after about 1700 training epochs where $\mathcal{L}_{\text{test}}$ starts increasing but $\mathcal{L}_{\text{train}}$ decreases monotonously. 28

3.6. **Schematic representation of an atomic neural network describing one water molecule.** For each atom i the descriptor $\boldsymbol{\sigma}_i$ is computed. The $\boldsymbol{\sigma}_i$ are then used as an input for the atomic NNs for the respective element. There is a unique NN for each chemical species, $\mathbf{NN}_{\text{O}}$ for oxygen (blue) and $\mathbf{NN}_{\text{H}}$ for Hydrogen (green). For atoms of the same chemical species identical NNs are used. . . 30

3.7. **Common activation functions and their derivatives** 39

5.1. **Workflow for force training with the Taylor expansion approach.** The training set is extended by generating additional training structures from known training structures by displacing atoms. The corresponding energy is approximated by a Taylor expansion of first order. After generating additional structures the workflow is identical to the workflow of force training: first structures are represented by descriptors and then a NN is trained to map the structure as described by the descriptor to its potential energy. . . . 74

6.1. **Distribution of training- and test set structures on the PES (open symbols).** Instanton paths are shown as lines, stationary points are given as filled symbols. All coordinates are given along normal modes 1 and 11. 84

6.2. **Normal modes 1 and 11**, on which the geometries are projected to result in figure 6.1. 85

6.3. **Average potential energy (black) ± 20 standard errors (red)** along the instanton at $T = 65\,\mathrm{K}$. The energy is given relative to the energy of the first image. Energy predictions from all 103 NN-PESs entered the average potential as well as the standard error. 89

6.4. **Comparison of bimolecular reaction rate constants** for a representative selection of individual NN-potentials (gray) to the rate constants obtained for the average NN-PES obtained by averaging over all 103 NN-PESs available. 91

6.5. **Rate constants** for reactions R 1-R 6 for an incoming H atom. . . 96

6.6. **Rate constants** for reactions R 1-R 6 for an incoming D atom. . . 97

6.7. **Instanton path for H-R 1 at** 30 K**.** The red atom positions refer to the reactant state CH_3OH + H, whereas the blue atom positions refer to the turning point of the instanton path closest to the product, CH_2OH+H_2. The molecule is displayed such that the OH group is located at the bottom of the image. 98

6.8. **Instanton paths for H-R 1 at Different Temperatures.** The changes in the C–H and H–H distances of the abstraction are displaced. 99

6.9. **Instanton paths for different mass combinations at** 260 K**.** . 100

6.10. **Heavy-atom KIEs** for the title reaction, $^{12}C/^{13}C$ in green, $^{16}O/^{18}O$ in orange. 101

6.11. **Bimolecular rate constants** for H-R 1 obtained from a micro-canonical formulation (solid line). Experimental data is shown as black point, simulation data from literature is given as red points. . 104

6.12. **Kinetic model** to explain the contribution of the hydrogen abstraction reaction from the C atom of methanol to the deuteration of methanol in the ISM. 104

7.1. **Exemplaric water cluster** 107

7.2. **Error in direction of predicted force vector acting on atom φ.** The angle α enclosed by the reference force vector $\boldsymbol{F}_{\varphi}^{\mathrm{REF}}$ (black) and the force vector predicted by the NN $\boldsymbol{F}_{\varphi}^{\mathrm{NN}}$ (green) is used as one error measure for force predictions. 108

7.3. **Distribution of the error in the absolute value of the energy** for training on all three training sets applying the energy training approach. 113

7.4. **Distribution of the error in the absolute force** for training on all three training sets applying the energy training approach. 113

7.5. **Distribution of the error in the force direction** for training on all three training sets applying the energy training approach. . . 114

7.6. **Relative mean absolute error (MAE) of the atomic forces for different force training parameters.** The MAE is reported relative to the MAE obtained by energy training (solid red line). Results are shown for displacement strategy (R) applied to reference data set `train_0500`. The dashed red line indicates the MAE obtained by direct force training. 116

7.7. **Distribution of the error in the absolute force** for training on the `train_0500` training set applying the energy training approach (gray) and approximate force training by the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy. 118

