Neural Network Potential Simulations of Copper Supported on Zinc Oxide Surfaces

Dissertation

for the award of the degree

“Ph.D."

of the Georg-August-Universität Göttingen

within the doctoral program Chemistry

of the Georg-August University School of Science (GAUSS)

submitted by

Martín Leandro Paleico

from **Buenos Aires, Argentina**

Göttingen, **18. März 2021**

Thesis Committee

Supervisor:
Prof. Dr. Jörg Behler
Theoretische Chemie
Institut für Physikalische Chemie

Second Supervisor:
Prof. Dr. Ricardo A. Mata
Computerchemie und Biochemie
Institut für Physikalische Chemie

Examination Board

Reviewer:
Prof. Dr. Jörg Behler
Theoretische Chemie
Institut für Physikalische Chemie

Second Reviewer:
Prof. Dr. Ricardo A. Mata
Computerchemie und Biochemie
Institut für Physikalische Chemie

Further Members of the Examination Board
Prof. Dr. Burkhard Geil
Biophysikalische Chemie
Institut für Physikalische Chemie

Jun. Prof. Dr. Daniel Obenchain
Physikalische Chemie
Institut für Physikalische Chemie

Prof. Dr. Martin Suhm
Physikalische Chemie II
Institut für Physikalische Chemie

Prof. Dr. Alec Wodtke
Physikalische Chemie I
Institut für Physikalische Chemie

Date of the oral examination: 20.05.2021

Oath

I hereby declare that I have prepared this thesis all by myself, did not use any sources or tools except for those explicitly stated, and marked all quotes, be it in literal or analogous form, accordingly.
I declare that this thesis has not, nor in excerpts, been submitted to this or any other university in the context of a failed examination.

Göttingen, **18. März 2021**

(**Martín Leandro Paleico**)

Quotes

There are more things in Heaven and Earth, Horatio,
Than are dreamt of in your Philosophy.

(Hamlet, Shakespeare)

They have a curse. They say: May you live in interesting times.

(Apocryphal quote)

Give me an algorithm and a point of fulcrum and I will move the world.

(An Archimedes for the informatic age)

The story so far:
In the beginning the Universe was created.
This has made a lot of people very angry and been widely regarded as a bad move.

(The Restaurant at the End of the Universe, Douglas Adams)

Much human ingenuity has gone into finding the ultimate Before. The current state of knowledge can be summarized thus: In the beginning, there was nothing, which exploded.

(Lords and Ladies, Terry Pratchett)

“All right," said Susan. "I’m not stupid. You’re saying humans need... fantasies to make life bearable."
REALLY? AS IF IT WAS SOME KIND OF PINK PILL? NO. HUMANS NEED FANTASY TO BE HUMAN. TO BE THE PLACE WHERE THE FALLING ANGEL MEETS THE RISING APE.

(Hogfather, Terry Pratchett)

By and large, the only skill the alchemists of Ankh-Morpork had discovered so far was the ability to turn gold into less gold.

(Moving Pictures, Terry Pratchett)

Windle shook his head sadly. Five exclamation marks, the sure sign of an insane mind.

(Reaper Man, Terry Pratchett)

And therefore education at the University mostly worked by the age-old method of putting a lot of young people in the vicinity of a lot of books and hoping that something would pass from one to the other, while the actual young people put themselves in the vicinity of inns and taverns for exactly the same reason.

(Interesting Times, Terry Pratchett)

Strange events permit themselves the luxury of occurring.

(Fads & Fallacies in the Name of Science, Martin Gardner)

(Economists are known to admire theoretical proofs; thus the old quip: Sure, it works in practice, but does it work in theory?)

(Levitt and Dubner)

There are now many invisible people on stage.

(Stage directions in The Chairs, Eugène Ionesco)

Abstract

Heterogeneous catalysis is an area of active research, because many industrially relevant reactions involve gaseous reactants and are accelerated by solid phase catalysts. In recent years, activity in the field has become more intense due to the development of surface science and simulation techniques that allow for acquiring deeper insight into these catalysts, with the goal of producing more active, cheaper and less toxic catalytic materials.

One particularly crucial case study for heterogeneous catalysis is the synthesis of methanol from synthesis gas, composed of H_2, CO and CO_2. The reaction is catalyzed by a mixture of Cu and ZnO nanoparticles with Al_2O_3 as a support material. This process is important not only due to methanol's many uses as a solvent, raw material for organic synthesis, and possible energy and carbon capture material, but also as an example for many other metal/metal oxide catalysts. A plethora of experimental studies are available for this catalyst, as well as for simpler model systems of Cu clusters supported on ZnO surfaces. Unfortunately, there is still a lack of theoretical studies that can support these experimental results by providing an atom-by-atom representation of the system.

This scarcity of atomic level simulations is due to the absence of fast but *ab-initio* level accurate potentials that would allow for reaching larger systems and longer simulated time scales. A promising possibility to bridge this gap in potentials is the rise of machine learning potentials, which utilize the tools of machine learning to reproduce the potential energy surface of a system under study, as sampled by an expensive electronic structure reference method of choice. One early and fruitful example of such machine learning force fields are neural network potentials, as initially developed by Behler and Parrinello.

In this thesis, a neural network potential of the Behler-Parrinello type has been constructed for ternary Cu/Zn/O systems, focusing on supported Cu clusters on the ZnO($10\bar{1}0$) surface, as a model for the industrial catalyst. This potential was subsequently utilized to perform a number of simulations. Small supported Cu clusters between 4 and 10 atoms were optimized with a genetic algorithm, and a number of structural trends observed. These clusters revealed the first hints of the structure of the Cu/ZnO interface, where Cu prefers to interact with the support through configurations in the continuum between Cu(110) and Cu(111). Simulated annealing runs for Cu clusters between 200 and 500 atoms reinforced this observation, with these larger clusters also adopting this sort of interface with the support. Additionally, in these simulations the effect of strain induced by the support can be observed, with deviations from ideal lattice constants reaching the top of all of the clusters. To further investigate the influence of strain in this system, large coincident surfaces of Cu were deposited on ZnO supports. Due to the lattice mismatch present between the two materials, this requires straining the Cu overlayer. This analysis confirmed once again that Cu(110) and Cu(111) are the most stable surfaces when deposited on ZnO($10\bar{1}0$). During this thesis a number of new algorithm and programs were developed. Of particular interest is the bin and hash algorithm, which was designed to aid in the construction and curating of reference sets for the neural network potential, and can also be used to evaluate the quality of atomic descriptor sets.

Table of Contents

List of Abbreviations (alphabetical order)

ACSF	Atom-Centered Symmetry Function
ASE	Atomic Simulation Environment (Python library)
BAH	Bin and Hash algorithm
BHMC	Basin Hopping Monte Carlo
CLM	Coincidence Lattice Match algorithm
DFT	Density Functional Theory
fcc	Face Centered Cubic
GA	Genetic Algorithm
GM	Global Minimum
GO	Global Optimization
hcp	Hexagonal Close Packed
HDNNP	High-Dimensional Neural Network Potential
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator
LM	Local Minimum
MC	Monte Carlo
MD	Molecular Dynamics
ML	Machine Learning
MLP	Machine Learning Potential
NN	Neural Network
NNP	Neural Network Potential
PES	Potential Energy Surface
PHTM	Polyhedral Template Matching
RMSE	Root-Mean Square Error
SA	Simulated Annealing
SF	Symmetry Function
VASP	Vienna Ab-initio Simulation Package

Part I

Introduction and Methods

Chapter 1

Introduction

1.1 The Copper and Zinc Oxide Catalyst

Heterogeneous catalysis [1–5] occurs when the reagents and the catalyst accelerating the reaction reside in different or immiscible phases. Many industrially relevant synthesis are possible thanks to the activity of this type of catalysts, and already in 2006 it was estimated that 80% of the reactions in industrial chemistry depend on catalysts of every type, contributing to up to 35% of the world's GDP [6]. One of the more emblematic examples of the progress of heterogeneous catalysis is the nitrogen fixation process into ammonia developed by Haber and Bosch in the first half of the 20th century [3]. In the preceding century, ammonia and nitrates were extracted from non-renewable and quickly depleting sources such as niter mineral deposits and bird guano. This latter source was particularly devastating to the ecology of many tropical islands at the time. The advent of the Haber-Bosch process meant not only that these sources could be abandoned, but also lead to an explosive increase in the production of fertilizer, explosives, and many products from organic chemistry, with the associated benefits to human life and economic and industrial progress.

Study of these catalysts is therefore of relevance, with the goal of discovering new catalytic materials and pathways, or of improving those already existing catalyst by replacing them with more active, cheaper, or less toxic components [2]. The field of heterogeneous catalysis has advanced exponentially since the days of Mittasch and Bosch, who in the first half of the 20th century while working for BASF tested thousands of catalyst combinations for the nitrogen fixation process until the most adequate one could be found. This is thanks to the development over the last century of a multitude of experimental techniques [7], including among them many surface microscopy methods, ultra-high-vacuum study chambers, surface spectroscopy techniques, techniques to follow the course of a reaction in real time with devices such as mass spectrometers, structural determination with X-ray diffraction, as well as many other specific solid materials and surface science techniques.

In spite of this abundance of experimental tools, solid catalysts present a number of challenges that make them hard to probe at the atomic level, such as the presence of a variety of morphologies and preparations methods for the industrial version of the catalyst, different behaviors between industrial and experimental conditions, the necessity for most of these techniques to work in high vacuum and low temperature situations, among others. To overcome some of these limitations and to gain more insight into the catalytic process, increasingly in the last years contributions from the simulation community have become more and more important [8]. This has been guided by the rapid improvement of computer power in the last decades, as well as the continuous development of new electronic structure and classical algorithms, and simulation setups [5, 8] that attempt to mimic reaction conditions.

The production of methanol follows a similar historical development to that of ammonia: originally produced inefficiently by pyrolysis from large amounts of wood, an industrial scale process involving heterogeneous catalysts was developed in the first half of the 20th century (coincidentally, also with the involvement of Mittasch at BASF). The current industrial synthesis of methanol requires a catalyst composed of intermixed Cu and ZnO nanoparticles with an Al_2O_3 (alumina) support [9, 10]. The reaction for the synthesis of methanol starts from a readily available mixture of H_2, CO_2 and CO

known as synthesis gas, and can be summarized as

$$2 \cdot H_2 + CO \rightarrow CH_3OH \tag{1.1}$$

The available experimental evidence points to the formation of CO_2 as an intermediate species in the reaction, and thus synthesis gas mixtures that already contain CO_2 are preferred [9]. This reaction takes place at high temperatures (200-400 °C) and pressures (50-200 atm.) [9, 10], depending on the specific version of the catalyst and gas mixture utilized. The catalyst is also utilized in other similar reduction reactions, such as the water gas shift reaction or production of higher order organic alcohols [11]. This catalyst is of great interest, not only due to the specific importance of methanol at the industrial scale as a solvent, synthetic raw material for other organic molecules and potential as an energy and atmospheric carbon storage material, but also in general as a model for heterogeneous catalysis and metal/metal-oxide catalysts.

The industrial catalyst itself is usually prepared [12–14] by mixing the inorganic nitrate salts of Cu (in the highest proportion), Zn, and Al, co-precipitating with sodium carbonate, followed by steps of ageing, drying, calcination, reduction, and annealing at high temperature. Many atomic microscopy images of the commercial catalyst with techniques such as transmission electron microscopy (TEM) and scanning tunneling microscopy (STM) are available [15–19], revealing a complex mixture of Cu and ZnO nanoparticles. Most of the nanoparticles can be distinctly identified as being either copper or zinc oxide, but the particles are often stuck to one another, and even at this scale the structure of the interface between the materials is hard to discern.

Copper plays a key role in the catalyst, being the likely active component for the reaction, with non-copper based versions presenting lower activity but better stability [20]. The alumina prevents sintering and poisoning of the catalyst and thus extends its active life. The role of the zinc oxide is more complicated. It is an important component since copper or zinc oxide on their own do not present a high catalytic activity [21], and substitution with other metal oxides leads to reduced activity [22]. A variety of hypotheses [9, 10, 23–26] regarding its actual function and the identity of the catalytic active site have been presented in the literature. To mention a few of the more relevant theories, it has been proposed that ZnO:

i. serves as a storage of hydrogen that then reacts on the copper [21]

ii. induces and stabilizes strain and defects on the copper which enhances the catalytic activity of the copper phase [16, 17, 27], while also preventing coalescence of the nanoparticles and thus increasing the Cu surface area [10]

iii. provides oxygen atoms to form a mixed copper oxide phase that is assumed to be the active material [28]

iv. stabilizes reactive Cu^+ species that substitute Zn atoms in the ZnO lattice [9, 11, 29]

v. induces shape changes on supported Cu particles under reaction conditions, which are responsible for catalytic behavior [30–32]

vi. provides Zn atoms to form brass alloy [33, 34], but yet other sources [30] mention that this was only observed under extremely reducing conditions

A number of recent papers further illustrate the difficulty of assigning a unique source to the activity of Cu/ZnO. Kuld *et al.* [35] suggested in 2016 that, once again, the catalytic activity is due to brass alloying on the external facets of copper nanoparticles, with Zn atoms migrating originally from the ZnO nanoparticles. Later in 2017, Kattel *et al.* [36, 37] proposed that this Zn is easily oxidized to ZnO, in which case brass is no longer the relevant species. Both papers are then somewhat contradicted by a previous 2015 paper by Lunkenbein *et al.* [19] in which they detected a thick graphite-like ZnO

overlayer on catalyst nanoparticles under industrial conditions. Due to the large number of possible reaction pathways for the formation of methanol from the gaseous reagents [38], and all the possible material combinations of a ternary Cu/Zn/O system, it is likely that many if not all of these proposed hypotheses are in the end partially responsible for the catalytic activity of the mixture.

Due to the complexity of the commercial catalyst, much more information is available for simpler systems and models. Copper and zinc oxide have of course been extensively studied on their own. Zinc oxide is interesting as an example of a widely available and utilized metal oxide model. Janotti and Van de Walle [39] explored the semiconductor characteristic of the material and presented an in-depth view of its electronic structure. With regards to ZnO surfaces, Wöll [40] presented an excellent review of experimental and theoretical results, while an earlier paper by Meyer and Marx [41] serves a useful reference for the in-detail structure of such surfaces and the challenges related to DFT calculations involving them. On the experimental side, Dulub *et al.* [42] presented an extensive study of the common low index surfaces of ZnO with a variety of surface science techniques. Wang [43] showed how zinc oxide exhibits a wide variety of nanoparticle shapes and behaviors, while Moezzi *et al.* [44] provided a good review of their synthesis, characterization and uses. To highlight the complexity of zinc oxide at the small cluster level and the general difficulty of cluster optimization, Wang *et al.* [45] calculated and provided structures for clusters between 2 and 18 formula units of ZnO. Finally, Kołodziejczak-Radzimska and Jesionowski [46] presented an even wider list of the available synthetic pathways for zinc oxide materials, and describe many of its industrial applications. For a general overview of metal oxides, their interfaces, and interactions with molecules and metals, the book "The Surface Science of Metal Oxides" by Henrich and Cox [47] is a good but outdated resource.

The literature for copper is naturally prolific, due to copper's historical, chemical, biological and industrial significance. The main reference for copper as a material is a 2001 ASM review book [48] centering on copper and its alloys, with most of the rest of the available literature concentrating on copper in combination with other materials or reagents. Of interest to the Cu/ZnO catalyst system are multiple papers on the synthesis and characterization of copper nanoparticles, with specific examples such as those presented by Dhas *et al.* [49] and Khanna *et al.* [50]; and more general reviews produced by Khodashenas and Ghorbani [51], and Din and Rehan [52].

Combined models of both components usually consist of copper deposited on zinc oxide, most commonly the polar surfaces ZnO(0001) and ZnO(000$\bar{1}$), and the non-polar ZnO(10$\bar{1}$0) surface. Many of these studies report the formation of clusters even at low monolayer coverages, and go on to describe the morphology of these clusters and their growth modes, such as the initial presence of 2D flat islands that then transition into 3D clusters or full Cu overlayers at larger coverage proportions, depending on the particular surface and conditions involved [53–55]. But many studies have also reported a number of interesting behaviors even for such a simple model system, such as the preferential nucleation of clusters on steps and kinks [56], formation of trenches and valleys at large coverages when a continuous sheet would be expected [56, 57], disappearance of a proportion of the deposited copper when annealing at high temperatures, presumably because of diffusion into the support or possible formation of an alloy [18, 56, 58], changes in the shape of the clusters when exposed to reaction gases [31, 59], and entrenching of the clusters into the ZnO support, deep enough to leave an imprint when the clusters are removed with a STM tip [60].

Many of the surface science techniques utilized to study these model systems present a number of limitations [7] that make it also necessary to rely on theoretical approaches. To mention a few problems, most surface science techniques need to be performed under ultra-high vacuum conditions and at low temperatures, have trouble imaging mobile atoms, cannot distinguish Cu and CuZn due to the similarity in lattice constants [61], nuclear charge and electronic density, which are properties on which many of these techniques rely. Thus insights can also be gained from theoretical studies and simulations [8], but this is difficult for the Cu/ZnO case due to the absence of cheap and accurate force fields capable of treating this system. This is apparent from the available theoretical literature for the catalyst. Because of this limitation, theoretical studies have been limited to small *ab-initio*

calculations containing a few Cu atoms or pre-designed clusters, a small ZnO support, and sometimes a handful of gas molecules [24, 62, 63]; or to simplified thermodynamical approaches such as *ab-initio* thermodynamics to explore the formation of vacancies and defects and the presence of adatoms on the ZnO substrate [24, 64, 65]. The few papers that attempt to simulate larger systems rely on *ad hoc* parameterized potentials [66, 67]. To mention a few recent relevant theory papers, Hellström *et al.* [68] deposited multiple pre-designed small clusters on ZnO and studied their electronic structure and charge distribution. Cheng *et al.* [66] performed a parametric fit with the COMB3 (third-generation Charge Optimized Many Body) potential [69]), and then deposited copper clusters on ZnO slabs, studying the distribution of Cu atoms on the surface, being able to reach systems with more than 500 atoms in the simulation. Beinik *et al.* [59] studied the wetting of supported continuous Cu overlayers with electronic structure methods, and how this changes when defects are introduced into the ZnO layer, but used small periodic cells for this purpose. Another later paper by Hellström *et al.* [70] studied the behavior of very small supported Cu clusters in the presence of water molecules. Mora-Fonz *et al.* [67] developed a parameterized force field, and utilized this in an attempt to globally optimize small deposited clusters and to study the interaction of larger, nanometer scale clusters with steps on a ZnO surface. Wan et al. [71] deposited vacuum optimized Cu clusters on a CeO_2 support and studied their interaction with CO including single atom diffusion, utilizing electronic structure calculations. More recently, Higham *et al.* [72] presented a study of the structure of copper clusters on the polar surfaces of ZnO by combining parameterized potentials (which seem to be the same from Mora-Fonz *et al.*) and electronic structure calculations.

For direct atom-per-atom simulations, a potential is required that is both accurate, but fast enough to be able to simulate either thousands of atoms or long simulation times. This has led to a lack of large-scale atomistic level studies. Electronic structure calculations can give the accuracy required, but are restricted to single point calculations, calculations with no support (which are thus of dubious use), or very short dynamic runs. Classical simulations, on the other hand, are constrained by the lack of a potential that can accurately treat this system. A force field is required that is not only reactive, but that can treat two very different materials together (one a conducting metal, the other a semiconductor metal oxide), and in a variety of configurations.

Commonly utilized and widely available force fields and parameterization methods are not suitable for the Cu-ZnO system or supported clusters in general, and only a few potentials for general metal/metal oxide systems are available [73, 74]. A particular strong requirement is that no fixed bonds should be present in the model, which already discards a large majority of simple force field setups. Force fields designed for solids, suffer from either being too simple (for example, the Stillinger-Weber potential [75]), or being able to only handle a limited number of configurations or phases (for example, the EAM method [76]), whereas a potential for the system under study requires handling different environments such as solids, surfaces and clusters. More complex force fields such as ReaxFF [77], which do allow for bond breaking, are available. Although originally intended for molecules, it can also be used to treat metals and metal oxides (see for example, refs. 74 and 78), but still requires careful parameterization and selection of reference structures. Another common parameterization scheme, particularly suited to density functional theory calculations and solids, is the DFTB (Density Functional Tight Binding) [79, 80] method. Potentials for ZnO [81] as well as for Cu [82, 83] are available in the literature, but no parameterization for ternary CuZnO appears to be available. DFTB is notoriously difficult to parameterize [84], and also rather computationally demanding.

As a specific example of parameterized potentials for Cu and ZnO, ref. 66 presents a parameterization with the COMB3 potential, and ref. 67 presents an *ad hoc* parameterization based on Buckingham and Morse terms. Ideally, a generally applicable solution is desired. Most of the parameterizable force fields available function best with one particular system (molecules, or metals, or metal oxides, or solids, or clusters) but not multiple systems at the same time. Worse still, the potentials need to be parameterized for each individual case, carefully selecting which particular potential is applicable to the problem at hand, and introducing new species into a parametrization can be difficult or in some

cases impossible given the particular construction of the potential.

For this purpose, machine learning potential and in particular neural network potentials [85–87] have appeared in the last few years as an attractive option. These potentials can reproduce *ab-initio* data with high accuracy but also high computational efficiency. More importantly for this system, no hard-coded concept of bonds is present in the NNP *ansatz*, which makes them naturally reactive, and they can contain a multitude of chemical environments. NNPs are parameterized directly on information about the potential energy surface (PES) of the system, as sampled by high-quality reference electronic structure calculations. They can work for a variety of systems, combining information from solids and molecules, and adding additional elemental species into the potential is much easier than in many parameterized potentials which might require new *ad hoc* potential terms or extra parameters. NNPs have been already found success in being applied to a series of related systems such as copper [88] and zinc oxide [89] on their own, as well as the interface between water and copper [90], and water and zinc oxide [91]. More recently, as part of research related to this thesis, also brass nanoparticles have been investigated [61, 92]. Finally, Artrith *et al.* [93] demonstrated that an accurate NNP was also possible for the ternary Cu/ZnO system.

The goal of this thesis is then to generate a neural network potential for Cu and ZnO, and utilize this potential to gain new insights into the structure of the combined Cu and ZnO system, with a focus on taking advantage of the power of NNPs to go beyond the capabilities of the previously available simulation-based studies. It seeks to address the following questions:

- Can we provide a framework for reliably generating reasonable structures for supported small clusters? Previous studies have centered on depositing pre-optimized or pre-designed clusters, or on optimizing supported clusters with parameterized potentials. Such small clusters would not only provide the first insights into the structure of the combined catalyst, but could also function as templates for future catalytic studies by addition of reactant molecules.

- Can we perform simulations reaching the experimental scale? All previous theoretical studies have centered either on very small clusters, coarse or abstracted simulations, or used parameterized potentials to simulate larger clusters. Can we utilize a NNP to approach *ab-initio* quality, but also simulate experimental scale systems in the nanometer range?

- Can we provide new information regarding the Cu and ZnO interface? This is difficult to access experimentally with surface science techniques since it is of course covered by both materials. Of particular interest is the presence of a mismatch between the two materials, which restricts the kind of interfaces that are energetically favorable and thus possible/observable.

1.2 Outline of this Work

This thesis is organized into four parts. Part I contains this introduction (chapter 1), a methods and theory chapter (chapter 2) describing the theory of electronic structure calculations, neural network potentials, and a variety of other simulation algorithms utilized in the course of this work; and finally a computational details chapter (chapter 3) detailing the setting and parameters utilized for a variety of methods as well as describing the generation of the reference structures for the Cu/ZnO potential. Part II contains the main results portion of this thesis. First (chapter 4), a new algorithm for analysing structural databases and atomic environment descriptors known as the bin and hash algorithm (BAH) is described and examples of its application are provided. Then (chapter 5) a genetic algorithm global optimization algorithm is utilized to generate small Cu clusters (between 4 and 10 atoms) deposited on ZnO($10\bar{1}0$). This is extended in the following chapter (chapter 6), with an analysis of larger Cu clusters (up to 500 atoms) as obtained from simulated annealing runs. Finally, the interface between Cu(100), (110), and (111) and ZnO($10\bar{1}0$) is studied from the point of view of the coincidence lattice match

algorithm. Part III includes a summary and conclusions (chapter 8), acknowledgements (chapter 9), and a bibliography. Finally, a series of appendices are included in part IV, labeled A through F.

Chapter 2

Methods and Theory

2.1 Density Functional Theory

As the method of choice for generating the reference electronic structure data in this work density functional theory (DFT) has been chosen. Here the necessary fundamentals of electronic structure calculation and DFT are introduced.

The state of a quantum system can be described by a wave function $\psi(\mathbf{R},\mathbf{r})$, which depends on the position of all nuclei ($\mathbf{R}$) and electrons ($\mathbf{r}$) in the system, as obtained from the solutions to Schrödinger's time-independent, non-relativistic differential equation

$$\mathbf{H}\psi = E\psi, \tag{2.1}$$

where E is the energy associated to that wave function, and $\mathbf{H}$ is the Hamiltonian operator, which contains kinetic and potential energy terms. For a system of nuclei (also known as ions) and electrons, the Hamiltonian can be expressed in atomic units as

$$\mathbf{H} = -\frac{1}{2}\sum_{i=1}^{N_\mathrm{e}}\nabla_i^2 - \frac{1}{2}\sum_{A=1}^{N_\mathrm{N}}\frac{1}{M_A}\nabla_A^2 + \sum_{i=1}^{N_\mathrm{e}}\sum_{j>i}^{N_\mathrm{e}}\frac{1}{r_{ij}} - \sum_{i=1}^{N_\mathrm{e}}\sum_{A=1}^{N_\mathrm{N}}\frac{Z_A}{r_{iA}} + \sum_{A=1}^{N_\mathrm{N}}\sum_{B>A}^{N_\mathrm{N}}\frac{Z_A Z_B}{R_{AB}} \tag{2.2}$$

$$= \mathbf{T}_\mathrm{e} + \mathbf{T}_\mathrm{N} + \mathbf{V}_\mathrm{ee} + \mathbf{V}_\mathrm{Ne} + \mathbf{V}_\mathrm{NN}, \tag{2.3}$$

where the terms represent the kinetic energy of electrons ($\mathbf{T}_\mathrm{e}$) and nuclei ($\mathbf{T}_\mathrm{N}$), and the electrostatic Coulomb interaction between electrons ($\mathbf{V}_\mathrm{ee}$), electrons and nuclei ($\mathbf{V}_\mathrm{Ne}$), and between nuclei ($\mathbf{V}_\mathrm{NN}$). N_e and N_N are the number of electrons and nuclei in the system, ∇^2 is the Laplacian differential operator, M_A and Z_A are the mass and charge of a given nucleus A, and r_{ij}, r_{iA}, R_{AB} represent the distances between the corresponding particles.

With the exception of a few simple cases, it is not possible to solve this differential equation exactly and directly. A first approximation to aid in this endeavor was provided by Born and Oppenheimer [94], by separating the nuclear and electronic degrees of freedom by

$$\psi = \psi(\mathbf{R},\mathbf{r}) \approx \psi_\mathrm{N}(\mathbf{R})\cdot\psi_\mathrm{e}(\mathbf{r}). \tag{2.4}$$

In this approximation, the electrons are said to respond immediately to changes in the positions of the much heavier nuclei, and as such Hamiltonian terms that depend on ion positions can be considered constant, with the exception of $\mathbf{V}_\mathrm{Ne}$ where now the nuclei position enter parametrically. This gives rise to the concept of a potential energy surface (PES): a function that uniquely correlates, for a given electronic state, atomic positions with the energies and forces in the system. The Hamiltonian associated to the electronic wave function thus becomes

$$\mathbf{H}_\mathrm{elec} = \mathbf{T}_\mathrm{e} + \mathbf{V}_\mathrm{ee} + \mathbf{V}_\mathrm{Ne}. \tag{2.5}$$

Computationally, the electronic Schrödinger equation can be solved with the Hartree-Fock [95] (HF) approach by making use of the variational principle, which states that the correct proposed wave func-

tion minimizes the energy of the system, expressing the trial wave function as a Slater determinant of basis functions, and solving a series of linear algebra equations.

A disadvantage of this approach is that the wave function depends on $3N_e$ coordinates. Since the square of the wave function can be interpreted as the probability of finding an electron in a given region of space, which can be related to the more classical concept of electronic density, a possibility emerges of developing a quantum framework based only on the electronic density. This is the goal of density functional theory.

The relationship between wave function and electronic density is given by

$$\rho(\mathbf{r}) = N_e \int ... \int |\psi(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{N_e})| d\mathbf{r}_2 d\mathbf{r}_3 ... d\mathbf{r}_{N_e}, \tag{2.6}$$

where the integral is over all electron spatial coordinates except for one, and the equation can be interpreted as the probability of finding any of the electrons in the volume $d\mathbf{r}_1$ while the system is in state ψ. Integrating over $d\mathbf{r}_1$ returns the total number of electrons in the system,

$$\int \rho(\mathbf{r}) d\mathbf{r}_1 = N_e. \tag{2.7}$$

The advantage of approaching the problem from the point of view of the electronic density is that this quantity only depends on 3 spatial coordinates, adopting a given value for every point in space, and thus its dimensionality remains constant regardless of system size. The theorems required for utilizing the electron density were proved by Hohenberg and Kohn [96]. They proved that i) the electronic density in fact uniquely determines a Hamiltonian, and consequently all other properties of the system; and ii) the variational theorem is also valid for the electronic density, that is, the "correct" electronic density for a system is also the one that minimizes the energy of the system.

With these theorems, the energy of the system can be expressed as a "functional" of the electronic density

$$E = E[\rho] = \mathbf{T}_e[\rho] + \mathbf{V}_{ee}[\rho] + \mathbf{V}_{Ne}[\rho]. \tag{2.8}$$

Now the problem becomes finding a functional form for each of the terms on the right hand side with respect to ρ. The classical Coulomb electrostatic contributions can be easily expressed as:

$$\mathbf{V}_{ee}^{\text{classical}}[\rho] = \frac{1}{2} \int \int \frac{\rho(\mathbf{r}_i)\rho(\mathbf{r}_j)}{\mathbf{r}_{ij}} d\mathbf{r}_i d\mathbf{r}_j \tag{2.9}$$

$$\mathbf{V}_{Ne}[\rho] = \int \rho(\mathbf{r}_i) \sum_{A=1}^{N_N} \frac{Z_A}{r_{iA}} d\mathbf{r}_i \tag{2.10}$$

which are just the classical expressions of an electron density interacting with another electron density or a nucleus, but given the relationship between electronic density and wave function, can also be expressed in terms of a wave function.

But this leaves some unknowns: the electron kinetic energy term, and non-classical electron contributions such as exchange (due to the required anti-symmetry of fermion/electron wave functions) and correlation (due to the fact that wave functions cannot actually be expressed as a single Slater determinant as in the HF approximation). These remaining terms are not as trivial to evaluate. Kohn and Sham [97] (KS) proposed the following *ansatz* for the unknown terms, as well as a series of equations that allow to solve the electronic density quantum problem computationally [98]. In the KS approach, the kinetic energy of the electrons is approximated as that of a non-interacting system of electrons, described by orbitals, which is known exactly and given by

$$\mathbf{T}_S = -\frac{1}{2} \sum_{i=1}^{N_e} \int \phi_i^\star \nabla^2 \phi_i d\mathbf{r}_i, \tag{2.11}$$

where ϕ is the orbital of the non interacting system, and $\phi^\star$ is its complex conjugate. An extra unknown term $\mathbf{T}_\mathrm{C}$ covers the gap between the non interacting and the interacting system. $\mathbf{T}_\mathrm{C}$, together with the non classical electron-electron interactions (E_ncl), is gathered into a single term, the exchange correlation functional E_XC. Notice that despite its name, it includes not only "potential" energy terms, but also a part of kinetic energy.

The final open question in density functional theory is, then, the expression for this exchange and correlation functional. The functional needs to fulfill a number of conditions such as reproducing exchange and correlation holes [98], and many possible forms have been proposed which can be organized in a hierarchical ladder [99] of increasing accuracy but also complexity and computational cost. The simplest approach, known as the local density approximation (LDA), obtains the exchange from an electron gas of uniform density, equal to that of the real system at a given point in space; while the correlation part still needs to be numerically estimated [98]. Due to their simplicity, LDA methods are nowadays mostly obsolete. Generalized gradient approximation (GGA) methods include terms that depend on the gradient of the electronic density. Meta-GGA methods include further derivatives of the electronic density. Hybrid functionals combine parts of the exchange energy from HF calculations, where it can be exactly calculated, with a concomitant increased computational cost. Due to the good compromise between computational cost and accuracy found in GGA functionals, they are the functional of choice in this work.

2.2 Neural Network Potentials

2.2.1 Introduction and Historical Development

Neural networks (NN) are one method within the ever-growing field of machine learning [100] (ML), which seeks to develop algorithms that can learn from experience and excel at tasks such as classification and prediction. Neural networks have been particularly successful, with applications ranging from image classification [101] to game playing such as Go [102], to cite only a few impressive examples. Of interest for this work, neural networks are also capable of reproducing functions, such as potential energy surfaces. Neural networks have found applications in chemistry and physics not only as potentials, but also as tools for NMR spectra classification [103] and drug design [104], to once again cite only a few examples. Other ML methods have also found success in these fields.

As their name implies, neural networks started as an analogy to biological neurons with the work of McCulloch and Pitts in 1943 [105], where they first formalized a system of individual artificial neurons that take an input, and yield a given output when a threshold is reached. In 1958, Rosenblatt [106] further developed neural networks by introducing input and output layers to create a binary classifier, the perceptron. In 1969 Minsky and Papert [107] introduced further layers into the network, called hidden layers, with the goal of reproducing all possible logic functions, a presumed requirement for general computability. Werbos in 1975 [108] developed the theory of backpropagation, allowing for the training of arbitrary, multi layer networks. Still, other methodological problems and the substantial computational effort required to train and implement neural networks delayed the field until better computers and algorithms were developed in the 80's and 90's, giving rise to the recent explosive growth of the machine learning field.

Of interest here is the use of neural networks as machine learning potentials (MLP), also known as a neural network potential (NNP). That is, utilizing neural networks not as classifiers but as interpolators, that can learn to reproduce the complex potential energy surface (PES) of a system of atoms. It can be proven [109] that a neural network of arbitrary size can reproduce any "well behaved" function from known values of that function, making neural networks a class of universal approximators. Here, the target function is the PES as a function of atom coordinates. That such a function linking coordinates and energies and forces exists is suggested by the Born-Oppenheimer approximation [94]. NNPs are usually constructed by sampling the PES with an "expensive" but accurate electronic structure method,

and then utilizing the forces and energies predicted by a NNP trained on this data to perform any number of classical simulations. The first potentials with NNs at their core were published in the 90's (see 86, 110, 111 for comprehensive reviews), starting with the work of Doren *et al.* in 1995 [112], but most of these consisted of simple feed-forward NNPs with methodical limitations. They constitute the first generation [113] of NNPs, which could only treat low dimensional molecular systems. A breakthrough in the field appeared in 2007, with the introduction of high dimensional neural network potentials by Behler and Parrinello [114–116] (HDNNP), inaugurating the second generation of NNPs, which allowed for the construction of a NNP for systems of arbitrary size and composition while respecting all the necessary invariants required for a force field. More recently, third and fourth generation NNPs have been introduced, the former ones capable of predicting local charges and thus electrostatic interactions [89] for long range interactions, and the latter making use also of non-local information for a similar purpose [113].

Research in the field of MLPs is highly active, with many new methods and algorithms introduced since the advent of HDNNPs. This includes developments that adopt other forms of NNs, such as convolutional [117] or deep [118, 119] networks. Others still rely on simpler NNs but adopt different atomic environment descriptors such as permutation invariant polynomials (PIPs) [120], smooth overlap of atomic positions (SOAPs) [121], spherical harmonics [122], electronic structure based descriptors [123], weighted atom centered symmetry functions (wACSFs) [124], and many others [125–127]. It has recently been shown that the predictive power for many of these descriptors is comparable [128]. Other methods from machine learning, not based on NNs, have also increasingly been adopted as PES replicators. Among them we can find Gaussian approximation potentials (GAPs) [129], spectral neighbor analysis (SNAP) [130], moment tensor potentials [131], and kernel ridge regression (KRR) [125, 126].

The following sections describe in more detail first simple feed forward neural networks in their roles as potentials, discussing their limitations, and then the development of the more complex high dimensional potentials that solve these problems.

2.2.2 Feed Forward Neural Networks as Potentials

A simple feed-forward neural network potential is presented in fig. 2.1. The network is divided into layers: the input layer, where the coordinates of the system under study are entered into the network, pre-processed in the form of for example internal coordinates; one or more hidden layers, which process the values obtained from the previous layer and pass the results forward to latter layers; and an output layer, which outputs the prediction of the NNP, in this case the energy.

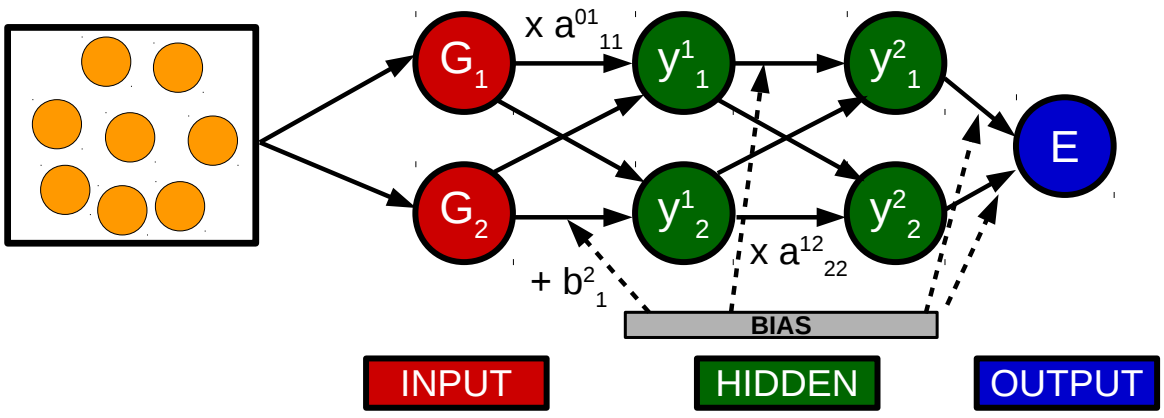

Figure 2.1: Diagram of a simple feed forward neural network potential, with an input layer, two hidden layers, and an output layer, each respectively with two, two, two and one node. This makes up a 2-2-2-1 architecture.

Each layer is made up of individual nodes or neurons, with each node usually standing in for a given mathematical operation or function. Each node outputs a value, and is fully connected to all nodes in the next layer. Each node in a given layer receives a value from all nodes in the previous layers, and takes them as an input to its own function, combined with weights and biases. Nodes in the input layer usually consists of functions of the atomic coordinates, the output node consists of a linear sum of the incoming nodes in the last hidden layer to prevent constraining the possible output values in the network, and the nodes in the hidden layers contain so called activation functions. The total number of layers and the number of nodes in each layer define the architecture of the NNP.

Weights and biases are present between the connections of every node. Weights are multiplicative and biases are additive modifications to the values going from one node to another. NNPs with the same architecture will have the same number of weights and biases, but their value depends on the training of the NNP. The objective of weights and biases is to moderate the response between nodes: trivially, if a weight is zero, the connection between two corresponding nodes will be severed, while if the weight is positive/negative, there will be a positive/negative correlation between those nodes.

The goal of the activation functions is to moderate the response of each node in the network, and introduce non-linearity into the NN *ansatz*, since otherwise the network could be reduced to a simple series of linear equations that would not be effective as a universal function approximator. Many types of activation functions are available, which in general share an asymptotic behavior for extreme input values (very positive or negative), and a non-linear region between the asymptotes. One of the more commonly used ones is the hyperbolic tangent, plotted in fig. 2.3 and given by the equation

$$f = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}}, \tag{2.12}$$

where, in the context of a NN, x is the sum of outputs from nodes in a previous layer modified with weights and biases, and f is the output from this node to the next layer. Due to its behavior, the hyperbolic tangent models a threshold function, with a binary output (the asymptotes), at both sides of the turning point of the function.

For feed forward neural network potentials it is possible to express the network also as a closed analytic equation. For a NNP with one input layer, two hidden layers and an output layer (the energy E), numbered in that order from 0 to 3, and with $N_{\mathrm{I}}, N_{\mathrm{H1}}, N_{\mathrm{H2}}$, and 1 node respectively, this expression looks like

$$E = b_1^3 + \sum_{k=1}^{N_{\mathrm{H2}}} a_{k1}^{23} \cdot f_k^2 \left(b_k^2 + \sum_{j=1}^{N_{\mathrm{H1}}} a_{jk}^{12} \cdot f_j^1 \left(b_j^1 + \sum_{i=1}^{N_{\mathrm{I}}} a_{ij}^{01} \cdot G_i(R) \right) \right), \tag{2.13}$$

where f_c^{α} is the activation function of node c in layer α, b_c^{α} are additive biases on the input of node number c in layer α, $a_{cd}^{\alpha\beta}$ are multiplicative weights connecting node c in layer α, with node d in layer β, R are the coordinates of the atoms in the system, G_i are functions of these input coordinates, and i, j, k count over the nodes in the input layer, first, and second hidden layers, respectively.

As described, these first generation neural networks are capable of reproducing target sampled PESs. Their form is analytic, which enables not only the use of backpropagation for fitting, but also the use of derivatives of the output energy to also predict forces. The method also exhibits a number of limitations: many of the early neural networks did not exhibit the invariances required for a more universal potential (translation, rotation, and permutation invariance) that could be applied to any system, had a fixed system size (that is, they would only work for the exact number and type of atoms that the NNP was trained for), and would scale badly if applied to larger systems due to the increasing number of input nodes. To overcome these limitation, the second generation of high-dimensional neural network potentials was introduced.

2.2.3 High-Dimensional Neural Networks

The Behler-Parrinello second generation potentials, depicted in fig. 2.2, solve the limitations of the first generation methods in two main ways. The first modification is that the total energy of the system is partitioned into individual, fictitious atomic energies, and the energetic contribution from each atom is calculated by an individual NNP. These individual NNPs are shared across atoms of the same element, but since each atom will exhibit different chemical environments, their predictions will differ even for atoms of the same element. This solves in part the problem of system size, since adding atoms to a system just means adding more individual networks and adding another term to the total energy of the system.

The second big modification is the change to a symmetry function (SF) based approach for the input layer of each elemental NNP. These symmetry functions take into account the local environment around each atom up to a given cutoff radius, and are constructed in such a way to respect translational, rotational and permutational invariance. This approach not only fixes the problem with invariances, but also further solves the size problem, since now the networks only look at the environment around each atom instead of the system as a whole.

Symmetry functions achieve spatial invariance by utilizing internal coordinates (distances and angles between atoms), permutation invariance by summing over contributions from pairs or triplets of atoms, and locality by including only atoms up to a given cutoff radius R_C. Due to the presence of a cutoff radius, continuity of the SFs functional form for the purpose of derivation needs to be ensured. For this purpose, a cutoff function is included in every SF. This cutoff function can have multiple expressions as long as it is continuous and differentiable, and goes to zero for values of $R_C > R_{ij}$, with R_{ij} the distance between a given atom j and the atom i at the center of the symmetry function. One possible example is a split function based on the cosine

$$f_C(R_{ij}) = \begin{cases} 0.5\left[\cos(\frac{\pi R_{ij}}{R_C}) + 1\right] & R_{ij} \leq R_C, \\ 0 & R_{ij} > R_C. \end{cases} \tag{2.14}$$

Many types of SFs exist [110], but two of the simpler ones are commonly used, referred as radial (G^2) and angular (G^4) symmetry functions. These are plotted in fig. 2.3, and their functional form for a given central atom i is

$$G_i^2 = \sum_j \mathrm{e}^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}), \tag{2.15}$$

$$G_i^4 = 2^{1-\zeta} \sum_{j,k} (1 + \lambda \cos(\theta_{ijk}))^{\zeta} \cdot \mathrm{e}^{-\eta(R_{ij}^2+R_{jk}^2+R_{ik}^2)} \cdot f_c(R_{ij}) \cdot f_c(R_{jk}) \cdot f_c(R_{ik}), \tag{2.16}$$

where j,k are the indices of all atoms within the cutoff radius of the central atom, R_{ij} is the distance between atoms i and j, θ_{ijk} is the angle between atoms i,j,k, and $\eta, R_s, \zeta, \lambda$ are parameters that in effect generate different symmetry functions out of the same initial functional form. These parameters are defined *before* a fit takes place, and in a way also form part of the architecture of the NNP. The collection of symmetry functions generates a set of descriptors for the atomic environment around each atom, usually called a symmetry function vector, or the atomic environment fingerprint. For radial SFs, one SF is present for each elemental pair present in the system (for atomic NNPs of element A, G^2_{AA}, G^2_{AB}, G^2_{AC}, etc., if the first element mentioned is considered the central atom of the SF), which may or may not share parameters. The same applies to angular SFs, but with more copies of the SF present due to relying on two neighboring atoms (for atomic NNPs of element A, G^4_{AAA}, G^2_{AAB}, G^2_{ABB}, G^2_{AAC}, G^2_{ABC}, G^2_{ACC}, etc., if the first element mentioned is the central atom of the SF).