7.8. **Distribution of the error in the force direction** for training on all three training sets applying the Taylor expansion training approach employing the random displacement strategy. 120
7.9. **Distribution of the absolute error in the energy predictions** obtained by training on the `train_0500` training set applying the energy training approach (gray) and the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy. Panel **a)** shows the distribution for the training set, panel **b)** shows the distribution for the validation set. 121
7.10. **Distribution of the error in the absolute force** for training on all three training sets applying the direct force training approach. . 122
7.11. **a) Distribution of the error in the force direction** for training on all three training sets applying the direct force training approach. Panel **b)** shows a zoomed in version of the graphs given in panel **a)** to show that the error distribution is really nearly independent of the reference data set. 123
7.12. **Distribution of the absolute error in the energy** for training on all three training sets applying the direct force training approach. 124
7.13. **Distribution of the error in the absolute force** for training on the `train_0500` training set applying the energy training approach (gray), the direct force training approach (black) and approximate force training by the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy. 125
7.14. **Distribution of the error in the energy predictions** for training on the `train_0500` training set applying the energy training approach (gray), the direct force training approach (black) and approximate force training by the Taylor expansion approach employing the Cartesian (green) and random (blue) displacement strategy. 126

List of Tables

6.1. Reaction rate constants for different representations of the PES and deviations from the CC reference. All values at $T = 65\,\mathrm{K}$ and with 60 images. 92
6.2. Unimolecular rate constants (s^{-1}) for incoming H 95
6.3. Unimolecular rate constants (s^{-1}) for incoming D 95
6.4. Data for the reactions R 1 to R 8 for an incoming H atom. $E_{\mathrm{uni,act}}$ refers to the unimolecular activation energy including ZPE, T_{c} is the crossover temperature. The KIE is given with respect to H-R 1, values in parentheses refer to powers of 10. 96
6.5. Data for the reactions R 1 to R 8 for an incoming D atom. $E_{\mathrm{uni,act}}$ refers to the unimolecular activation energy including ZPE, T_{c} is the crossover temperature. The KIE is given with respect to D-R 1, values in parentheses refer to powers of 10. 97
6.6. Bimolecular rate constants for R1 with incoming H, obtained from a microcanonical formalism. 103

7.1. Optimal meta-parameters for applying the Taylor expansion approach 117
7.2. MAE of the atomic forces for all training approaches and all reference data sets. For the Taylor expansion based approach displacement strategy (R) was employed. 126

Bibliography

[1] B. Hartke and S. Grimme, *Reactive Force Fields Made Simple*, Phys. Chem. Chem. Phys. 17, 16715 (2015).

[2] T. P. Senftle, S. Hong, M. M. Islam, S. B. Kylasa, Y. Zheng, Y. K. Shin, C. Junkermeier, R. Engel-Herbert, M. J. Janik, H. M. Aktulga, T. Verstraelen, A. Grama, and A. C. T. van Duin, *The ReaxFF Reactive Force-Field: Development, Applications and Future Directions*, npj Comput. Mat. 2, 15011 (2016).

[3] J. Behler and M. Parrinello, *Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces*, Phys. Rev. Lett. 98, 146401 (2007).

[4] J. Behler, S. Lorenz, and K. Reuter, *Representing Molecule-Surface Interactions with Symmetry-Adapted Neural Networks*, J. Chem. Phys. 127, 014705 (2007).

[5] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, *Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons*, Phys. Rev. Lett. 104, 136403 (2010).

[6] J. Behler, *Atom-Centered Symmetry Functions for Constructing High-Dimensional Neural Network Potentials*, J. Chem. Phys. 134, 074106 (2011).

[7] J. Behler, *Neural Network Potential-Energy Surfaces in Chemistry: a Tool for Large-Scale Simulations*, Phys. Chem. Chem. Phys. 13, 17930 (2011).

[8] K. V. J. Jose, N. Artrith, and J. Behler, *Construction of High-Dimensional Neural Network Potentials Using Environment-Dependent Atom Pairs*, J. Chem. Phys. 136, 194111 (2012).

[9] A. P. Bartók, R. Kondor, and G. Csányi, *On Representing Chemical Environments*, Phys. Rev. B 87, 184115 (2013).

[10] A. P. Bartók and G. Csányi, *Gaussian Approximation Potentials: A Brief Tutorial Introduction*, Int. J. Quantum Chem. 115, 1051 (2015).

[11] S. De, A. P. Bartók, G. Csányi, and M. Ceriotti, *Comparing Molecules and Solids Across Structural and Alchemical Space*, Phys. Chem. Chem. Phys. 18, 13754 (2016).

[12] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, *Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning*, Phys. Rev. Lett. 108, 058301 (2012).

[13] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, *Machine Learning of Molecular Electronic Properties in Chemical Compound Space*, New J. Phys. 15, 095003 (2013).

[14] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, *Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies*, J. Chem. Theory Comput. 9, 3404 (2013).

[15] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, *Machine Learning Predictions of Molecular Properties: Accurate Many-Body Potentials and Nonlocality in Chemical Space*, J. Phys. Chem. Lett. 6, 2326 (2015).

[16] M. Rupp, R. Ramakrishnan, and O. A. von Lilienfeld, *Machine Learning for Quantum Mechanical Properties of Atoms in Molecules*, J. Phys. Chem. Lett. 6, 3309 (2015).

[17] V. Botu and R. Ramprasad, *Learning Scheme to Predict Atomic Forces and Accelerate Materials Simulations*, Phys. Rev. B 92, 094306 (2015).

[18] M. J. Hirn, N. Poilvert, and S. Mallat, *Quantum Energy Regression using Scattering Transforms*, arXiv abs/1502.02077 (2015).

[19] L. Vlcek, W. Sun, and P. R. C. Kent, *Combining Configurational Energies and Forces for Molecular Force Field Optimization*, J. Chem. Phys. 147, 161713 (2017).

[20] J. S. Langer, *Theory of the Condensation Point*, Ann. Phys. (N.Y.) 41, 108 (1967).

[21] W. H. Miller, *Semiclassical Limit of Quantum Mechanical Transition State Theory for Nonseparable Systems*, J. Chem. Phys. 62, 1899 (1975).

[22] C. G. Callan Jr. and S. Coleman, *Fate of the False Vacuum. II. First Quantum Corrections*, Phys. Rev. D 16, 1762 (1977).

[23] S. Coleman, *Fate of the False Vacuum: Semiclassical Theory*, Phys. Rev. D 15, 2929 (1977).

[24] J. O. Richardson and S. C. Althorpe, *Ring-Polymer Molecular Dynamics Rate-Theory in the Deep-Tunneling Regime: Connection with Semiclassical Instanton Theory*, J. Chem. Phys. 131, 214106 (2009).

[25] S. C. Althorpe, *On the Equivalence of Two Commonly Used Forms of Semiclassical Instanton Theory*, J. Chem. Phys. 134, 114104 (2011).

[26] J. Kästner, *Theory and Simulation of Atom Tunneling in Chemical Reactions*, Wiley Interdiscip. Rev.-Comput. Mol. Sci 4, 158 (2014).

[27] J. O. Richardson, *Derivation of Instanton Rate Theory from First Principles*, J. Chem. Phys. 144, 114106 (2016).

[28] R. P. Feynman, *Space-Time Approach to Non-Relativistic Quantum Mechanics*, Rev. Mod. Phys. 20, 367 (1948).

[29] J. B. Rommel, T. P. M. Goumans, and J. Kästner, *Locating Instantons in Many Degrees of Freedom*, J. Chem. Theory Comput. 7, 690 (2011).

[30] S. Andersson, G. Nyman, A. Arnaldsson, U. Manthe, and H. Jónsson, *Comparison of Quantum Dynamics and Quantum Transition State Theory Estimates of the* H + CH_4 *Reaction Rate*, J. Phys. Chem. A 113, 4468 (2009).

[31] J. B. Rommel and J. Kästner, *Adaptive Integration Grids in Instanton Theory Improve the Numerical Accuracy at Low Temperature*, J. Chem. Phys. 134, 184107 (2011).