The goal of the SF set is to generate a unique input vector to the NNP for each possible atomic

environment, so that the NNP can differentiate the various atomic configurations and assign a unique energy and force to each one. In a sense, NNP are not only working as universal approximators, but also still as classifiers, associating an energy "label" to each SF vector. Generating these sets of symmetry functions is often a trial and error process that takes into account as a starting point the usual geometries of the system under study, although some more automated processes to aid in SF selection have appeared in recent years (see sec. 4.1).

As mentioned in the previous section, forces can be obtained as a derivative of the output node of the network, since the whole network can be expressed analytically into a closed equation. For a HDNNP, this is given by

$$F_{k,x} = -\frac{\partial E_{\text{tot}}}{\partial R_{k,x}} = -\sum_{i=1}^{N_{\text{atoms}}} \frac{\partial E_i}{\partial R_{k,x}}, \tag{2.17}$$

where $F_{k,x}$ is the force on atom k in the Cartesian direction x, E_{tot} is the total energy output by the neural network, $R_{k,x}$ is the x coordinate of atom k, N_{atoms} is the total number of atoms in the system, and E_i is the partial energy contribution of the individual atom i to the total NNP energy. Since only atoms in the cutoff radius of atom k actually see this atom, only their individual atomic networks have a functional dependence on the position of k through the symmetry functions. As such, the sum can be constrained to only atoms within one cutoff radius of k. Then, the chain rule can be applied to separate into derivatives containing the symmetry functions as terms. Since these symmetry functions depend on atoms neighboring those atoms in the first neighbor shell of k, force calculation depends actually on atoms up to two cutoff radii away from k

$$F_{k,x} = -\sum_{i=1}^{N_{R_C}} \frac{\partial E_i}{\partial R_{k,x}} = -\sum_{i=1}^{N_{R_C}} \sum_{j=1}^{M_i} \frac{\partial E_i}{\partial G_{i,j}} \frac{\partial G_{i,j}}{\partial R_{k,x}}, \tag{2.18}$$

where atom i is a neighbor of k, N_{R_C} is the number of neighbors to atom k within a cutoff radius of R_C, M_i is the number of symmetry functions of atom i, and $G_{i,j}$ is symmetry function number j of atom i, which might depend on the coordinates of atoms up to $2R_C$ away from k.

The fitting procedure for a NNP begins by proposing an error function, which for the case of the energy prediction of a HDNNP is

$$\Gamma = \frac{1}{N_{\text{structs}}} \sum_{i}^{N_{\text{structs}}} (E_i^{\text{NNP}} - E_i^{\text{REF}})^2, \tag{2.19}$$

where N_{structs} is the number of structures in the training data, i counts over these structures, E^{NNP} is the energy predicted by the NNP with a given architecture and utilizing the current set of weights and biases, and E^{REF} is the reference target energy as provided by some electronic structure method. The goal then is to minimize the quantity $\frac{\partial \Gamma}{\partial a}$ with a standing in for the weights and biases in the network, which results in equations that give the necessary updates to weights and biases given the current error in predictions. Weights and biases are initialized in a range utilizing the Nguyen-Widrow scheme [132], and the update to these weights is improved by utilizing the Kalman filter [133] procedure. Training of the network proceeds in distinct epochs, where within one epoch all relevant pieces of information (energies and forces) have been used a maximum of once, or have not been used if the prediction for that particular quantity is already good enough within a threshold. A simple quality measurement for the fit is the root mean square error (RMSE), which can be calculated for energies and forces and as the name implies, calculates the current mean error in a predicted quantity. For the energy, this quantity is

$$\text{RMSE}(E) = \sqrt{\frac{1}{N_{\text{structs}}} \sum_{i}^{N_{\text{structs}}} (E_i^{\text{NNP}} - E_i^{\text{REF}})^2}. \tag{2.20}$$

Unsurprisingly, this looks similar to the error function Γ. As the fitting epochs progress, overfitting can occur: the NNP fits the known data points with ever increasing accuracy at the cost of losing interpolation capacity and prediction capability between the known data. To avoid this, a portion of the data is reserved as a testing set. The fit is only performed with information from the training set, but errors are still calculated for the testing set. If no overfitting is present, the RMSE for both the training and testing set should be similar. Overfitting can be detected when the error for testing set is significantly higher than that for the training set: the NN reproduces the learned data better but predictive power for new data is lost in the process. In practice, during a fitting procedure, the errors for both data sets are tracked, and the training can thus be stopped before the errors start diverging, or the training can be allowed to continue for a set number of epochs and the weights and biases of the best fitting epoch chosen. This "best epoch" should be a combination of the lowest error for the training data, but also with low divergence between this error and the testing set error.

As described, HDNNPs are capable of learning the PES of a wide variety of chemical systems, without the need for crafting specific coordinate systems for each case and without a size limit. They are thus highly transferable and expandable. They of course also exhibit some limitations: due to still being nonphysical, without an underlying physical expression or safe physical limits, care should be taking to avoid unsampled configurations or extreme distances when performing simulations. For example, the energy and forces of a dimer that comes too close together should increase, but if this region is not sampled, the NNP does not have an infinite energy limit for overlapping dimers. Due to the local construction of the SFs, the NNP cannot capture all long range interactions. That said, it is possible to fit NNPs to data with dispersion corrections for systems where these interactions are important, such as in water [90, 91, 134–136]. In those cases, the NNP has been successful in reproducing properties that depend on these long range interactions, although there is no explicit long range term in the basic HDNNP *ansatz*. Long range interaction capabilities are currently being solved with the recent advent of third and fourth generation [113] NNPs, where charges are also predicted and thus long range charge interactions are taken into account explicitly.

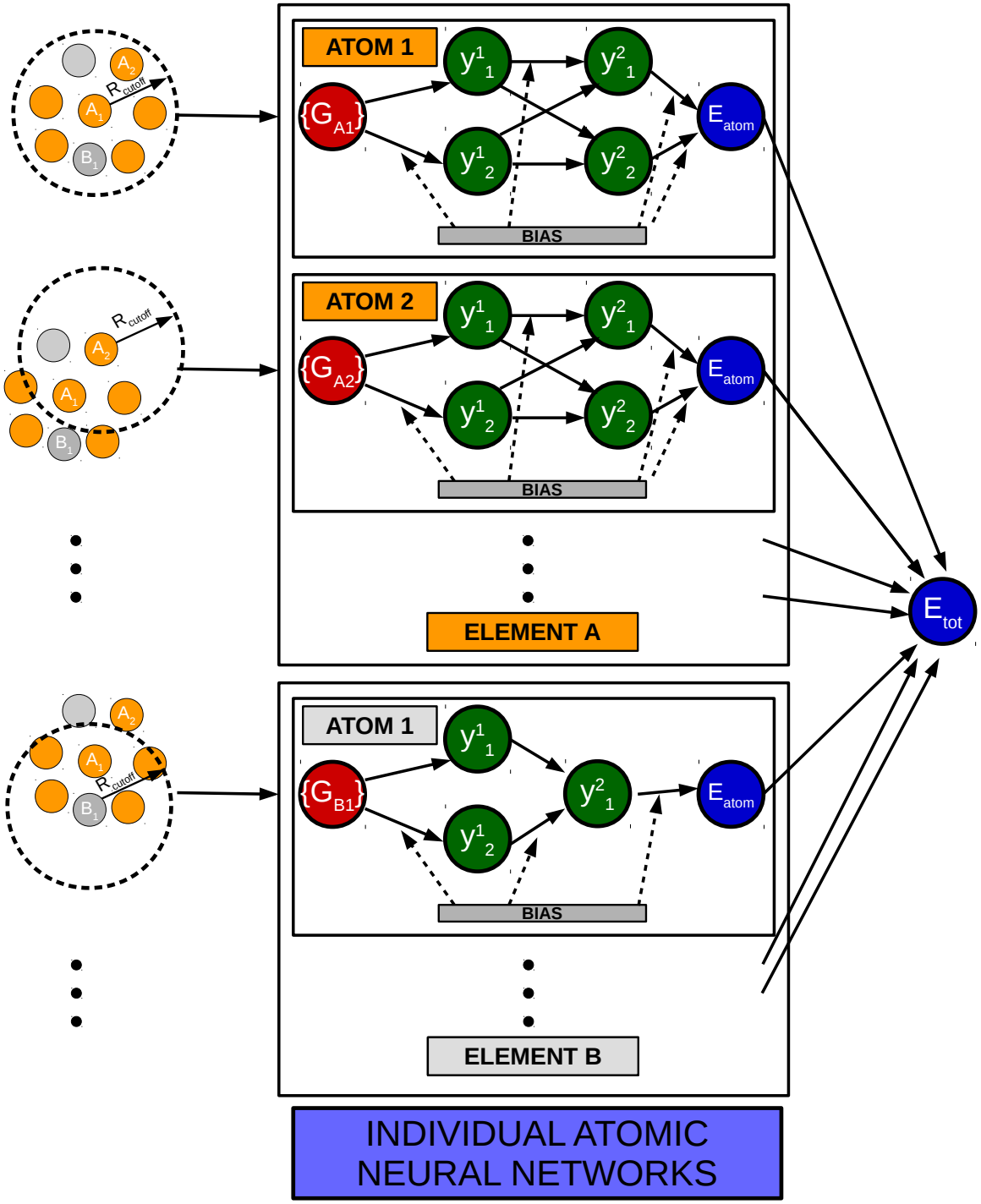

Figure 2.2: Diagram of an example high dimensional neural network potential for a system with two elements. The energy contribution of each atom to the total energy is based on atomic neural networks, which depend on a series of local symmetry function which include neighbors up to a given cutoff radius. The atomic networks are different (different architecture, weights, symmetry functions) for different elements.

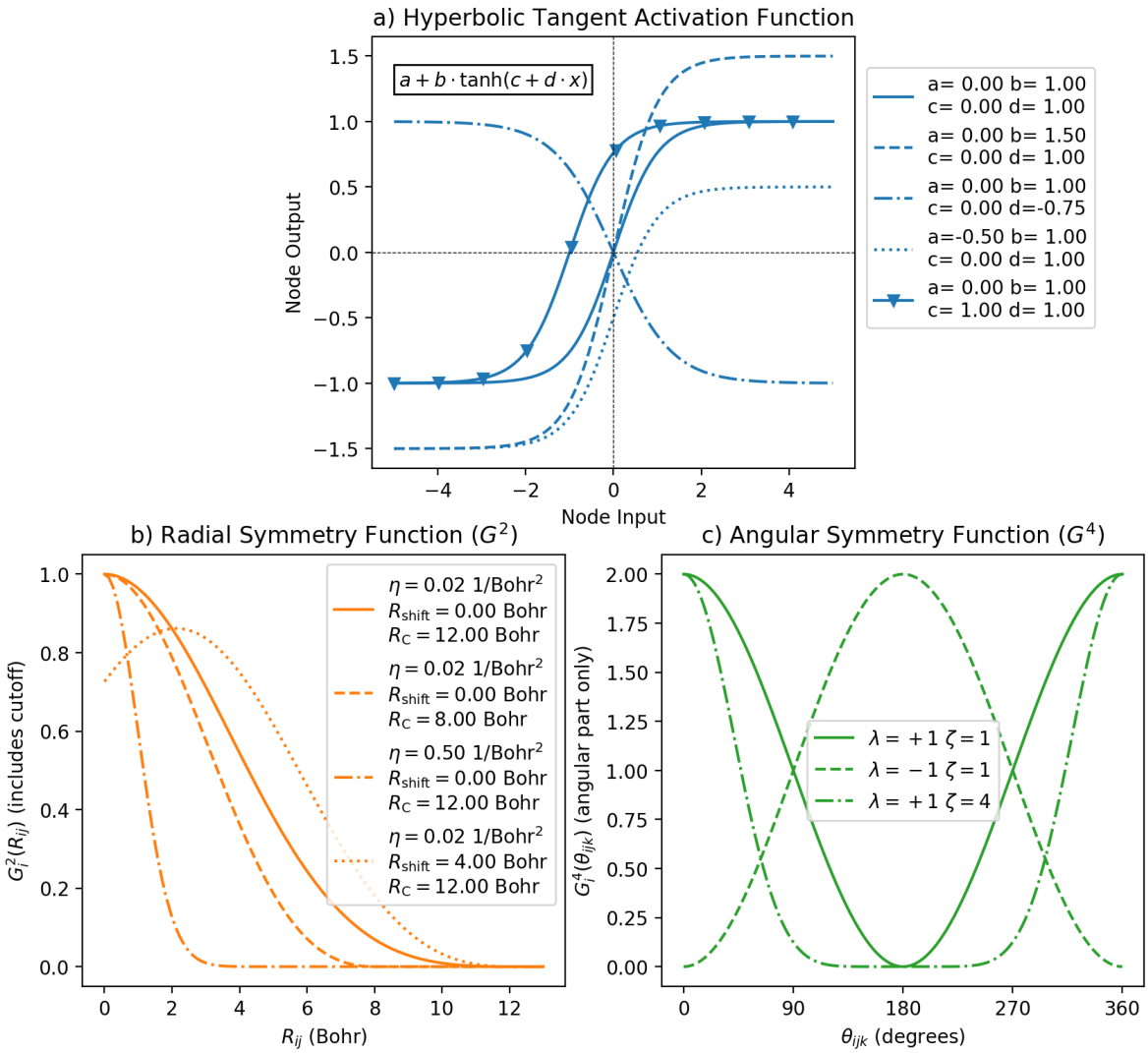

Figure 2.3: Different functions relevant for the construction of NNPs: a) Hyperbolic tangent activation function with different parameters showing its functional flexibility. b) Radial SF including cutoff function modifier, with different parameters. c) Angular part of the angular SF with different parameters.

2.2.4 Environment Decomposition

Due to the locality of the HDNNPs, it is possible to employ an environment decomposition approach, where information from smaller systems can be utilized in simulations containing larger configurations, both for fitting and simulation purposes. In a way, the local environments described by vectors of symmetry functions values can be thought of as building blocks, that can be put together to construct a larger system, adding up to configurations that might not have even been explicitly present in the sampled dataset. This facilitates the sampling required to construct a NNP potential.

The environment decomposition approach has been successfully utilized for both molecular [137] and atomistic [61, 93] systems. In the case of molecular systems, the approach was successful even when cutting through bonds between atoms, although it of course requires saturating the split bonds to avoid dangling bonds. This behavior could in principle be exploited in the future to construct a dataset that actually consists of a library of atomic environments, with the help of other algorithms that can curate the database (see sec. 4.1 for more on this).

Two problems arise with this approach for the study of copper deposited on ZnO. The first is that the simulation code of choice, VASP (see sec. 3.1), is a strictly periodic code, due to utilizing planewaves as its basis set for performing DFT calculations. Non-periodic structures can be approximated by allowing for enough vacuum in the non-periodic direction so that interactions between periodic images are minimized. This works well in the case of slabs, and for clusters with atoms of very similar elements [61], but presents problems of non-convergence of the electronic calculation for materials where a net dipole or charge distribution can be expected. An example of such a material is of course, zinc oxide: two of the main ZnO facets, the (0001) and (000$\bar{1}$) Miller cuts, present an inherent dipole moment in the direction perpendicular to the surface; and for the other non-polar ZnO surfaces, any attempt at extracting an environment by performing spherical cuts around a central atom will inevitably result in a mismatched sphere, with section where either zinc or oxygen atoms prevail.

This is related to the second problem: for such a ionic material, irregular cuts can lead to forces that do not converge as the size of the spherical cut increases, electronic structure problems notwithstanding. Small scale tests with a ternary system consisting of Cu deposited on ZnO show that this is the case. As the radius of the spherical cut increases, different layers of Zn and O are cut through, resulting in very different looking environments, and a force on the central atom that oscillates. This force also never reaches the same value as compared to a calculation with the full system included.

To solve these problems in the case of ZnO slabs, the cube cut algorithm was developed and utilized, as described in sec. 2.6.1. This allows for cutting environments around a central atom, while maintaining correct periodicity for VASP calculations. Figure 3.2 in the computational details section shows an example of environment decomposition applied to large supported copper clusters on zinc oxide.

2.2.5 Sampling

A central problem in the construction of NNPs is that of sampling. Relevant regions of the PES need to be sampled, but these can not always be known *a priori*, without already running simulations that require a NNP. At the same time, wasting computational time on repeated configurations or physically not accessible or irrelevant sections of the PES is to be avoided. A random sampling of configurations (for example, for an atomic cluster), would be inefficient, since a majority of the visited configurations would correspond to high-energy, irrelevant positions of the PES. A NNP trained on this data would not be able to reproduce the behavior of lower energy, thermally accessible configurations. Another possibility would be to sample the PES with a simple and fast method, such as an already available force field. The problem with this is of course that for most systems, force fields might not be available, or they might only be available for certain phases (e.g.: bulk but not surfaces) of the material, or they might be much less accurate than feasible *ab-initio* calculations. A final option would be to run, for example, *ab-initio* molecular dynamics simulations to sample directly the PES. The problem here is that although the obtained data would already have the required *ab-initio* quality, the generated

configurations would be highly correlated, that is, they would only cover a small portion of the PES for a significant investment in computational time.

Instead, an iterative sampling procedure is adopted [88]. An initial NNP is created from known experimental structures thus sampling known points on the PES; this potential is utilized to perform simulations, these simulations return new relevant and (hopefully) physically correct structures that center around the accessible sections of the PES, which are utilized to fit a new NNP, and so on until the network achieves the desired convergence level for the target PES. More detail is given in the computational methods sec. 3.3.

In this iterative approach, a method is required to detect new configurations that actually need to be recalculated with an expensive reference method. This is also sometimes known as "active learning", where the machine learning algorithm actually influences which data is to be taken into consideration instead of only passively accepting all the available information. A first indication is found when a symmetry function in a simulations adopts a value that is higher or lower than the known range present in the available reference dataset. This is known as an extrapolation [116], and it is simple to detect. Unsampled areas within the known symmetry function range, also known as "holes", are not as easy to detect since here a continuum of values is present. Additionally, such holes can appear in multidimensional space [116], since what is important is not any single value of a SF but the whole SF vector. As such, an approach that takes the whole simulation structure or atomic environment at once is preferred, such as utilizing an ensemble of NNPs [88, 115], or the bin and hash method (BAH) described in chapter 4, or other methods also described in the introduction to that section.

2.3 Genetic Algorithm Global Optimization Search

This section is adapted from Reference [138] M. L. Paleico and J. Behler, "Global optimization of copper clusters at the ZnO($10\bar{1}0$) surface using a DFT-based neural network potential and genetic algorithms," J. Chem. Phys. **153**, 054704 (2020), with the permission of AIP Publishing.

Evolutionary and genetic algorithms [139, 140] (GA) belong to the wide family of global optimization (GO) methods. As their name implies, these algorithms attempt to emulate biological evolution as a way of optimizing the solution to a given problem. For chemical systems [141–146], the algorithm encodes the structure of the system into "genes", a fitness value is assigned to the structure (usually a function of the energy, but other targeted properties are possible [147]), and finally these genes and structures are combined and variability introduced in the form of mutations, with the goal of generating better and fitter candidates. In this way, the configuration space is quickly explored and evaluated at specific points. Fitter candidates are chosen more often for crossover operations, so the configuration space around well-scored structures gets explored in more detail. Thus, GA searches provide a good balance between exploration of the PES at large, and exploitation of known good candidates [148]. In this work, a GA is utilized to search for the global minimum structures of small Cu clusters on ZnO.

In the particular case of atomistic systems, the GA operates directly on the coordinates of the atoms. This is often described more precisely as an evolutionary algorithm (since it operates directly on the "phenotype" of the configuration), but the names are often used interchangeably in the literature and the genetic algorithm nomenclature will be the preferred one in this text. Crossovers in coordinate space are performed by dividing the two candidate clusters with a random plane [140], and combining one half from each cluster while ensuring that system size and stoichiometry is maintained. After crossover, a geometry minimization is performed, to relax the high-energy configurations that might arise from this sort of random recombination and to be able to evaluate all candidates under the same conditions. This is reminiscent in some ways of basin hopping Monte Carlo [149] (BHMC), and it has been long acknowledged that both algorithms in fact sample the same transformed PES and thus lead to similar results [149]. Mutations for atomistic systems consist in large scale modifications of the atomic coordinates, reminiscent of Monte Carlo (MC) trial moves, which attempt to jump into different areas of the PES.

Furthering the biological analogy, some GA implementations make use of the idea of a generation: candidates are generated from pairings, mutated, evaluated and ranked in batches. Only the fittest candidates in one generation give rise to the next generation. This approach is useful for GA searches that rely on electronic structure calculations, since all the batched structures can be calculated slowly at the same time. Otherwise the first structures generated and available would dominate the generation of future structures. This *ansatz* has been shown to not be required [150] for a successful search, and here we adopt a continuous generation and evaluation of structures, that is made possible by the speed of the NNP.

Mutations should apply large changes to the atomic structure of the system, allowing for sideways jumps in configuration space, but still generate "reasonable" structures so time is not wasted calculating hopeless candidates. For supported clusters, moves only affect the atoms defined as belonging to the cluster, while the support is only affected by geometry minimizations and its direct interaction with the cluster. The mutation moves implemented for this case are: rattle (a proportion of the atoms is displaced a random amount), twist (the whole cluster is rotated a given angle, with the rotation axis perpendicular to the surface), angular (atoms are moved to positions at the surface of the cluster), mirror (a random plane is used to mirror the whole cluster), molecular dynamics (MD) (a short MD run at high temperature followed by a rapid cooling). Moves have been inspired by a variety of sources [151], including basin hopping [149], simulated annealing [152] and minima hopping [153]. More detail of the implemented moves can be found in ref. 138.

Given the setup of GA optimization, it is possible to repeatedly encounter the same configurations,

particularly for smaller systems. This is problematic because it can lead to the same structural pattern "poisoning" the breeding population, and efficient exploration of the PES stops. To avoid this, some structural diversity needs to be enforced [150]. Here we utilize a simple comparator based on interatomic distances [141], further detailed in sec. 2.6.2.

The breeding population is a fixed size collection of the fittest, structurally distinct configurations at any given point. Fitness is defined as [154]

$$f_i = 0.5 * (1 - \tanh(2 \cdot (E_{\max} - E_i)/(E_{\max} - E_{\min}) - 1)), \tag{2.21}$$

where f_i is the fitness of a particular candidate, E_i, $E_{\min}$ and $E_{\max}$ correspond to the energies of the candidate and the minimum as well as maximum energies in the whole population. This creates a fitness distribution that falls sharply as we get away form the optimal candidate and has a constant low value for the worst candidates.

A random roulette wheel algorithm [155] is utilized to extract breeding pairs from this restricted population, by evaluating the criterion

$$f_i > f_{\max} \cdot \mathrm{ran}(0.0, 1.0) \tag{2.22}$$

to decide if a candidate is kept, where f_i is the fitness of the randomly chosen candidate, $f_{\max}$ is the fitness of the fittest candidate, and ran generates a random number between 0.0 and 1.0. With this equation, fitter candidates will tend to get chosen more often since their ratio of $f_i/f_{\max}$ will be closer to 1.0, but all candidates in the active population have a non-zero chance of being picked.

2.4 Simulated Annealing

2.4.1 Introduction

Simulated annealing [152, 156] (SA) is a method for global optimization. A system is given enough kinetic energy to overcome potential energy barriers, and slowly cooled so that it can find its way into lower energy configurations. In this work, SA is utilized to optimize large copper clusters on ZnO, which are too large for global optimization with other approaches.

A simulated annealing optimization run can be divided into three sections: the heating up phase, the temperature hold or randomization phase, and the cooling down phase. Figure 2.4 a) shows a scheme of a planned SA run, and b) the real temperature profile obtained from simulations with a Cu_{300} cluster. The heating phase is intended to take the system from its initial configuration (which will just be a guess) up to beyond the melting temperature. The speed with which this phase can be completed depends mostly on the structural and numerical stability of the system: a completely periodic system might have problems with a sudden increase in temperature from 0 K (starting from a geometry minimization) past 1000 K. In the case of the supported copper clusters, this is not really a problem, and the heating phase can be safely skipped.

The temperature hold phase allows for the atomic configuration to be randomized. If this phase is not long enough, the resulting cooled down clusters will depend on the starting conditions. Since the starting conditions are just a guess, this would not be ideal. Additionally, this phase allows for different clusters to be obtained: since molecular dynamics simulations are actually deterministic, if we were to cool down starting from the same point we would obtain the same configurations. One could change this by randomly resetting the temperature/atomic velocities before the cooling down, but it is safer to invest more steps into the hold phase.

The cool down phase is the critical part of the simulation, and usually the one that takes the longest. Instead of performing a linear temperature ramp down, it is better to slowly reduce the temperature in constant temperature steps. This makes it easier to assign a defined temperature to each step, and allows for a better analysis of the system as the simulated annealing progresses. Here there are two

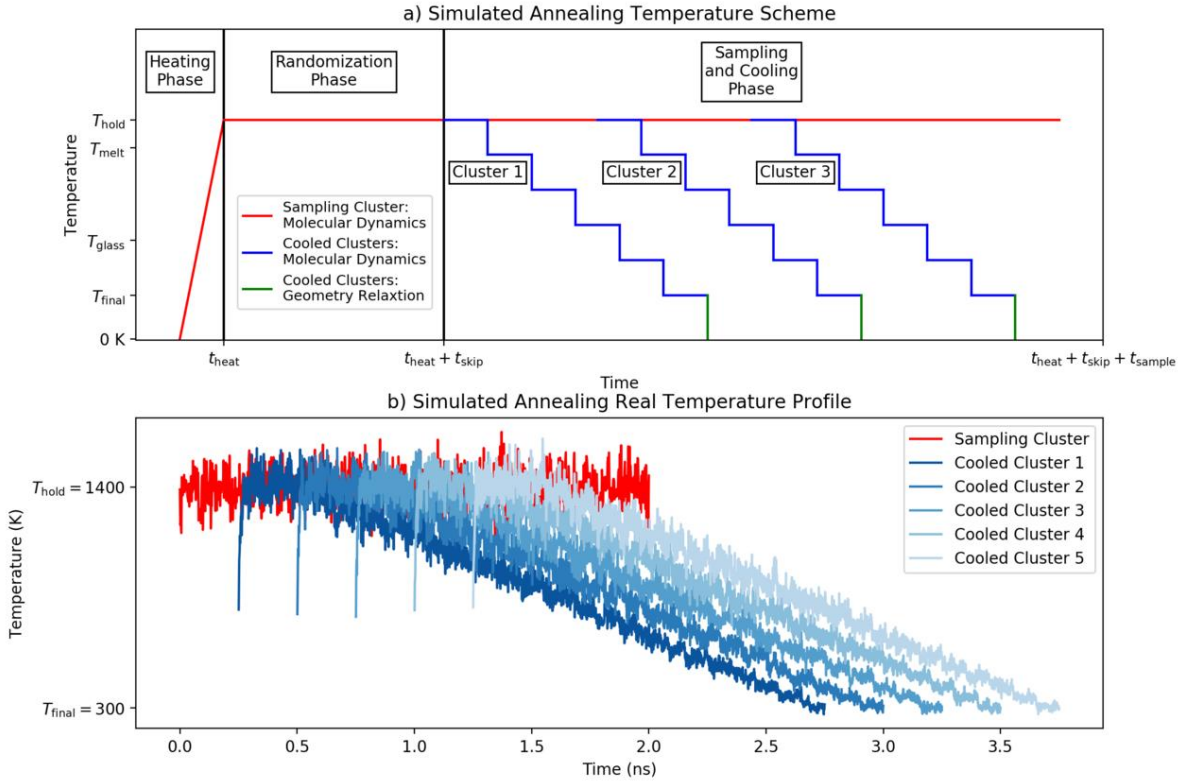

Figure 2.4: a) Scheme of a simulated annealing simulation. In the heating phase, the system is brought up to the target temperature for the randomization phase. During this phase, the system is maintained at a high temperature, beyond the melting point, to randomize atomic positions for t_{skip} time. Finally, in the cooling phase, structures are extracted from the high temperature simulation and slowly cooled down in temperature steps. b) Real temperature profiles from 5 SA runs for a Cu_{300} cluster. No heating phase is required, the system is randomized for 0.25 ns, and afterwards sampled every 0.25 ns a total of 5 times. Each cooled down cluster also has its own randomization period.

critical points: the melting temperature of the system, and the "glass temperature": the temperature below which atomic jumps between crystal lattice positions become vanishingly rare. If the system is cooled down too fast below the melting temperature, an amorphous configuration will result, since the atoms are "frozen" into a liquid-like configuration. Between the melting and the glass temperature atomic jumps are still often observed, which help the system optimize further, so the temperature ramp should still be slow in this section. Once the glass temperature has been reached, the system can be cooled down more quickly, and finally a geometry relaxation performed if the minimized structure of the obtained local minimum is required.

This setup is somewhat complicated by the fact that these temperatures are not known beforehand, but they can be estimated. The melting temperature depends on the potential being utilized, and can be modified by parameters such as cluster size [157–159], but we can usually predict that it will be below the experimental value. The glass temperature can be estimated by observing previous simulations, and detecting when atomic movements beyond vibrations cease to take place. Estimating these temperatures precisely beforehand is not critical, but can save on computation time since we can cool rapidly before the melting point and after the glass point.

Figures 2.5 and 2.6 show the trajectory of a single copper atom in a Cu_{500} cluster during the cool down process. The atom starts at the bottom of the cluster in a), and at the high temperature of 1400 K quickly travels the whole length of the cluster (b) through e)). As the temperature decreases, atom

mobility also decreases, and between subfigures n) and o), the atom finds its final position. The movements of the atom between a) and e) show that our randomization period is long enough, since the highlighted atom (and others in the cluster) has had enough time to travel through the cluster multiple times. Figure 2.6 allows for a better visualization of the atom going up and down in the layers of the cluster, as well as the regular disposition of the Cu atoms in the cluster in the final subfigure o).

2.4.2 Size-dependent Melting Point

It is well known that due to the increasing influence of surface atoms as the size of a nanoparticle decreases, many properties depend on the nanoparticle size [157–159]. In particular, the melting temperature is lower for smaller systems, since the under-coordinated atoms at the surface melt before the corresponding fully coordinated atoms in a bulk. The result is that the melting temperature follows the relationship

$$T_{\mathrm{m}} = T_{\mathrm{m}}^{\mathrm{bulk}}\left(1 - \frac{\beta}{R}\right), \tag{2.23}$$

where T_{m} is the melting temperature at a given size, $T_{\mathrm{m}}^{\mathrm{bulk}}$ is the melting temperature of the bulk (or more precisely, a corresponding infinite size nanoparticle), β is a complex constant that depends on the nanoparticle shape, the elements involved, etc. Thus a plot of observed melting temperature vs. $1/R$ will result in a line, whose intercept it the melting temperature at infinite size. If we assume that the nanoparticle can be replaced by an equivalent sphere of volume $V = 4/3\pi R_{\mathrm{effective}}$ and density $\delta = N/V$, the equation becomes

$$T_{\mathrm{m}} = T_{\mathrm{m}}^{\mathrm{bulk}}\left(1 - \frac{\beta'}{N^{-1/3}}\right), \tag{2.24}$$

$$\beta' = \beta\left(\frac{4}{4\pi\delta}\right)^{-1/3}. \tag{2.25}$$

This second version is easier to work with since the radius of the nanoparticle can be hard to estimate, particularly for supported nanoparticles where the nanoparticle has a specific contact angle with the support, which means we have to define an effective radius from the observed height and diameter of the nanoparticle [160]

$$R_{\mathrm{effective}} = \frac{\sqrt{(D/2)^2 + H^2}}{H}, \tag{2.26}$$

where D is the diameter of the particle and H its height when deposited on the support.

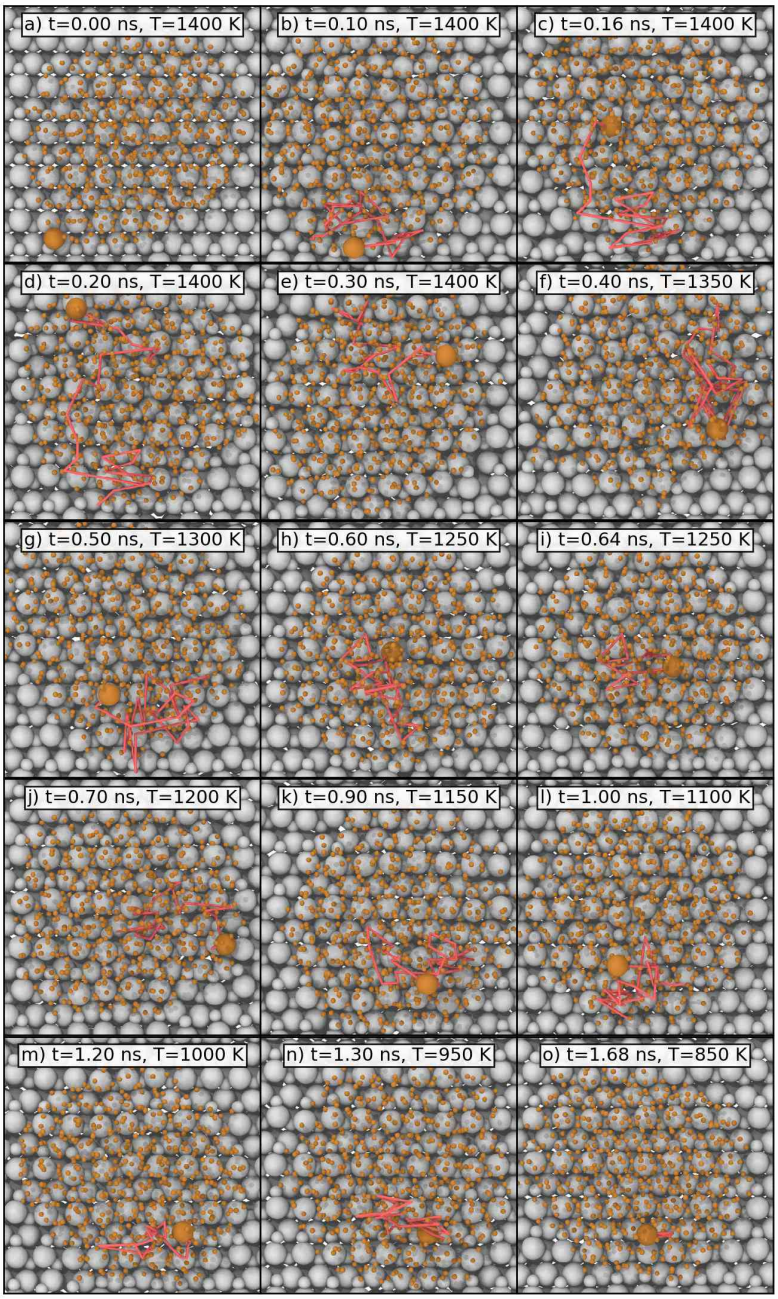

Figure 2.5: 0.1 ns trajectory line for a selected atom in a simulated annealing cycle for a Cu_{500} cluster. The titles in each figure show the corresponding target temperature in this step of the simulation, and elapsed time since beginning of simulation. a) Initial configuration, atom starts at the bottom of the cluster in contact with the support. d) Temperature is above the melting point of the cluster, so the atom easily manages to travel through the whole cluster. n) and o) Atom finds its final position, as the temperature is now below the melting point and atomic movements become rare.

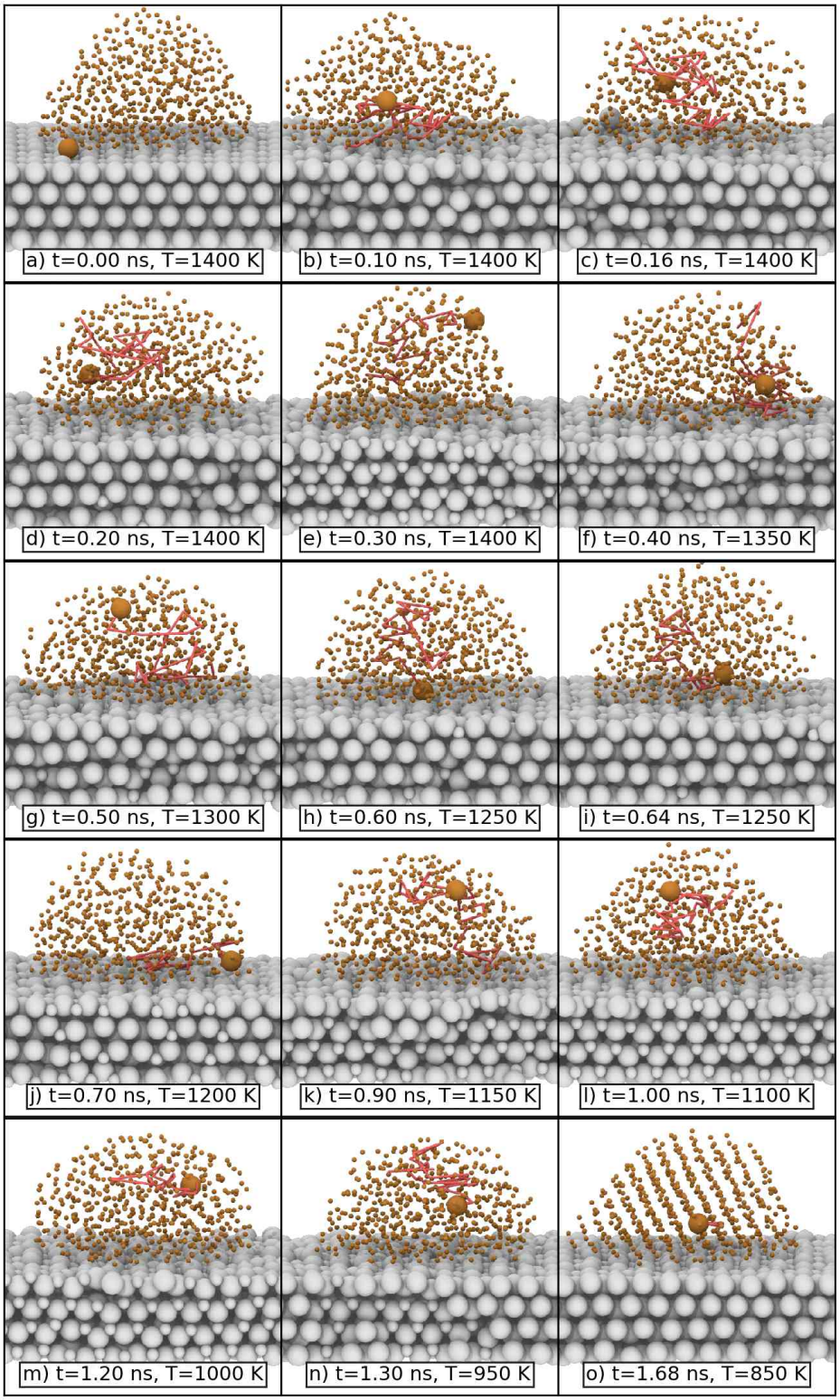

Figure 2.6: As in fig. 2.5, but viewed from the side. Here the height changes of the atom within the cluster can be appreciated, as well as the regular arrangement of Cu atoms in o).

2.5 Coincidence Lattice Match

2.5.1 The Coincidence Lattice Match Algorithm

A problem arises when attempting to investigate the Cu and ZnO interface. Due to the values of their equilibrium lattice constants and lattice geometry, as with many other pairs of materials, any common interface between them presents a mismatch. That is, it is not possible to easily find a single, periodic structure that can encompass both materials at the same time. This becomes a problem for extended surfaces, and is an issue that not only the system has to solve physically in the real world, but also an impediment to carrying out calculations in a computer within the constraints of a simulation box. As an algorithm for finding possible coincident surfaces we have adopted the coincidence lattice match (CLM) algorithm as described in ref. 161. Here follows a brief explanation of the steps involved, and the limitations of the algorithm. As a helpful guide to understanding the procedure, we also present a detailed worked example of matching two surfaces in the appendix (A.2). In this work, the CLM is utilized to generate matching Cu and ZnO surfaces, and study the effect of lattice mismatch on the different Cu low Miller index surfaces.

The CLM algorithm attempts to find a common supercell to the two-dimensional surface lattices that generate the interface of a pair of slabs. It additionally takes into account possible rotations between the lattices by including a rotation matrix. If both lattices are described by 2×2 matrices (the extension to 3×3, if required, is trivial, but in this case all the third vectors of the lattices are perpendicular to the X-Y plane) containing the lattice vectors as columns, any point in space belonging to the lattice is given by the equation

$$P = \mathbf{A}\vec{m}, \tag{2.27}$$

where P is a point in space that belongs to the periodic lattice, that is, it can form a vertex of a parallelogram corresponding to a supercell of the original lattice; $\mathbf{A}$ contains the lattice vectors as columns (the procedure can also be carried out with row vectors by transposing where appropriate, but vectors as columns is the conventional approach); and $\vec{m}$ is a column vector containing a pair of integer numbers. In effect, P is a linear combination of the lattice vectors contained in $\mathbf{A}$.

This equation is still valid if the orientation of the lattice changes by being multiplied with the usual rotation matrix,

$$\mathbf{A}_{\text{rot}} = \mathbf{M}(\theta)\mathbf{A}, \tag{2.28}$$

$$\mathbf{M}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{2.29}$$

where θ is an anti-clockwise rotation angle of the lattice vectors in the X-Y plane.

To generate a pair of coincident lattices, eq. 2.27 needs to be valid for at least three points simultaneously for both lattices. These three points are enough to define a parallelogram, by describing an origin and the two end points of two vectors composing the sides of the lattice parallelogram. Since we can arbitrarily and without loss of generality match both lattices at the Cartesian origin (the trivial solution to this equation with all the integers in $\vec{m}$ equal to 0), we need two further coincident points, which will form the generating vectors of a superlattice parallelogram. This results in the following equation, where the rotation matrix has been added to cover any possible rotations between the two generating lattices

$$P(\mathbf{A}) = P(\mathbf{B}), \tag{2.30}$$

$$\mathbf{M}(\theta)\mathbf{A}\vec{m} = \mathbf{B}\vec{n}, \tag{2.31}$$

where $\mathbf{B}$ and $\vec{n}$ are the lattice matrix and integer vector corresponding to the second material. The rotation angle θ and the matrices describing each lattice are already known or provided, and the challenge is to find integer solutions for the vectors $\vec{m},\vec{n}$ that make the equation valid. Non-integer solutions correspond in principle to strained (see sec. 2.5.2) lattices. This type of linear equation requiring integer solutions is known as a linear Diophantine equation [161, 162], and despite its apparent simplicity the requirement for the solutions to be integer values makes their behavior quite complex.

As mentioned, each integer solution of this equation corresponds to *one* side of a coincident superlattice parallelogram. To obtain the desired matching superlattice, *two* such solutions are required. Additionally, many solutions need to be filtered out by testing for linear dependency between the generated vectors, and for equivalent lattices (for more details on this, see the solved example in the appendix).

Three problems with this approach appear. The first is that it is not possible to directly solve this equation just for integer numbers, although it is of course exactly and easily solvable for real based results. We need to try all possible combinations of 4 integers, positive, negative and zero, with higher integers resulting in larger matching supercells. This combinatory approach to finding solutions explodes quickly, so we limit our analysis to integers with absolute value up to 10.