[32] M. Kryvohuz, *Semiclassical Instanton Approach to Calculation of Reaction Rate Constants in Multidimensional Chemical Systems*, J. Chem. Phys. 134, 114103 (2011).

[33] S. McConnell and J. Kästner, *Instanton Rate Constant Calculations Close to and Above the Crossover Temperature*, J. Comput. Chem. 38, 2570 (2017).

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (The MIT Press, 2016).

[35] N. Artrith, A. Urban, and G. Ceder, *Efficient and Accurate Machine-Learning Interpolation of Atomic Energies in Compositions with Many Species*, Phys. Rev. B 96, 014112 (2017).

[36] T. M. Mitchell, *Machine Learning*, 1st ed. (McGraw-Hill, Inc., New York, NY, USA, 1997).

[37] D. C. Liu and J. Nocedal, *On the Limited Memory BFGS Method for Large Scale Optimization*, Math. Prog. 45, 503 (1989).

[38] R. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A Limited Memory Algorithm for Bound Constrained Optimization*, SIAM J. Sci. Comput. 16, 1190 (1995).

[39] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, *Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization*, ACM T. Math Software 23, 550 (1997).

[40] H. Robbins and S. Monro, *A Stochastic Approximation Method*, Ann. Math. Stat. 22, 400 (1951).

[41] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv abs/1412.6980 (2014).

[42] S. J. Reddi, S. Kale, and S. Kumar, *On the Convergence of Adam and Beyond*, International Conference on Learning Representations (2018).

[43] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, *Neural Network Models of Potential Energy Surfaces*, J. Chem. Phys. 103, 4129 (1995).

[44] N. Artrith and J. Behler, *High-Dimensional Neural Network Potentials for Metal Surfaces: A Prototype Study for Copper*, Phys. Rev. B 85, 045439 (2012).

[45] J. Behler, *Constructing High-Dimensional Neural Network Potentials: A Tutorial Review*, Int. J. Quantum Chem. 115, 1032 (2015).

[46] 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. 8, 3192 (2017).

[47] H. Wu, *Global Stability Analysis of a General Class of Discontinuous Neural Networks with Linear Growth Activation Functions*, Inf. Sci. 179, 3432 (2009).

[48] C. J. Willmott and K. Matsuura, *Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance*, Climate Research 30, 79 (2005).

[49] B. Zoph and Q. V. Le, *Neural Architecture Search with Reinforcement Learning*, CoRR abs/1611.01578 (2016).

[50] C. Liu, B. Zoph, J. Shlens, W. Hua, L. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy, *Progressive Neural Architecture Search*, CoRR abs/1712.00559 (2017).

[51] Yao, X., *Evolving Artificial Neural Networks*, Proc. IEEE 87, 1423 (1999).

[52] D. Floreano, P. Dürr, and C. Mattiussi, *Global Stability Analysis of a General Class of Discontinuous Neural Networks with Linear Growth Activation Functions*, Evol. Intel. 1, 47 (2008).

[53] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, *Designing Neural Networks through Neuroevolution*, Nat. Mach. Intell. 1, 24 (2019).

[54] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Efficient BackProp. Neural Networks: Tricks of the Trade (2nd ed.)* Vol. 7700, edited by G. B. Montavon G.and Orr and K.-R. Müller, Lecture Notes in Computer Science (Springer, 2012), p. 9.

[55] X. Glorot and Y. Bengio, *Understanding the Difficulty of Training Deep Feedforward Neural Networks*, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Vol. 9, edited by Y. W. Teh and M. Titterington, Proceedings of Machine Learning Research (2010), p. 249.

[56] A. M. Saxe, J. L. McClelland, and S. Ganguli, *Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks*, arXiv abs/1312.6120 (2013).

[57] D. Sussillo, *Random Walks: Training Very Deep Nonlinear Feed-Forward Networks with Smart Initialization*, arXiv abs/1412.6558 (2014).

[58] N. Artrith and A. Urban, *An Implementation of Artificial Neural-Network Potentials for Atomistic Materials Simulations: Performance for* TiO_2, Comput. Mater. Sci. 114, 135 (2016).

[59] A. M. Cooper, J. Kästner, A. Urban, and N. Artrith, *Efficient Training of Accurate Neural Network Potentials Including Atomic Force Information: Application to Water and Transition-Metal Oxides*, npj Comput. Mater. 6, 54 (2020).

[60] A. M. Cooper, P. P. Hallmen, and J. Kästner, *Potential Energy Surface Interpolation with Neural Networks for Instanton Rate Calculations*, J. Chem. Phys. 148, 094106 (2018).

[61] A. M. Cooper and J. Kästner, *Low-Temperature Kinetic Isotope Effects in* $CH_3OH + H \rightarrow CH_2OH + H_2$ *Shed Light on the Deuteration of Methanol in Space*, J. Phys. Chem. A 123, 9061 (2019).

[62] A. Ratajczak, V. Taquet, C. Kahane, C. Ceccarelli, A. Faure, and E. Quirico, *The Puzzling Deuteration of Methanol in Low- to High-Mass Protostars*, Astron. Astrophys. 528, L13 (2011).

[63] B. Parise, A. Castets, E. Herbst, E. Caux, C. Ceccarelli, I. Mukhopadhyay, and A. G. G. M. Tielens, *First Detection of Triply-Deuterated Methanol*, Astron. Astrophys. 416, 159 (2004).

[64] J. L. Linsky, A. Diplas, B. E. Wood, A. Brown, T. R. Ayres, and B. D. Savage, *Deuterium and the Local Interstellar Medium Properties for the Procyon and Capella Lines of Sight*, Astrophys. J. 451, 335 (1995).

[65] J. T. Jodkowski, M.-T. Rayez, J.-C. Rayez, T. Bérces, and S. Dóbé, *Theoretical Study of the Kinetics of the Hydrogen Abstraction from Methanol. 3. Reaction of Methanol with Hydrogen Atom, Methyl, and Hydroxyl Radicals*, J. Phys. Chem. A 103, 3750 (1999).

[66] B. Kerkeni and D. C. Clary, *Ab Initio Rate Constants from Hyperspherical Quantum Scattering: Application to* $H+C_2H_6$ *and* $H+CH_3OH$, J. Chem. Phys. 121, 6809 (2004).

[67] T. P. M. Goumans and J. Kästner, *Deuterium Enrichment of Interstellar Methanol Explained by Atom Tunneling*, J. Phys. Chem. A 115, 10767 (2011).

[68] R. Meana-Pañeda, D. G. Truhlar, and A. Fernández-Ramos, *High-Level Direct-Dynamics Variational Transition State Theory Calculations Including Multidimensional Tunneling of the Thermal Rate Constants, Branching Ratios, and Kinetic Isotope Effects of the Hydrogen Abstraction Reactions from Methanol by Atomic Hydrogen*, J. Chem. Phys. 134, 094302 (2011).

[69] A. Nagaoka, N. Watanabe, and A. Kouchi, H-D *Substitution in Interstellar Solid Methanol: A Key Route for D Enrichment*, Astrophys. J. L29, 624 (2005).

[70] A. Nagaoka, N. Watanabe, and A. Kouchi, *Efficient Formation of Deuterated Methanol by* H-D *Substitution on Interstellar Grain Surfaces*, AIP Conf. Proc. 855, 69 (2006).

[71] J. Meisner, T. Lamberts, and J. Kästner, *Atom Tunneling in the Water Formation Reaction* $H_2+OH\rightarrow H_2O+H$ *on an Ice Surface*, ACS Earth Space Chem. 1, 399 (2017).

[72] T. Lamberts, P. K. Samanta, A. Köhn, and J. Kästner, *Quantum Tunneling During Interstellar Surface-Catalyzed Formation of water: the reaction* $H+H_2O_2\rightarrow H_2O+OH$, Phys. Chem. Chem. Phys. 18, 33021 (2016).

[73] L. Song and J. Kästner, *Formation of the Prebiotic Molecule* NH_2CHO *on Astronomical Amorphous Solid Water Surfaces: Accurate Tunneling Rate Calculations*, Phys. Chem. Chem. Phys. 18, 29278 (2016).