Related to this, the second problem is that we need to try every possible rotation matrix (although this number can be reduced taking into account the rotational symmetries present in the slabs). We limit our analysis to rotations spaced 1 degrees from one another. This ensures we cover a reasonable configuration space without making such small rotations that we in effect always find the same matches. If the angle change is too small, sometimes the same match is repeatedly generated.

The third and final problem is that a match with arbitrary proposed integer solutions will, for a majority of cases, not be exact (the left and hand sides of eq. 2.31 will not be the same). The paired surfaces, except for cases where one lattice is an exact integer multiple of the other, will always be slightly mismatched. This means that one of the two materials needs to be deformed to adapt to the final configuration (or both materials need to be deformed a certain amount). Thus the matches need to be evaluated for quality (minimum of deformation/best matching) by a given parameter. Ideally this parameter should be quick to evaluate (that is, it should not depend on any energy evaluations), descriptive, and be based on the resulting match geometry.

Two such parameters can be defined. The first is the quality of the solution, as defined in the original CLM implementation with

$$\delta_{\mathrm{QOS}} = \mathbf{M}(\theta)\mathbf{A}\vec{m} - \mathbf{B}\vec{n}, \tag{2.32}$$

which in effect asks how close are the proposed integer numbers to generating an exact solution to 2.31, in which case δ_{QOS} is zero. The farther away this is from zero, the more the materials will have to deform when actually matching them.

This δ_{QOS} parameter is a good measure for single solutions, but it obviates the extra deformation that happens when combining two solutions into a superlattice. For this reason in this work we also concurrently utilize the degree of lattice distortion [163–165], as obtained from the eigenvalues $\varepsilon_{1,2}$ of the strain tensor (see sec. 2.5.2)

$$\varepsilon_{\mathrm{DOLD}} = \frac{\sqrt{(\varepsilon_1^2 + \varepsilon_2^2)}}{2}. \tag{2.33}$$

Notice that if the strain is to be calculated for a 3D case, this needs to be divided by 3 instead of 2. In effect, $\varepsilon_{\mathrm{DOLD}}$ measures an average deformation for the generated superlattice. This quantity is related to the second invariant of the strain tensor, and is one half of the spontaneous strain [165]. Alternative measures are for example the von Mises strain/yield criterion, which is also related to the second invariant of the strain tensor. Any particular global measure can be used, as long as it is clear which one in particular is being applied.

The original CLM paper also utilizes strain at some point, but due to the geometry of their lattices (only combining hexagonal lattices with the exact same intra-lattice angle but different lattice constants), only uniaxial strain was considered. The definition of degree of lattice distortion utilized here also takes into account any possible shear generated by the deformation, since $\varepsilon_{1,2}$ depend on γ_{xy}, the shear component of the strain tensor.

As a result from this algorithm, for every starting pair of lattices, hundreds to thousands of matching candidates can be obtained. To be able to analyze this in an *ab-initio* approach in a reasonable amount of time, the candidates can be first filtered out by the mentioned strain (assuming that larger strains/deformations correspond to larger stresses/forces and thus more unstable configurations) or number of atoms in the supercell (since larger systems will be more expensive to calculate).

In summary, the CLM procedure is as follows:

1. Determine:
   a) Materials involved
   b) Miller index cuts of those materials, and also which particular geometry of the lattice cell to take. For example, for Cu(111) we can use the usual hexagonal cell or the orthogonal rectangular supercell. The latter will lead to a subset of matches, but might be more convenient for some purposes.
   c) Angle between the materials
   d) Maximum integer allowed in the match search
   e) Number of layers for each material
2. Find all matches up to the selected integer, discard those that are linearly dependent (see appendix)
3. Pair the matches together, discard those new lattices that are supercells of others (see appendix)
4. Filter and discard matches by:
   a) Strain: with for example a global measure such as $\varepsilon_{\mathrm{DOLD}}$
   b) Number of atoms: If there is a limit for the simulation method of choice, can also be a minimum limit to avoid highly strained structures
   c) Aspect ratio of the generated cell: The algorithm sometimes generates structures with extreme, needle-like aspect ratios, which can be inconvenient for simulations or lead to numerical stability problems (due to the majority of the atoms lying too close to the edge of the periodic boundary). These structures can be filtered out, or supercells constructed to make the cell more orthogonal, at the cost of increased number of atoms.
5. Fill in the matches with atoms, distort one (or both) materials
6. Simulate and analyze

A result of this procedure is that larger interfaces tend to show lower values of strain. Infinite surfaces of course posses a geometrical strain of zero, since they do not have to share a common boundary. Large enough supercells can also exhibit a strain of zero, since it is always possible to eventually find a common multiple for rational numbers. For example, if we try to match two square cells, one with sides of 2.55 Å and the other 3.2 Å, we need to find an integer expression of the ratio 3.2/2.55, which is 64/51. This means that the lattices match if we repeat the smaller one 64 times, and the other one 51 times. But, this supercell might be in the order of thousands of repeats (64^2 = 4096 in our example), and thus thousands of atoms per layer per material, impractical for any simulation purposes (hard even for classical force fields).

Lower strain usually implies a lower stress and thus a more stable configuration. The problem then is that larger cells are hard to fit into electronic structure calculations, with cells easily reaching hundreds or thousands of atoms due to the number of layers required to reach any well converged results. As is well known, *ab-initio* methods scale very poorly with the number of atoms in the system. In contrast, the NNP scales linearly since each extra atom means an extra individual atomic neural network is added to the calculation, so larger cells are a manageable problem.

A limitation from this approach is that it is purely geometrical. Only the lattice vectors of the 2D cells forming the interface are relevant. If one were to follow this line of thought, the best matching surfaces would be those which generate the least geometric strain and thus least stress. But this approach obviates many structural and atomistic details that could also compensate the generated stress. In particular, the energy loss induced by straining one of the materials might be compensated by a better interaction between both of them.

The generated matched cells have a number of degrees of freedom still available to them that are important to consider, and are not covered by the CLM procedure which is purely geometrical. The first degree of freedom is the distance between the interfaces. This is simple to find (by relaxing the two slabs that constitute the interface or scanning across multiple distances), and expected to be similar for different configurations. Thus it is the simplest characteristic to simulate and analyze.

A second relevant degree of freedom is the translation of the atomic lattices *within* the lattice vectors, with respect to each material. To a given lattice match correspond an infinite number of atomic dispositions between the materials (although many of them related by symmetry, of course). This can lead to different stacking of atoms at the interface level and thus potentially very different energy levels for different matching surfaces. This requires checking all possible translations, as restricted by the symmetry of the system. This is easiest done with a grid pattern search, but the number of points to test can grow rapidly. According to experience, after relaxation, many close points in the grid relax to the same configuration, so dense grids are not required.

The third degree of freedom is the possibility of surface reconstructions as the two systems come into contact and relax. The strain from being compressed/expanded in the periodic direction can be relieved by plane shifting in the direction perpendicular to the surface. Additionally, surfaces might relax to better accommodate the atomic level surface roughness of each other (the surfaces are not perfectly flat). Some surfaces might interconvert while remaining atomically flat. These kinds of reconstructions are easy enough to simulate with geometry relaxations, as long as the initial configuration corresponds to some sort of saddle point in the potential energy surface of the system, which will then find a path and relax to another configuration. It is not guaranteed that such a path will exist, and if a more complex reconstruction study is desired, global optimization techniques need to be utilized.

Finally, gross reconstructions can take place in the interface region, with atoms migrating across the interfacial plane to relieve stress and achieve a better mixing (if such interaction is favorable). This would require some sort of simulation that can optimize the atomic structure at the interface and jump over the large energy barriers present for atoms to come out of their ideal lattice positions, such as simulated annealing [152] or a genetic algorithm [140], which is beyond the scope of this work, but could be easily achieved with a NNP.

Analyzing these degrees of freedom is where the NNP capabilities are useful, since hundreds of single point evaluations and relaxation steps are required for each matching candidate. This would become expensive to perform with an *ab-initio* approach, but the myriad of very similar configurations that appear in such an analysis are ideal for a NNP.

2.5.2 Strain Theory

Strain theory attempts to codify the behavior of deformed materials. Strain theory is quite complex, combining notions of tensors, derivatives, material properties and stress. Here a short introduction into the topic is presented, and a more detailed explanation is provided in the appendix (see sec. A.3).

Our requirements from strain theory concentrate on obtaining the strain for a deformed lattice starting from the matrices containing the initial and final lattice vectors, which is not the usual approach to strain. Previous attempts at this exist [166, 167] for cells described in the crystallographic convention of three angles and three vectors, but here we want to extend this to cells described by arbitrary vectors. Additionally, some tools already exist that provide strain calculation and operation facilities. The Bilbao Crystallographic Server [165] provides the STRAIN tool, but this seems to be available only online (with no stand alone program), and relies on the crystallographic convention. The Pymatgen [168, 169] library for Python allows for straining already known cells, but not for the reverse of calculating strain from a pair of related cells. Despite their missing features, these tools can be and have been used for testing and validating our implementation of strain calculations.

To begin this analysis, the deformation tensor $\mathbf{e}$ is defined connecting the initial and final configurations

$$\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{e}\mathbf{A} \tag{2.34}$$
$$= (\mathbf{I} + \mathbf{e})\mathbf{A}, \tag{2.35}$$

where $\mathbf{A}$ is a matrix containing the vectors of our lattice cells as columns, $\widetilde{\mathbf{A}}$ is the same but for the deformed lattice, and $\mathbf{I}$ is the identity matrix. This definition proposes that the deformed configuration is equal to the original configuration plus some disturbance, the deformation tensor. Alternative definitions are possible, depending on whether one post or pre-multiplies with $\mathbf{e}$, or whether the undeformed or deformed configurations are taken as starting points, but all the definitions tend to be equivalent for small strains, and any of them can be used as long as it is clear which particular one is the starting point. From this equation we can extract a definition of $\mathbf{e}$ from known quantities as

$$\mathbf{e} = \widetilde{\mathbf{A}}\mathbf{A}^{-1} - \mathbf{I}. \tag{2.36}$$

From this deformation tensor, a symmetric tensor corresponding to the strain tensor can be constructed. There are actually several definitions for the strain tensor, each one convenient for different applications. The two easiest to work with, and the ones we will make use of, are the infinitesimal strain tensor ε (also known as the linear, Cauchy, or small strain tensor) and the finite strain tensor η (also known as the Green-Lagrange strain tensor), as given by

$$\varepsilon = 0.5(\mathbf{e} + \mathbf{e}^T), \tag{2.37}$$
$$\eta = 0.5(\mathbf{e} + \mathbf{e}^T + \mathbf{e}^T\mathbf{e}), \tag{2.38}$$

where $\mathbf{e}^T$ indicates the transpose of the deformation tensor.

The disadvantage of the infinitesimal strain tensor is that it becomes affected by rigid body rotations between the starting and final configurations. This is fixed in the finite strain tensor [166, 167], at the cost of this tensor no longer being linear in the generated strain (second order terms appear on the diagonal and off-diagonal). For more information, please see the appendix.

The degree of rigid body rotation between starting and ending configurations can also be estimated from a skew-symmetric tensor ω with

$$\omega = 0.5(\mathbf{e} - \mathbf{e}^T) = \begin{bmatrix} 0.0 & \omega \\ -\omega & 0.0 \end{bmatrix}, \tag{2.39}$$

which is related to the rotated angle. Notice that $\mathbf{e} = \varepsilon + \omega$.

The components of the strain tensor (either infinitesimal or finite) are

$$\varepsilon \text{ or } \eta = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} \\ \gamma_{yx} & \varepsilon_{yy} \end{bmatrix}, \tag{2.40}$$

where ε_{xx} and ε_{yy} are the relative increase or decrease in distance for a pair of points initially lying on the x- or y-axis, identified with the uniaxial strain; and $\gamma_{xy} = \gamma_{yx} = \gamma$ is related to the angle change of an originally orthogonal pair of lines and is identified with the shear strain. Importantly, these are all unitless quantities, and as such do not depend on the units or the size of the lattice cells involved, they are relative changes. An ε_{xx} of ± 0.01 for example indicates that distances in the x-direction have been increased or decreased by 1%.

The eigenvalues of the strain tensor allow us to express it as a diagonalized matrix. As such, the eigenvalues can be interpreted as the uniaxial strain experienced in a coordinate system where no shear strain is present (since the off diagonal terms in this basis are zero). This coordinate system corresponds to the eigenvectors of the tensor. These are sometimes also known as principal strain directions. The eigenvalues for a 2×2 matrix can be easily calculated from its components, and for the strain tensor (either infinitesimal or finite) are

$$\varepsilon_{1,\,2} = \varepsilon_{\text{max, min}} = \frac{\varepsilon_{xx} + \varepsilon_{yy}}{2} \pm \sqrt{\left(\frac{\varepsilon_{xx} - \varepsilon_{yy}}{2}\right)^2 + \gamma_{xy}^2}, \tag{2.41}$$

Notice that the eigenvalues combine information from both uniaxial strain directions and the shear strain. As such they are a useful quantity for extracting an "average deformation" from the strain tensor, as in the previously defined $\varepsilon_{\text{DOLD}}$ (eq. 2.33).

A problem is that the strain tensor as defined actually measures strain with respect to the *x- and y-axis*. This causes unexpected behavior (see appendix) if our lattice vectors are not aligned with the Cartesian coordinate axis. To solve this, we can project into our lattice vectors material basis [170] utilizing a cosine matrix

$$\varepsilon' \text{ or } \eta' = \mathbf{C}^T(\varepsilon \text{ or } \eta)\mathbf{C}, \tag{2.42}$$

$$\mathbf{C} = \begin{bmatrix} \cos\alpha & \cos\beta \\ \sin\alpha & \sin\beta \end{bmatrix} = \begin{bmatrix} \hat{a_1}\cdot\hat{x} & \hat{a_2}\cdot\hat{x} \\ \hat{a_1}\cdot\hat{y} & \hat{a_2}\cdot\hat{y} \end{bmatrix}, \tag{2.43}$$

where ε', η' are the versions of the strain tensors in material coordinates, $\mathbf{C}$ is the so-called cosine matrix, measuring the cosine of the angle between our lattice vectors and the Cartesian axis, α and β are the angles between the lattice versors $(\hat{a_1}, \hat{a_2})$ and the x-axis versor $\hat{x}$. In effect, this measures the dot product projection between the Cartesian unit versors and the unit versors parallel to our lattice vectors. In our case, this misalignment occurs when rotating one of the materials to match the other through the use of $\mathbf{M}(\theta)$. If the misalignment is not removed, the interpretation of the components of the strain tensor is no longer straightforward, but the eigenvalues and other related derived properties of the tensor should remain invariant.

This coordinate conversion also gets rid of the problem of the rotation between our initial configuration and the Cartesian axis, which is **not** solved in the finite strain tensor. Another possible solution to this is to align every lattice the same way. This is done implicitly in those derivations that begin from a conventional crystallographic cell definition, which is always aligned (see appendix).

In this work it has been decided to make use of the finite version of the strain tensor, and to report all strains after projecting into material coordinates as in eq. 2.43. If the strain tensor and values are required in Cartesian coordinates, they can easily be recovered from the data.

2.6 Structural Methods and Tools

2.6.1 The Cut Cube Method

Extracting periodic substructures from another structure is not a trivial task, considering that the original structure might include distortions due to for example thermal movements, or the presence of ad-atoms or vacancies. To cut a sub-slab form a large slab configuration, the key realization is that cuts with the same size as the original lattice vectors that generated the slab, or a supercell of these, will still maintain the periodicity of the slab at its side, no matter where the cut is applied, due to the translational periodicity of the solid. If the cut is larger than the cutoff radius of the NNP, the atomic environment around a target atom can be preserved.

Due to the mentioned possible deviations from the perfect crystal structure, the algorithm needs to take into account the following points: i) the stoichiometry of the cell needs to be preserved, for example, for a pure ZnO slab the ratio of Zn and O needs to be 1:1; ii) distances at the PBC of the new configuration need to be carefully checked, so that they are not too large or too small compared to the original structure, which would indicate a bad configuration that does not correctly tile periodically.

Fulfilling these two requirements is not always possible. For this purpose, the algorithm can center the cut on different atoms of the structure, for example in the case of supported clusters, different atoms of the cluster. The cut attempts are repeated until one is successful. To help this, the lattice vectors are slightly modified, up to 3% of their original lengths. With all of this, it is possible to obtain periodic substructures with a minimum of mismatch across the PBC, and that are small enough to be calculated with the electronic structure code of choice. The atomic environment is in this way slightly modified due to removing atoms beyond the cutoff and the possible modifications to the lattice vectors, but the resulting environment should still be close enough to the configuration that it samples from. The whole algorithm is exemplified in fig. 2.7 for a small copper cluster supported on zinc oxide.

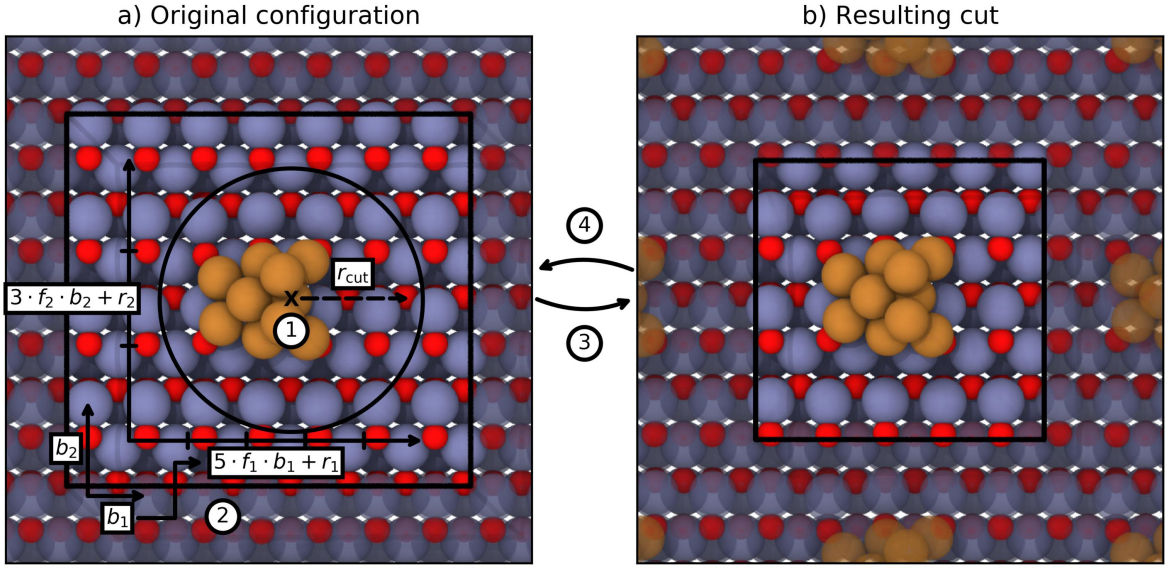

Figure 2.7: Periodic cut algorithm. a) Initial configuration. b) Resulting structure. 1) The atom closest to (or farthest from) the center of geometry (COG) of the cluster is chosen. 2) A superlattice of the original lattice vectors b_i is created that completely contains the cutoff radius (r_{cut}) around the chosen atom. Ticks on the superlattice show multiples of the original lattice vectors. 3) The created structure is checked for consistent stoichiometry, and reasonable interatomic distances at the periodic boundary. 4) If rejected, modify lattice vectors slightly by a factor f_i ($f_1 = 0.97, f_2 = 1.03$) and try again. If the factor is already too large/small, pick the next atom according to the distance from the COG and repeat the procedure.

2.6.2 Cumulative Distance Metric

At two points in this work, first for the comparison of small clusters obtained by GA optimization, and second for the comparison of the initial and final structure of relaxed CLM matches, it is necessary to measure the structural distance between an initial and final configuration. This measure needs to be objective, quantitative and significant (larger value means a larger distance between configurations), while also being fast and simple to calculate. For this purpose, the cumulative difference distance, as defined in 141 and implemented in the ASE [171] Python library has been chosen. This distance measure makes use of internal atom to atom distances as

$$\delta_{\mathrm{CD}}(1,2) = \sum_{\alpha=\mathrm{O,Cu,Zn}}^{\text{elements}} \frac{\sum_{i,j} |D_i^{\alpha}(1) - D_i^{\alpha}(2)|}{\sum_i D_i^{\alpha}(1)} \frac{N_{\alpha}}{N_{\text{atoms}}}, \tag{2.44}$$

where $\delta_{\mathrm{CD}}(1,2)$ is the cumulative distance (CD) factor between configurations 1 and 2, the first sum is over the elements present in the system, N_{α} is the number of atoms of a given element α, N_{atoms} is the total number of atoms in the system, $D^{\alpha}(1,2)$ is a sorted list of interatomic distances for atoms of element α in configurations 1 or 2, and D_i is the distance at position i of that sorted list. Thus the numerator of the first term calculates the difference between the sorted lists of interatomic distances, and the denominator sums over all the interatomic distances of structure 1. Notice that thanks to the factors in the denominators, the value is effectively normalized, so δ_{CD} can also be compared across different system sizes. The interatomic distances can be calculated taking the minimum image convention [156] into account or not, as needed.

The parameter δ_{CD} is simple to interpret, easy and fast to calculate, and gives a good enough measure of structural change in a given relaxation. As implemented it unfortunately only takes into account distances for atoms of the same element, so care needs to be taken and the algorithm may need to be modified if elements end up mixing between parts in the system (for example, a brass cluster on ZnO).

2.6.3 Lindemann Parameter

The Lindemann parameter is a structural factor often utilized to estimate the melting point of clusters and nanoparticles [172]. Since melting is a first order transition, discontinuities in the potential energy and specific heat would be expected at the transition point. This fact is exploited in experiments to perform differential scanning calorimetry [173–176] to measure melting points. However, in simulations, due to the small size of finite systems such as clusters, this discontinuity in potential energy is hard to observe and is usually covered by the normal potential energy fluctuations in the NVT canonical ensemble. One could make the system larger, thus making thermodynamic fluctuations smaller, but this is of course not useful if the goal is to estimate the melting point of clusters of a specific size.

The Lindemann parameter is purely structural, and is defined as

$$q_i(T) = \frac{1}{N-1} \sum_{j \neq i}^{N} \frac{\sqrt{< r_{ij}^2 >_T - < r_{ij} >_T^2}}{< r_{ij} >_T} = \frac{1}{N-1} \Big(\frac{\sigma_T^{r_{i1}}}{< r_{i1} >_T} + \frac{\sigma_T^{r_{i2}}}{< r_{i2} >_T} + ... \Big), \tag{2.45}$$

$$< q >_N (T) = \frac{\sum_i^N q_i}{N}, \tag{2.46}$$

where $q_i(T)$ is the Lindemann parameter for the ith-atom in the system at a given temperature T, N is the number of atoms under consideration, r_{ij} is the distance between atoms i and j, $< r_{ij} >_T$ is the average of this value across a simulation at a temperature T, and the denominator is the standard deviation of r_{ij}, ($\sigma_T^{r_{ij}}$). Notice that the division is by $N-1$ since the sum does not consider the case where $j = i$, so there are only $N-1$ sum terms. Since this value is only for one atom, we define $< q >_N (T)$ as the system average Lindemann value at a given temperature, which is just an average of the Lindemann value of all the atoms in the system.

Here we see the advantage of performing a SA run with defined temperature steps instead of a ramp, since the Lindemann averages need to be performed at a specific (average) temperature, which would not be well defined in a ramp.

The Lindemann parameter for a single atom can be interpreted as the average relative fluctuation of the interatomic distances between the given atom and all other atoms in the simulation. If the system is in a solid state, atoms will be stuck vibrating in their lattice positions, and interatomic distances will fluctuate little. When the system melts, atoms are no longer bound to a point in space, and interatomic distances fluctuate much more. The parameter generally increases with temperature, as atom mobility always increases with increasing kinetic energy. However, right at the melting point the system experiences a sudden increase of the parameter.

The behavior of the parameter can be modeled and fitted to a sigmoid or logistic function plus a linear term as in

$$< q > (T) = H + \frac{L}{1.0 + 1.0 \cdot e^{(-k \cdot (T - T_{\mathrm{m}}))}} + l \cdot T, \tag{2.47}$$

where H defines the lower asymptote of the sigmoid (when $T << T_{\mathrm{m}}$), $L + H$ is the upper asymptote (when $T >> T_{\mathrm{m}}$), k defines the curvature/steepness of the curve, T_m is the turning point of the sigmoid, and l defines the linear term due to the normal increase in atom mobility as temperature rises. It is important to understand the meaning of each fit parameter, since this is a non-linear fitting procedure which is highly sensitive to the initial guesses of each value. In particular, L, H, and T_m need to be guessed with at least the correct order of magnitude or the fit result will not be good, but this can be easily done by inspecting the $< q >$ vs. T plot. The melting temperature for the cluster can be extracted from the fitted T_{m} parameter.

2.6.4 Polyhedral Template Matching

Polyhedral template matching [177] (PHTM) is a member of a family of algorithms [178, 179] that attempt to automatically classify crystal structures into one of the main known crystal patterns. This is a non-trivial task, since the algorithms have to take into account possible deviations from the perfect crystal structure, such as thermal distortions, grain boundaries, different orientations, etc.

The core of the PHTM relies on matching the set of points corresponding to the nearest neighbors of an atom, with the set of points corresponding to the neighbors of an atom in a given lattice crystal (face-centered cubic, hexagonal close packed, etc.). This match is achieved by minimizing the distance between the points, taking into consideration translations between the sets of points (which is simple by just operating on the center of geometry of the set of points), all possible rotations around a central point, as well as scalings (for example, due to thermal distortions or different lattice constants). The best matching pattern is the one with the smallest possible root mean square deviation (RMSD) between the set of points, considering all possible rotations and scalings. This is expressed in the equation

$$\mathrm{RMSD}(v, w) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} |s(v_i - \bar{v}) - Q(v_i - \bar{v})|}, \tag{2.48}$$

where N is the number of points being tested, v is the set of points being tested, w is the reference set of points, $\bar{v}$ and $\bar{w}$ is the center of geometry of both sets, s is a scaling factor of the test points, and Q is a rotation matrix.

The main problem with this equation is that the number of possible matches to try grows rapidly with the number of points. In PHTM, this is solved by utilizing a convex hull (generating a polyhedron that contains all the points in the set), and graph theory to reduce the number of matches that need to be tested. The results is a fast identification algorithm, that is also robust to deviations from the perfect match. The algorithm is conveniently implemented in the OVITO [180] visualization software.

Chapter 3

Computational Details

3.1 Density Functional Theory

3.1.1 Settings

Density functional theory [96, 97] (DFT) is the method of choice for sampling the PES of the Cu-ZnO system, as implemented in the Vienna Ab-initio Simulation Package (VASP) [181, 182] version 5.4.4. With this program it is possible to perform periodic, planewave-based calculations with PAW (Projector Augmented Wave) [183, 184] to treat the core electrons. This algorithm is required to efficiently simulate the rapidly changing orbital probability distributions of the core electrons, which would be computationally expensive to describe with a planewave basis set. The related pseudopotentials for the different elements were selected according to the VASP manual recommendations. The generalized gradient approximation as proposed by Perdew, Burke, and Ernzerhof (PBE) [185] has been chosen as the exchange and correlation functional.

Convergence tests have been run for the number of k-points and the maximum energy of the planewaves basis functions, to achieve an energy and force convergence below the usual NNP RMSE of approx. 1 meV/atom for total energies and 100 meV/Bohr for forces. Testing with a 3.61 Å cubic unit cell of fcc Cu containing 4 atoms yields a planewave cutoff of 500 eV with a k-point grid of $12 \times 12 \times 12$. For larger cells, the number of k-points has been scaled to maintain at least the same k-point density and thus the same level of convergence. For finite systems such as slabs and isolated clusters, at least 13 Å total vacuum has been used in the cell directions that are meant to be non-periodic. This means that for example, for a slab, 6.5 Å of vacuum are allowed above and below the slab. Tests have shown that this amount of vacuum is enough to minimize the interaction between periodic images, while at the same time it does not excessively increase the computational cost of the simulations. The Fermi level has been treated with a Gaussian smearing method, with a σ factor of 0.1 eV. An example input file for VASP with a complete list of the settings is provided in appendix C.

According to initial tests, adding dispersion corrections to the generated DFT data results in overbinding, with lattice constants being smaller than in the experiment. In particular, the charge density dependent Tkatchenko-Scheffler [186] method as implemented in the VASP program was utilized. This is a known property of the PBE functional [187–190]. If required, data with dispersion corrections is also available, and it has been shown in the past that the NNP is capable of reproducing dispersion corrected results [90, 136, 191] in spite of the locality of its atomic environment descriptors. Since predicting the bulk properties of Cu and ZnO is a minimum requirement for the NNP, all simulations in this work have been performed with uncorrected potentials.

3.1.2 k-point Grids for Non-orthogonal Cells

A problem arises when attempting to calculate the slabs generated with the CLM algorithm utilizing periodic DFT. Since these slabs are not just simple supercells of the original slabs, the usual method of scaling down the number of k-points as a cell is replicated in that direction is no longer applicable. To solve this, one can either utilize a larger than required k-point grid (leading to much slower calculations, particularly for large slabs such as those generated with the CLM algorithm in sec. 7); or adopt one

of the automatic k-point grid generation algorithms available. The latter approach has been chosen, utilizing the algorithm of generalized regular k-point grids as described in refs. 192, 193 and 194, which also conveniently provides a server (and a stand-alone version) to calculate a k-point grid for arbitrary slab structures. The key parameter in this algorithm is r_{min}, which sets the density of points in reciprocal space. Larger r_{min} corresponds to a denser grid, and thus also to a slower calculation. This parameter has been tested against calculations for more "normal" shaped slabs, and a value of 55 Å results in a similar convergence level as obtained with the initial $12 \times 12 \times 12$ point grid.

3.2 Construction of the Neural Network Potential

The NNP utilized in this work was trained on a reference dataset containing a total of 73,136 structures covering about 5 million atomic environments. Therefore, in total 73,136 energies and about 15 million force components are available for training the NNP. As explained in sec. 3.3, this library contains multiple configurations of copper, zinc oxide, and ternary systems.

The NNP is constructed with the in-house program RuNNer [115, 116], which is made available under GPL3 license. The hyperbolic tangent activation function has been used for all layers, except the output layer which consists of a linear function. Each element in the system presents the same architecture and symmetry functions, which simplifies the structure of the NNP. Other cases where atoms exhibit very different typical distances, such as water, might require different per-element symmetry functions. The parameters of the symmetry functions have been derived from previous work on this system [93], and are presented in appendix D.

The root mean squared errors between the predicted and reference energies and forces are presented in table 3.1 for 4 different NN architectures. Architecture number 1 exhibits the smallest errors, with a 2.5 meV/atom RMSE for the training energies and 59.7 meV/Bohr for the training forces, with very similar values for the testing data not included in the fitting process indicating the absence of significant overfitting. This architecture and fit was thus selected to perform the simulations presented in this work. Figure E.1 in the appendix shows a comparison between reference and predicted energies and force components for this NNP architecture, in which the degree of accuracy of the selected NNP can be appreciated.

Architecture	Training Set RMSE		Test Set RMSE	
	E (meV/atom)	F (meV/Bohr)	E (meV/atom)	F (meV/Bohr)
15-15	**2.5**	**59.7**	**3.1**	**59.6**
20-20	2.7	70.5	3.5	69.4
15-15-15	7.4	254.3	9.5	257.3
20-20-20	2.6	73.5	3.4	74.4

Table 3.1: NNPs obtained for different architectures (neurons per hidden layer are given) and root mean square errors (RMSE) for the predicted energies E and forces F. In bold, the chosen architecture for the simulations in this work is highlighted.

3.3 Generation and Composition of the Reference Dataset

3.3.1 Introduction

A reliable NNP reference structure set needs to cover all the sections of configuration space that are relevant to the simulation method of choice, while at the same time, due to the cost of electronic structure calculations, remaining as small as possible.

Tuning the sampling of the PES to the simulation algorithm of choice is thus vital. For example, Monte Carlo simulations will more often visit high energy configurations due to the behavior of the usual trial moves, when compared to a Molecular Dynamics simulations that follow the trajectory of the atoms step by step and can avoid high energy states. If possible simulation and sampling should be integrated together [88]. A good sampling strategy will cover a wide section of the PES, without sampling randomly generated, irrelevant, or high-energy configurations. Structures should also contain a restricted number of atoms so that they can fit into the required electronic structure calculations, about 200-300 maximum atoms for the current DFT setup. Additionally, recalculating already well-sampled environments should be avoided.

All in all, a sampling strategy needs to take into account three main points:

1. A way to generate structures: Structure generation should be as automated as possible due to the sheer amount of sampling points required, but at the same time being careful not to generate "bad", nonphysical or high-energy configurations.

2. A way to extract smaller configurations from larger structures: Due to the locality of the NNP and the proven principle of environment decomposition [61, 137], it is possible to perform simulations for larger systems and then extract smaller structures for electronic structure programs to process. This is easy for some systems and some *ab-initio* codes, where a sphere of atoms around a central atom can be extracted, thus preserving the atomic environment. From test simulations this is not the case with supported clusters on ZnO and with the VASP code: spherical cuts around a target atom results in layers of dangling Zn and O atoms in the support, which results in calculations that have trouble converging the self-consistent iteration steps, or the force on the central atom never converging because it is affected by unbalanced Zn/O atoms further away than the NNP cutoff radius. For this purpose, the cube cut algorithm (see sec. 2.6.1) has been developed.

3. A way to identify repeated configurations: The utilized structure generation methods result in the same configurations being visited repeatedly, particularly genetic algorithm simulations. To avoid repeatedly recalculating already well-sampled configurations, some sort of filtering needs to be applied. One possibility is the NNP ensemble approach [88, 115], where new structures are tested with already fitted potentials, expecting the predicted energies and forces to differ for those structures in poorly sampled configuration space. Another possibility, not requiring pre-fitted potentials, is presented in sec. 4.

With these requirements in mind, a sampling procedure in three phases has been adopted throughout this work. In the first phase, structures are generated systematically by modifying known configurations such as bulk structures. In a second phase, more configurations are generated by utilizing basic simulations methods such as MC or MD. The final phase depends on the specific final purpose of the NNP, and is tailored to obtaining desired results. This last phase is different for example when sampling GA structures for global optimization of small clusters, when compared to structures for large supported clusters obtained from simulated annealing.

3.3.2 Phase I: Systematic Modification of Known Structures

For phase I, well-known and -determined configurations are generated and then systematically modified. For the case of Cu (and brass [61]) and ZnO, this consists of generating bulk structures, low Miller index surfaces, and Wulff [195] shaped nanoparticles. All of these are well characterized from experiments, or are simple to generate by following geometric and energetic rules from other known configurations.

These base structures can then be modified systematically: experimental lattice constants are contracted and expanded to sample a range of possible crystal lattices, atomic positions shifted randomly

to emulate thermal distortions, different number of layers are utilized for slabs, and so on. Once these structures are calculated with the reference method of choice and a NNP trained, this results in a potential that has sampled and can accurately predict the behavior of a system around its equilibrium position for a variety of specific configurations. At this point the NNP can reproduce or predict simple structural and energetical parameters [93], such as lattice constants, bulk moduli, and surface energies.

3.3.3 Phase II: Simple Simulations

In phase II, the systems are evolved beyond these simple equilibrium conditions by making use of simple simulation procedures, such as molecular dynamics in the NVT and NPT ensembles, utilizing the NNP generated in phase I. The systems can also be molten and cooled to generate high temperature configurations, liquid and amorphous structures, all of which would not be simple to systematically generate following the procedure of phase I. Structures are sampled at regular intervals, utilizing techniques such an ensemble of NNPs [88, 115] or the bin and hash algorithm (sec. 4) to scan for new, unsampled configurations. After phase II, the NNP is usually able to perform long NVT and NPT MD simulations without visiting unsampled regions of the PES. For many projects this is enough, but in this work further simulations targeting specific simulation methods have been necessary, as detailed in the following sections. Other simulation methods that allow the NNP to explore reasonable sections of configuration space are also useful, such as Monte Carlo atom swaps [61] that exchange elements, simulated annealing [152], and semi-grand canonical ensemble simulations [196].

3.3.4 Phase IIIa: Genetic Algorithm Optimization of Small Clusters

This section is adapted from Reference [138] M. L. Paleico and J. Behler, "Global optimization of copper clusters at the ZnO($10\bar{1}0$) surface using a DFT-based neural network potential and genetic algorithms," J. Chem. Phys. **153**, 054704 (2020), with the permission of AIP Publishing.

For the genetic algorithm optimization of small clusters, high-energy configurations are also required, since these often appear when crossing over candidates or performing mutations on known candidates. As such, phase III for these simulations consists of two parts.

First, random supported clusters are generated and processed with DFT, taking care to not generate configurations where atomic distances are below 20% of the usual nearest neighbor distance. Below this threshold, problems can arise in DFT calculations and NNP fitting due to huge repulsive forces.

From this a provisional NNP can be obtained with which trial GA runs can be performed. In an iterative approach, as detailed in fig. 3.1, GA searches are performed for different cluster sizes. Minimized and unrelaxed structures generated by crossovers or mutations are extracted, filtered with a NNP ensemble or the bin and hash algorithm, calculated with VASP, introduced into the reference dataset for a new NNP fit, and the process repeats until no more new low-energy configurations can be detected. With this procedure, hundreds of new configurations per GA run can be easily extracted.

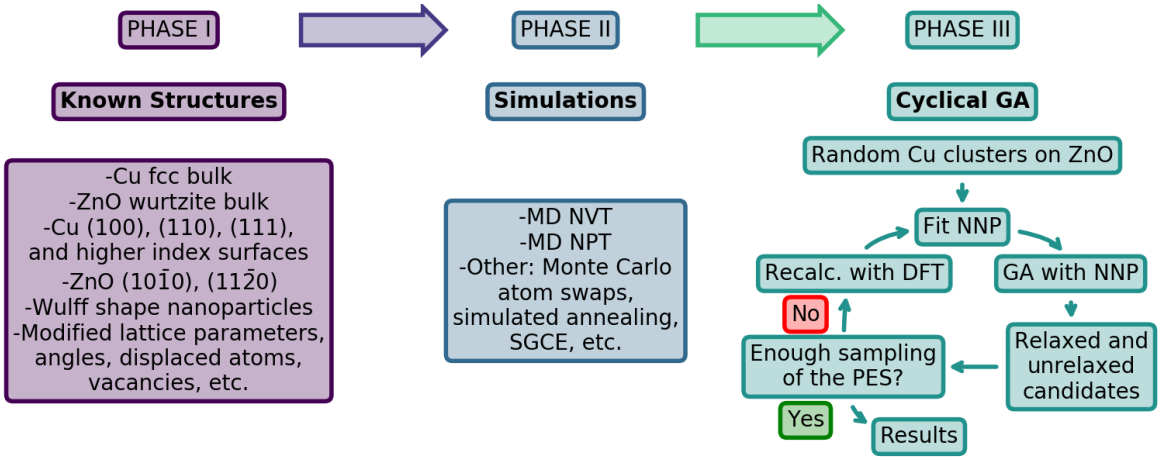

Figure 3.1: Phases followed in the construction of the NNP dataset for the GA optimization of small supported Cu clusters, as described in more detail in the main text. Phase I corresponds to generating known structures with small modifications, in phase II we perform simulations with these structures with the usual simulation methods such as MD and MC, and in phase III we proceed to perform a cyclical GA search for new structures. Reproduced from Reference [138], with the permission of AIP Publishing.

To avoid the supported clusters interacting with themselves through the PBC, which results in wires or sheets instead of distinct clusters, large ZnO slab surfaces need to be utilized. To fit these structures into DFT, they are extracted utilizing the cut cube algorithm (see sec. 2.6.1). This preserves the immediate environment of the clusters, as well as the periodicity of the ZnO slab.

3.3.5 Phase IIIb: Simulated Annealing of Large Supported Clusters

The initial NNP trained on distorted base structures is utilized to run SA simulations. Structures are extracted from these large SA configurations with the cube cut algorithm (see sec. 2.6.1). Since it is not always possible to extract periodically and stoichiometrically correct cubes, the algorithm was applied to every frame of the simulation twice: once starting from the atoms farthest away from the center of geometry of the cluster, and once from the closest atoms. If the algorithm cannot find a safe cube, it

moves on to the next atom in the list, until a good cut is found or the structure is exhausted. This results in a collection of extracted structures that sample either the core or the edges of the supported clusters, following the spirit of the environment decomposition process (see sec. 2.2.4). Figure 3.2 shows a possible environment decomposition approach for this system, comprised of structures obtained as described here, and information from structures from other simulations presented in this work. To avoid recalculating every extracted configuration, they are pre-filtered by the NNP ensemble procedure [88, 115], and/or by utilizing the BAH algorithm (see sec. 4) to rapidly evaluate their possible presence in the available dataset. The structures that pass this filtering are recalculated with the electronic structure method of choice, a new NNP is fitted with the expanded dataset, and the process can then start anew. This procedure was repeated a total of three times.

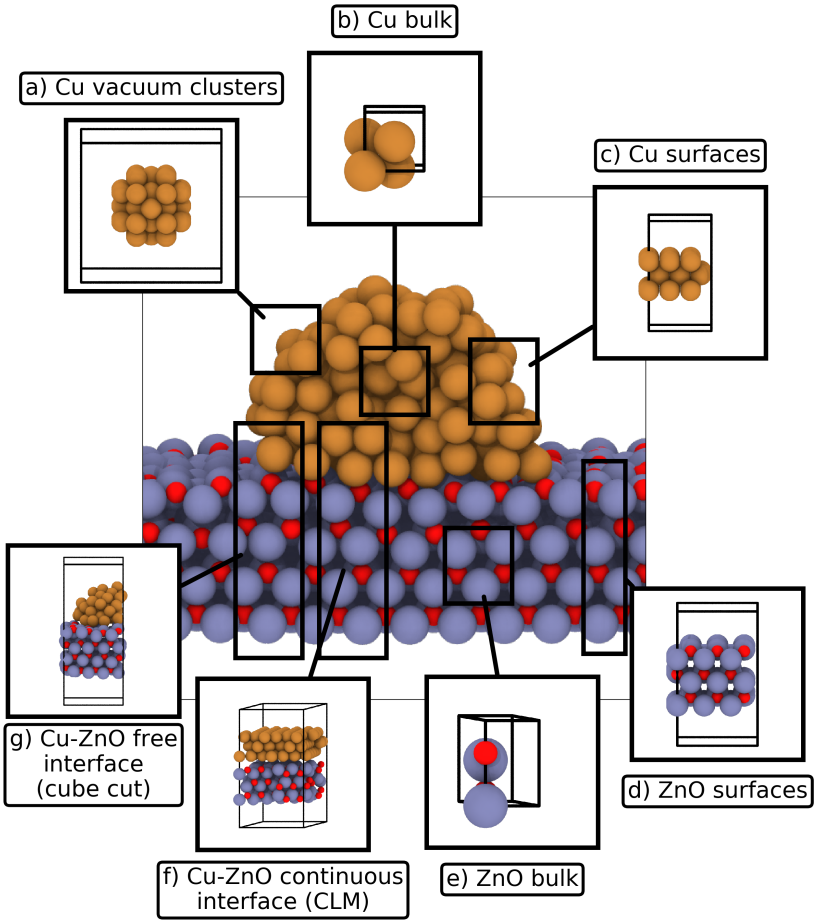

Figure 3.2: Possible environment decomposition for large copper clusters on zinc oxide: Central figure: Target system. a) Information for cluster atoms far away from the surface can be obtained from simulations of free-standing clusters [61]. b) Information for atoms deep inside the cluster can be obtained from simulations of bulk copper. c) Information for atoms on surface facets can be obtained by simulating copper surfaces. d) Information for the ZnO support far away from the cluster can be obtained from simulations of clean ZnO slabs. e) Information for deep layers of the ZnO support can be obtained from simulations of bulk ZnO. f) Information for atoms right at the interface between the two materials can be obtained from CLM structures, and also from structures obtained with the cut cube algorithm. g) Information from atoms at the edge of the cluster near the support can be obtained from the cube cut algorithm.

3.3.6 Phase IIIc: Large Coincident Surfaces

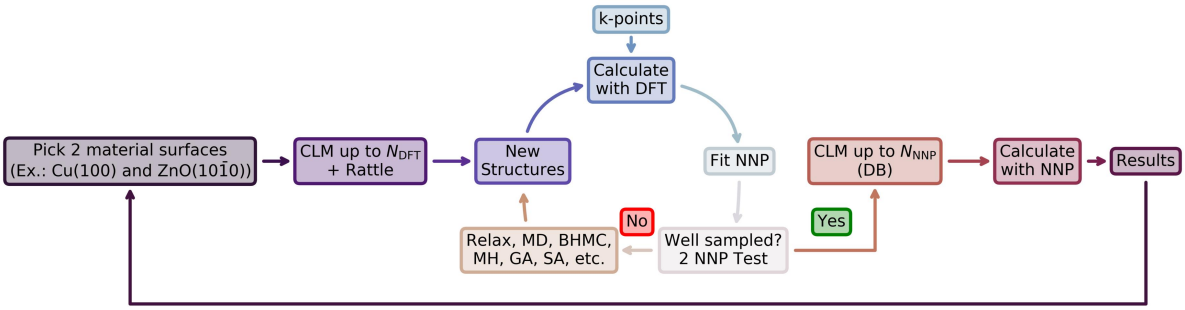

Figure 3.3: Proposed CLM and NNP fitting scheme for the search of coincident surfaces: After a pair of materials is chosen, structures up to a given number of atoms (N_{DFT}) are calculated with the electronic structure method of choice. A NNP is fitted, and the fit checked for predictive capabilities by comparing results from different fits. If the fit is deemed not sufficient, extra structures are generated with good PES exploration methods and recalculated until the fit is satisfactory. Once the fit is good enough, it is used to predict the behavior of larger matches (N_{NNP}).

For coincident surfaces, a number of structures have been generated and calculated as detailed in fig. 3.3. Two materials and specific Miller indices are chosen (in this case, ZnO(10$\bar{1}$0) with any of the low index Cu surfaces). Matches are generated with a thickness of 2 to 6 layers per material and a total maximum of $N_{\mathrm{DFT}} = 250$ atoms (which as shown in sec. 7.2.1, fairly restricts the number of possible structures). Additionally for some structures the atoms are rattled to generate thermally distorted structures. These structures are calculated with DFT (taking into account k-point generation as described in sec. 3.1.2), and afterwards a number of NNPs are fitted to this data. A NNP ensemble test [88, 115] is performed on larger structures, and if the fit is found to be satisfactory, larger matches are generated and analyzed with the NNP. If not, further structures can be generated with a variety of other simulation algorithms that can produce new "reasonable" looking structures and are applicable to slabs: relaxing the available slabs, performing molecular dynamics (MD), basin hopping Monte Carlo [149] (BHMC), simulated annealing [152] (SA), minima hopping [153] (MH), and evolutionary or genetic algorithms [140, 197] (GA).

3.4 Genetic Algorithm Search Settings

This section is adapted from Reference [138] M. L. Paleico and J. Behler, "Global optimization of copper clusters at the ZnO($10\bar{1}0$) surface using a DFT-based neural network potential and genetic algorithms," J. Chem. Phys. **153**, 054704 (2020), with the permission of AIP Publishing.

The supporting slab for the global optimization of small copper clusters consists of a ZnO($10\bar{1}0$) surface with a size of 23.03 $\times$ 21.19 Å^2, or a 6 $\times$ 4 supercell, which is large enough that cluster atoms in this size range do not interact across the PBC. The slab is generated with enough layers to have a minimum thickness of 7 Å, so atoms at the top of the slab cannot detect the other side as this is beyond the NNP cutoff radius. In total the slab has 448 atoms. At least 13Å of vacuum are added in the direction perpendicular to the slab, also taking into account the top of the deposited cluster, once again to avoid self-interactions. The slab is pre-optimized at the beginning of the GA search, and then is optimized again with each generated cluster. In many DFT-based GA simulations the supporting slab is left frozen because of the added computational costs, but as explained further elsewhere [138], this can lead to different minimized structures.

For geometry relaxations the LBGFS algorithm (limited memory BFGS [198]) is used, with a maximum force criterion of 0.005 eV/Å and a limit of 1000 evaluation steps, as implemented in the ASE library [171]. This optimizer was chosen mainly for being the fastest for our particular setup and being rather robust. Care should be taken that all structures are completely relaxed, otherwise false local minima can find their way into the candidate pool, that then relax into other different local minima if reoptimized.

The GA runs are seeded by generating 15 structures with random geometries while maintaining a minimum distance between cluster atoms, that are then geometrically relaxed. These structures are rapidly abandoned by the search due to their relatively high energy.

The active breeding population is kept at 50 structures. Using an interatomic distance comparator [141] (see also sec. 2.6.2), the breeding population is kept structurally distinct. Three criteria are required for the comparator: first the energy of the structures is compared, taking into account a threshold for minimum total energy difference of 0.1 eV. If the energy difference between two structures is larger than this value, they are assumed to be different. A maximum correlation distance of 0.7 Å, and a maximum cumulative difference of 0.025 Å are also utilized for the comparison of interatomic distances. The larger these two values are, the less "strict" the comparator is. If the comparator is too strict, small differences between optimized clusters are deemed significant and the breeding population might become poisoned with the same structural motif. If too lax, structures that are evidently different are falsely detected as similar. The values mentioned here were suitable to keeping the breeding population varied.

Each search for each cluster size is repeated 4 times, until 3000 structures have been evaluated in each run, for a total of 12.000 structures for each size. Clusters at each cluster size are then merged into a single dataset before continuing with the analysis performed in Sec. 5.2.1. The same structures are usually found in each run (see fig. 3.4 a), described below), but the independent runs are kept as a consistency check, and we can expect this situation to be different for larger clusters where finding a given putative global minimum (GM) becomes more difficult [199].

The GA search has been implemented taking as a base the routines already present in the ASE (Atomic Simulation Environment) Python library [171], version 3.17.0. Some mutations needed to be modified (rattle and mirror), or implemented from scratch (the rest). The mutation moves employed have a number of parameters that can be adjusted. These parameters are detailed in ref. 138. Energy and force evaluations were provided by the n2p2 neural network library [200] in combination with LAMMPS [201].

The typical course of a GA search is illustrated in figure 3.4 a), for four GA searches for the GM of Cu_{10}. The putative GM is quickly found in fewer than 300 attempts for all four independent searches,

with no lower energy structure found in the subsequent 2700 steps of the search. Figure 3.4 b) shows how the different mutations (or no mutation, just crossover between two parent structures) affect the energy of the generated relaxed candidates, for two GA searches for Cu_{10}. The energy is expressed relative to the parents' average energy, as given by

$$P = 100 * \left(\frac{E_{\text{candidate}}}{(E_{\text{parent 1}} + E_{\text{parent 2}})/2} - 1 \right), \tag{3.1}$$

where $E_{\text{candidate}}, E_{\text{parent 1}}$ and $E_{\text{parent 2}}$ are the energies of the relaxed candidate and its two parents, respectively. If the relaxed candidate has a lower energy than the average of its parents, P is negative, and we can say that the mutation was "successful" in progressing the GA search.

Performing only pairing shows no particular energy improvement or worsening, centering the energy changes around the 0% mark. In contrast, angular mutation moves tend to generate clusters with high energies, probably due to moving copper atoms from their favorable positions interacting with the support to on top of the cluster. Short molecular dynamic runs seem efficient at generating lower energy configurations, being the only mutation with an average energy change below 0%. The other mutations exhibit a similar behavior between them, sometimes improving the energy of the offspring cluster, but with a long tail towards higher energies.

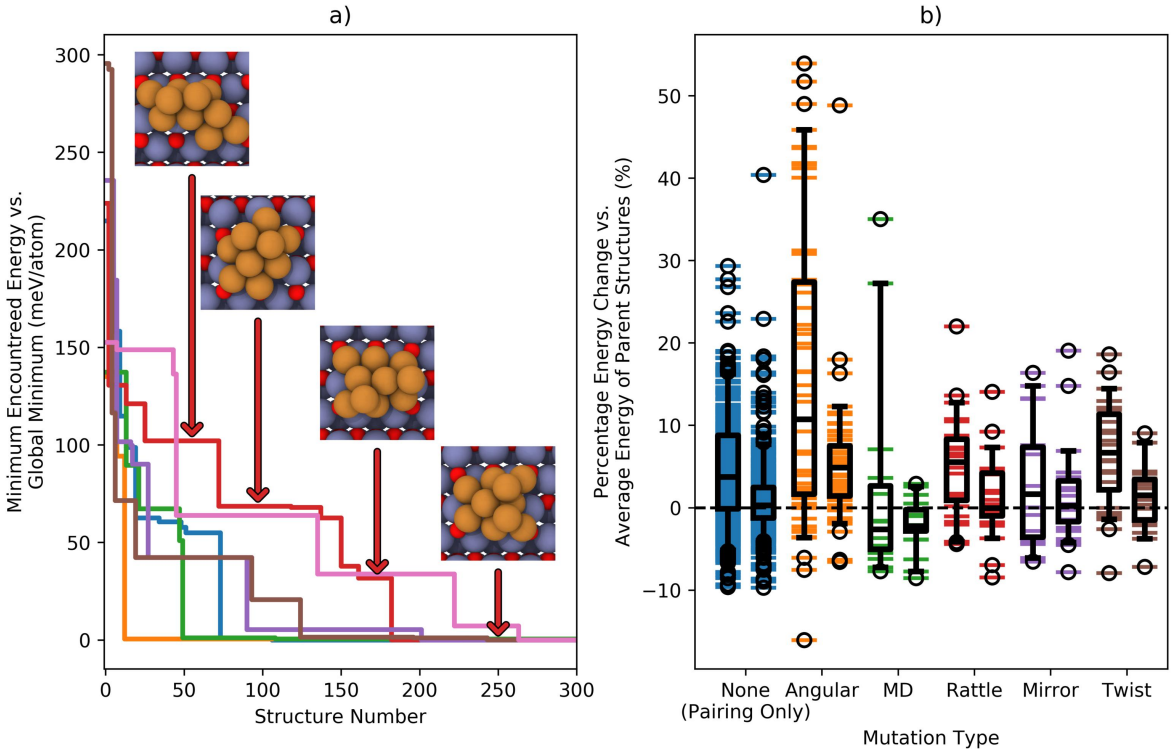

Figure 3.4: a) Energy progression of seven GA searches for the Cu_{10} cluster. Lines indicate the energy of the lowest energy configuration found in a given GA search until that point, with respect to the putative GM. All runs find the minimum in less than 300 structures, and no lower energy configuration is observed for the following 2700 structures for a total of 3000 structures in each independent run. Arrows and inserts show selected cluster structures for the red run. b) Percentage energy changes compared to the average energy of the parent structures (see Eq. 3.1), for different mutations for two GA searches on the Cu_{10}. "None" means that only a cut and splice pairing has been performed, with no added mutation. "MD" is the molecular dynamics move described in sec. 2.3. The lines inside the boxplots shows the median energy change, the boxplots extend from the lower to the upper quartile of the data, whiskers extend from the 5th to the 95th percentile of the data, circles show data points outside of the whisker range. Reproduced from Reference [138], with the permission of AIP Publishing.

3.5 Simulated Annealing Settings

Copper atoms are initially deposited on a simple regular cubic grid on top of the ZnO support. For simplicity and to avoid biasing the initial configurations, an fcc grid was not chosen. The size of this cube is pre-calculated taking into account copper's density (4 atoms in a 3.6 Å side cubic cell). The ZnO slab consists of 4 layers (enough so that taking into account the NNP's cutoff radius, atoms at the top of the slab do not see the vacuum at the bottom of the slab), and sized to be as big as the initial copper cube, plus at least an extra 10 Å. This extra lateral size adds more atoms to the simulation and results in a slower computation, but it is vital to avoid the cluster merging with itself through the periodic boundary conditions of the simulation.

The ZnO slab is relaxed before depositing the copper atoms, and the atoms in the slab are held in place by adding a soft spring potential binding them to their relaxed positions. Although the melting point of ZnO is much higher than that of copper (experimental values: 2248 K vs. 1358 K [202]), the surface of the slab can melt at a lower temperature, which risks atoms from the support incorporating into the cluster. The goal of the simulations is to study simple supported clusters. Dissolution of the support into the cluster is also of interest as mentioned in the introduction, but would be a much more complex topic.

The molecular dynamics part of the simulation was performed with a Nosé-Hoover [203, 204] thermostat, and a timestep of 0.004 ps. The clusters are initially randomized at 1400 K, above the copper bulk melting temperature. Configurations are sampled starting 37.500 steps (0.15 ns) every 62.500 steps (0.25 ns). Five configurations are extracted in this way and then cooled down. The cool down process begins with another small randomization phase of 0.3 ns at 1400 K, and then the clusters are cooled down to 300 K in 50 K steps of 0.1 ns, for an average cooling rate of approx. 500 K/ns and a total simulation time of 2.6 ns. Slower cooling rates were tested, but the structures of the obtained clusters do not seem to change significantly with more cooling time below this value.

3.6 Coincidence Lattice Match

3.6.1 Settings and Implementation

As mentioned in sec. 2.5.1, some parameters need to be defined for a successful CLM run. Here all the relevant parameters are summarized. The angle between the two materials has been swept by 1 degree at a time, from 0 degrees up to when the results start repeating due to the geometry of the surfaces involved (see sec. 7.2.1). The maximum allowed solution tolerance has not been used, and instead $\varepsilon_{\mathrm{DOLD}}$ is the deciding criterion for accepting matches, with a maximum allowed value of 0.1. Each slab has been generated with 6 layers per material, since this ensures that the atoms at the bottom of the material do not feel the effects of the interface, due to the cutoff of the NNP (slab thickness of at least 6 Å). The thickness of each material is different, particularly for the different Cu surfaces, so the required amount of layers needs to be checked for each case. The maximum number of atoms for DFT calculations has been limited to 250, and for NNP calculations to 2000 (in principle a larger limit is possible for the NNP, but this number already allows for thousands of valid matches for each material pair).

The CLM algorithm has been implemented in Python version 3.5, with some helper functions from the Atomic Simulation Environment (ASE) [171] library, version 3.17. In particular, heavy use is made of the database functionality of the library, as well as the structure generation functions. These new functions will hopefully be made available in a public repository in the future.

3.6.2 Starting Surfaces

The resulting matches are strongly dependent on the structure of the initial materials taken into consideration. As such, in fig. 3.5, the structure of the 4 materials of interest is presented: ZnO($10\bar{1}0$), and the

three low index surfaces of Cu(100), (110), (111). The structures were obtained utilizing as a starting point the bulk structures of the respective materials. The bulk structures have been optimized with the NNP and DFT, obtaining comparable results in both cases (see table F.2). For Cu the 1 atom cell has been chosen as a starting point instead of the more usual 4 atom cubic orthogonal cell, since this is the smallest possible cell, thus maximizing the amount and type of matches obtained with CLM. A 4 Cu atom basis cell would seem in principle to be equivalent to the smaller basis, but in fact the range of possible geometric matches it can generate is constrained to a subset of the matches of the smaller cell. As a simple example, if we consider the Cu_1 cell as the base 1×1 cell and Cu_4 as 2×2, it is not possible to generate a 3×3 cell from the latter utilizing integer combinations of lattice vectors. A 6×6 cell is possible from both and in some ways equivalent to the desired 3×3 cell, but unnecessarily larger.

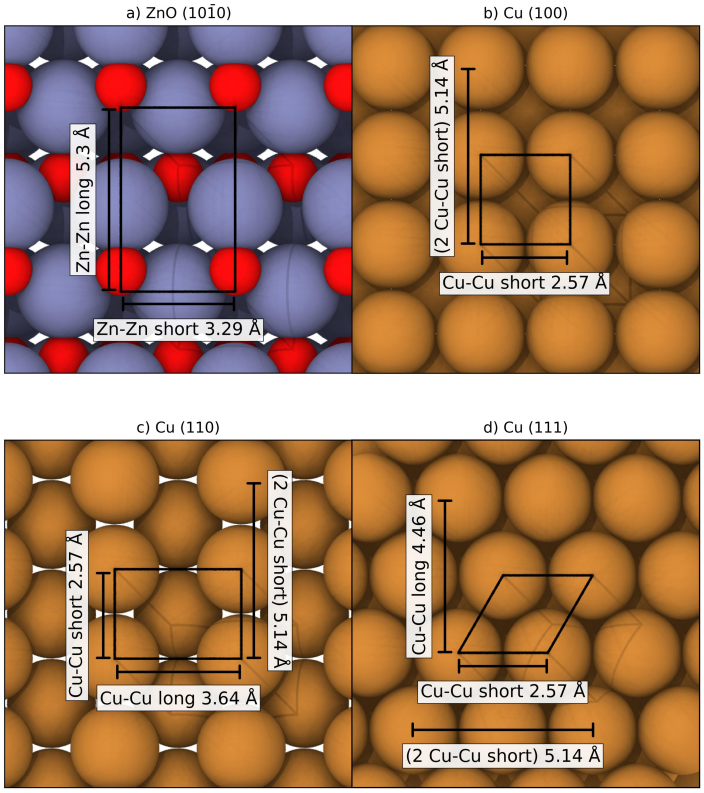

Figure 3.5: Lattice cells and relevant interatomic distances for the main surfaces utilized in the matching algorithm. a) ZnO(10$\bar{1}$0) b) Cu(100) c) Cu(110) d) Cu(111)

Attempting to create a match between these cells requires finding rectangles in the Cu configurations that can be matched to the rectangle in ZnO. As it can be seen from the figure, there is no way of creating this matching cell for the involved lattices without straining one (or both), and without creating a huge coincident cell where the numbers match. For Cu(100), a long Cu-Cu distance that is twice that of the unit cell almost matches the rectangle in the ZnO cell, but the remaining direction is too short to match with the corresponding vector in ZnO (2,57 Å in Cu vs. 3.29 Å in ZnO). In Cu(110), the opposite problem can be found, with a long Cu-Cu distance of 5.14 Å once again almost matching the 5.3 Å distance in ZnO, but in contrast to the previous case, the remaining direction is too long (3.64 Å in Cu vs. 3.29 Å in ZnO). The rectangular cell of Cu(111) is not so easy to detect at first glance, since the usual representation for this surface is a hexagonal configuration. The rectangular cell in this case is similar to that of Cu(110), but with a longer distance between the Cu atoms at the long edge of the rectangle. Due to this, this second distance is once again too long to match with the corresponding distance in ZnO.

The previous analysis shows that all the cells are close to matching, but one direction is always

too long or too short. The advantage of the CLM algorithm is that it performs this analysis, but for every possible supercell (up to a given maximum integer), which means we are no longer matching just rectangles but many possible parallelograms; and for every possible angle between the mentioned parallelograms, instead of only in the default orientation.

Part II

Results

Chapter 4

The Bin and Hash Algorithm

This chapter is adapted from Reference [205] M. L. Paleico and J. Behler, "A Bin and Hash Method for Analyzing Reference Data and Descriptors in Machine Learning Potentials", Mach. Lean.: Sci. Technol., accepted, (2021), with the permission of IOP Publishing.

4.1 Motivation

The quality of machine learning potentials is dependent on the availability of reference data obtained from electronic structure programs. These datasets can become quite large, with hundreds of thousands of single structures and millions of atomic environments contained within them. This is required in order to be certain that the structures in the dataset cover the necessary sections of configuration space. Handling such datasets can be problematic, both when they are being generated by a human, in which case it can be complicated to keep track of which structures are already present in the collection; or by an algorithmic automatic exploration of a PES [206–208], in which case repeated or bad configurations can find their way into the structure library due to lack of human supervision.

In this chapter the bin-and-hash (BAH) algorithm is presented. The algorithm is based on the widely utilized hash table [209] data structure. The goal of this algorithm is the rapid identification and comparison of large number of vectors containing atomic environment descriptions, such as the data present in the usual MLP datasets. Such an algorithm can be utilized to detect redundant information in the dataset, thus reducing the need for costly repeated electronic structure calculations or fitting onto redundant data. As will be shown, the algorithm can also be used as a rough qualitative measure of the description of a given dataset by the selected descriptors corresponding to a ML method. Finally, the algorithm can be utilized to identify unreliable derived values associated with the atomic fingerprint vectors (such as forces or charges).

Here the BAH algorithm is applied to the atom-centered symmetry functions (ACSF) [110] that form the basis of the Behler-Parrinello HDNNP approach as a descriptor of the local atomic environment, but can in general be applied to any method based on well-behaved, meaningful and ordered atomic descriptors [121–127], and it has recently been shown that many descriptors perform equally well [128]. The analysis allowed by the BAH method can be applied before a NNP has been trained, and even before reference *ab-initio* calculations have been performed, which results in a large time save when compared with other methods that require one or the other.

One example of previous methods that attempt to process structural datasets include SketchMap [210, 211], which is a complex dimensionality reduction algorithm that groups structures by similarity, from which structures can be discarded or added to the obtained clusters. Other algorithms attempt to find meaningful measures of distance in structural space [212, 213]. This distance can once again be used to prune large datasets if structures that are too similar are detected. Methods relying on ML descriptors to establish similarity measures are also available [214]

The mentioned methods are in the end versions of the well known problem of finding distances and nearest neighbors in multi-dimensional data. More basic approaches rely on complex binary tree data structures such as kDtrees [209, 215], that can efficiently store data points according to their mutual

distance in multi-dimensional space and rapidly reduce a search space due to their binary structure; and other dimensionality reduction algorithms such as principal component analysis (PCA) [216, 217] that reduce the size of the space under consideration. These algorithms are often too complex and slow when applied to the typical MLP reference dataset sizes and dimensionality.

Being the cornerstone of most MLP methods, automatic atomic fingerprint selection has also been an area of research interest. Hyperparameter optimization [218–220] can be used for simpler forms of machine learning, but they are usually too complex and slow for MLPs with large datasets, relying on multiple fitting rounds. The MLP community itself has developed some methods, for example relying on genetic algorithm based optimization searches [124, 221] to find the best descriptor configuration, or dimensionality reduction through importance estimation with CUR decomposition [222] to discard those descriptors less relevant to the overall training process.

Dataset maintenance has also been an area of focus. Active learning [88, 223–226] has been the tool of choice in this regard, which attempts to reduce the size of a dataset by the relative importance of each structure to the overall description of the PES.

This chapter is structured as follows. The algorithm is described in Sec. 4.2. The first step of the algorithm is binning the vector of ACSF corresponding to a given atomic environment. This is described in Sec. 4.2.2. After binning, a hash function (sec. 4.2.2) is relied upon to create a numerically unique representation of each environment, and by combining this with a hash table, searches for representations become fast and do not scale with the number of environments being studied. The procedure is contrasted with a naive direct comparison approach in Sec. 4.2.2, with big O notation [209] scaling discussed in Sec. 4.2.3.

In Sec. 4.3, results from the application of the algorithm are shown. Concrete timings are presented in Sec. 4.3.1, confirming the scaling expected based on theoretical considerations. Section 4.3.2 demonstrates how the BAH algorithm reproduces distances in ACSF vector space, while Sec. 4.3.3 shows the behavior of the algorithm when changing the number of binning subdivisions and the ACSF set description of the dataset, and how this can be utilized to qualitatively evaluate the quality of a given ACSF set, without requiring a previous potential fit. Finally, Sec. 4.3.6 shows how the method can be applied to find similar atomic environments and contradicting or unreliable information in a dataset.

4.2 The Bin and Hash Method

4.2.1 Description of the Algorithm

```
1  divs =  number of subdivisions in ACSF space
2  for atom_env_i in dataset
3      for acsf_j in acsf_set
4          calculate symmetry function vector Gi={Gj}
5  find Gjmax and Gjmin across each acsf component Gj
6  initialize empty hash table Ht
7  for each Gi vector
8      bin Gi vector Bi={Bj},
9          Bj=divs*(Gjmax-Gj)/(Gjmax-Gjmin)
10     calculate hash Hi=hash(Bi)
11     if Hi not in Ht
12         store it Ht[Hi]=j index
13     else
14         count as collision ncolls+=1
15         add to existing record in hash table
16         Ht[Hi] append(j index)
```

Code Block 4.1: Pseudocode for the bin and hash algorithm

The bin and hash algorithm is explained in pseudocode in code block 4.1. Here follows a brief

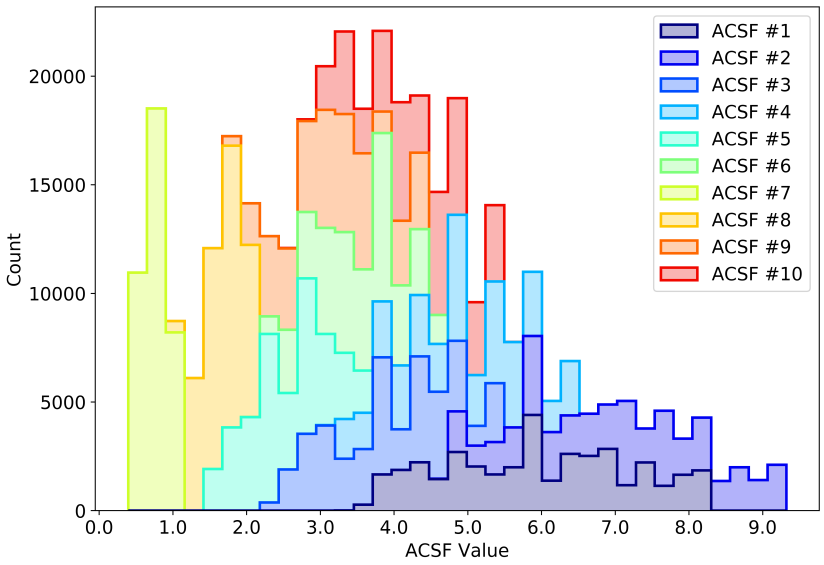

Figure 4.1: Stacked histogram plot of the values of the first 10 radial ACSFs in the ZnO dataset describing the atomic environments of the oxygen atoms. Reproduced from Reference [205], with the permission of IOP Publishing.

description of the procedure, and then sections detailing the different parts of the algorithm.

As a test dataset structures representing a zinc oxide slab have been chosen. A typical distribution of ACSFs for such a dataset is presented in Fig. 4.1 in the form of a stacked histogram plot, for the first 10 ACSFs of a small dataset containing 1192 configurations of a ZnO($10\bar{1}0$) surface slab with in total 75360 atomic environments. The structures included in the dataset consist of bulk cut slabs, relaxed slabs, and configurations extracted from MDs, with different number of layers. The same 58 atom-centered symmetry functions per elements utilized in the rest of this work (see appendix D) describe this dataset. Even for such a relatively small and simple dataset describing just one structural motif, the distribution of the ACSFs already has a rather complex form.

The individual steps of the BAH algorithm are illustrated in Fig. 4.2. Here the system is described by two arbitrary ACSFs, represented by the histograms in Step 1. The range of each ACSF is split into a predefined number of subdivisions, typically between 10^1 and 10^7 bins, taking into account the maximum and minimum values present in each range. The purpose of this is to transform the ACSF vector $\mathbf{G}_i$ for a given atomic environment i, from a float-based continuous representation to an integer-valued binned vector $\mathbf{B}_i$ of the same dimensionality (Step 2). This binned vector is then processed with a hash function, resulting in a one-dimensional hash key H_i (Step 3), which is then used for indexing into a hash table (H_t) (step 4).

The binning solves two problems at once: it does away with the need for a floating point representation, which does not allow for an accurate calculation of a hash, since the hash would be numerically unstable to the round-off errors of the floating point values; and it bins similar ACSF vectors (whose floating point representation is close) to the same $\mathbf{B}_i$ vector, ultimately resulting in the same hash key. Hashing this binned vector makes it possible to construct a hash table, where the advantage of such a data structure is that each vector carries in itself, after hashing, its position in the table. Binning and hashing are thus central for the performance and functioning of the method. In each bucket the ID of the corresponding environments is stored, which makes it easier to retrieve information on the results inside each bucket and to export the hash table for other uses.

$\mathbf{G}_i$ vectors that end up in the same hash table bucket, are said to be in a hash collision. These vectors are by necessity similar, and depending on the number of subdivisions, even exactly the same save for floating-point round off noise. The number of such collisions in general and for each bucket can be

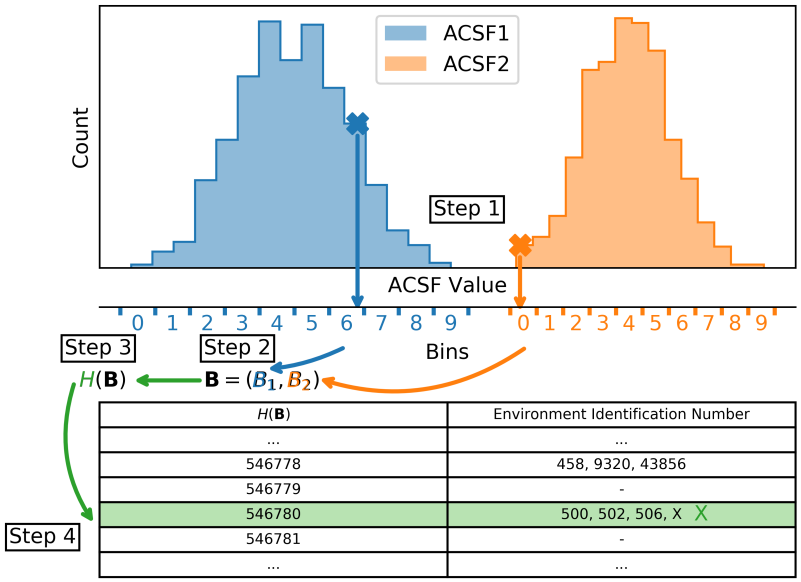

Figure 4.2: Illustration of the BAH approach: Each atomic environment in this example is characterized by a two-dimensional ACSF vector $\mathbf{G} = (G_1, G_2)$. In step 1, the histograms corresponding to the ACSFs are generated as a visualization aid. The values of G_1 and G_2 are highlighted by the crosses for one particular example environment. This ACSF vector is then binned to a pair of integer values, forming the binned vector $\mathbf{B} = (B_1, B_2)$ in step 2. In step 3 the hash $H(\mathbf{B})$ of this binned vector is calculated. Finally, in step 4 this hash is used (directly or indirectly) to index into the hash table, and add the atomic environment to a counter for similar environments. Reproduced from Reference [205], with the permission of IOP Publishing.

kept track of, and this is a measure of how many similar environments are present in the dataset.

The usefulness of different subdivision values will be shown later, but at this stage different subdivision attempts can be justified as the solution to a predictable problem. It is possible for one component of the ACSF vector for two environments to fall at either side of a subdivision boundary. If this is the case, the binned version of the ACSF vector will be off by one, resulting in different hash value and different position in the hash table. If the original environments where actually similar, this is undesirable behavior. Multiple binning solves this problem by moving the bin boundaries around. In this way, it can be avoided that very similar environments are converted to different hash keys. Despite the need for multiple bin subdivisions, the algorithm is so fast that it remains efficient.

4.2.2 Analysis of the Algorithm

Here the scaling of each section of the algorithm is discussed and estimated. Scaling is often ignored by the more sophisticated algorithms mentioned in the motivation section, resulting in algorithms that are in practice too slow given the typical size of ML reference sets. As reference algorithms, neighbor lists and naive approach brute force comparison are presented. Finally, binning and hashing operations are explained in more detail and the scaling in big O notation [209] is derived.

Cell-Based Neighbor Lists

A simple approach to efficiently calculating distances is commonly present in molecular dynamics, where cell lists [156] are implemented. Here they are useful because most force fields depend in some

way on the distance between atoms. The system is divided into smaller cubic cells, and atoms are assigned to these cells according to their coordinates. If the size of the cells is chosen properly with respect to the cutoff radius of the potential, finding neighbors becomes simple: only the current cell the atom occupies and the adjacent cells need to be checked.

How could one extend this procedure to atomic environment descriptors? Cells would be created not in coordinate space but in the higher-dimensional ACSF space. Unfortunately, this procedure would be unfeasible as the computational costs scale badly with dimensionality: in a one-dimensional system we need to check the central cell plus two neighbor cells, in two dimensions it is the central cell plus eight cells organized in a square, and so on with the total number of cells to be checked scaling as 3^D with D the dimensionality of the space. Since the typical ACSF set has at least 20 descriptors, the method is not useful at all.

The binning step in BAH is similar to this cell-based approach, and has a similar goal of reducing the degrees of freedom of the problem. In BAH however, we only look at neighbors within one cell, avoiding the scaling problem as dimensionality increases. Additionally, hash tables allow for performing this check within one cell very fast.

The Naive Approach

Naively, atomic environment vectors could be compared one by one against each other, taking into account the need to only compare atoms of the same element. This approach scales linearly with the number of descriptors in a given atomic environment, but quadratically with the total number of environments present in the dataset. To see why this is, consider that for environment number N, we need to compare it with the previous $N-1$ environments that have already been compared.

This poor scaling of lookup times is avoided in BAH by utilizing a hash table, where finding a member of the table is, in principle, a constant time operation [209] that does not scale with the amount of data already present in a hash table.

Binning

The BAH proper starts with binning of the available data. The range of each ACSF in the dataset is known beforehand, and the space between the maximum and minimum values of the ACSF is divided into bins whose borders are given by

$$B_j = \mathrm{nint}\left(\mathrm{divs} * \frac{\mathrm{ACSF}_{\mathrm{max}} - \mathrm{ACSF}_{\mathrm{val}}}{\mathrm{ACSF}_{\mathrm{max}} - \mathrm{ACSF}_{\mathrm{min}}} \right), \tag{4.1}$$

where B_j is the bin value for the j-th ACSF, nint is the nearest integer function, i.e., a round-off to the closest integer; and $\mathrm{ACSF}_{\mathrm{max}}$, $\mathrm{ACSF}_{\mathrm{min}}$, and $\mathrm{ACSF}_{\mathrm{val}}$ are the maximum, minimum, and current value of the ACSF under consideration, respectively. The number of subdivisions is kept the same for all ACSF, although some of them might have larger or smaller ranges. A possible modification would be to change the number of subdivisions to obtain a uniform density of values for each ACSF.

As mentioned previously, the binning serves various purposes. It converts floating point numbers, whose representation is imprecise due to limited prevision round-off errors during mathematical operations or due to limited precision when outputting data to a text format, into integers which are much more robust. Due to their imprecise nature, floats cannot (or rather, should not) be hashed directly, since small variations in the number will result in wildly different hashes.

Binning, in a way, also preserves a notion of distance between ACSF vectors. Although Euclidean distances in multi-dimensional space tend to lack significance [227], it can still be assumed that ACSF vectors that are close enough correspond to similar atomic environments. As the number of subdivisions increases, only more and more similar vectors bin to the same representation, increasing the sensitivity of the BAH process until only those vectors that are exactly the same (barring the limitations of floats mentioned in the previous paragraph) remain.

Binning is not enough for a full rapid identification of environments on its own, since binned vectors would still need to be compared one by one against each other. The solution to this come from the adoption of a hash table.

Hashing and Hash Tables

Hash functions [209] are a family of functions that can map data of arbitrary size to data of fixed size. They are one-way functions, that can assign a single numerical value to arbitrary data. Since the output space of a hash function is smaller than its input space, the assignment is not unique. Two different objects can result in the same hash value, an event known as a hash collision. Due to this domain reduction and the use of operations such as modulo and per-byte logic functions that result in loss of information, the hash function operation is not usually reversible other than by brute force. If well designed, the output space of a hash function is uniformly covered, and the behavior of the hash is not continuously smooth. That is, if a collision occurs, the hashed objects are either exactly the same, or completely different; small modifications to the input results in very different outputs. On their own, thus a hash value cannot be used to measure distance in input space. All of these properties make hash functions useful in a variety of fields, such as cryptography (e.g.: password storage) where the non-reversibility is taken advantage of, or in the realm of data validation (e.g.: checksums, credit card numbers, blockchain), where the discontinuity with respect to input data is exploited.

The properties of hash functions also give rise to the hash table data structure. A hash table behaves like an array, but the index of an object into an array is "contained" by the object itself, instead of being arbitrary or following some sort of numerical or lexicographical ordering. The index is "extracted" from the object by the hash function, and thus the position of an object in the hash table "buckets" is always known. The index is calculated as

$$\text{index} = \text{hash}\%\text{array\_size}, \tag{4.2}$$

where "index" is the index to be used when accessing the hash table array, "hash" is the hash function value of the object of interest, "array_size" is the size of the array holding the hash table, and % is the modulo operator. If constructed in this way, the hash will always index an array position, no matter the size of the array.

One question appears at this point: How is it possible to map the 10^7 integers possible for a single ACSF value into an array of finite space? Or even worse, the multitude of combinatorial arrays that arise when increasing the size of the ACSF vector. As mentioned before, hash functions already map larger domains into smaller ones, so collisions are bound to happen. Various implementation dependent solutions exist for solving this problem [209]. In separate chaining, all collided keys are kept in the same bucket in the form of a list. Attempting to assign to the hash table then consists of finding the corresponding bucket as in eq. 4.2, and afterwards looking through the list of collisions stored in the bucket for the exact match in a naive comparison approach. This of course incurs an overhead, but the collision list will usually be small enough that the hash table efficiency is maintained. In another implementation, known as open addressing, if the current bucket is already occupied, objects are assigned to the next free bucket. Assignment now becomes a fast search utilizing eq. 4.2, followed by testing each occupied bucket following it. This of course also has an extra computational overhead associated with it. Since MLP datasets usually correspond to well-structured data and not to completely random numbers, the worst case scenario of maximum collisions is avoided and the hash table can be predicted to exhibit optimal behavior.

As described, when collisions do not become a burden, search, assignment and insertion in a hash table are constant time operations that do not depend on the amount of data already stored in the table. Instead of requiring the algorithm to perform a naive one by one comparison until the target object is found as with an array, in a hash table the object already carries its index and can thus be immediately located.

This desirable behavior comes with some associated overhead: if the table is constructed in sequential memory, space will be wasted since buckets might be empty due to the spread output of hash indexes, hashes need to be pre-computed for each new operation on the table (but they are usually fast), and extra care needs to be taken to avoid hash collisions. This overhead is usually justified by the advantages the data structure provides, and as such it is often utilized for efficient storage and retrieval of data.

Hash tables can easily be stored in text form for post-processing or to compare future structures without the need for reprocessing the whole dataset. For BAH, this is implemented by outputting to a file every bucket, with its associated binned vector and the numerical ids of the atomic environments contained in the bucket. The whole table can be reconstructed by reading these binned vectors and ids and populating a new table with them.

4.2.3 Scaling

Algorithm	Scaling
Naive Comparison	$O(M*N^2)$
Binning	$O(M*N)$
Hashing	$O(M*N)$
Hash Table Lookup	$O(N)$

Table 4.1: Big O notation scaling of the different algorithms under consideration. N is the number of atoms corresponding to the number of atomic environments in the dataset. M is the number of functions in the atom-centered symmetry function vector.

Here the expected big O notation [209] scaling for the relevant algorithms is discussed. The expected formulas are summarized in Table 4.1. The operation performed here is searching once through a whole dataset, attempting to find repeated atomic environments.

In the following discussion, N is the number of environments in the dataset, which is also the total number of atoms across all structures. M is the number of functions in each ACSF vector, and also the dimensionality of our problem. The scaling with respect to N can be more important than regarding M, since the number of ACSF in a HDNNP is usually less than 100 per element for most systems, while the number of atomic environments can reach millions and has no upper bound.

The following scaling is predicted:

- Naive comparison and lookup: Comparison scales at worst as $O(M)$, since we need to compare each element in one ACSF vector to the corresponding element in another ACSF vector, but we might end early if a mismatch is detected. We then need to compare environment 1 with the next $N-1$ environments, environment 2 with the next $N-2$ environments and so on until environment $N-1$ for the last single comparison with environment N. This is a mathematical series that in the end scales as $O(N^2)$. Both parts of the algorithm together scale as $O(M*N^2)$.

- Binning: Binning scales with both the number of elements in each ACSF vector – since we need to bin each element individually – as $O(M)$. Additionally, it has to be done for each of the N atomic environments ($O(N)$). Combined it scales as $O(N*M)$. This operation is usually fast.

- Hashing: Hashing scales weakly with the size of the object being hashed ($O(M)$). There is some dependence on the specific implementation of the hash function (see Sec. 4.3.1) and the hashing needs to be repeated for each ACSF to be compared ($O(N)$). It is a relatively fast operation but highly implementation dependent since there are many possible hashing functions.

- Hash tables: Addition of data to a hash table and lookup are constant with respect to the size of the stored dataset (except for hash collisions) so this scales as $O(1)$ (no scaling/constant scaling). This is where the main time saving for the whole algorithm comes from. We have to repeat this N times, once per hashed array, resulting in a scaling of $O(N)$.

From these, we can calculate the total time for the described operation. For the naive, we need to compare $M*N^2$ elements to process the whole library of environments. In the case of BAH, first all environments in the dataset are binned, then hashed, and finally stored into a hash table, detecting repeated environments if present. These operations are independent and sequential, so their times are additive. Thus, the timings of each algorithm are

$$\begin{aligned} t_{\text{naive}} &= k_{\text{comp and lookup}} * O(M*N^2), \\ t_{\text{bah}} &= k_{\text{binning}} * O(M*N) + \\ &\quad + k_{\text{hashing}} * O(M*N) + k_{\text{hash lookup}} * O(N), \end{aligned} \tag{4.3}$$

where each k is the timing constant to perform that operation once, which depends on the actual implementation of each algorithm, the programming language of choice, and the CPU architecture. As expected the naive approach shows a polynomial $O(N^2)$ scaling, which is usually undesirable. In contrast, BAH consists of three linearly scaling sub algorithms. These scalings are tested in Section 4.3.1 for the case of random data (worst case scenario for all the algorithms), and the different scaling constants are estimated.

4.2.4 Implementation

The algorithm has been implemented in Python 3.5, using the dict [228] (short for dictionary) data structure, which is a hash table with the possibility to associate arbitrary data to each hash bucket. Hash tables are easy to implement in many languages, since all that is needed are pointers and allocatable arrays. The dict object in Python already hashes the data, so no explicit hash function has been defined or implemented nor is required.

The BAH algorithms can be ported to a parallel implementation for the processing of very large datasets, or in the case asynchronous operation is required, such as a compute cluster associated with a database. A master process can hold a central copy of the hash table at the core of the BAH method, and relegate binning and hashing to slave processes; or each process can hold its own copy of the hash table, update it as new chunks of data are read, and periodically combine the results into a master table.

4.3 Results

4.3.1 Performance and Timings

For illustrative purposes, we present the timings and scalings of the naive and BAH algorithms on randomly generated values, as obtained from Python3.5 on a Intel Core i5-5300U CPU 2.30GHz. Fig. 4.3 plots the behavior of the different algorithms for increasingly large amounts of said data.

As previosuly predicted, the naive algorithm scales as N^2 (fig. 4.3a), while BAH scales a N (fig. 4.3b). The logarithmic plot in fig. 4.3c combines the data of both previous subfigures, and here it can be seen that the naive algorithm rapidly becomes unfeasible while the BAH timing only increases at much larger data sizes. We define a speedup as the relative time gain for any algorithm in BAH when compared with the naive approach, as $t_{\text{algo}}/t_{\text{naive}}$. This is plotted in fig. 4.3d. As the size of the data increases, the speedup become larger, because the naive approach scales polynomially compared to the linear scaling of BAH. The larger the data becomes, the more advantage is gained from utilizing BAH over the naive approach.

Naive		
Constant	Value (s/op^2)	op^2/s
$k_{\text{comp and lookup}}$	8.8E-8	11.000.000
BAH		
Constant	Value (s/op)	op/s
k_{binning}	3.0E-6	336.000
k_{hashing}	1.8E-7	5.500.000
$k_{\text{hash lookup}}$	2.9E-7	3.400.000
$k_{\text{BAH global}}$	4.2E-6	238.000

Table 4.2: Estimated scaling constants for the different parts of the naive and BAH algorithms, at a constant $M = 10$ (scaling is assumed linear for other M values, in the cases where relevant). Units are in seconds required per operation (s/op). The inverse constant is also given providing the number of operations per second (op/s). Note that the naive algorithm only seems "faster" because it is expressed in terms of op^2.

The final two subfigures, e and f, show the behavior of the hash table itself. In this implementation, hashing scales linearly with the size of the ACSF vector under consideration, but is extremely fast for the size of vectors considered here. As expected, operations regarding the hash table, such as assignment and lookup, do not depend on data size.