[74] T. Lamberts and J. Kästner, *Influence of Surface and Bulk Water Ice on the Reactivity of a Water-Forming Reaction*, Astrophys. J. 846, 43 (2017).

[75] L. Song and J. Kästner, *Tunneling Rate Constants for* H_2CO+H *on Amorphous Solid Water Surfaces*, Astrophys. J. 850, 118 (2017).

[76] T. B. Adler, G. Knizia, and H.-J. Werner, *A Simple and Efficient CCSD(T)-F12 Approximation*, J. Chem. Phys. 127, 221106 (2007).

[77] K. A. Peterson, T. B. Adler, and H.-J. Werner, *Systematically Convergent Basis Sets for Explicitly Correlated Wavefunctions: The Atoms H, He, B–Ne, and Al–Ar*, J. Chem. Phys. 128, 084102 (2008).

[78] G. Knizia, T. B. Adler, and H.-J. Werner, *Simplified CCSD(T)-F12 Methods: Theory and Benchmarks*, J. Chem. Phys. 130, 054104 (2009).

[79] Y. Zhao, B. J. Lynch, and D. G. Truhlar, *Development and Assessment of a New Hybrid Density Functional Model for Thermochemical Kinetics*, J. Phys. Chem. A 108, 2715 (2004).

[80] P. Sherwood, A. H. de Vries, M. F. Guest, G. Schreckenbach, C. R. A. Catlow, S. A. French, A. A. Sokol, S. T. Bromley, W. Thiel, A. J. Turner, S. Billeter, F. Terstegen, S. Thiel, J. Kendrick, S. C. Rogers, J. Casci, M. Watson, F. King, E. Karlsen, M. Sjøvoll, A. Fahmi, A. Schäfer, and C. Lennartz, *QUASI: A General Purpose Implementation of the QM/MM Approach and its Application to Problems in Catalysis*, J. Mol. Struct. (THEOCHEM) 632, 1 (2003).

[81] S. Metz, J. Kästner, A. A. Sokol, T. W. Keal, and P. Sherwood, *ChemShell—a Modular Software Package for QM/MM Simulations*, WIREs Comput. Mol. Sci. 4, 101 (2014).

[82] Valiev, M. and Bylaska, E. J. and Govind, N. and Kowalski, K. and Straatsma, T. P. and van Dam, H. J. J. and Wang, D. and Nieplocha, J. and Apra, E. and Windus, T. L. and de Jong, W. A., *NWChem: A Comprehensive and Scalable Open-Source Solution for Large Scale Molecular Simulations*, Comput. Phys. Commun. 181, 1477 (2010).

[83] H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, M. Schütz, P. Celani, T. Korona, R. Lindh, A. Mitrushenkov, G. Rauhut, K. R. Shamasundar, T. B. Adler, R. D. Amos, A. Bernhardsson, A. Berning, D. L. Cooper, M. J. O. Deegan, A. J. Dobbyn, F. Eckert, E. Goll, C. Hampel, A. Hesselmann, G. Hetzer, T. Hrenar, G. Jansen, C. Köppl, Y. Liu, A. W. Lloyd, R. A. Mata, A. J. May, S. J. McNicholas, W. Meyer, M. E. Mura, A. Nicklass, D. P. O'Neill, P. Palmieri, D. Peng, K. Pflüger, R. Pitzer, M. Reiher, T. Shiozaki, H. Stoll,

A. J. Stone, R. Tarroni, T. Thorsteinsson, and M. Wang, *MOLPRO, Version 2012.1, a Package of ab Initio Programs*, see http://www.molpro.net, 2012.

[84] J. Kästner, J. M. Carr, T. W. Keal, W. Thiel, A. Wander, and P. Sherwood, *DL-FIND: An Open-Source Geometry Optimizer for Atomistic Simulations*, J. Phys. Chem. A 113, 11856 (2009).

[85] S. R. McConnell, A. Löhle, and J. Kästner, *Rate Constants from Instanton Theory via a Microcanonical Approach*, J. Chem. Phys. 146, 074105 (2017).