From these plots the different scaling constants of eq. 4.3 can be calculated. The values are gathered in table 4.2, where the naive and BAH halves of the table are expressed with different units. The fastest part of the BAH algorithm is the hash calculation (k_{hashing}), while the bottleneck in the current implementation seems to be the binning (k_{binning}). This is probably due to the division and rounding nearest integer operations involved in binning, and it could probably be improved with some vectorization or better numerical libraries. Not considered here is the required I/O to read ACSF data from a file, which might become a more serious bottleneck for larger datasets, but is however common to both algorithms. The values obtained here represent only an approximate order of magnitude since this will change significantly for different implementations and computer architectures.

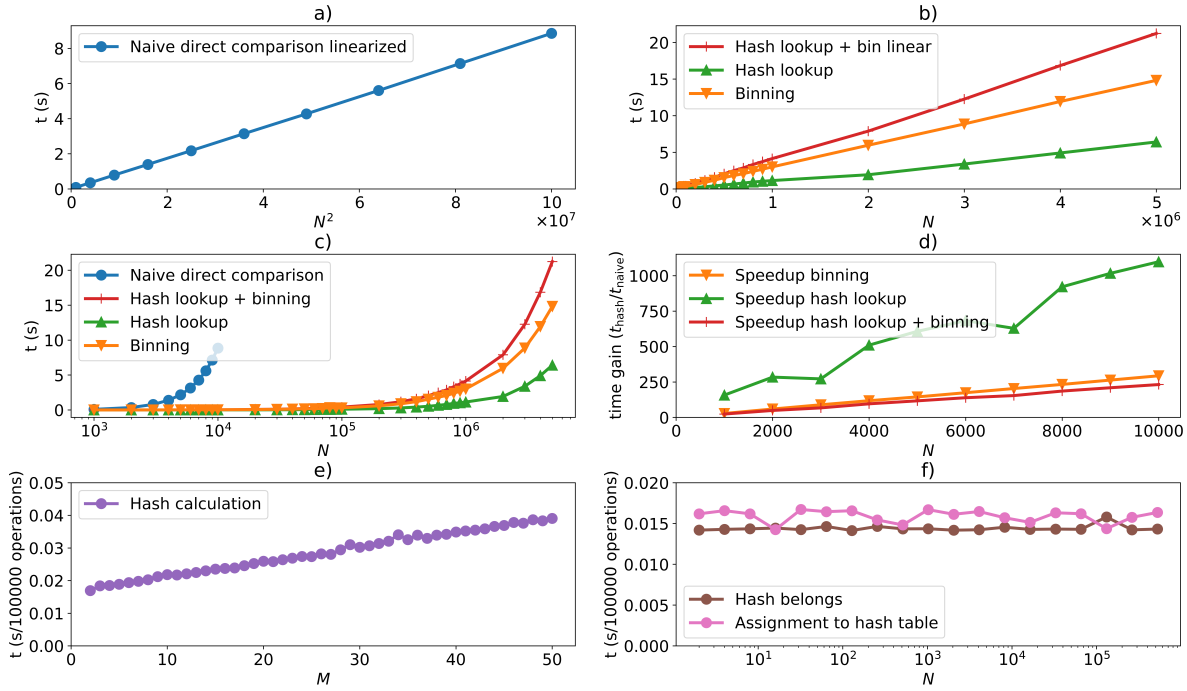

Figure 4.3: Plots of the timing of the different algorithms with increasing system size. a) Naive lookup vs. squared size of dataset. b) Different parts of the BAH algorithm vs. size of dataset. c) All algorithms together in log scale for comparison. d) Relative speedup or time gain of the different parts of the BAH algorithms compared to the naive approach, calculated as $t_{\text{algo}}/t_{\text{naive}}$, with t_{algo} the timings of the different parts of the algorithm from b). e) Scaling of the hash calculation with ACSF vector size, per 100.000 operations. f) Behavior of hash table operations with dataset size, per 100.000 operations. Reproduced from Reference [205], with the permission of IOP Publishing.

4.3.2 Analysis of the Distance in Symmetry Function Space

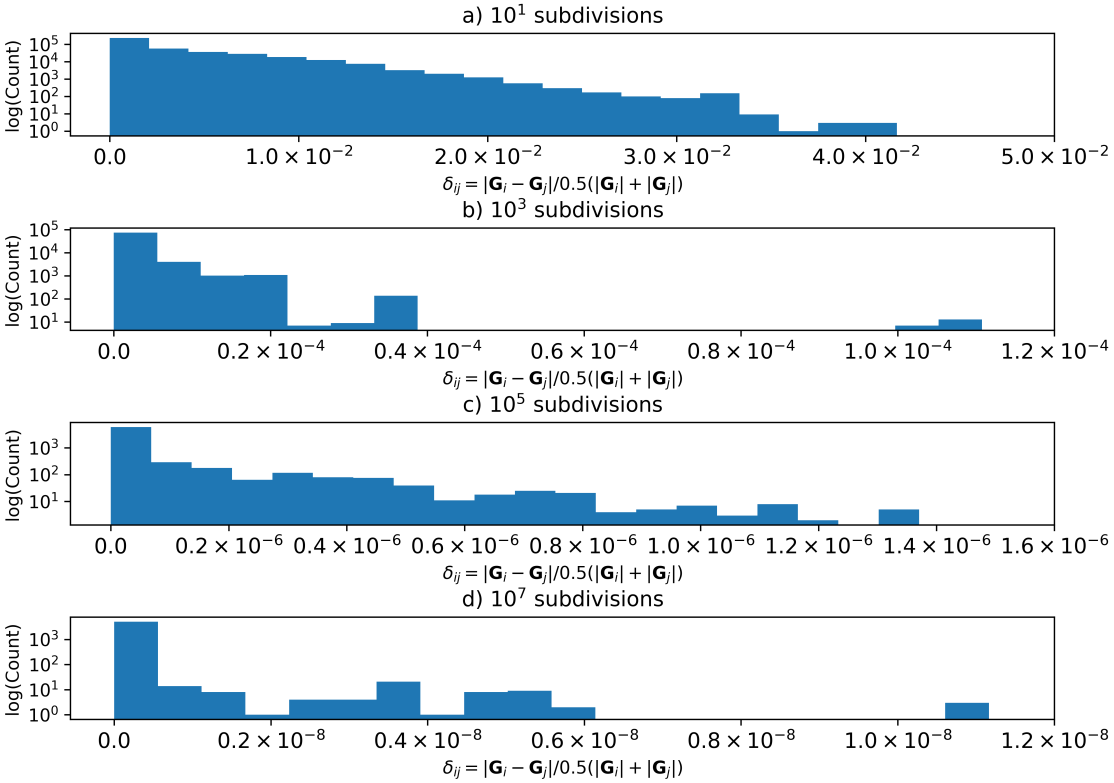

Figure 4.4: a)-d) Histograms for the typical intra-bucket ACSF relative distance (δ) values for different subdivisions ($10^1, 10^3, 10^5, 10^7$) in the ZnO slab dataset. Other intermediate subdivisions ($10^2, 10^4, 10^6$) exhibit similar behaviors. The counts axis is logarithmic for better visualization. Reproduced from Reference [205], with the permission of IOP Publishing.

Since hash functions cannot be utilized directly to measure distances in ACSF space, and information is otherwise lost in the process of binning and hashing, the question arises whether the BAH algorithm is even capable of reproducing distances between atomic environment descriptions. Certainly ACSF vectors that are exactly the same will land in the same integer representation and thus bucket, but what happens with vectors that are just close?

To investigate this, a relative distance in ACSF space is defined as

$$\delta_{ij} = \frac{|\mathbf{G}_i - \mathbf{G}_j|}{0.5(|\mathbf{G}_i| + |\mathbf{G}_j|)}, \tag{4.4}$$

where $\mathbf{G}_i$ and $\mathbf{G}_j$ are a pair of symmetry function vectors corresponding to atomic environments that ended up in the same bucket, and which are thus similar for the BAH algorithm. This distance is calculated between atoms i and j of the same element.

This distance is calculated on the same dataset for different number of subdivisions, from which a histogram can be produced, as appears in fig. 4.4. As expected from environments in the same bucket, most of the distances calculated are zero. As the number of subdivisions increases, the maximum intra-bucket distance (right-most bar of the histogram) rapidly becomes smaller, because the binning imposes a stricter criterion for similarity. For the maximum number of subdivision in the plot, the maximum distance corresponds to numerical noise inherent in the floating point representation. The behavior of the maximum and average intra-bucket distance surprisingly follows a linear relationship with the number of subdivisions in a double logarithmic plot (fig. 4.5). Changing the number of subdivisions

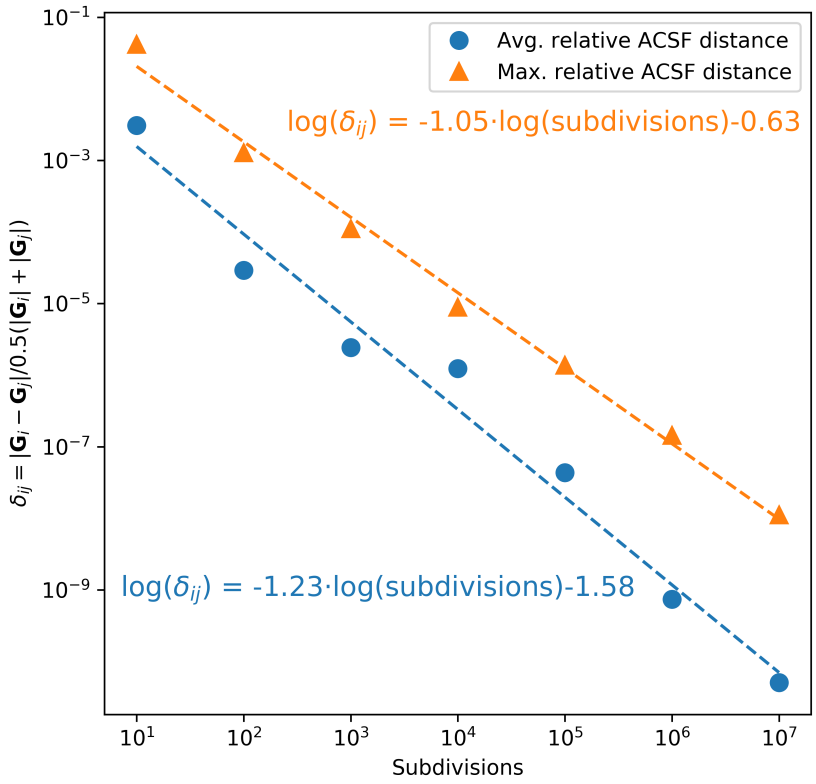

Figure 4.5: Maximum and average intra-bucket relative distances for the histograms in Fig. 4.4 versus number of subdivisions, in log scales. Notice that they follow approximately linear relationships, and trendlines with corresponding fitting equations are included. Reproduced from Reference [205], with the permission of IOP Publishing.

therefore allows for tuning how strict the BAH algorithm is, and what the maximum distance between two ACSF vectors can be before a similarity is no longer detected.

Given this behavior of the distances in ACSF space, it is also of interest to study the corresponding distances in associated property space, such as forces. In fig. 4.6 the difference in force magnitude [1] vs. the ACSF relative distance δ is plotted, for different subdivisions. It can be seen in a) that there is a relationship between both quantities, albeit weak. This is expected, since atoms with similar environments and thus similar ACSF vectors should also exhibit similar properties, such as forces. Distances in "force space" do not necessarily transfer linearly into ACSF space [128], which makes the dependence weak, once again highlighting the problems with distances in high dimensional spaces. As the number of subdivisions increases and only atoms with more and more similar atomic environments are considered, the distance in force space also decreases. In the final subfigure, d), the force distance corresponds only to DFT noise, since the detected environments are the same for all practical purposes.

[1] When comparing force components directly, care should be taken. ACSF vectors are invariant with respect to rotations and translations in coordinate space, but forces are *not*. This is due to the derivatives involved in going from energy to forces, which add a direction component. The result is that with the same ACSF vector, one can have different force vector orientations, that is, the components of the force vector might not match. The predicted magnitude of the force vector should on the other hand remain consistent since it is directionless. A trivial example of this is an unrelaxed unmodified slab with two interfaces: atoms in the top and bottom surfaces will have identical environments as described by their ACSFs, but the Z-component of their force vectors will necessarily, due to symmetry, be opposite. This becomes more complicated for more homogeneous systems such as liquids and amorphous solids, where the same atomic environment might be found in a variety of orientations. Thus only force vector magnitudes should be compared, or a consistent orientation of the environments should be achieved in some way.

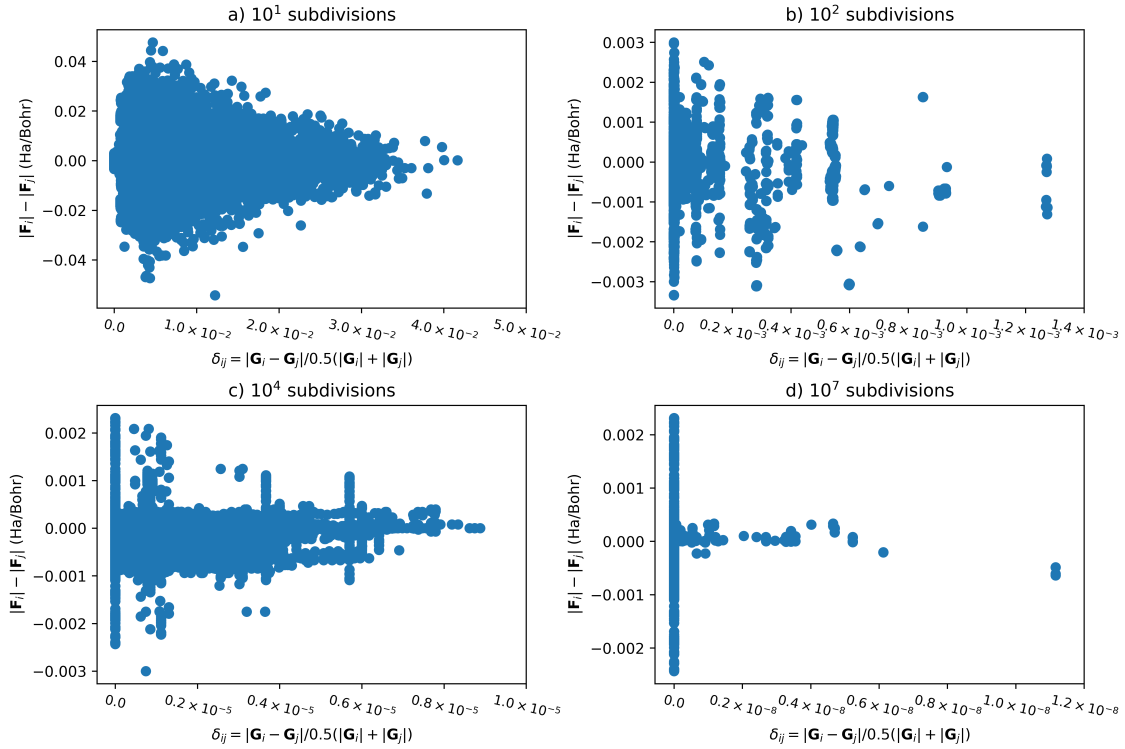

Figure 4.6: Difference in force magnitude vs. the ACSF relative distance, δ, for different subdivisions of the BAH algorithm applied to the ZnO slab dataset. The points present in each subplot are not always the same, since the plots are generated from environments that collided for a given number of subdivisions. Notice the difference in the scale of the x- and particularly the y-axis for a) when compared to b)-d); the force spread for structures with $\delta_{ij} \approx 0$ is due to remaining numerical noise in the DFT data. Reproduced from Reference [205], with the permission of IOP Publishing.

4.3.3 Results for Different Symmetry Functions

In the previous section it was shown how the resolution power of the algorithm changes as the number of subdivisions is altered. The question is then, what happens if we instead change the ACSF descriptor set? For this purpose, the number of total bucket collisions and the number of collisions in the largest bucket was counted for different subdivisions and different ACSF sets.

It is expected that as the number of subdivisions and ACSF functions increases, the number of collisions should go down. More divisions means that environments need to be more similar in ACSF space to collide (see Sec. 4.3.2) and more ACSFs lead to a more granular and precise description of each environment. This count should eventually converge since datasets sometimes include, by necessity, environments that are exactly the same. For example, this happens in a slab for atoms in the center of the slab or at the top and bottom of the slab, which have by symmetry the same atomic configurations. This rough prediction is confirmed in fig. 4.7. For this plot the ACSF set has been systematically increased in the order listed in the appendix.

In subfigures a) and b), collisions initially go down quickly as the number of members of the ACSF set increases, across all subdivision numbers. After this initial fall, the number of collisions plateaus with a slightly downward trend. The line with 10^5 divisions offers the highest sensitivity, showing changes across the whole ACSF range. Being able to differentiate chemical environments in this way is a necessary (but not sufficient) condition for a good HDNNP fit, in which case the BAH algorithm could be utilized to identify a floor to the size of the ACSF set.

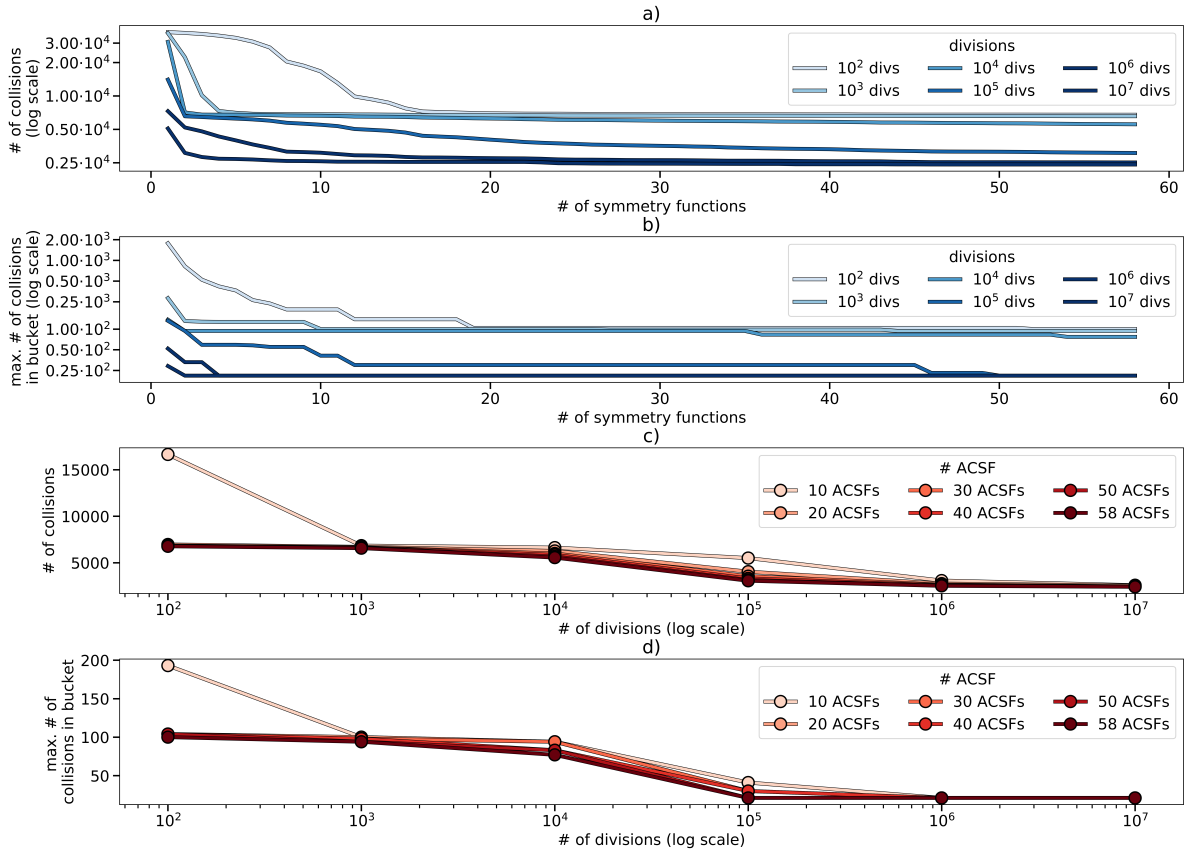

Figure 4.7: Panels a) and b) show the total and maximum number of hash table collisions, i.e., configurations that hash into the same bucket due to similarity of their ACSF vectors, vs. number of ACSFs, for different binning divisions. Panels c) and d) show the same properties as a function of the number of binning divisions, for different numbers of ACSFs. Reproduced from Reference [205], with the permission of IOP Publishing.

Going a step further, fig. 4.8 illustrates the three-way relationship between number of ACSF utilized for a fit, number of detected collisions, and the fitting accuracy as measured by the root-mean squared error (RMSE) of energies and forces. Subfig. a) shows that, as expected, increasing the number of atomic descriptors improves the fit accuracy, up to about 30 ACSF where the error plateaus. Subfigs. b) and c) show that RMSE can also be correlated to the number of detected collisions, since these in turn depend on the number of ACSF. The curve for 10^5 subdivisions exhibits the greatest sensitivity, that is, it covers the largest range in the x-axis. These series of plots demonstrate that the number of collisions can indeed be utilized as a rough measure of the final quality of a given descriptor set, with the benefit that no actual fit is required. The approach is then to propose different sets of ACSFs, and find those that minimize the number of detected collisions, and then perform expensive fits for only the best few.

4.3.4 Curating a Dataset

An important goal for the BAH algorithm is to be able to "curate" a dataset, that is, detect and remove repeated structures or atomic environments that do not provide new information about the PES of the system. This can be used to remove structures already in a dataset, making future fits faster; or to filter new configurations before processing them with an expensive electronic structure reference method, saving computational time.

To demonstrate how this would work with BAH, structures were removed from the ZnO dataset with two different methods and the resulting predictive power of NNPs trained on these datasets compared. The first method was a random removal of structures, as a control comparison. This method was repeated multiple times to average out effects from randomly selecting structures. The second method utilized the BAH algorithm as a guide: structures in the dataset were ranked according to the number of shared environments, structures with the highest number removed, then this number was recalculated (since removing one structure means all those who shared environments with it change their ranking), and the procedure repeated until a predefined percentage of the dataset had been removed. The random selection method also removed this same amount of structures for a fair comparison. All removed structures were placed in the testing set of each fit, with fitting errors then available for the removed structures and the remaining dataset (the training set).

The results from this procedure are presented in figs. 4.9 and 4.10. The first figure compares the fitting error for energies and forces for the BAH or random (RAN) approach. Up to 8% of removed structures, the BAH approach overperforms (smallest predictive error for the testing set/removed structures) most or all of the random removals, particularly for forces. Beyond this point, BAH still usually performs averagely when compared to random removal. The second figure shows a more detailed view of the fitting process, comparing the RMSE between the methods for forces and energies, for both training (remaining dataset) and testing (removed structures) sets. As can be seen in subfig. a), the training RMSE for energies is similar for both removal methods ($\Delta\text{RMSE}_{\text{BAH}-\text{RAN}}$ as defined remains around zero), but favors the BAH approach for the removed structures in the testing set. A similar behavior can be appreciated in subfig. b). This shows that both removal procedures lead to the same quality of fit on the training data, and so the effect of the quality of fit of the testing data can be attributed to the BAH approach. Subfigs. c) and d) show a similar result, but with a relative error. The properties of the structures removed with the BAH method can be predicted 20-40% better than with a random removal approach, showing once more that BAH is capable of detecting those atomic environments that are repeated and do not have new information that is required for a robust fit.

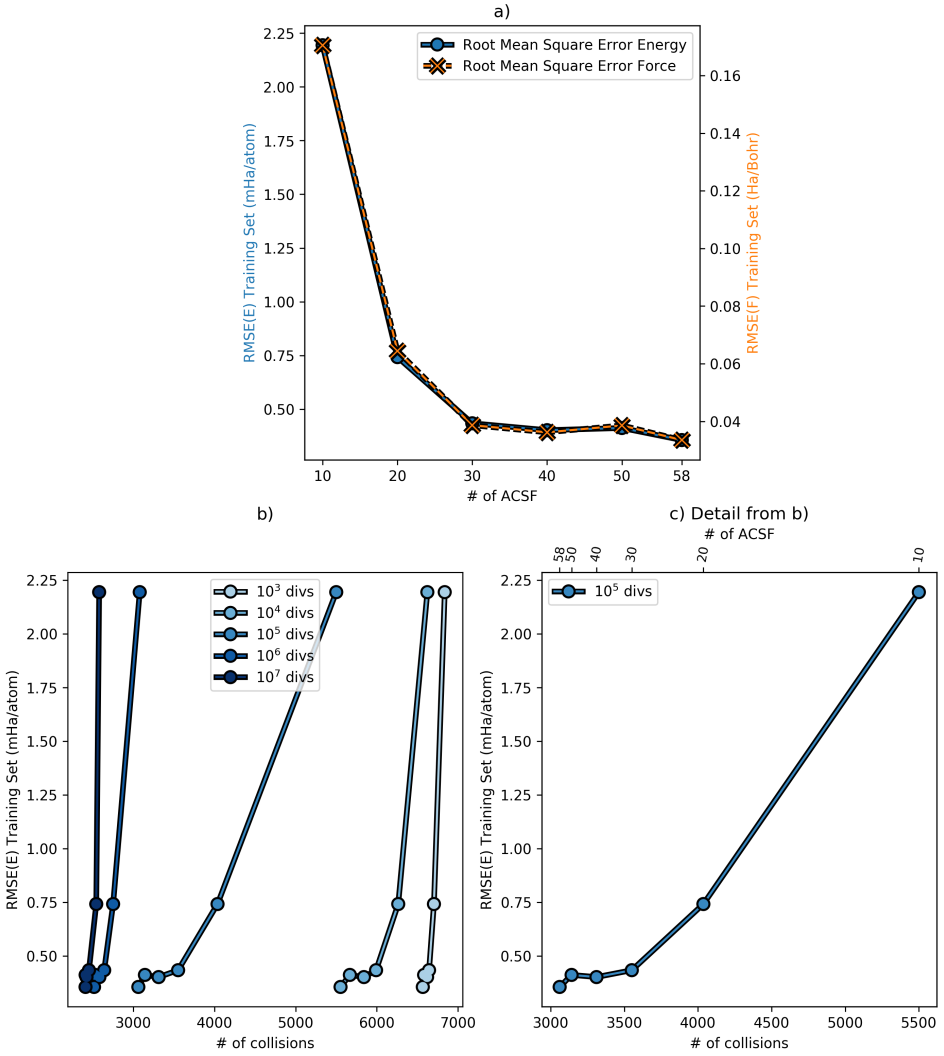

Figure 4.8: a) Root mean squared error (RMSE) for energy and forces for the ZnO dataset with different number of ACSFs as descriptors. b) and c) Energy RMSE at different ACSF numbers vs. number of collisions at the same ACSF number detected by the BAH algorithm with different numbers of divisions. Plots of force RMSE have a similar shape. The dependence of the number of collisions and RMSE on the number of ACSF descriptors is made more explicit in c) for the 10^5 divisions curve. Reproduced from Reference [205], with the permission of IOP Publishing.

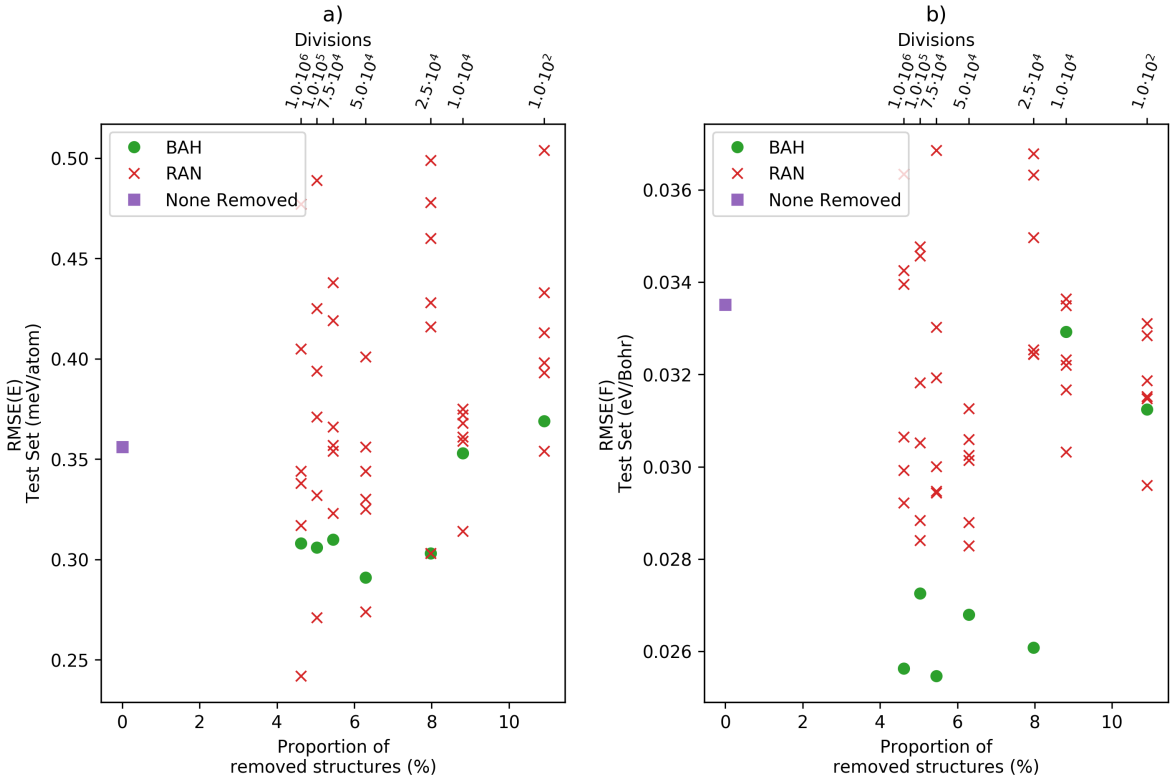

Figure 4.9: RMSEs for a) energies and b) forces, for structures removed from the ZnO dataset utilizing the BAH algorithm at different numbers of divisions, or a random selection approach. A reference point for the accuracy of the NNP on the whole dataset is also presented. BAH performs similar or better to random sampling for energies, and better than random until 8% of removed structures for forces. Reproduced from Reference [205], with the permission of IOP Publishing.

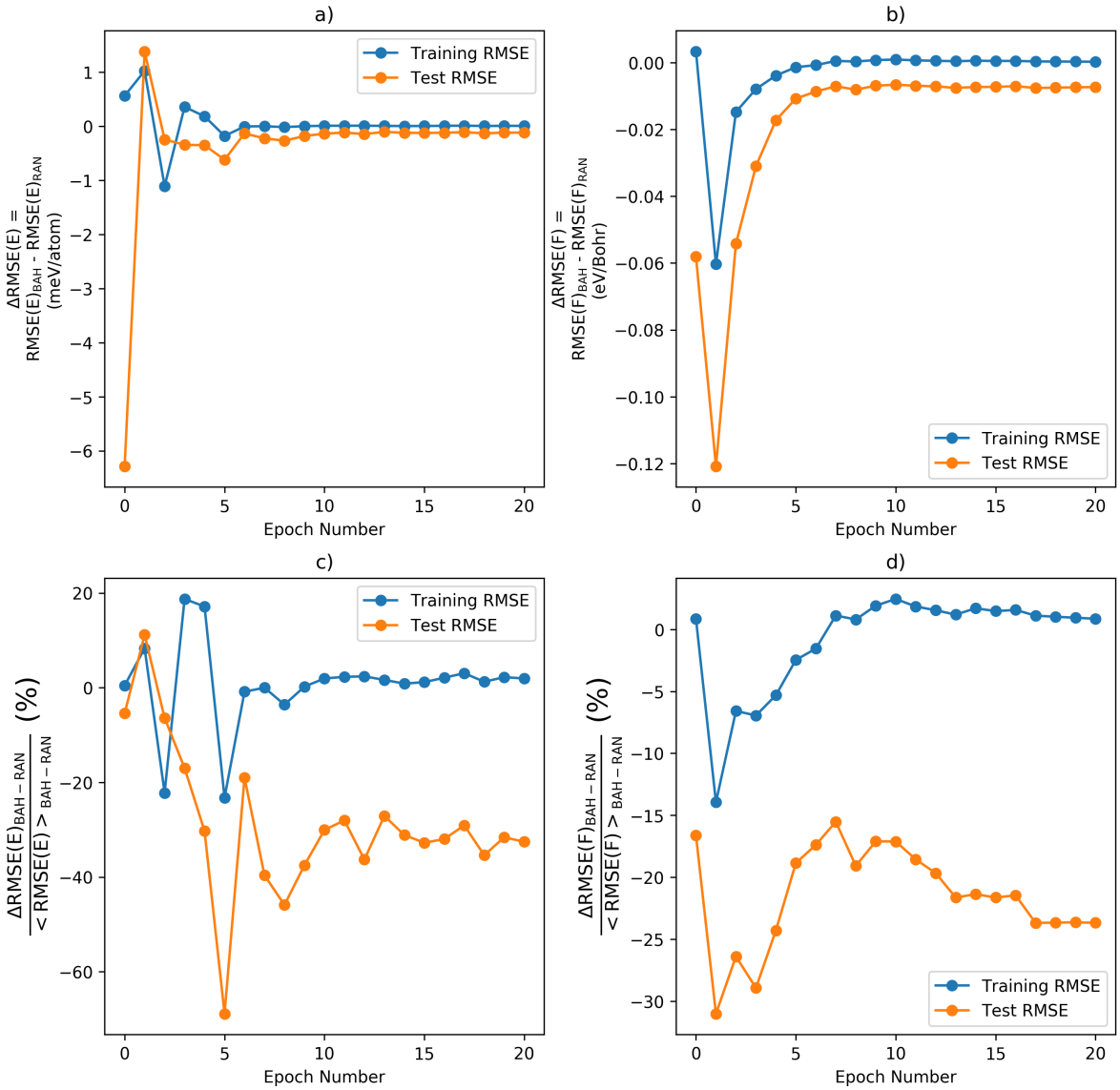

Figure 4.10: Comparison of the accuracy of a NNP trained on a modified dataset by removing 5% of the structures with a BAH (with 10^5 divisions) or a random approach. The removed structures are put into a test dataset. a) and b) Absolute difference in energy and force RMSE vs. training epoch ($\Delta\text{RMSE}_{\text{BAH}-\text{RAN}} = \text{RMSE}_{\text{BAH}} - \text{RMSE}_{\text{RAN}}$, negative values mean that BAH is more accurate). Both approaches result in a similar accuracy for the training set, but BAH is favored in the removed/test set. c) and d) As in the previous subfigures, but a percentage difference (with $< \text{RMSE}_{\text{BAH}-\text{RAN}} >= 0.5 \cdot (\text{RMSE}_{\text{BAH}} + \text{RMSE}_{\text{RAN}})$, once again negative values mean BAH is more accurate). Once again both methods perform similarly for the train data, but BAH performs up to 20% better on the test set. Reproduced from Reference [205], with the permission of IOP Publishing.

4.3.5 Effective Number of Subdivisions

After all these results, the question is what is the best number of subdivisions to utilize for the BAH analysis. The number and type of ACSF will usually be fixed beforehand for a given dataset, but the number of subdivisions is intrinsic to BAH. As can be seen from fig. 4.5, the number of subdivisions roughly reproduces the symmetry function space distance between atomic environments. As such the "right" subdivision range depends on the distance that we need to detect between environments, and as such multiple ranges are valid. A lower number of subdivisions (in the range of 10^2 to 10^4) appears to provide a more granular behavior in the number of collisions vs. symmetry functions utilized. For detecting contradictions (see sec. 4.3.6) we require environments that are either extremely similar or exactly the same, in which case the higher number of subdivisions (10^6 to 10^7) is better suited.

Whether the number of subdivisions required depends on the specific dataset is harder to evaluate. Since MLP datasets are derived from physically "reasonable" configurations, they share roughly the same distribution of data, with some differences depending on the involved elements, states of matter present, energy ranges covered, etc. The parameters of the trendlines in Fig. 4.5 might depend on the specific composition of the data in the dataset, but as long as the relationship with ACSF space distance remains, the specific parameters are not crucial.

In the end the specific number of subdivisions needs to be tested with each dataset and adapted to each desired analysis, but the BAH process is so fast that binning a dataset multiple times is not a problem. From the results presented here, the recommendation is to test three widely separated orders of magnitude of subdivisions ($10^3 - 10^5 - 10^7$ for example), and refine the algorithm following the results.

4.3.6 Conflicting Information

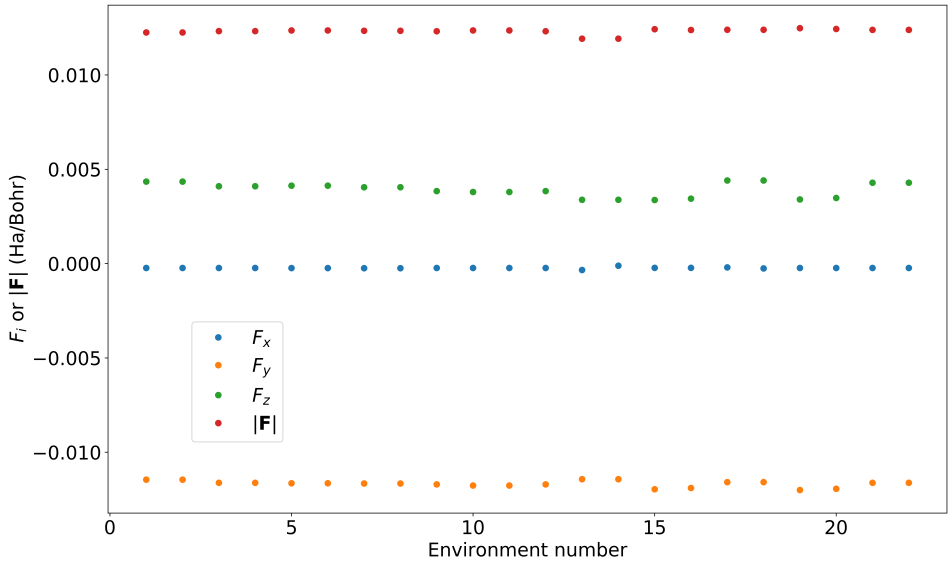

Figure 4.11: Force components and force vector magnitude for 22 environments found in a collision bucket. Note that although the ACSF vector for all environments is similar, there are slight differences in force values arising from numerical noise in the DFT calculations. Reproduced from Reference [205], with the permission of IOP Publishing.

Contradictions arise in a MLP dataset when atoms are described by similar environments, but the derived predicted properties such as force, spin, charge, etc. differ by more than a reasonable threshold (that is, beyond the error inherent to the calculation method). These contradictions can arise due to small ACSF sets or cutoff radius, in which case the environment around atoms is not sufficiently

described and factors beyond the environment alter the derived properties. Another possible source is bad electronic structure data, such as non-converged calculations that can appear when calculating the thousands of structures required for a dataset. Contradictions are detrimental to the fitting process, since in case of conflicting data the HDNNP cannot reach a high fitting accuracy [110].

With 10^5 binning divisions on the ZnO dataset, the most populated bucket contains 22 environments. The ACSF vector of these configurations is identical, but plotting their DFT force components [2] and magnitude results in fig. 4.11. The forces for the central atoms in these environments are not exactly equal due to the limited accuracy of DFT forces, but is within the expected error margin for the HDNNPs [61], about 100 meV/Bohr. In this case, no contradiction is detected. For larger datasets, the points within buckets can be automatically analyzed, and a contradiction warning raised if a specified property differs more than a certain threshold.

4.4 Conclusions

The BAH method allows for a simple and efficient comparison of large amounts of atomic environments, such as those required in the rest of this work. The sensitivity of the algorithm can be systematically tuned by changing the number of subdivisions of the ACSF space.

The speed of the algorithm allows for processing millions of atomic environments in minutes, permitting many different analysis and operations on a dataset. Redundant atomic environments can be detected and thus repeated electronic structure calculations or fits with bloated reference data can be avoided. New configurations obtained from validation simulations employing an initial HDNNP can be screened with the algorithm, to see if they are sufficiently distinct form those structures already present in the reference training set, and decide whether it is worth it to employ resources on performing new electronic structure calculations. For the purpose of maintaining reference datasets, the BAH algorithm is complementary to the use of other procedures such as active learning, which present the disadvantage of requiring already trained potentials. Searching for repeated environments can also lead to identification of conflicting information in the dataset, where atoms identified by the same descriptor vector posses different derived properties. This can be due to errors at the electronic structure calculation level, which due to the amount of calculations needed would otherwise be hard to single out.

Finally, the number of similar environments is a way to validate a given descriptor set, and a three way correlation can be found between detected number of collisions, number of descriptors utilized, and quality of a fit as quantified by the RMSE. Poor descriptor sets will results in a large number of erroneously identical environments, but as the set grows, it should be able to differentiate more and more environments. This is a minimal (but certainly not sufficient) condition for a good fitting procedure.

[2] See previous footnote regarding force comparisons.

Chapter 5

Genetic Algorithm Global Optimization of Small Supported Copper Clusters

This chapter is adapted from Reference [138] M. L. Paleico and J. Behler, "Global optimization of copper clusters at the ZnO($10\bar{1}0$) surface using a DFT-based neural network potential and genetic algorithms," J. Chem. Phys. **153**, 054704 (2020), with the permission of AIP Publishing.

5.1 Motivation

Supported clusters, due to their relevance for the study of more complex catalytic models, are an active area of research [43, 229–232], both from the theoretical [233–235] and experimental [236–239] point of view. It is not trivial to determine the structure of such clusters with simulations, since even for the smallest cluster sizes, there is a large number of available degrees of freedom and the PES dictating the interaction between cluster and support can be quite complex. For this purpose, often global optimization techniques are utilized. In this chapter, the goal is to utilize a genetic algorithm [139, 140] (GA) (see also sec. 2.3 and 3.4 for a more in-depth explanation of the GA approach) search to optimize copper cluster with 4 to 10 atoms, deposited on ZnO($10\bar{1}0$).

Global optimizations of such small deposited clusters face a number of challenges. The complexity of the PES for the system demands an *ab-initio* level of accuracy, but the number of degrees of freedom and consequent cost of global optimization techniques makes such searches very expensive. Additionally, since the clusters are supported, atoms from the slab material have to be taken into consideration, and the simulation cell needs to be large enough that periodic images of the cluster do not interact with themselves. This adds a large amount of "dead weight" atoms that need to be there for structural stability reasons, but do not really provide much to the simulation. NNPs are ideal in this case, since the extra atoms are cheap to calculate, the structures generated by the GA search are all closely related (and thus ideal for NNP training), and as pointed out recently in the literature [207, 208], the GA search can be combined with a NNP to generate a self-training algorithm which learns the PES "on the fly" by studying the candidates generated by the GO search.

Previous GO searches in the literature for the Cu-ZnO system utilized either parametrized force fields [67], or worked with *ab-initio* calculations by first pre-optimizing the clusters in an isolated configuration in vacuum [71] before depositing them on the target surface for further study. Here, results are based directly on DFT data of supported clusters, as reproduced by the NNP. As mentioned, this allows for easy inclusion of the support, but also allows to extend the number of independent GA searches, and how many candidates are generated in each search.

Other GO methods were implemented and tested before arriving at GA as the ideal candidate. Grid-based searches [240] would seem ideal for such small systems, but the grid needs to be known or guessed beforehand, and as seen from our results a fixed grid approach would not work for this system since small clusters strongly deviate from a perfect face centered cubic grid. A flexible grid approach as developed in the course of this thesis (see reference 199) could work in this case. A grid is introduced into the system, whose sites can be either occupied or unoccupied by atoms in the cluster. The

grid points interact among themselves with a simple potential, which guides the grid into regular arrangements without over-biasing the system. Occupied and empty positions in the grid are exchanged following a BHMC approach (see next paragraph), and successful exchanges are following by an optimization of the empty grid points around the new system configuration. This approach combines the advantages of BHMC with the dimensionality reduction of a grid, without needing to assume a given shape for said grid.

BHMC [149] is often utilized for small clusters, but is according to personal experience too slow, since it attempts to perform a linked Monte Carlo simulation. GAs inherently parallelize their search, and additionally utilize information from previous structures to bias future ones, which greatly speeds up PES exploration when compared with basic BHMC. Either way it, has been suggested by the creators of BHMC themselves that both approaches actually produce similar results [149].