[86] J. F. Meagher, P. Kim, J. H. Lee, and R. B. Timmons, *Kinetic Isotope Effects in the Reactions of Hydrogen and Deuterium Atoms with Dimethyl Ether and Methanol*, J. Phys. Chem. 78, 2650 (1974).

[87] D. L. Baulch, C. T. Bowman, C. J. Cobos, R. A. Cox, T. Just, J. A. Kerr, M. J. Pilling, D. Stocker, J. Troe, W. Tsang, R. W. Walker, and J. Warnatz, *Evaluated Kinetic Data for Combustion Modeling: Supplement II*, J. Phys. Chem. Ref. Data 34, 757 (2005).

[88] E. F. V. Carvalho, A. N. Barauna, F. B. C. Machado, and O. Roberto-Neto, *Theoretical Calculations of Energetics, Structures, and Rate Constants for the* H+CH_3OH *Hydrogen Abstraction Reactions*, Chem. Phys. Lett. 463, 33 (2008).

[89] T. Morawietz, A. Singraber, C. Dellago, and J. Behler, *How van der Waals Interactions Determine the Unique Properties of Water*, Proceedings of the National Academy of Sciences 113, 8368 (2016).

[90] I. T. Todorov, W. Smith, K. Trachenko, and M. T. Dove, *DL_POLY_3: New Dimensions in Molecular Dynamics Simulations via Massive Parallelism*, J. Mater. Chem. 16, 1911 (2006).

[91] S. Grimme, C. Bannwarth, and P. Shushkov, *A Robust and Accurate Tight-Binding Quantum Chemical Method for Structures, Vibrational Frequencies, and Noncovalent Interactions of Large Molecular Systems Parametrized for All spd-Block Elements (Z = 1–86)*, J. Chem. Theory Comput. 13, 1989 (2017).

[92] C. Bannwarth, S. Ehlert, and S. Grimme, *GFN2-xTB—An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions*, J. Chem. Theory Comput. 15, 1652 (2019).

[93] A. D. Becke, *Density-Functional Exchange-Energy Approximation with Correct Asymptotic Behavior*, Phys. Rev. A 38, 3098 (1988).

[94] C. Lee, W. Yang, and R. G. Parr, *Development of the Colle-Salvetti Correlation-Energy Formula into a Functional of the Electron Density*, Phys. Rev. B 37, 785 (1988).

[95] S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, *A Consistent and Accurate ab Initio Parametrization of Density Functional Dispersion Correction (DFT-D) for the 94 Elements H-Pu*, J. Chem. Phys. 132, 154104 (2010).

[96] F. Furche, R. Ahlrichs, C. Hättig, W. Klopper, M. Sierka, and F. Weigend, *Turbomole*, WIREs Comput Mol Sci 4, 91 (2014).

[97] N. Artrith, T. Morawietz, and J. Behler, *High-Dimensional Neural-Network Potentials for Multicomponent Systems: Applications to Zinc Oxide*, Phys. Rev. B 83, 153101 (2011).

[98] N. Artrith, B. Hiller, and J. Behler, *Neural Network Potentials for Metals and Oxides – First Applications to Copper Clusters at Zinc Oxide*, Phys. Status Solidi B 250, 1191 (2013).

[99] A. Singraber, T. Morawietz, J. Behler, and C. Dellago, *Parallel Multistream Training of High-Dimensional Neural Network Potentials*, J. Chem. Theory Comput. 15, 3075 (2019).

Erklärung über die Eigenständigkeit der Dissertation

Ich versichere, dass ich die vorliegende Arbeit mit dem Titel *Accurate Force- and Hessian Predictions from Neural Network Potentials* selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe; aus fremden Quellen entnommene Passagen und Gedanken sind als solche kenntlich gemacht.

Declaration of Authorship

I hereby certify that the dissertation entitled *Accurate Force- and Hessian Predictions from Neural Network Potentials* is entirely my own work except where otherwise indicated. Passages and ideas from other sources have been clearly indicated.

Name/Name: April Mae Cooper

Unterschrift/Signed: _____

Datum/Date: October 2, 2020