This chapter is organized as follows: Section 5.2 presents results addressing the global minimum (GM) and low-energy local minima (LM) from Cu_4 to Cu_{10} (5.2.1), the interface structures (5.2.2), and the geometric properties of the optimized clusters (5.2.3). Finally, conclusions are drawn in Section 4.4.

5.2 Results

5.2.1 Global Optimization Results

Figure 5.1 shows the results of running the GA algorithm for Cu clusters between 4 and 10 atoms on ZnO($10\bar{1}0$). The results from independent runs have been combined to obtain this collection of structures. Cluster atoms appear to interact strongly with the oxygen atoms of the support, with most copper atoms making contact at positions on top of oxygen atoms. The clusters exhibit a rich variety of patterns. A number of geometrical families can be detected that are consistently present among the lower energy configurations at each cluster size.

Starting at the smallest cluster size of Cu_4, clusters already exhibit 3D growth instead of laying flat on the surface of the ZnO. The same is true for larger cluster sizes, with completely flat configurations where every cluster atom is directly in contact with the support only observed at high energies, and as such are not depicted in this ranking. Trivially, the clusters do in fact form a cluster and are not spread across the support, in agreement with experimental results even at low monolayer coverage [42] and theory [241]. Due to the applied GA mutations, clusters sometimes do separate, but these configurations end up in the high energy range. For no size range are extended periodic structures across the PBC observed such as wires and sheets, as has been proposed in other studies [67] based on parametrized potentials. These kinds of structures only appear in cases where the simulation cell is too small for the amount of copper deposited, and the cluster atoms start interacting with themselves across the PBC.

Due to the reduced number of atoms in the Cu_4 cluster, it appears to form only two main structural patterns at low energies. The first pattern is a truncated square pyramid with only 3 atoms making contact with the support, which is found in the GM in Fig. 5.1 and in LM 5. Notice that the GM and LM 5 are in fact different with respect to the support, as the adsorption footprint on the ZnO($10\bar{1}0$) surface is not symmetric because of the orientation of the ZnO dimers corresponding to the [0001] and [$000\bar{1}$] directions. In this direction, two characteristic distances are present, a short Zn-O distance within the dimers, and a long distance between different dimers. This leads to a different interaction pattern with the surface: in the GM, two copper atoms interact with the oxygen at the end of a dimer and one lies between the Zn and O of a dimer, while for LM 5 the bonding pattern is the other way around. The cluster-surface interaction thus exhibit a preferential direction, since rotating the cluster 180° results in a non-equivalent environment. The second pattern for the Cu_4 clusters found in the rest of the LM consists of small, more or less distorted tetrahedra. These structures also exhibit this preferential direction.

For Cu_5, many of the structures are continuations of the Cu_4 structures. The GM completes the square pyramid of the previous GM, and LM 1 is a distortion of this pyramid. LM 2 contains an added

atom on top of the LM 1 of Cu_4. LM 3 is a flattened version of the GM, and is reminiscent of the rectangular patterns found in Cu(111). LM 4 and 5 are similar to the GM of Cu_4, with an extra atom on top instead of on the basis of the pyramid.

For Cu_6, the GM now corresponds to the previous LM 3 with an atom on top, a pattern that will be known as a saddle. LM 3 is the first cluster with a five-fold symmetry, and LM 2 and 5 show some of the few examples of flat clusters at low energies. LM 2 is also the first cluster to extend beyond a (1×1) surface cell. This growth happens in the "short" direction of the surface, perpendicular to the direction of the Zn-O dimers. Since this direction is shorter, it is easier to cover the gap without straining the cluster excessively, until the cluster gains enough atoms to grow in the long direction.

For Cu_7, once again we can find many repeated structural patterns from the previous cluster size. The GM continues the trend of the saddle configuration, with an atom on top and to the side. LM 5 also belongs to the saddle family, but with the extra atom on top and at the end of the cluster. LM 1 and 4 are related to the five-fold symmetry cluster at the LM 3 of Cu_6, and LM 3 continues the previous triangular flat conformation, showing that the clusters prefer not remaining flat.

For Cu_8, the GM is a moderately distorted version of the cluster at the LM 1 or LM 4 of Cu_7, with an extra atom added at the corner. LM 1 and 5 show new modifications of the saddle, and the first cluster with a hexagonal interface contact pattern appears at LM 4, related to the LM 5 of Cu_6. For Cu_9, the saddle-based pattern once again becomes the GM. Finally, for Cu_{10}, by attaching an additional atom the previous LM 1 of Cu_9 becomes the GM, with the saddle configuration falling to LM 2.

The different structural families could be categorized more formally in a couple of different ways. Previous GA works [207] make use of the nudged elastic band [156] (NEB) algorithm to find the energy barrier between clusters within the same size category. From this a a map can be built, where clusters with low energy barriers will be close together, implying they belong to the same family. A number of problems are present with this approach i) it requires a NNP that has also been trained with data from transition path searches; ii) the approach is limited, as mentioned, to clusters of the same size, so families cannot be followed across different atom numbers; and, iii) as opposed to the case with molecules, atoms in a cluster are indistinguishable, which means it is not trivial to assign a correspondence between atoms in the initial and final configurations of the NEB.

A possible solution to this is to utilize an algorithm similar to the one in PHTM (see sec. 2.6.4), in particular eq. 2.48, to measure the structural distance distance between clusters. This of course would obviate the need for an extended NNP. It could also be applied to clusters of different size, by removing one atom from the larger cluster, finding the structural distance, and then testing again by removing a different atom, and so on until the best mapping is found. This aspect requires more development, as it would be of great interest to obtain a general classification of the structural families present in the GA results and their behavior as cluster size increases.

Another simpler but less robust algorithm is possible, which could be implemented easily with the available tools. The idea is to take a cluster of size $N+1$, remove one atom, minimize the resulting cluster with the NNP, and compare this structure one by one against all clusters of size N, using any tool that does not depend on atom identity (for example, the interatomic distance comparator in sec. 2.6.2). Then, start again with the original cluster, remove a different atom, compare again against all the smaller clusters, and repeat until a "best" match has been found. This approach is computationally expensive due to the extra minimizations required and the number of comparisons that scales as $O(N^2)$, but should be fast enough thanks to the NNP. It remains to be seen if this approach is robust enough, or how to define a "best" match under the circumstances.

From the structural patterns present across the different sizes a general conclusion can be derived: there is a compromise between the interactions of the cluster and the support, which would cause the clusters to lay flat and spread out onto the support at the cost of straining the Cu lattice; and the self-interactions of Cu, which tend to grow the cluster in a 3D fashion at the cost of reducing the contact area with the ZnO. This is complicated by the preferred adsortion positions at the support, as well as the atom-level roughness of the ZnO and the symmetries present on the surface.

In Fig. 5.2 the binding energies of the GM structures as a function of the cluster size are plotted. The binding energy is defined as

$$E_{\text{binding}}(N) = E_{\text{cluster + slab}} + \\ - E_{\text{ZnO slab clean relaxed}} - N \cdot E_{\text{Cu atom}}, \tag{5.1}$$

where N is the number of Cu atoms in the cluster under consideration, $E_{\text{cluster + slab}}$ is the energy of the whole system, including cluster and slab; $E_{\text{ZnO slab clean relaxed}}$ is the energy of the initial supporting slab, relaxed and without cluster atoms, and $E_{\text{Cu atom}}$ is the energy of an isolated Cu atom as obtained from DFT calculations, since the NNP has not been trained with this information. All other energies can be obtained from NNP calculations. Since the only term that changes between clusters is $E_{\text{cluster + slab}}$, this plot also shows the relative energies between all the clusters. As expected, a stronger binding is observed as a more negative binding energy with increasing number of copper atoms. As usually observed, the spread in energies within a cluster size becomes smaller as the clusters contain more atoms.

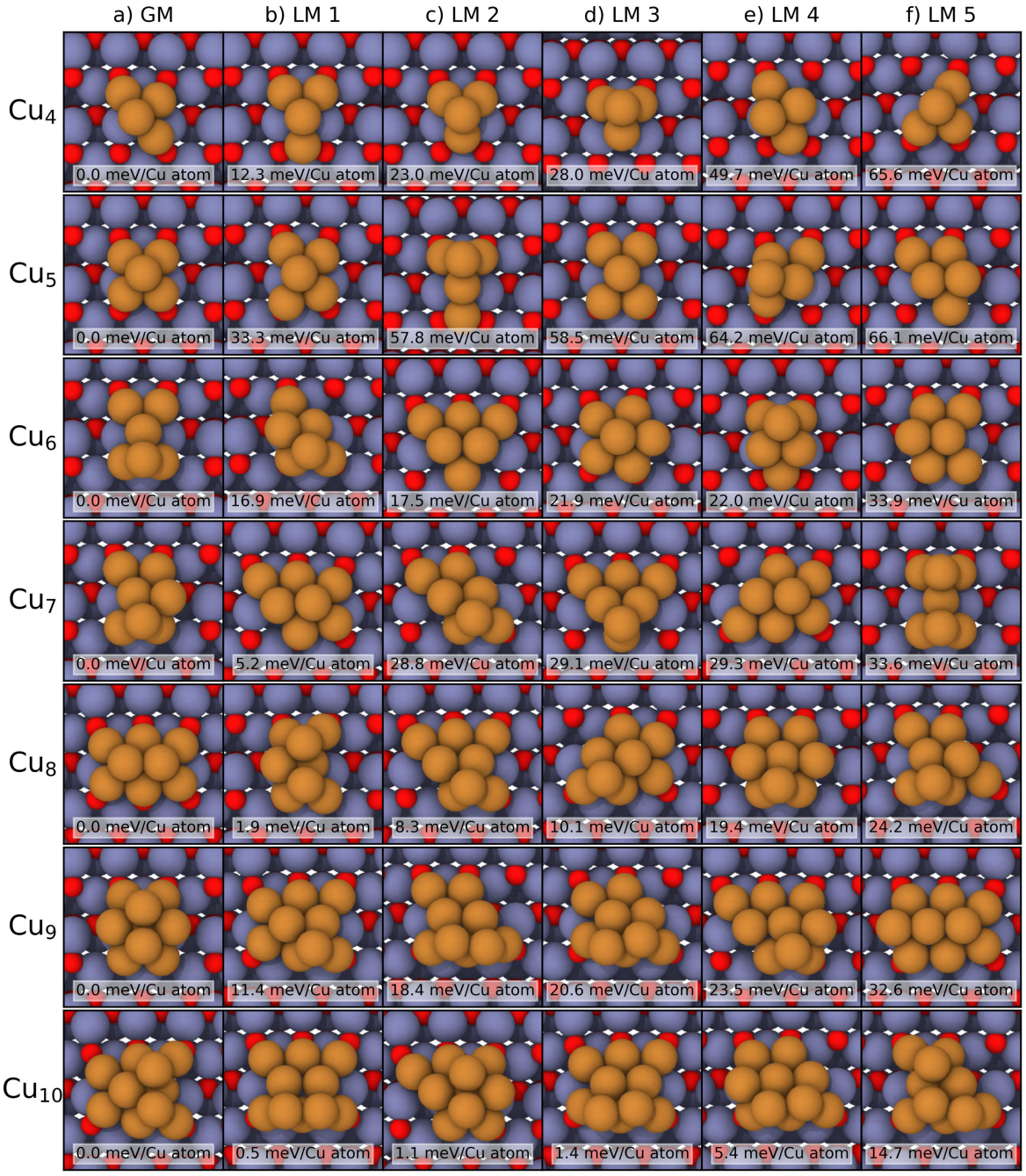

Figure 5.1: Global minimum (GM) and first five local minima (LM) for different copper cluster sizes from GA search on the ZnO($10\bar{1}0$) surface. Each row corresponds to a cluster size with a number of atoms as indicated on the left, each column corresponds to the ranking within a size. For each structure the relative energy per copper atom with respect to the GM is given. Reproduced from Reference [138], with the permission of AIP Publishing.

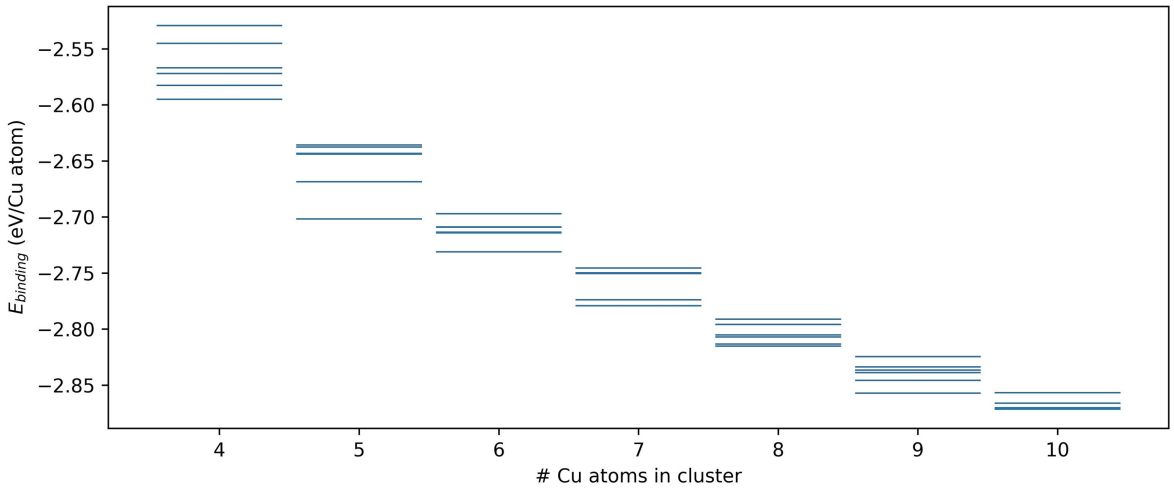

Figure 5.2: Binding energy per copper atom (see Eq. 5.1) for the GM and first 5 LM for the Cu_{4-10} clusters. Reproduced from Reference [138], with the permission of AIP Publishing.

5.2.2 Interface Structure

Since the presence of the ZnO support is the symmetry breaking object in the system, it is interesting to examine what the structure of atoms right at the interface between both materials is. For very large clusters, atoms far away from the interface should adopt the same configuration as in a free standing nanoparticle or as in bulk copper. However, right at the interface, a number of conflicting trends influence the disposition of atoms. Finite solids tend to adopt shapes that minimize their surface energy, as proposed by the Wulff construction [195]. This theory in principle only applies to mono-elemental particles in a vacuum, although extensions exist for interacting materials [241] and alloys [242]. Wulff's theory was proposed for the macro-scale, but it can be assumed as a starting point even for clusters of this small size.

The interaction between the cluster and the support can also induce new preferred geometries, due to the presence of strong adsortion sites and their distribution on the support. This can be interpreted in various forms, such as strong metal-support interaction[243–245], or the theory of lattice mismatch [161, 246]. Such a lattice mismatch is present for Cu and ZnO, since it is not possible to contain the low Miller index surfaces of each materials within a common supercell, without deforming one or both surfaces. This is solved by the system generating defects and strain, which can lead to enhanced catalytic effects [247–249]. For finite structures such as clusters this is still a problem, although somewhat alleviated by the fact that the clusters are finite.

The interfacial atoms of the minimized clusters of fig. 5.1 are shown in fig. 5.3. Here interfacial atoms are defined as those within a 1.6 Å vertical distance of the lowermost Cu atom of each cluster. This interval was chosen as it is thick enough to potentially contain a monolayer of all the low index Cu surfaces, and also include the sub-layer of Cu(110) which has a thickness of about 1.275 Å, i.e., one-half of the nearest neighbor distance, d_0. The distance criterion has been chosen about 0.3 Å larger than this value to accommodate also distorted positions of Cu atoms on top of the support, with vertical distances falling in the range of 1.3 to 1.6 Å. Atoms farther away from the substrate are displayed in a perspective view with larger radii, providing a sense of the 3D structure of these interface atoms. Identifying surface structures for such distorted configurations is not trivial, but a comparison can be attempted with the low Miller index surfaces of copper, (111), (100) and (110), which are known to be the most stable ones [250] for the material in vacuum, in the given order [251]. Surface energy calculations obtained with DFT and the NNP (see table F.2) show that Cu(111) is still the most stable with this particular DFT setup, but the other 2 surfaces are very close in energy.

Cu_4 clusters are too small to exhibit any easily identifiable surface pattern, presenting instead strange triangular patterns. Instead, starting with Cu_5, the GM shows an interface configuration reminiscent of the Cu(100) surface, organized in approximately square units of Cu atoms. This surface, despite being close to Cu(110) in energy terms, only appears for the smallest clusters.

In LM 3 of Cu_5 we observe the first example of a structural pattern that repeats for many of the larger clusters: five interfacial atoms, structured as a rectangle with an atom in the center that is slightly above the plane formed by the other atoms. This corresponds in fact to a configuration between that of Cu(110) and Cu(111). These two surfaces are related for fcc crystals. The (111) surface is usually described as planar and with atoms in a hexagonal configurations, but the pattern can also be interpreted as consisting of 4 atoms in a rectangular configuration with a fifth atom in the center. This rectangle has two characteristic distances: a short one equal to d_0, the nearest neighbor distance of 2.55 Å, and a long one equal to $\sqrt{3}a_0$, with a_0 the lattice constant of bulk copper, for a length of 4.42 Å.

The (110) surface, in contrast, is usually depicted as four atoms in the top layer in a rectangular configurations, with a fifth exposed atom in the center of the rectangle belonging to the next subsurface layer. In a similar fashion as for (111), the (110) surface can be thought as consisting of asymmetrical tilted hexagons, i.e., they are longer in one dimension than the other one. The distances in the rectangle consist of: a short one, which is again equal to d_0, 2.55 Å, and a long one equal to a_0, 3.64 Å. This shows that both surfaces are actually related: take the top two layers of the (110) surface, compress them in the vertical direction, and the (111) surface can be obtained. As such there is a continuum

between both surfaces.

This justifies why, for larger clusters, an intermediate behavior is actually obtained. Some of the interfaces of the clusters are flatter, with longer directions forming the long direction of the rectangle, which makes them closer to Cu(111). Others are less flat, with a central atom that is slightly above the other atoms in direct contact with the ZnO, forming a configuration closer to Cu(110). This can even change within one cluster, such as the GM at Cu_{10}, which exhibits two parts with the two different extremes.

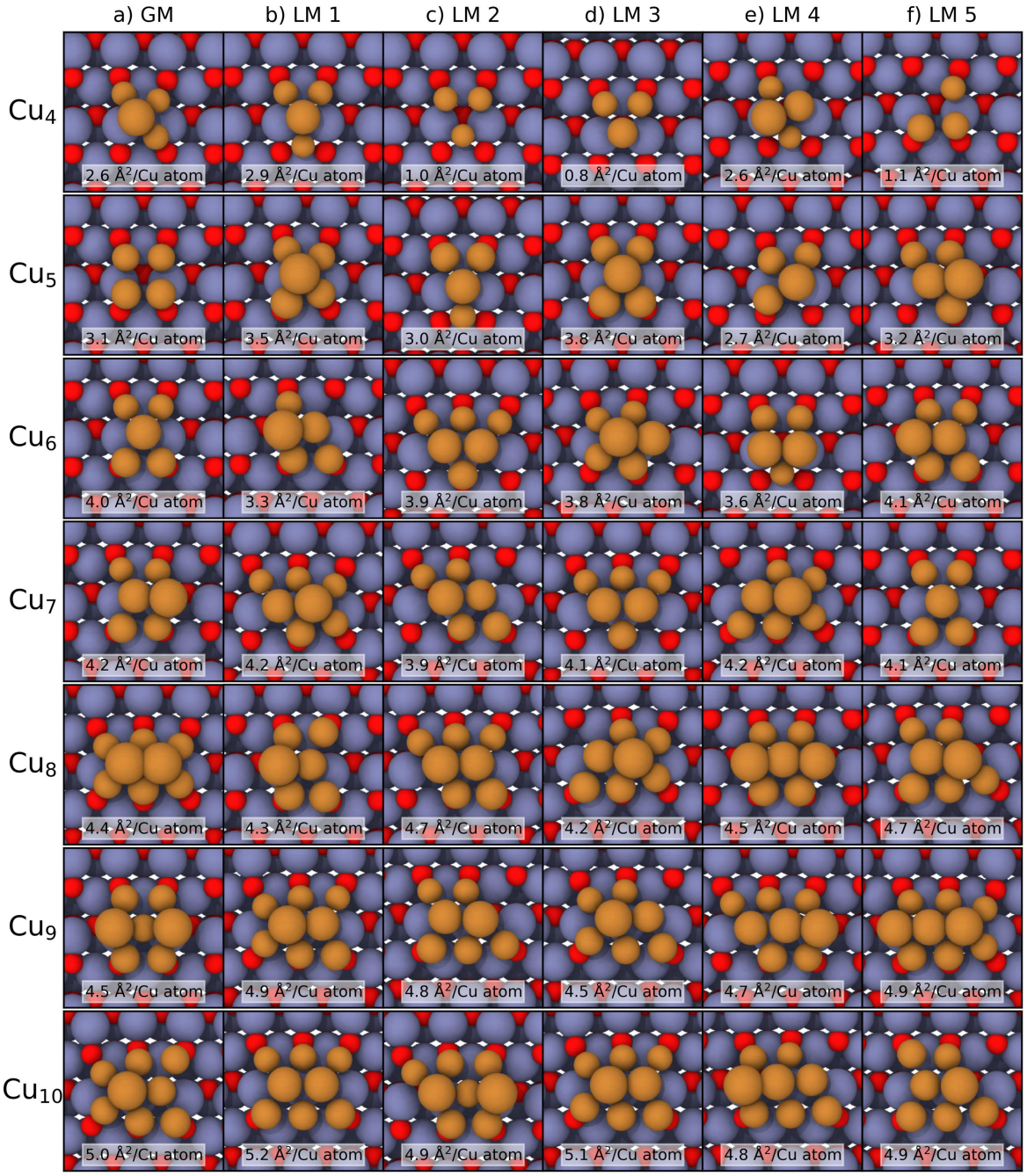

Figure 5.3: Interfacial Cu atoms of the optimized clusters in Fig. 5.1, with the number indicating the interface area per cluster atom as calculated with the convex hull algorithm. We have defined as interface atoms all those atoms belonging to the cluster that are within 1.6 Å vertical distance of the lowermost atom of the cluster, enough to encompass up to one monolayer of Cu(110) plus a margin of error. The radius for Cu atoms has been scaled by distance to the surface (larger as the distance increases) for visualization purposes. Reproduced from Reference [138], with the permission of AIP Publishing.

5.2.3 Properties

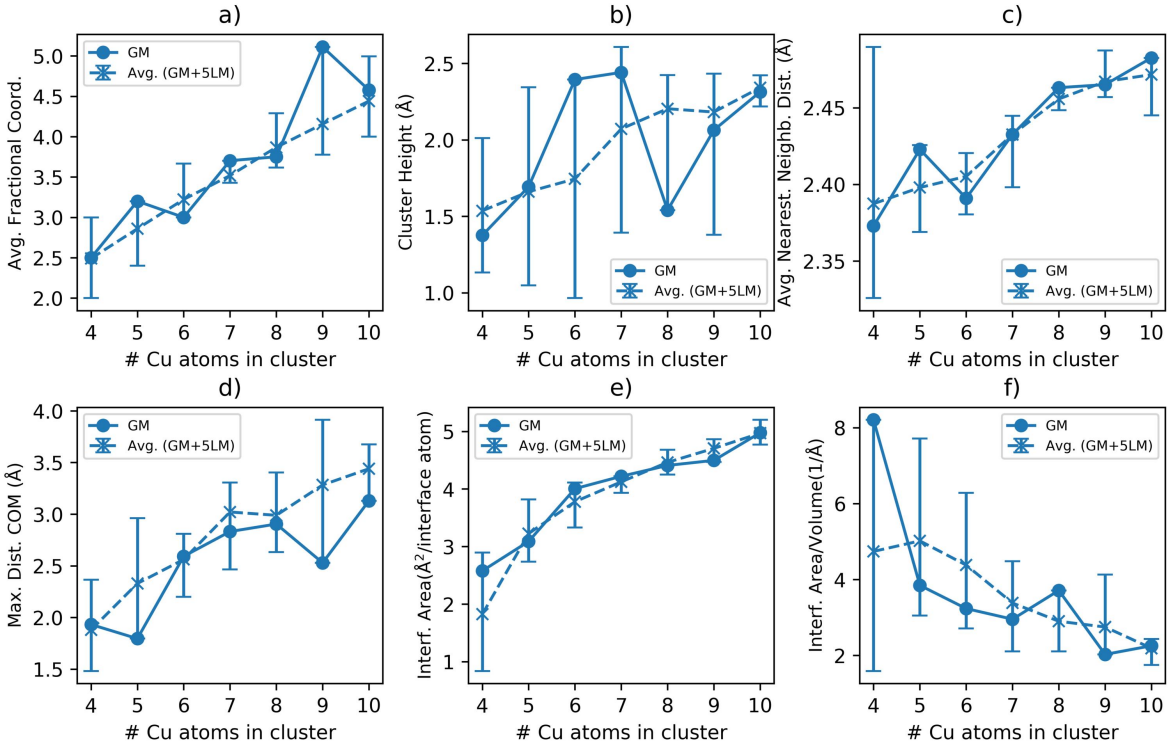

Figure 5.4: Various structural and geometrical properties of the clusters in Fig. 5.1 versus cluster size. Each plot contains two curves: the full line shows the values of the given property for the GM, the dashed line shows the average for the GM plus the 5 LM, with vertical bars indicating the minimum and maximum values within the group. In each subplot the following properties (explained in more detail in the main text) are plotted vs. cluster size: a) average fractional coordination number b) cluster height c) average nearest neighbor distance d) maximum distance to the COM of the cluster e) interfacial area as obtained from the QHull algorithm f) ratio of interfacial area to cluster volume, also from QHull. Reproduced from Reference [138], with the permission of AIP Publishing.

A number of structural properties have been analyzed for the obtained clusters, in an attempt to identify possible structural trends in the putative GM. These are:

- The fractional coordination number [252] of the atoms in the cluster. The fractional coordination number converts the usual integer-valued coordination number into a continuous value. Atoms beyond the nearest neighbor distance are considered as partially coordinating, instead of supplying a coordination of 0 as in the normal coordination analysis. This is a better measurement for such small clusters, and avoids coordination numbers varying wildly due to only being able to adopt integer values.
- The height of the cluster, as given by the difference in z coordinates of the top and bottom-most atoms in the cluster.
- The maximum distance of any atom to the center of geometry (COG) of the cluster. This shows the maximum extent of the cluster, and would be equivalent to the radius for spherical clusters.
- The average nearest neighbor distances, for each atom and for the whole cluster, with neighbors in the cluster up to 3.0 Å away.

- The volume of the cluster as obtained from the QHull convex hull algorithm library [253]. The convex hull algorithm returns the smallest convex polyhedron that encompasses a collection of points, which in this case are the positions of the cluster atoms. From this one can derive the properties of a polyhedron such as volume and surface area, but also other properties, for instance if an arbitrary point is inside or outside the constructed polyhedron.

- The area of the atoms at the interface, also using QHull. Interface atoms are chosen following the definition in sec. 5.2.2, i.e., within 1.6 Å vertical distance of the lowest cluster atom.

The values of these properties are plotted in fig. 5.4, showing the values for the GM, and the average value for the GM plus the first 5 LM, with error bars indicating the minimum and maximum values of the property within this group.

The average coordination number of the clusters (a) does not reach its saturation value, since all cluster sizes explored are smaller than the fcc coordination sphere of 12 atoms. Cluster height (b) for the GM shows no clear trend with cluster size, but the average across the GM and first few LM does exhibit an increasing trend. Average nearest neighbor distance (c) is well below the bulk value of 2.55 Å, but as expected trends upwards as cluster size increases. The cluster distance to the COG (d) is the only property where all GM exhibit the same behavior, being at the average or below which indicates a more compact shape. Interface area per atom (e) initially grows rapidly but becomes more or less constant starting at Cu_6 and Cu_7, indicating that extra atoms go to grow the cluster in height instead of expanding the area with the support. This could be related to a requirement for a minimum number of atoms in order to gap over the distances between oxygen atoms in the ZnO support, and sudden jumps in interface area could be expected for clusters beyond Cu_{10} at specific sizes. The interface area to volume of the cluster ratio (f), remains rather constant except for Cu_4.

Figs. 5.5 and 5.6 show the clusters colored by the per atom fractional coordination number and the shift from the bulk nearest neighbor distance, respectively. The first fully surrounded Cu atom with 8 other nearest neighbor Cu atoms at the center of the GM of Cu_9. This is of course the first size that can present such a coordination, but for this cluster size it also happens to be the GM. The only other fully coordinated cluster appears as a LM for Cu_{10}. Although it would be expected for fully coordinated structures to be energetically favored, this does not seem to be the case at these small cluster sizes. As for the nearest neighbor distances, only two clusters present positive deviations from the bulk value, LM 2 of Cu_4 and LM 3 of Cu_9.

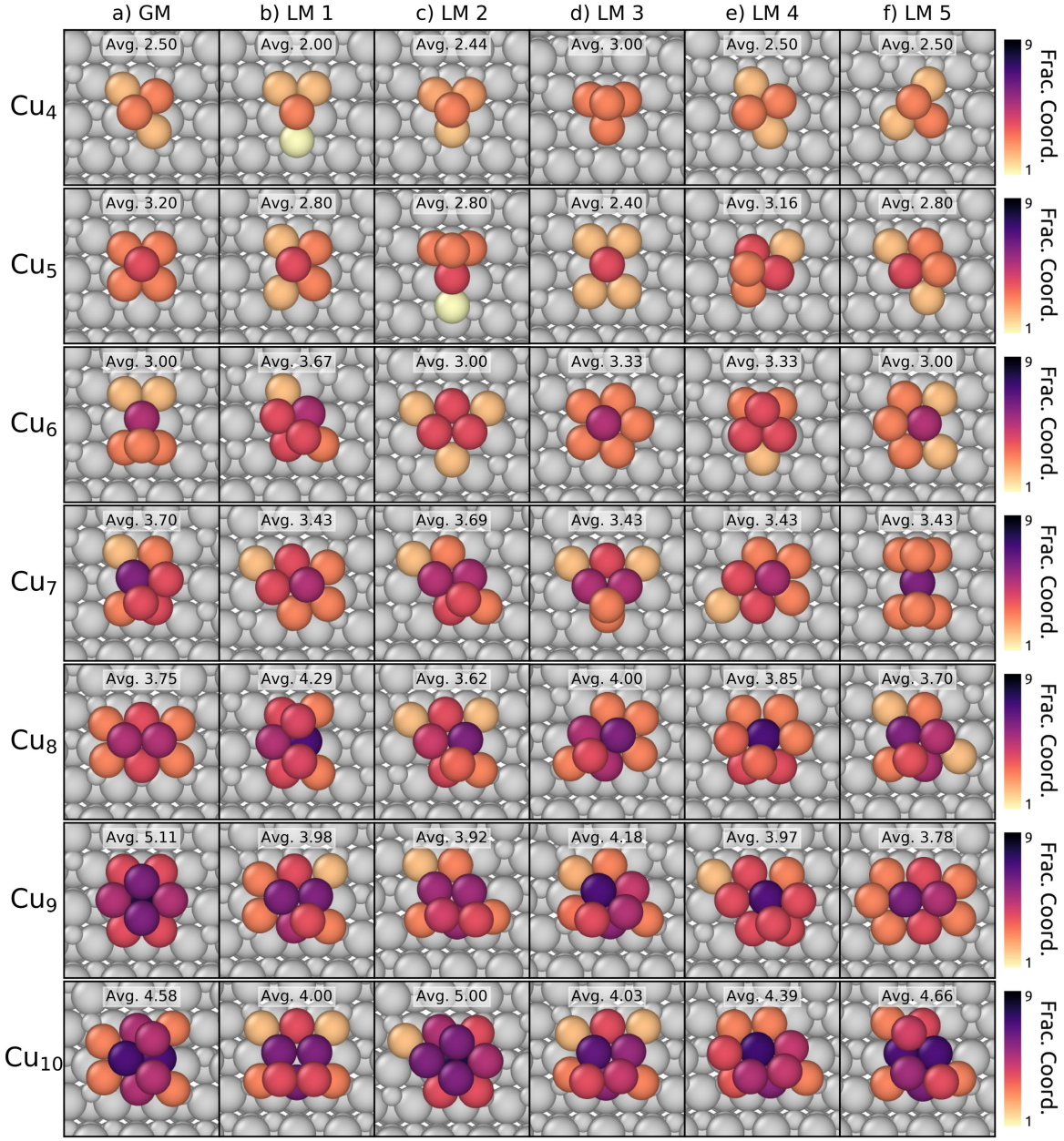

Figure 5.5: Minima from Fig. 5.1 colored by their fractional coordination number. Average coordination numbers are given as insets. Reproduced from Reference [138], with the permission of AIP Publishing.

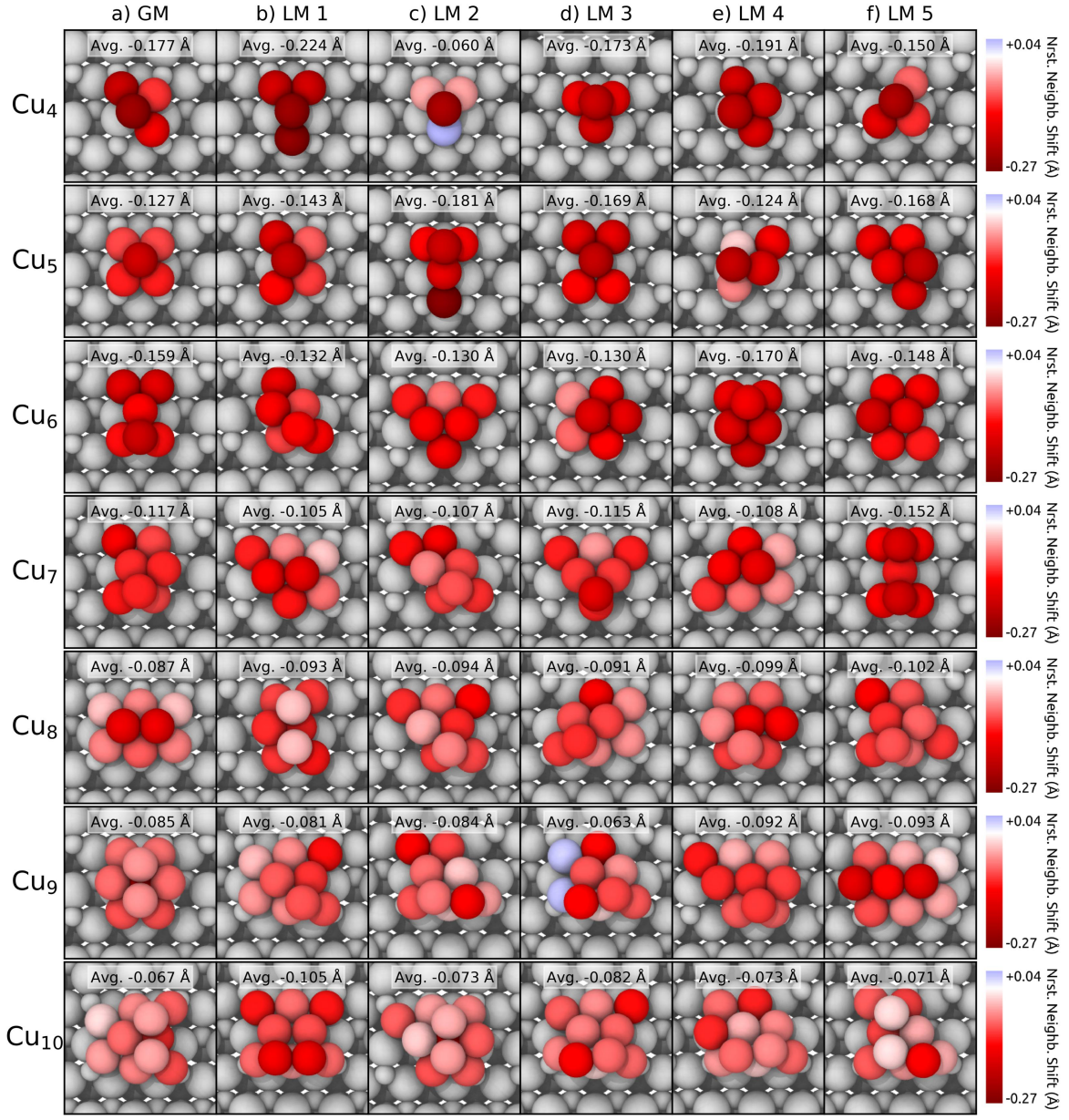

Figure 5.6: Minima from Fig. 5.1 colored by their shift from the bulk Cu nearest neighbor distance (2.55 Å), with average shift indicated in the insets. Red indicates atoms that have a negative shift from this distance, that is, atoms that are closer together than in the bulk. Blue (only visible in Cu_4 c and Cu_9 d) indicates the opposite. White indicates no shift. Reproduced from Reference [138], with the permission of AIP Publishing.

5.3 Conclusions

A successful genetic algorithm global optimization search on the most stable structures of copper clusters with 4 to 10 atoms on the $(10\bar{1}0)$ surface of zinc oxide has been performed, making use of a high-dimensional neural network potential. The search can be performed at a fraction of the computational cost of electronic structure based searches, and with the presence and concurrent relaxation of very large surface cells which are vital to obtaining the correct results [138]. A number of consistently present structural families can be detected across the cluster sizes The energy differences within a given cluster size are often very small, and thus clusters are expected (and in fact are observed during MD mutation simulations) to interconvert at moderate temperatures.

Of particular interest is the interface between both materials. It is found that the clusters favor interacting with the oxygen atoms of the ZnO surface. This, plus the competition between cluster-cluster and cluster-interface interactions restricts the possible footprints and shapes of the clusters. The interface itself adopts configurations in the continuum between Cu(111) and (110) structures. This kind of strong dependence between cluster morphology and support properties is already well known in the literature. Studies are available both for theoretical coarse models [241, 246], and more specific atomistic simulations and experimental results [243–245], and is known as strong metal-support interaction.

The presented tools and algorithms could in principle be utilized to expand the analysis to larger clusters. The available literature presents many examples of global optimizations with clusters containing up to hundreds of atoms, but this is usually performed with either no support or with a very simple potential that is fast to calculate. Personal experience has shown that it should be possible to optimize supported clusters of up to 20 to 30 atoms, which is still beyond the size range usually accomplished with electronic structure calculations. As system size increases, the local minima landscape of the clusters becomes more complex, and minimizations also become more computationally expensive. Each generated GA candidate samples a smaller portion of this landscape while also consuming more time. Eventually the combination of these two effects means that the GA approach becomes inefficient, and other algorithms with some sort of dimensionality reduction (such as the use of grids [199]) might perform better. More interesting than increasing cluster size is the possibility of optimizing binary brass and ternary Cu-Zn-O clusters which can be done without increasing the size of the clusters. With this, the distribution of atoms in the brass alloy could be studied [61, 92], as well as the possible formation of disordered ZnO phases enveloping the Cu clusters.

Chapter 6

Simulated Annealing of Large Supported Copper Clusters

6.1 Motivation

Although global optimization of small deposited Cu clusters reveals interesting information about the structures between the two materials at the small scale, and enables further simulations such as testing of adsorbate sites, it is hard to compare with experimental results and to extract trends for the larger scale. Deposited copper clusters rapidly and naturally coalesce into larger clusters [18] in a process known as Ostwald ripening, and it does not seem possible to study such small, size-selected clusters deposited on a support. This is a trend with many materials and obtaining small clusters is a challenge more often than not [254], exceptions being those cases in which the clusters can be trapped somehow [255, 256], or when clusters are previously generated in vacuum where they can be size selected and then deposited on the target substrate [257, 258].

Test molecular dynamics simulations also show a tendency for the supported clusters to coalesce into larger clusters: if a large ZnO slab is sparsely populated with Cu atoms and the system is given enough kinetic energy for atoms to be mobile, the clusters quickly combine until a number of large clusters are present and no single atoms or clusters below 5-10 atoms are present. Then the evolution of the system stops, since the process of further coalescing (be it through single atom detachment and attachment, or whole cluster migration) is extremely slow at the MD scale. Simulation methodologies exist that attempt to progress the Ostwald ripening process through kinetic Monte Carlo [259], coarse grain modeling [260], or grand canonical ensemble Monte Carlo [261]; all of which present their own technical and implementation difficulties.

Larger copper clusters are easier to compare with experiments, but on the other hand due to the increased degrees of freedom, it is hard to find the structure of a putative global minimum as the presumed most relevant structure. Global optimization techniques such as genetic algorithms have been used for clusters with hundreds of atoms [145, 262–264], but they usually make use of very simple, fast potentials, and/or only optimize the clusters in a vacuum without the added computational effort of an interface. From preliminary simulations and test GA searches with the current potential, 20 to 30 Cu atoms seems to be the limit for this system where "reasonably" shaped clusters can be observed. Beyond this size, the GA approach appears to become inefficient and generates only partially optimized clusters.

Another approach would be a grid-based Monte Carlo search [240], but as observed in sec. 7 with mismatched interfaces and as will be shown later, a fixed fcc grid would fail due to the changes induced in the cluster by the support (or rather, would lead to an artificial minimum that is constrained by the proposed atomic grid). Flexible grid approaches [199, 265, 266] would fare better in this regard.

The question arises whether the global minimum at this scale is even actually relevant: as size increases, so does the amount of structures with very similar energy levels but different atomic configurations. That is, at larger numbers of atoms, the configuration space (entropy) wins over the energy differences present between structures (energy/enthalpy). At this scale, the properties of the system are better described as an average of the collective ensemble of structures accessible at a given temperature. Now the problem turns from finding the global minimum to generating this set of structures. At

higher temperatures, past the melting point of the system, these structures are easy to obtain since the system has enough kinetic energy to sample all possible configurations in the timeframe allowed by an MD simulation. The problem is present below the melting point, where a simple MD will result in a solid structure (crystalline or amorphous) where atoms just vibrate in their equilibrium positions, with rare events where an atom might jump a lattice position.

Monte Carlo [156] is a possible solution to this new problem, but presents problems for condensed phases. For such a system, normal displacement Monte Carlo results in most atoms either jumping into the vacuum or into the repulsive zone in the vicinity of other atoms, in which case most of the trial moves are rejected and computational time is wasted. Basin hopping Monte Carlo [149] performs a geometry minimization after every trial move, which increases the acceptance ratio at the cost of increased computational cost, but would still not solve situations where an atom from the cluster tries to jump inside the support, for example. Another approach would be to simulate the equilibrium between the cluster and a gas reservoir with grand canonical Monte Carlo [156]. This has been successfully utilized in the past to study the surface equilibrium for solids such as ice with an appreciable quasi-liquid surface layer [267], but for a metallic system such as copper where such layer is practically non existent, it leads to unstable equilibrium simulations. According to preliminary simulations, a very low partial reservoir pressure/reservoir chemical potential needs to be utilized which can lead to numerical stability problems; and the equilibrium is unstable: for liquid systems a range of reservoir potentials can be found where the system neither grows nor shrinks in size [268], that is, the liquid and the reservoir are in equilibrium. For copper it has not been possible to find this point, with the system either filling completely or evaporating rapidly.

A simpler solution is to employ simulated annealing [152]: the system is given kinetic energy past its melting temperature and then slowly cooled down past its glass point, where no more reconstructions of the atomic structure can take place. The high temperature allows the cluster to reconstruct by jumping over the potential energy barriers present between atomic positions, while the slow cooling down allows the cluster to recrystallize slowly, into lower energy configurations. In this way, instead of simulating a long time at a low temperature hoping to observe structural changes, time is "accelerated" by providing kinetic energy, and then cooling down the system to observe the results at the desired lower temperature. If we perform this operation multiple times, we obtain a collection of cluster structures at low temperatures, from which we can attempt to extract general trends and properties. Since the simulated annealing *ansatz* is just a molecular dynamics simulation in the end, we can also make use of the usual MD analysis tools, which would not be possible in other algorithms such as MC where the trajectory of the cluster is broken up.

6.2 Results

6.2.1 Energy

Since the stated goal of simulated annealing is to find low energy configurations, it is of interest to look at the potential energy behavior of the clusters. Figure 6.1 a) shows the potential energy profile for one cluster of each size between Cu_{200} and Cu_{500} every 50 Cu atoms, during a simulated annealing run sweeping a temperature between 1400 K and 300 K. In each case, the energy of a relaxed ZnO slab of the same size has been subtracted from the absolute energy value, and then divided by the number of atoms in the Cu cluster. The black bars mark where the target simulation temperature changes, which are accompanied by potential energy drops. Notice that as cluster size increases (darker lines), the temperature fluctuations as well as the difference in potential energy steps becomes smaller. Subfigure b) shows the result for the 5 clusters at each size, after a geometry minimization has taken place. The resulting energies for each cluster size are clustered together, which shows that the SA has managed to optimize all the clusters of a given size to the same degree. As expected, the energy differences between cluster sizes decrease as the cluster sizes increase.

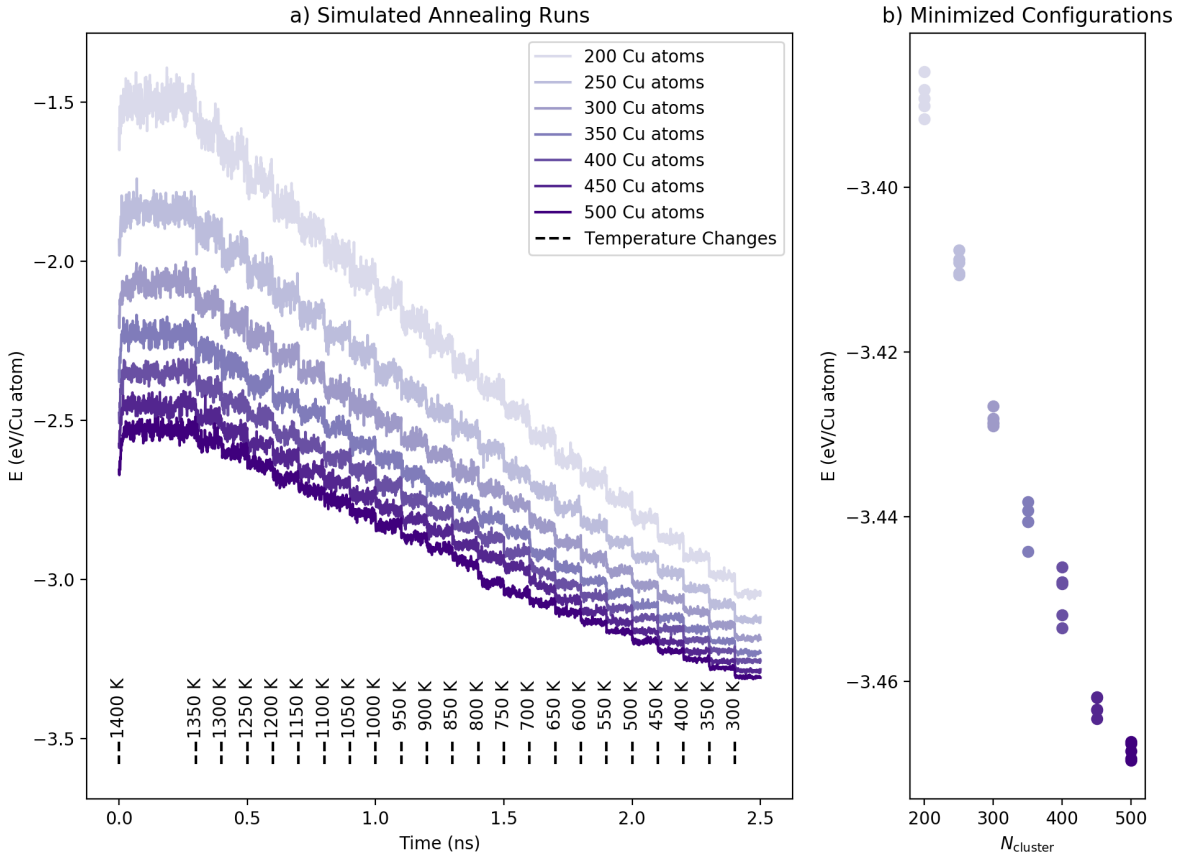

Figure 6.1: a) Potential energy profile of one selected cluster per cluster size during the SA run. The black bars mark where a target temperature change has taken place. b) Energy distribution vs. cluster size for the final, geometry minimized configurations of clusters between Cu_{200} and Cu_{500}, with 4 examples per cluster size.

As mentioned in sec. 2.6.3, melting points can sometimes be determined by discontinuities in the potential energy. Notice that no real potential energy discontinuity is visible in this case, except for maybe towards the middle of the Cu_{500} curve, and the steps seem to uniformly decrease with each temperature change. Even if such a drop where present, it would be hard to distinguish it from the fluctuations present in the energy values. This shows the problem with relying on potential energy to detect first order transitions in this kind of simulation.

6.2.2 Average Structural Property Plots

Figures 6.2 through 6.5 plot a number of average structural properties within each cluster's SA run. For each plot, one data point represents the average of that properties across all atoms in the cluster (or for the whole cluster if it is a global property such as its height), and also averaged within the given temperature window. So a point at 500 K averages the given property across all the configurations obtained in the 500 K window.

Figure 6.2 show the average volume and surface area (including the area in contact with the ZnO slab at the interface) per atom of the clusters as obtained with the QHull [253] algorithm. In a), the volume per atom decreases uniformly as temperature decreases, with a sharp decrease between 800 and 600 K, which as will be seen later, is around the melting point of the clusters. As expected, the volume per atom increases non-linearly with cluster size. The surface area of the clusters in c) behaves more erratically above the melting temperature, but stabilizes after 600 K and steadily decreases past

that point. As opposed to the volume, in d) the surface area per atom decreases as the cluster increases. That is, the clusters roughly maintain their shape (volume to area ratio) as they grow in size. As can be seen in b) and d), the spread of the properties within a given cluster size is large, when compared to a similar spread for the energy. This can mean that either the SA procedure has not properly optimized the clusters and as such many different structures are obtained, or that there is a large fluctuation in structural patterns at these sizes, in which case many more simulations (e.g. 100) at each cluster size are required for a proper ensemble average.

Figure 6.3 shows some simpler global cluster structure parameters. In a) and b), the height of the cluster is plotted, estimated as the maximum difference in Z position components between atoms in the cluster. In c) and d) the diameter of the cluster is plotted, estimated as the maximum distance between cluster atoms in the X-Y plane. Finally, in e) and f), the effective radius [160] is plotted. The effective radius is useful in some cases where the contact angle between the cluster and the substrate is extreme, so that the measured and actual radius of the particle do not coincide. In this case, $R_{\text{effective}}$ and D are close, so the effective radius is not necessary. All 3 properties behave in a similar way, fluctuating somewhat above 600 K and then adopting a constant value below this temperature. All three increase non-linearly with cluster size, and exhibit also a large dispersion of values.

Figure 6.4 shows the average of per atom properties, the fractional coordination number (as explained in 138) in a) and b), and the nearest neighbor distance (as explained in 138). Both properties exhibit distinct behaviors around the melting point: the fractional coordination number increases uniformly as temperature decreases, but presents a jump at around 800 K. The nearest neighbor distance remains constant at around 2.65 Å above 800 K, but then sharply decreases past this point. This shows why the Lindemann parameter, although it depends only on the structural parameter of interatomic distances, is such a good estimator of the melting point. When plotted against cluster size for the minimized clusters, the coordination number steadily increases with cluster size, while the nearest neighbor distance seems to already have stabilized close to the bulk values (approx. 2.55 Å) for the 200 atom cluster.

Finally, fig 6.5 shows the behavior of the interface between the cluster and the support. Interface atoms are defined as those within 1.6 Å of the lowest Cu atom of the cluster. In a) and b), the proportion of interface to whole cluster atoms is plotted. The proportion fluctuates at higher temperatures, and it is larger there than for lower temperatures. For lower temperatures, the proportion slowly decreases, and for the minimized configurations, larger clusters exhibit a similar ratio of 0.10 to 0.15. Subfigures c) and d) show the interface area per interface atom, as obtained with QHull. The interface area per atom presents large fluctuations within each cluster size. Subfigures e) and f) show the proportion between interface area and surface area of the cluster. For all sizes this remains at around 0.34.

All of these plots illustrate the complexity of the liquid to solid transition for the supported clusters. In particular, it can be appreciated that there is quite the spread in the behavior of different cluster even for the same size category, but the expected rough trends with system size are maintained. This makes it evident that at this system size multiple clusters are needed to achieve a converged average property, and the 5 clusters per size presented here might not even be enough. Thanks to the NNP, it should be possible to obtain up to one hundred sampled clusters per size, which would allow for a comparison of properties with experimentally observed clusters.

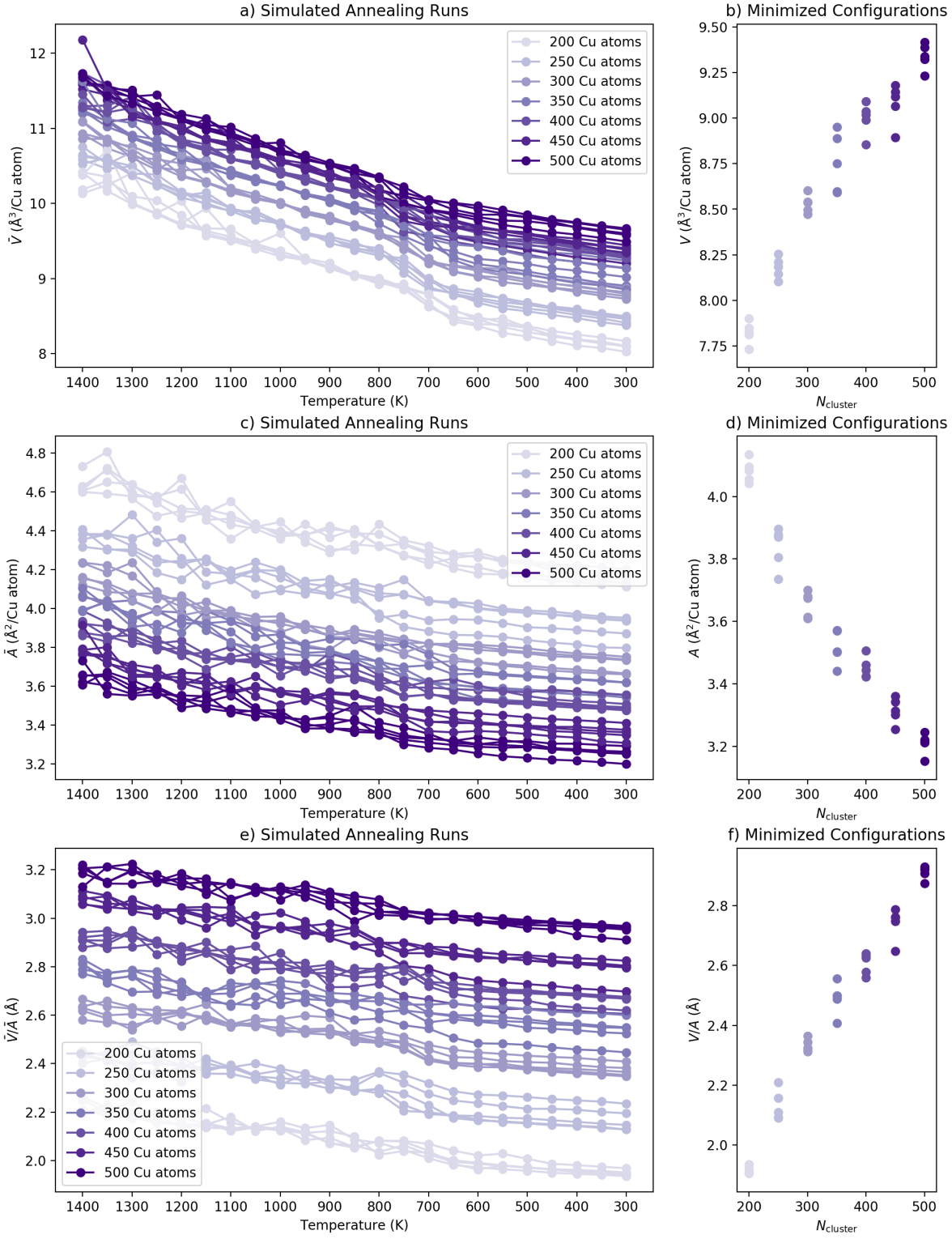

Figure 6.2: Subfigures on the left show properties averaged in each temperature window plotted against temperature, while subfigures on the right show those properties for the minimized clusters plotted against cluster size. a) and b) volume per atom as obtained with QHull. c) and d) surface area per atom, as calculated also with QHull.

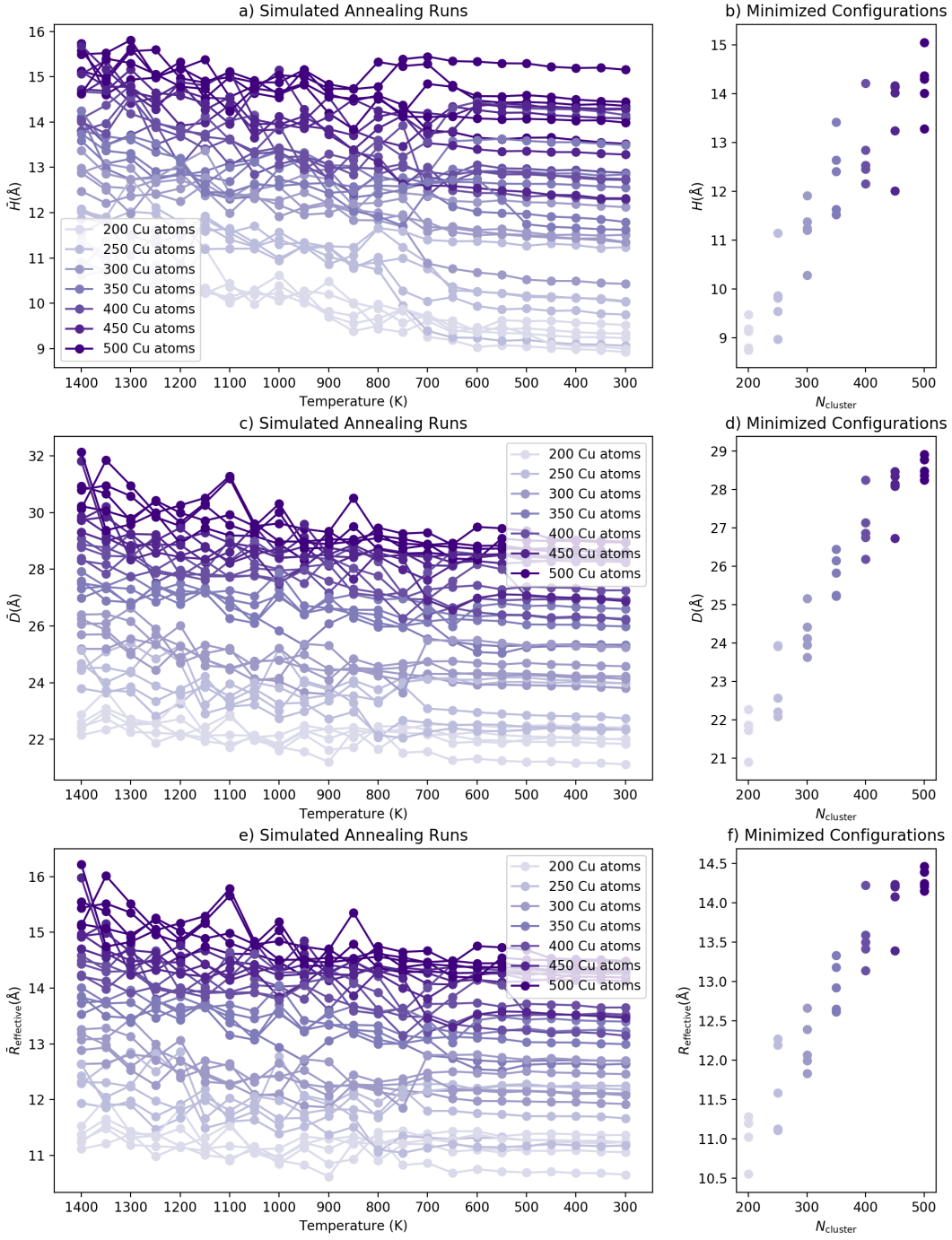

Figure 6.3: Subfigures on the left show properties averaged in each temperature window plotted against temperature, while subfigures on the right show those properties for the minimized clusters plotted against cluster size. a) and b) height of the cluster. c) and d) diameter of the cluster. e) and f) effective radius of the cluster, a function of the height and diameter.

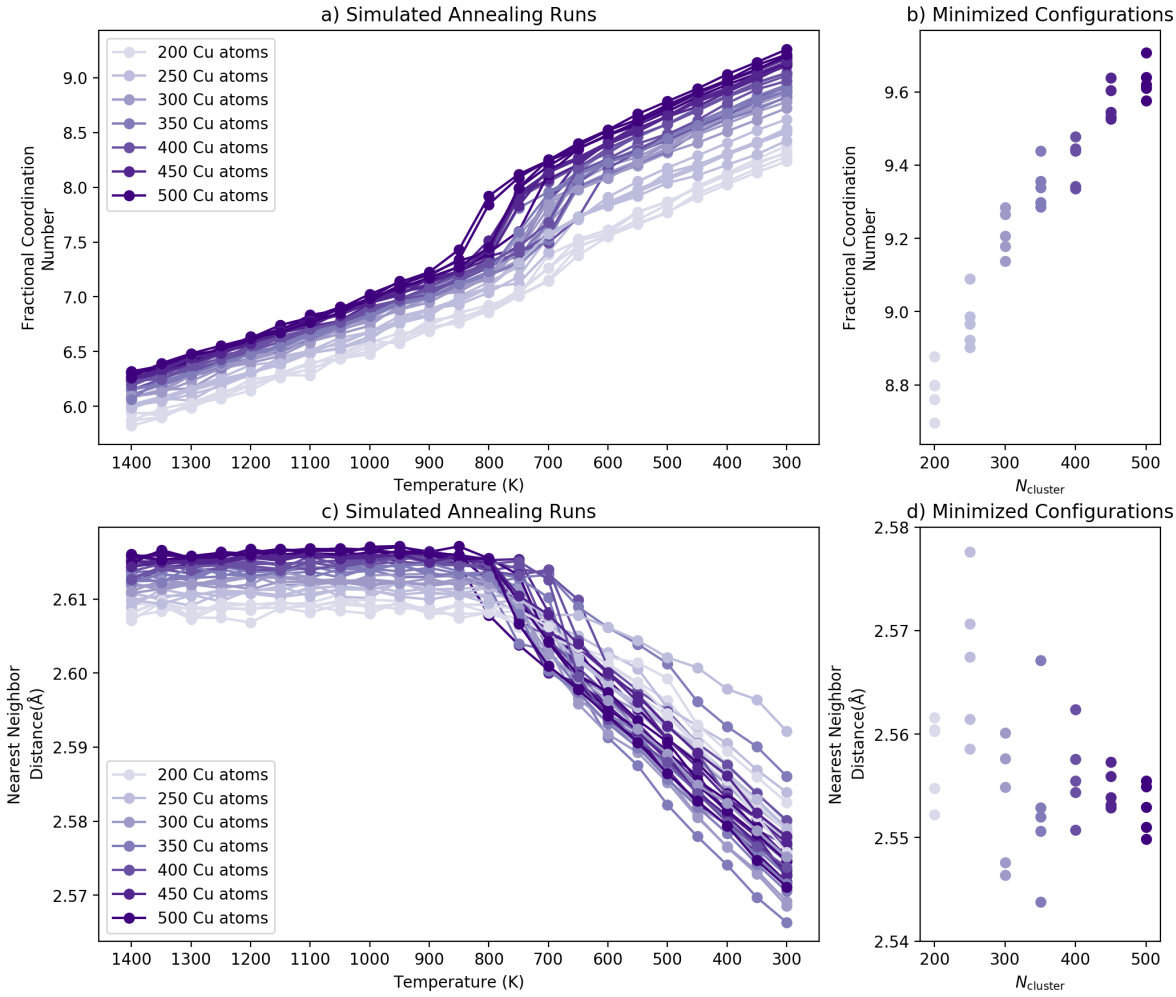

Figure 6.4: Subfigures on the left show properties averaged in each temperature window plotted against temperature, while subfigures on the right show those properties for the minimized clusters plotted against cluster size. a) and b) fractional coordination number. c) and d) nearest neighbor distance.

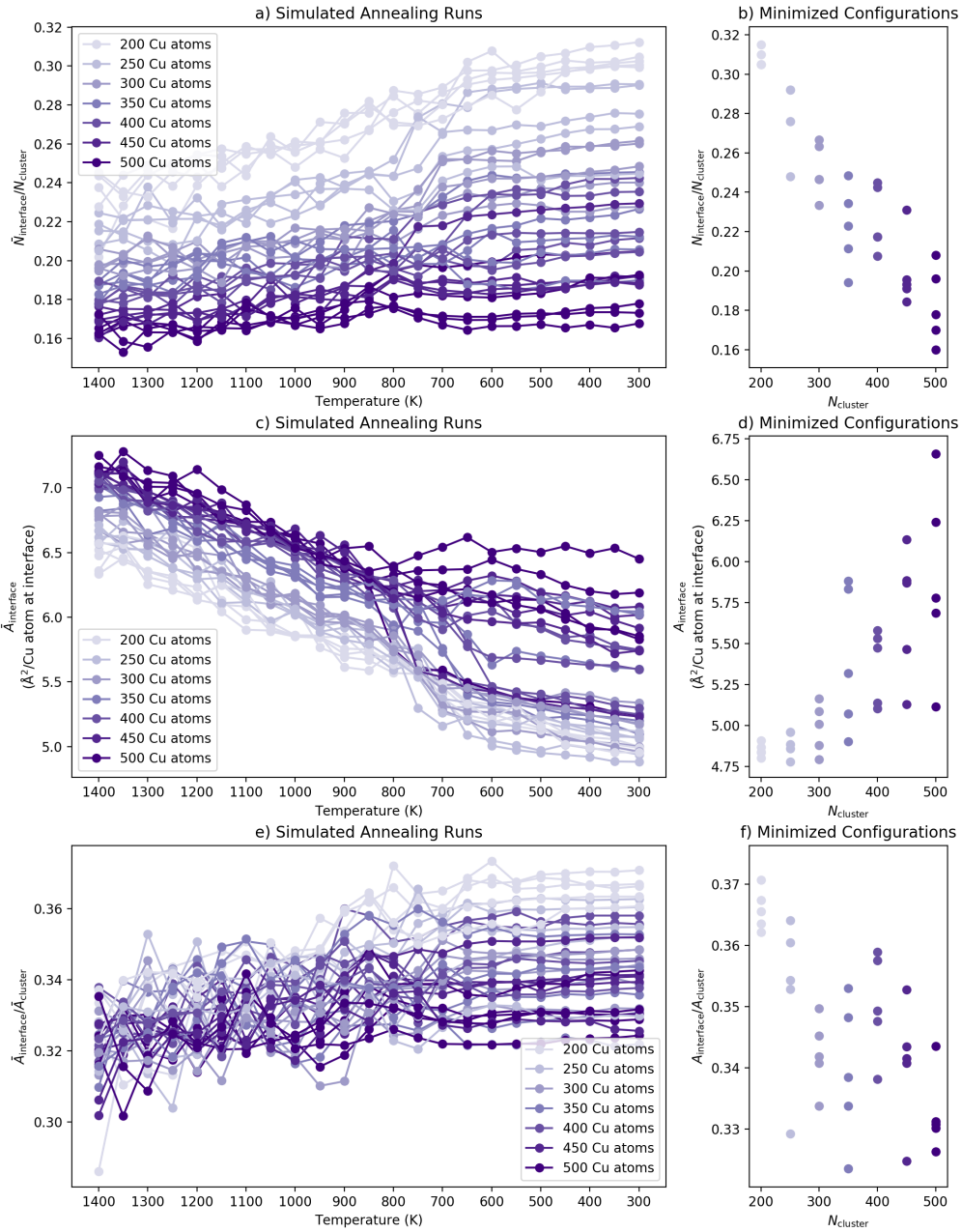

Figure 6.5: Subfigures on the left show properties averaged in each temperature window plotted against temperature, while subfigures on the right show those properties for the minimized clusters plotted against cluster size. a) and b) Proportion of interface to total cluster atoms. c) and d) interface area per interface atom. e) and f) proportion between interface area and the whole surface area of the cluster.

6.2.3 Structure of the Clusters

Figures 6.6 and 6.7 show the structure of the obtained minimized clusters, from the top and with a side cut view respectively. For convenience, the clusters have been sorted according to energy, but as shown in a previous section, the energy differences in this case are very small. The clusters in general look similar, exhibiting facets of all three low Miller index Cu interfaces, and with a vaguely hexagonal footprint on the substrate, particularly noticeable for Cu_{450} a). Two orientations are present between the clusters and the support, the cluster can be either parallel to the oxygen rows in the ZnO slab (Cu_{450} a)), or at a slight angle (Cu_{450} b)). As will be seen later, these patterns also repeat if only the interface atoms are observed. From the sliced clusters, we can see how the Cu atoms have arranged themselves into an ordered pattern.

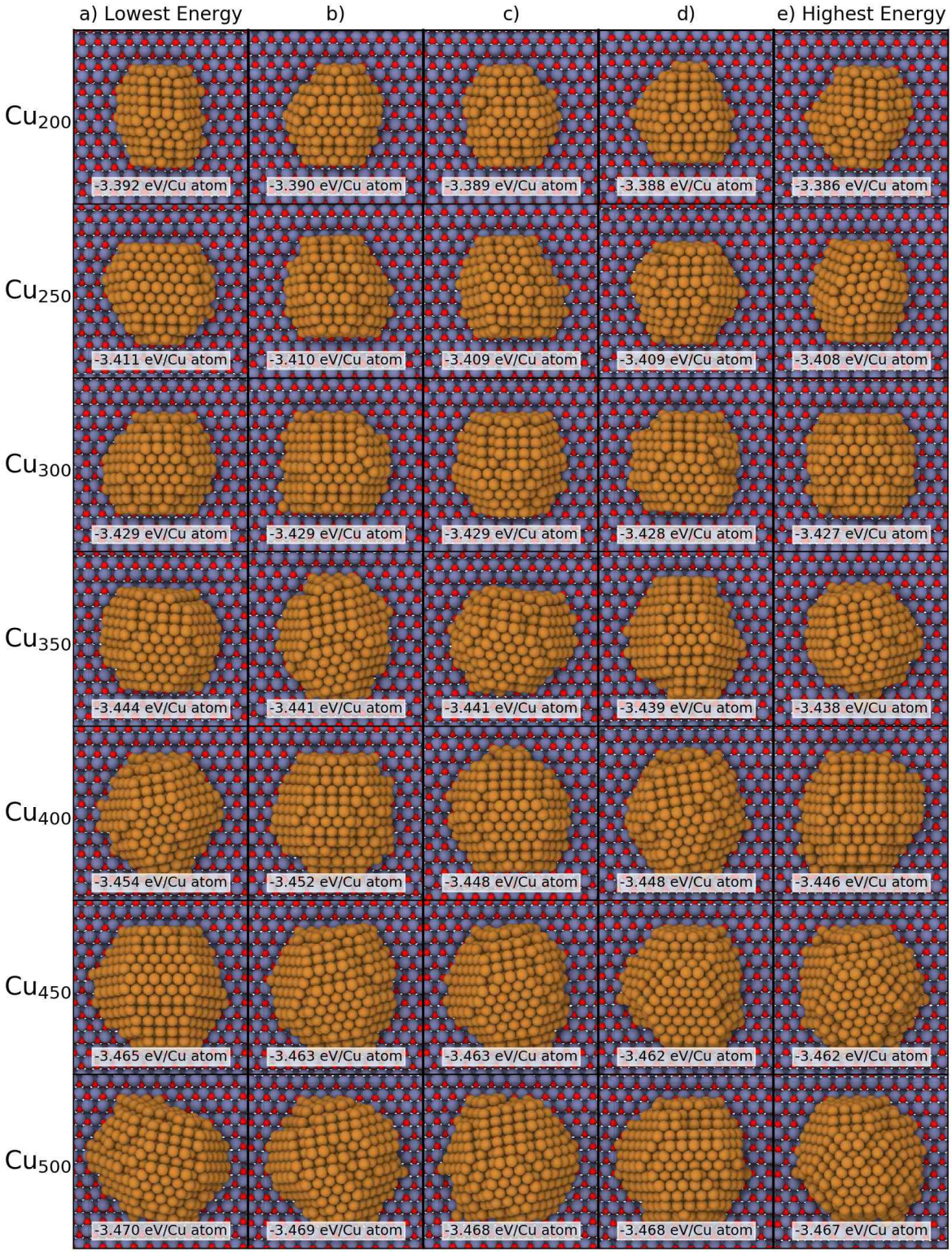

Figure 6.6: Top view of all the generated clusters for each cluster size, sorted by energy from left to right.

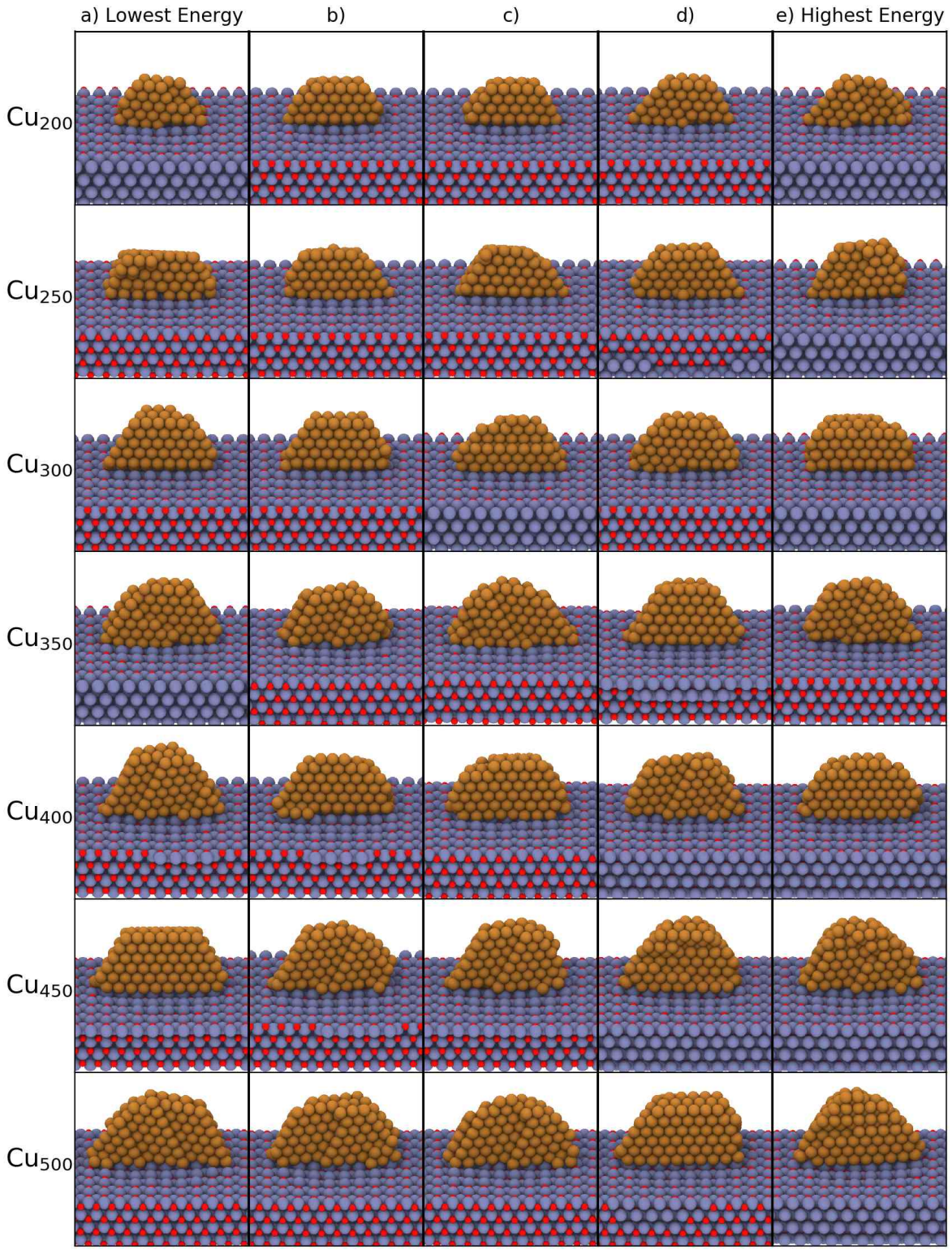

Figure 6.7: Clusters from fig. 6.6 are presented cut in half and from a side view, to highlight the inner structure and the interaction at the interface.

6.2.4 Polyhedral Template Matching

Using the PHTM algorithm as explained in sec. 2.6.4, it can be confirmed that the clusters have crystallized into the appropriate fcc configuration. Figure 6.8 shows the results of the PHTM analysis from a top view. The analysis has been performed taking into account only the cluster atoms. Atoms on the surface of the cluster, which cannot be properly classified by PHTM since their coordination sphere is not complete, have been colored white and their size reduced to make it possible to see inside the cluster.

The clusters here exhibit a similar crystallization pattern to that observed for unsupported clusters in the vacuum [61, 92]. Most of the cluster is fcc (green color), with atom planes colored as hcp indicating planes where a stacking fault has taken place. These hcp planes seem to be present in two types: perpendicular to the ZnO support (clearest examples is in Cu_{450} a)) which are also more abundant; and planes parallel to the cluster layers and at an angle with the support, such as at the top of Cu_{450} d). Also in some cases, particularly for Cu_{250}, some bcc (body centered cubic) atoms seem to be detected. Notice that these sections also coincide with positive deviations from the bulk nearest neighbor distance (see fig. 6.11), instead of the usually expected negative deviations for surface atoms. This can be either due to a problem with the PHTM algorithm (since these only appear at the edges of clusters, where interatomic distances are different from those in the core of the nanoparticle) or with the SA procedure (which has not been able to optimize these atoms into a better stacking). In contrast with the free standing clusters [61, 92], no hcp patterns radiating from the center of the cluster in an icosahedral shape appear. The presence of hcp stacked layers also seems to roughly coincide with higher energy for the cluster.

The PHTM results can also be utilized to track the process of cluster crystallization. Figure 6.9 shows top and side views of a Cu_{500} cluster at a cool down step with a target temperature of 850 K, which is right below its melting temperature. Fcc structures develop rapidly, radiating from the bottom and center of the cluster. Initially (a) and d)), these structures are small and quickly appear and melt completely away. Eventually (b) and e)), their size starts increasing, but they also fluctuate in extent. Finally (c) and f)), the fcc structure becomes permanent, and in subsequent frames grows to fill the whole cluster.

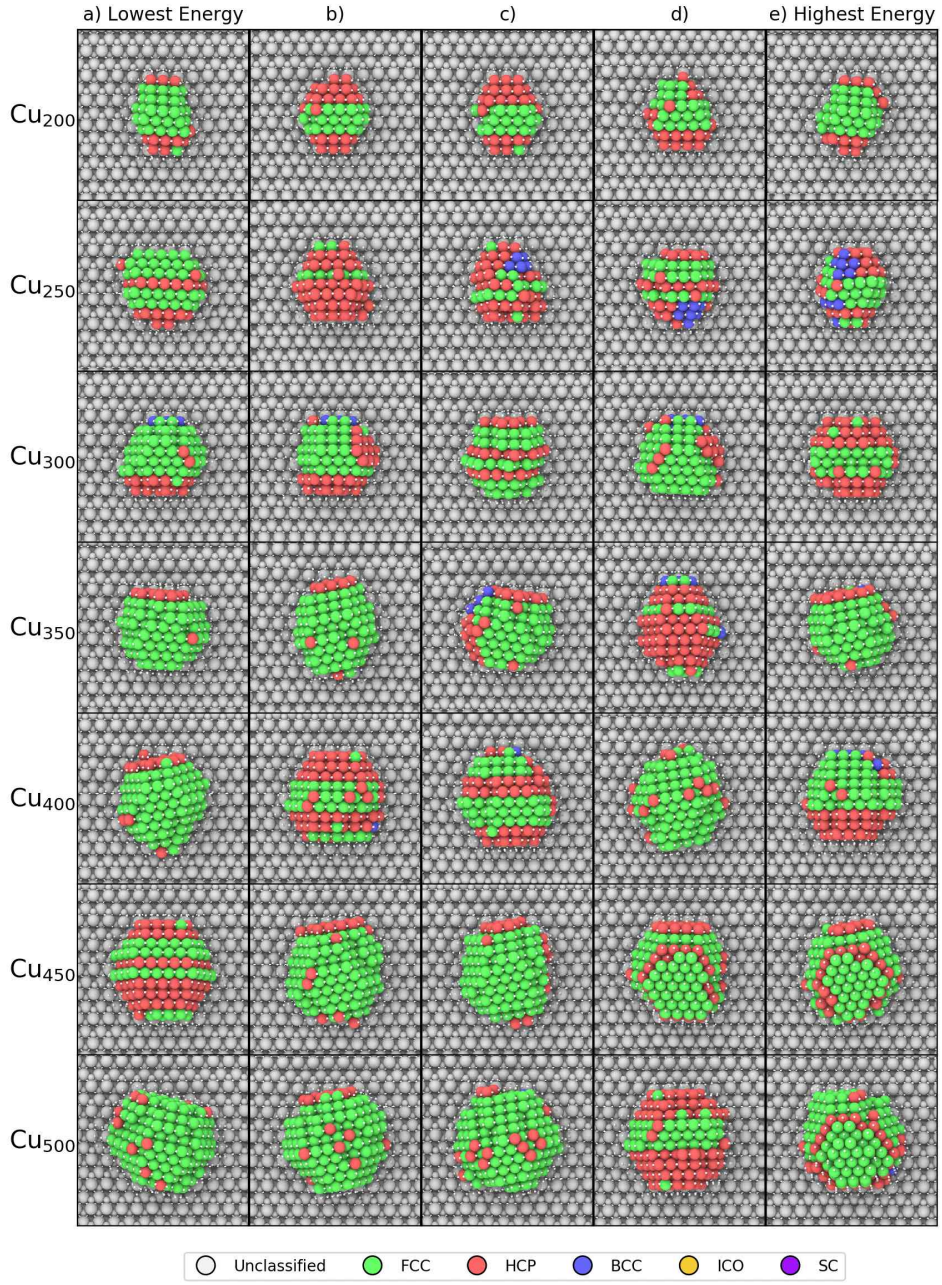

Figure 6.8: Top view of the clusters, colored by their PHTM classification. Atoms on the surface of the cluster are not classifiable, and they have been rendered small for better visualization. Legend: Unclassified: Not possible to classify, belonging to the surface of the cluster or to the support (white). FCC: Face centered cubic (green). HCP: Hexagonal close packed (red). BCC: Body centered cubic (blue). ICO: Icosahedral (purple). SC: Simple cubic (yellow).

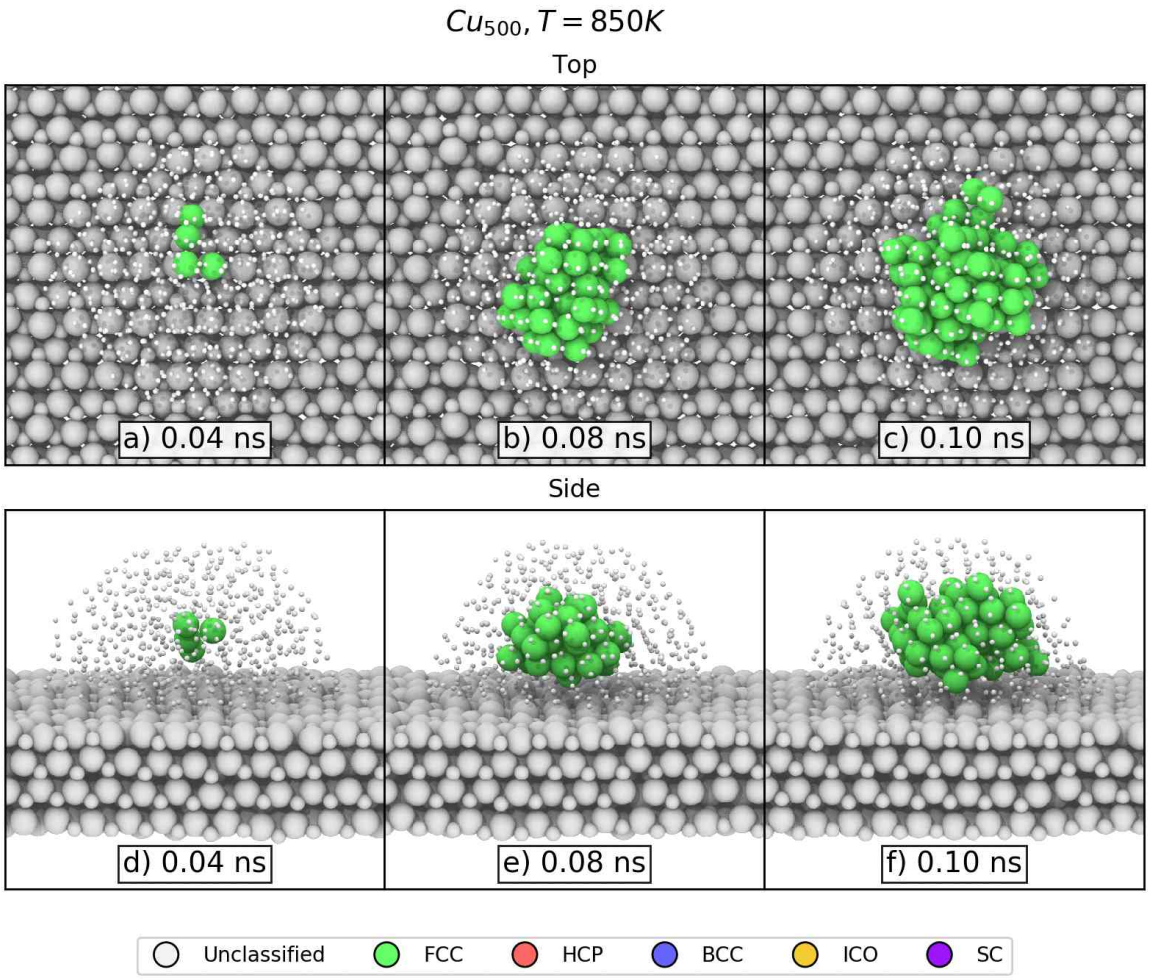

Figure 6.9: PHTM results close to the melting/crystallization temperature, for Cu_{500} at a 850 K target temperature step. Top figures show a top view of the cluster, while the bottom figures show a side view. Timestamps correspond to time since the start of this temperature step. a) and d) At the beginning of the crystallization, fcc areas appear and disappear quickly, at the center and bottom of the cluster. b) and e) As the crystallization progresses, the fcc areas grow in size and become more permanent. c) and f) A permanent fcc core has now formed, that will eventually grow to encompass the whole cluster.

6.2.5 Coordination Numbers and Nearest Neighbor Distances

Figure 6.10 shows the cluster atoms colored by their fractional coordination numbers. The coloring facilitates recognizing facets, edges and loose atoms, although other structural parameters could be used for this such as the centrosymmetry parameter [269] or the orientational order parameter [270].

More interesting is to analyze the changes in nearest neighbor distance in the clusters, as shown in figures 6.11 and 6.12 for top, and side and sliced views respectively. Here atoms are colored according to the nearest neighbor shift with respect to the bulk nearest neighbor distance of 2.55 Å, with red indicating a contraction of this distance and blue a larger average distance than in the bulk. In the top view figures, we can recognize that the atoms belonging to large planar facets are mostly white (close to bulk distances), while the atoms marking the edges of these facets are colored red: since they are under-coordinated, they tend to bind stronger and closer to the few neighbors available. The slice view presents a more revealing image: for the clusters, most of the atoms at the interface are farther apart than in the bulk due to the strain induced by the support, and this disturbance propagates upwards in the layers of the cluster. This is particularly noticeable for the larger clusters.

Figure 6.13 combines both views by showing a layer by layer rendition of Cu_{500} a). Layers have been sampled every 1.65 Å from the surface, atoms in layers below the one being considered are made transparent, and atoms in the current layer have their radius changed to exaggerate their size from the camera view point. As suggested in the previous paragraph, atoms at the interface (a)) have mostly been strained in the positive direction. As we move up the layers, the atoms at the edge of the layer remain in a nearest neighbor distance deficit (as explained before, due to their under-coordination), but towards the center of the layer the opposite is true and atoms present an excess in distance. This effect attenuates as we reach the final not fully exposed layers (f) and g)), where atoms are finally almost at the bulk neighbor distance but still present a small excess. This is remarkable considering that this is the largest cluster size and the layers at f) and g) are 10 Å away from the lowest Cu atom in contact with the support. That is, it takes almost 4 nearest neighbor distances and 6 or 7 layers for the effects of the substrate to disappear.

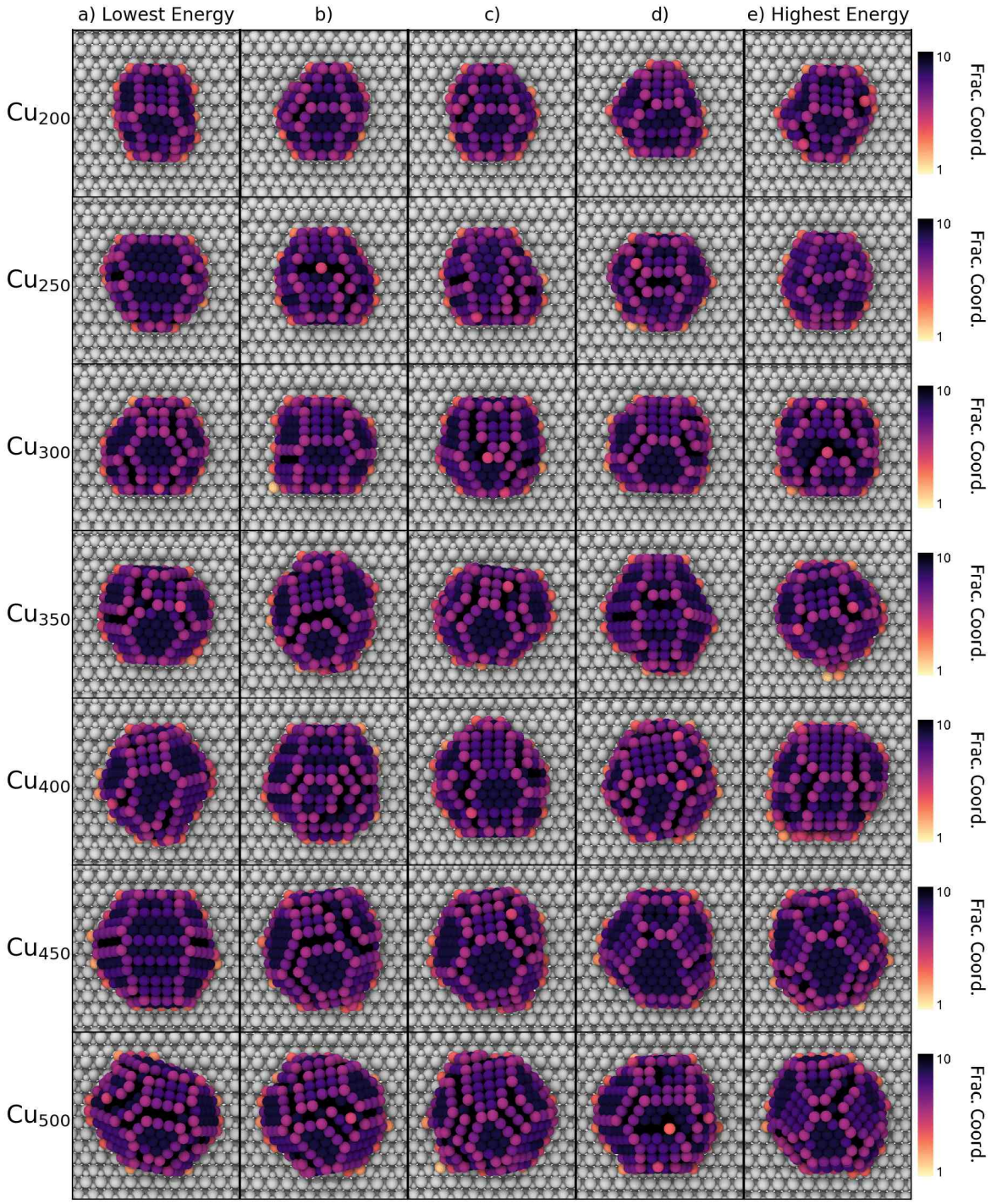

Figure 6.10: Top view of the optimized clusters, colored by coordination number. Coordination number scale ends at 10 to exaggerate the colors at the surface, but atoms inside the cluster are fully coordinated (12). This coloring scheme facilitates recognition of facets and facet edges.

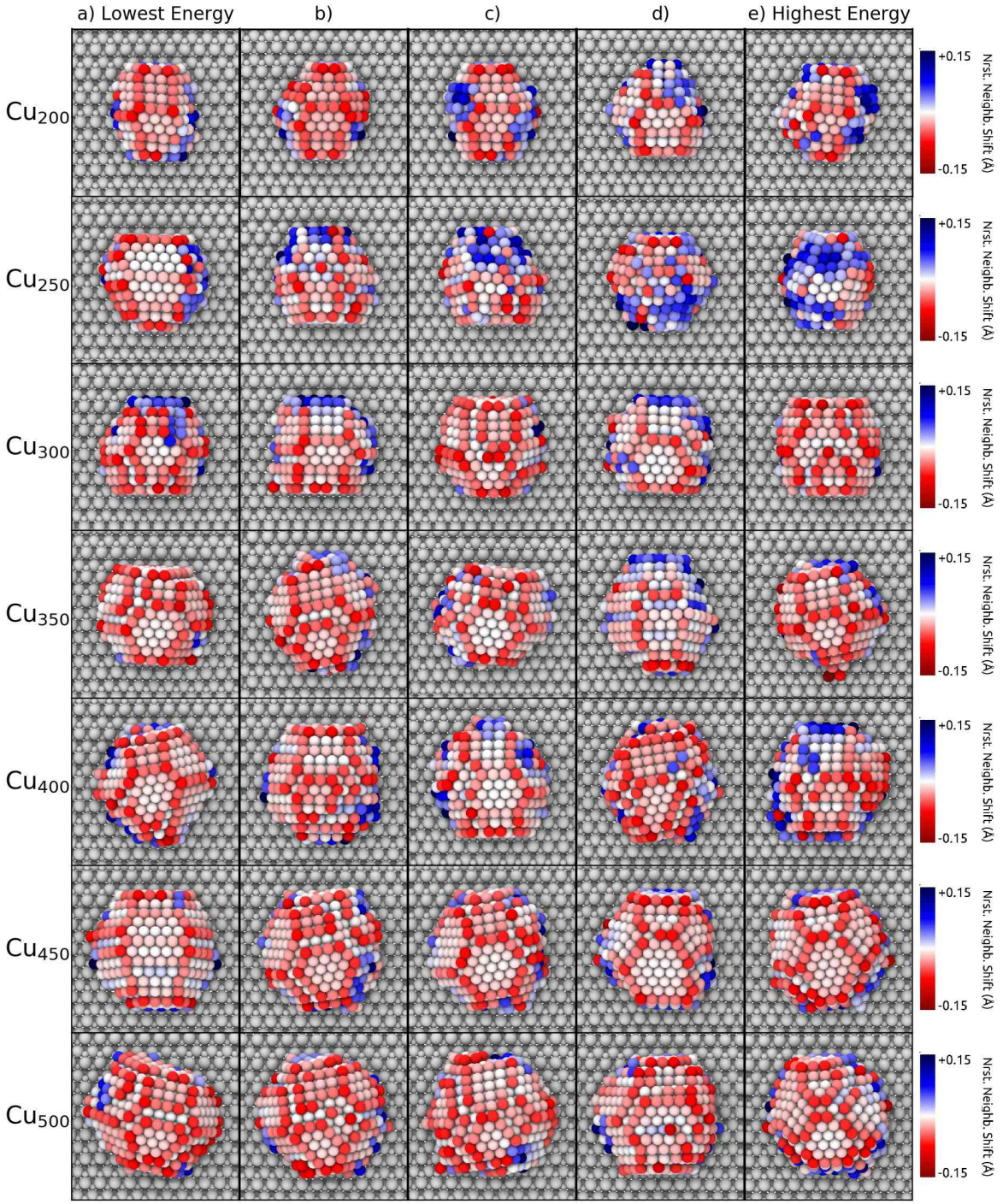

Figure 6.11: Top view of the minimized clusters, colored by the shift from the nearest neighbor distance in bulk copper (2.55 Å). As expected, most surface atoms of the clusters present a negative deviation from the bulk nearest neighbor distance, particularly at facet edges.

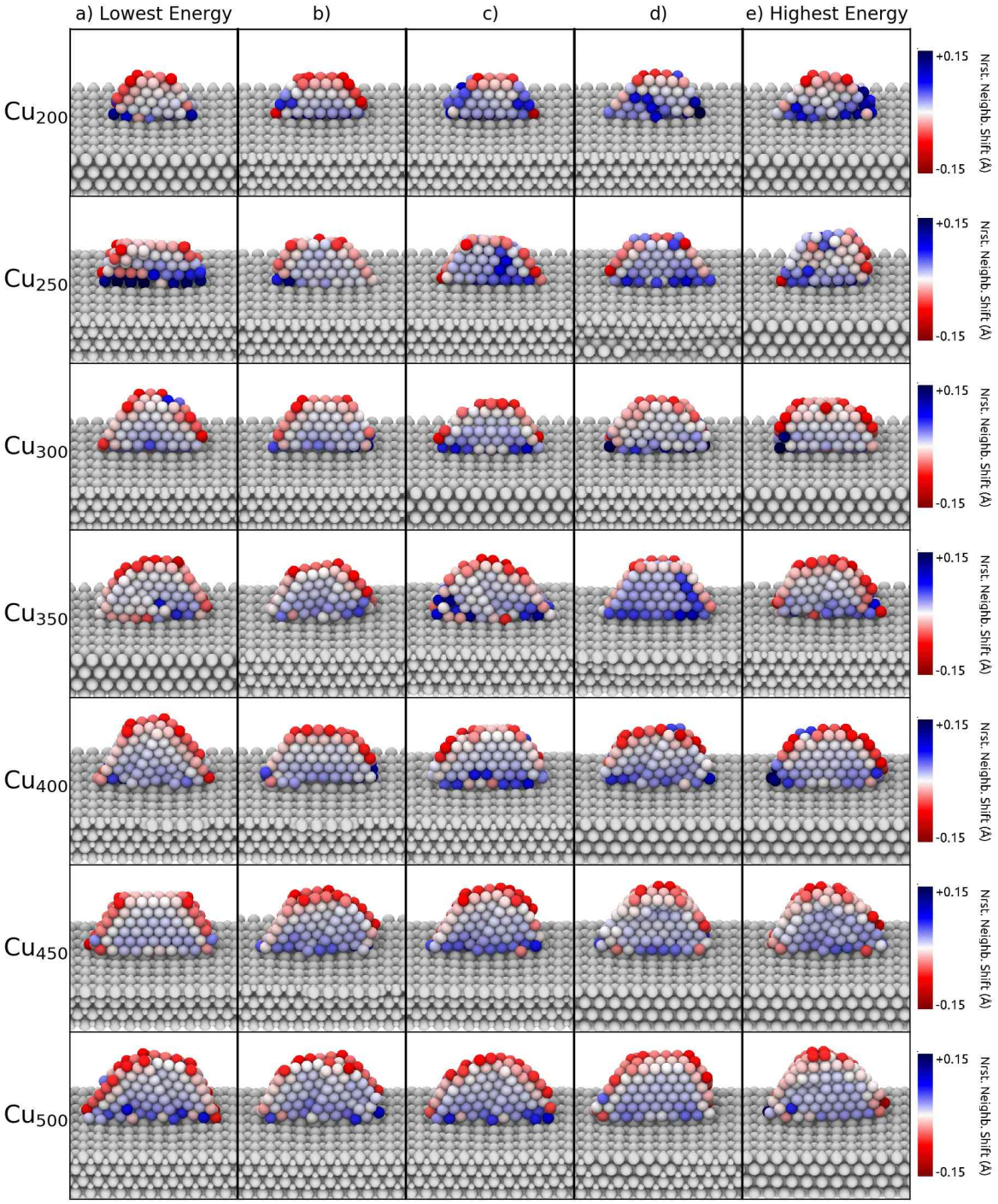

Figure 6.12: Clusters from fig. 6.11 are presented cut in half and from a side view, to highlight the inner structure. Atoms are colored by the shift from the nearest neighbor distance in bulk copper (2.55 Å).

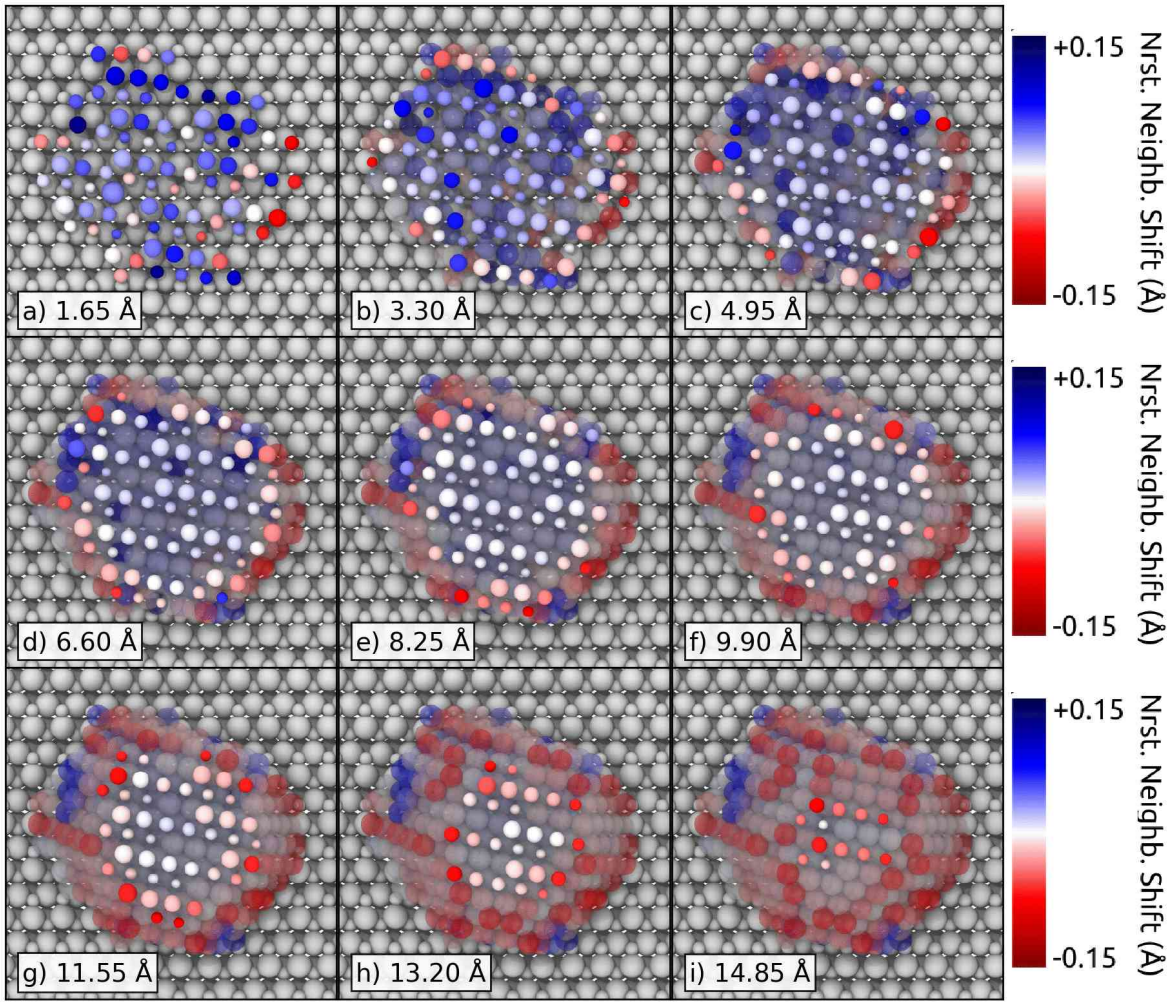

Figure 6.13: Per layer view of Cu_{500} a) in fig. 6.11, colored by the shift from the nearest neighbor distance in the bulk. The caption in each subfigure shows the ceiling height used to define the layer, with the floor corresponding to the lowest Cu atom in the cluster, spaced every 1.65 Å. Atoms in lower layers are made transparent, and atoms in the current layer have their radius changed according to the distance to the view point, to highlight the differences in atom position.

6.2.6 Lindemann Parameter and Melting Point

Figure 6.14 a) shows the Lindemann curve for supported clusters of different size. These curves were obtained by heating up one result from the SA runs at each size, and heating up particularly slowly around the critical melting point in increments of 10 K. It is better to estimate melting points from heating rather than cooling curves due to the possibility of hysteresis [271], which can manifest as overcooling and undershooting of the melting point, that can appear in phase transition simulations of nanoparticles. The Lindemann curves seem to be noisy, and somewhat overlap due to the small difference in atom counts (± 50 atoms), but it can be clearly seen that as expected, the melting temperature increases with cluster size. Longer simulations at each temperature or averaging results from multiple simulations might be required to obtain cleaner Lindemann curves.

These plots were fitted with eq. 2.47, and the inflection point of the sigmoid curves, T_{melt}, thus estimated. Subfigure b) shows the linear fit of an average T_{melt} vs. $N^{-1/3}$ plot. From this it is possible to estimate a melting temperature of 1405 K for infinite nanoparticle size, which is in the same order of magnitude as the experimental value for copper of 1358 K [202]. Considering the minimum 10 K error in the estimation of inflexion points due to the temperature jumps in the simulation, the noise apparent in the Lindemann curves, and any deviations from the experimental melting point due to DFT, this is still quite a close result.

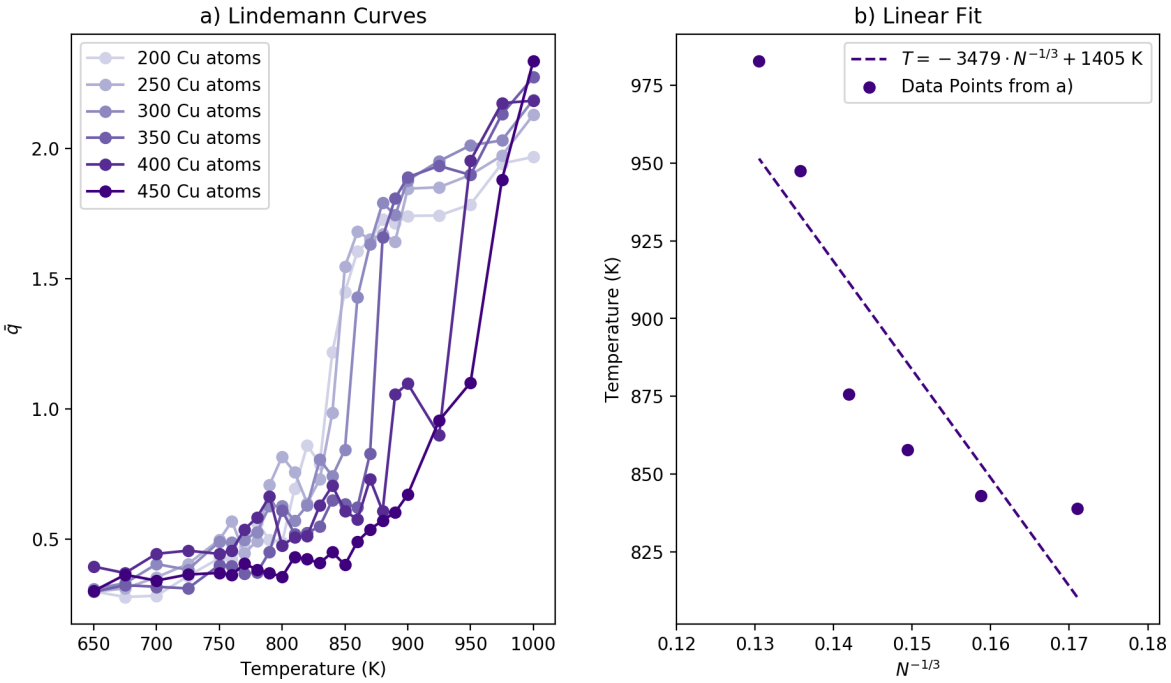

Figure 6.14: a) Lindemann curves for the heating and melting of clusters between Cu_{200} and Cu_{450}. As size increases, the turning point of the Lindemann curves appears at higher temepratures. b) Melting temeprature against cluster size ($N^{-1/3}$), as extracted from fitting the curves in a) to a sigmoid function. A linear fit of the values is also presented.

Obtaining the correct extrapolated infinite size melting temperature is not necessarily assured, since there is some theoretical and experimental results available that suggest that stron cluster-support interactions could lead to alterations in melting point [272, 273]. Analyzing the results of the nearest neighbor distance shifts presented in figures 6.11, 6.12 and 6.13, a mechanism for a possible melting point alteration could also be suggested. The distance between atoms in the cluster is heavily affected by the presence of the support, which leads to a destabilized solid cluster when compared with a proper bulk copper configuration. From this destabilized structure, the liquid phase would be easier to reach,

which could translate into an altered melting point when extrapolating the cluster results to bulk.

The Lindemann index per atom can also be utilized to show the behavior of atom mobility as the clusters cool down. Figures 6.15 and 6.16 show the accumulated average Lindemann parameter value for each atom in a Cu_{500} cluster at the end of each temperature step. To better appreciate the mobility within each temperature window, the color scale range in each subfigure is different.

From a) through c) (1000 to 900 K), the Lindemann index of all the atoms in the cluster is high and randomly distributed. At these temperatures the cluster is molten, and all atoms exhibit similar mobility. Between d) and f) (850 to 750 K) the cluster crystallizes, and regions of lower Lindemann index can be observed starting from the bottom center of the cluster and growing upwards (particularly from the side views). Below this temperature, atoms inside the cluster present the lowest Lindemann value due to their frozen state, while atoms on the surface of the cluster present slightly larger mobility (side view m), n) and o)), and finally atoms that do not belong to well formed facets and thus are more free to move also exhibiting large mobilities at middle temperatures (top views g), h) and i)).

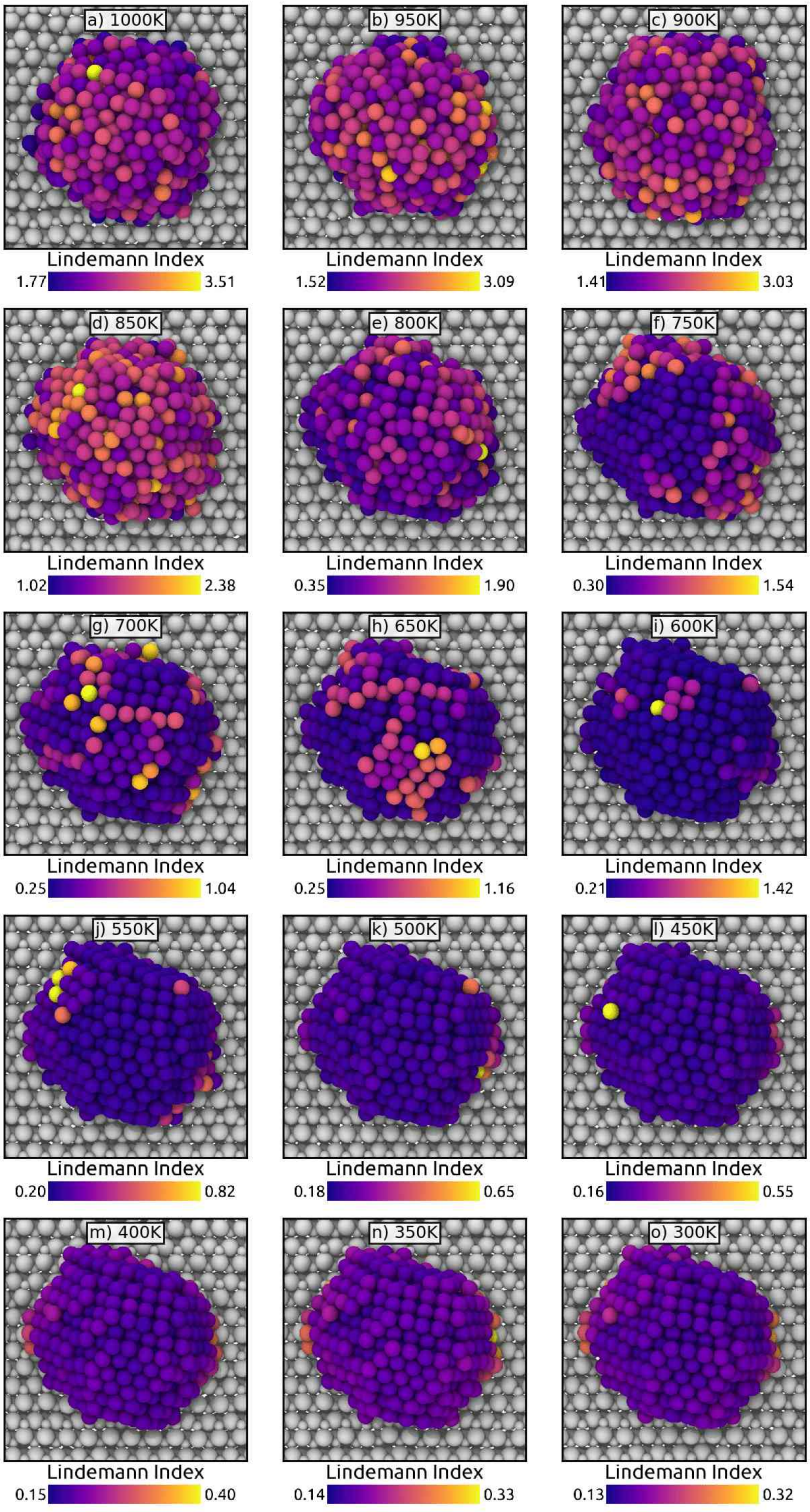

Figure 6.15: Top view of a Cu_{500} at the end of each temperature step in the SA run, colored by the average Lindemann parameter of each atom during that temperature step. The scale in each subfigure is different.

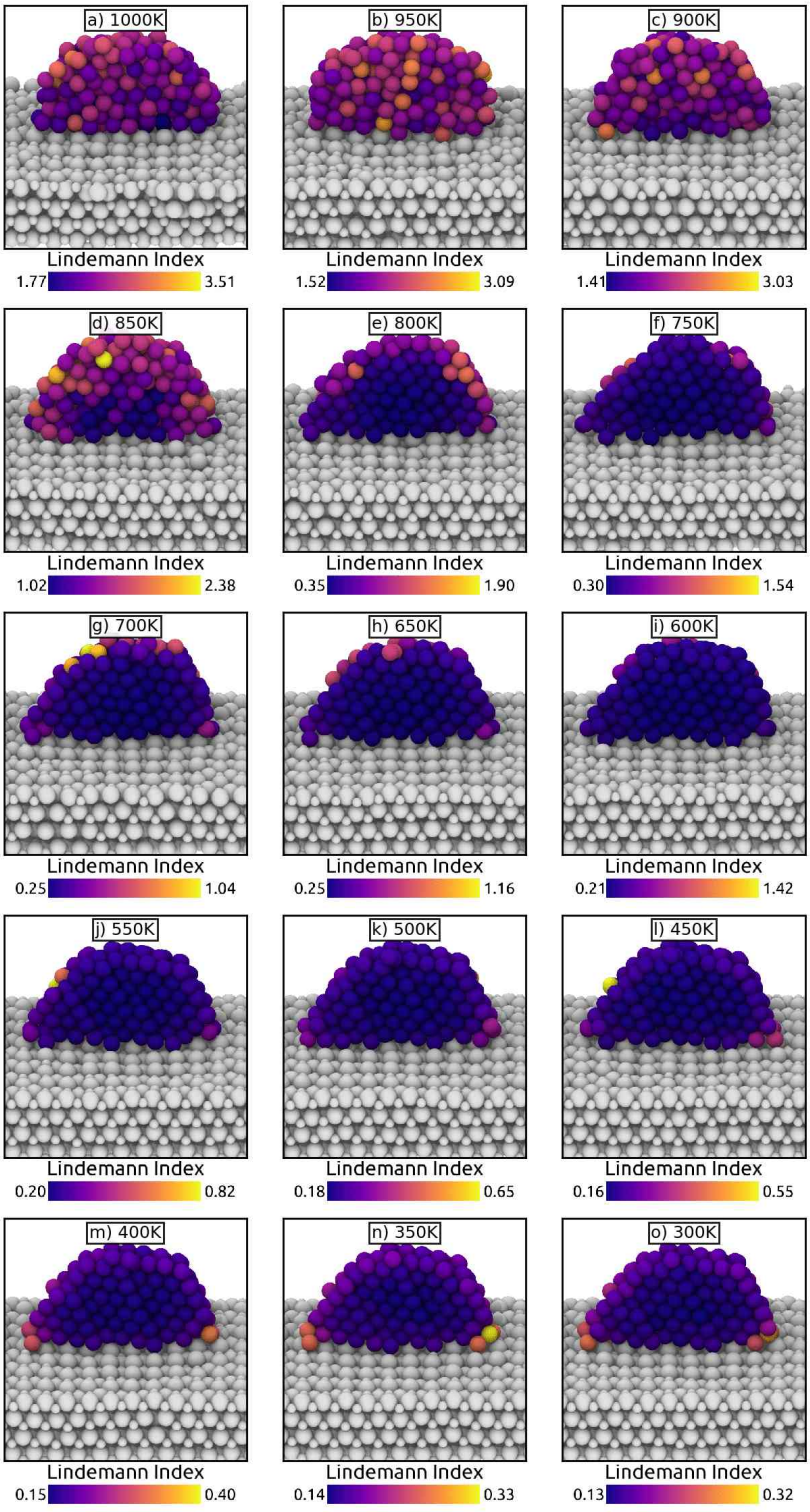

Figure 6.16: Cut-in-half and side view of a Cu_{500} at the end of each temperature step in the SA run, colored by the average Lindemann parameter of each atom during that temperature step. The scale in each subfigure is different.

6.2.7 Structure at the Interface

As discussed in previous sections, the interface between both materials is crucial due to the inherent lattice mismatch present between them. To study the structure of this interface, figures 6.17 and 6.18 present the atoms considered to be at the interface, with the second figure coloring them by the nearest neighbor distance shift as explained in a previous section.

In figure 6.17 the rough polygonal shape of the clusters can be confirmed, with many presenting a hexagonal shape (particularly Cu_{350} d), Cu_{400} b) and Cu_{450} a)), but with other shapes also present such as a square with rounded corners in Cu_{300} d). This behavior is also due to the interactions between atoms and support, since the edges of the polygons follow along principal directions of the ZnO support.

Another noticeable difference between the clusters is the presence of two orientations between them and the support, with the Cu rows either parallel to the oxygen rows (the more common pattern), or at an angle. This also affects the intra-layer structure of the interface atoms. Where the atom rows are oriented parallel to the support rows, the cluster alternates rows of atoms directly in contact with the support and rows above the other rows. In the case where the cluster forms an angle, these rows turn into ascending lines of atoms.

Why do these raised rows of atoms appear? As discussed in other sections, the closest Cu interfaces in structure with the ZnO($10\bar{1}0$) support are Cu(111) (flat) and Cu(110) (not flat). Both structures are actually related to one another, with the difference being whether the fifth copper atom at the center of a copper rectangle is in the same plane as the edges (Cu(111)), or raised (Cu(110)). Here we can clearly see that the clusters exhibit both structures, and also some structures in-between the perfect Cu surface extremes. For this reason, clusters alternate rows of Cu atoms flush with the support and those above the other rows.

This alternation can be used as a way to relieve the strain imposed by the pattern of the support. Instead of leaving a larger gap, or forcing more atoms to be in contact with the support, the cluster bridges the gaps between Cu rows with an out of plane Cu row. Additional strain relief defects appear sometime in the form of missing Cu atoms in a row, particularly close to the edges of the cluster.

This picture of the cluster-support interface is confirmed by looking at the nearest neighbor distance shifts of figure 6.18. Interface atoms mostly exhibit enlarged nearest neighbor distances, but defects usually show a return to the expected nearest neighbor distance(for example, Cu_{200} b), Cu_{300} b) and Cu_{400} b)).

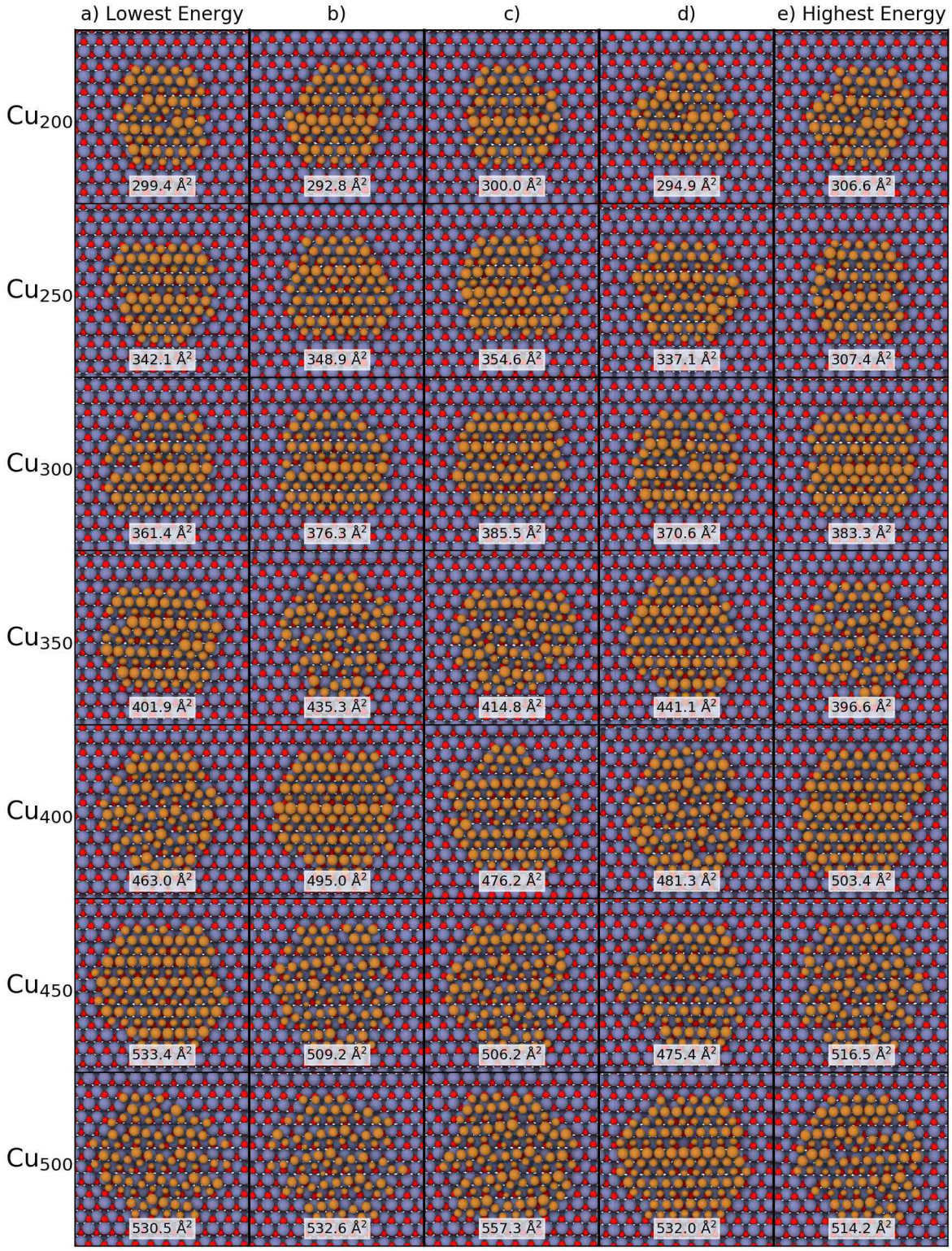

Figure 6.17: Interface structures of all minimized clusters. The caption shows the calculated interface area. Atoms have been resized by their distance to the viewpoint to better show the position of each atom within the layer.

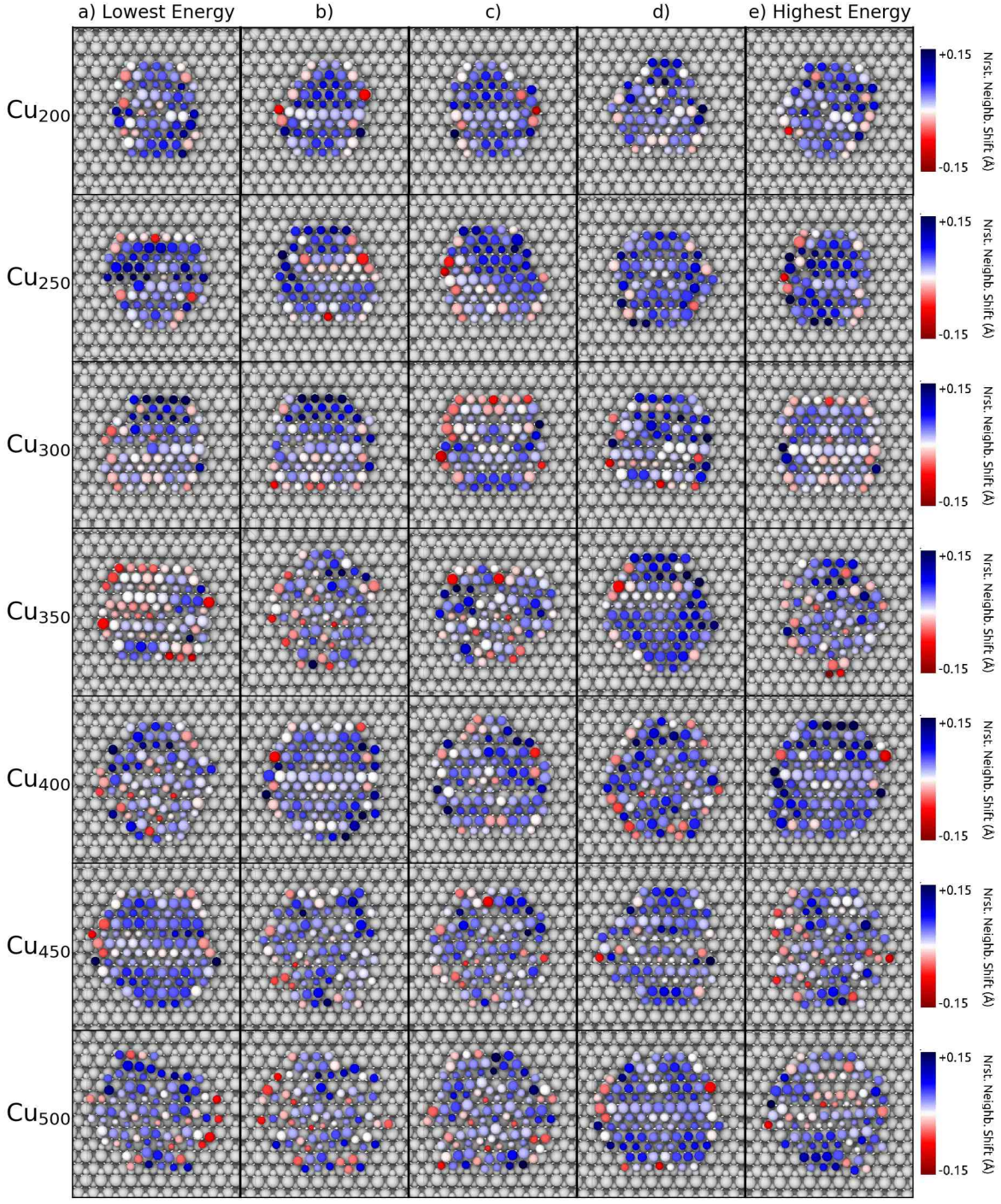

Figure 6.18: Interface structures of all minimized clusters, colored by their shift from the bulk nearest neighbor distance. The caption shows the calculated interface area. Atoms have been resized by their distance to the viewpoint to better show the position of each atom within the layer.

6.2.8 Facets

Figure 6.19 shows the three low Miller index surfaces of copper present in Cu_{350} e) from fig. 6.6, colored by the fractional coordination of each atom to highlight the boundaries between facets. All clusters across sizes show a similar behavior, with mainly Cu(100) and Cu(111) facets present, since these are the surfaces with the lowest surface energy (according to our NNP and DFT, see F.2). Cu(110) facets are rarely present on the outside of the clusters, and when found they usually form the boundary between two other facets, as in the figure. This makes it all the more surprising that the interface between clusters and support does indeed present atoms in the Cu(110) configuration.

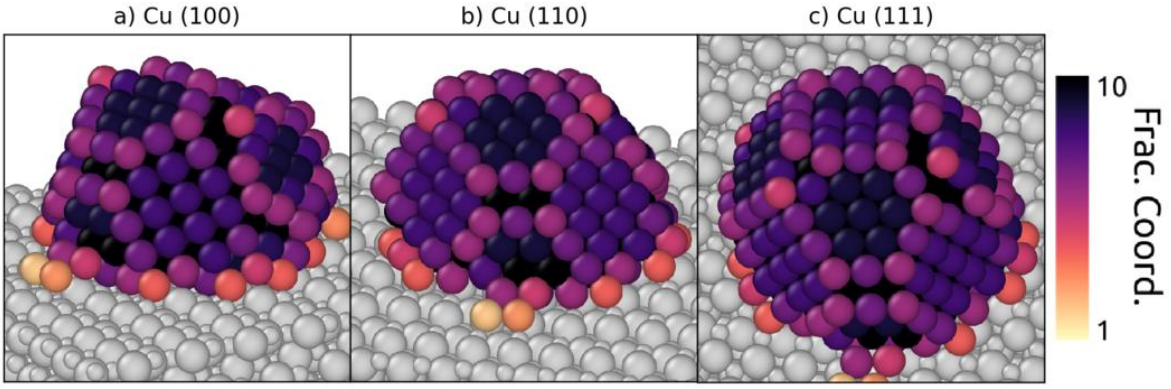

Figure 6.19: Low Miller index surfaces of copper present in Cu_{350} e) from fig. 6.6. a) Cu(100) surface at an angle. b) Row of Cu(110) rectangles, flanked at the top and bottom by Cu(111) and to the sides by Cu(100). c) Cu(111) surface with Cu(100) facets marking the sides of the hexagon.

It would be interesting to quantify the number of atoms belonging to each type of facet and test whether the proportion reproduces the ratio between the surface energies of each facet. This can also be done in experiments when analyzing microscopy images from nanoparticles. For this, some automated method of identifying structures would be required, since a manual assignment would be tedious and error-prone. A cursory review of the bibliography did not manage to find any specific method for classifying surface atoms, but many methods are available for classifying crystal structures. Some simple configuration based methods are available [178, 179], such as the PHTM algorithm [177] explained and utilized in previous sections. From an analysis of the available structural methods, it should in principle be possible to modify one of these to identify previously defined surface atom patterns corresponding to the low Miller index surfaces, but such modification would not be trivial. Thus this remains an open question, but such a tool would be useful for nanoparticle simulations in general.

6.3 Conclusions

The SA runs have successfully generated clusters with internal fcc structures and presenting facets corresponding to the low index surfaces of copper. The effects of the strong interaction with the support as exemplified in previous global optimization and coincidence lattice match simulations are reproduced at this scale. The interface between the cluster and the support presents a configuration corresponding to Cu(111), Cu(110) or structures in-between those two. This agrees with the results for small clusters (sec. 5), where all small clusters starting from a given size showed interfaces intermediate between Cu(111) and Cu(110); and the results for matching interfaces (sec. 7), where Cu(111) was shown to be the most stable of the three low index Cu surfaces when strained, followed by Cu(110). This also agrees with the suggestion by Kohler *et al.* [60], which from observations of the facets of deposited

large Cu clusters posited that the structure at the interface would correspond to Cu(110). These interfaces are distorted by the need to match with the ZnO support, with atoms at the interface in general being farther apart than in bulk copper. A new behavior is that for some cases the lattices are not aligned in the same way, which for the coincident surfaces was simulated by allowing a rotation angle between the matched lattices.

The generated clusters finally enter the nanoparticle range, with the Cu_{500} clusters possessing and average diameter of 2.85 nm. This opens the possibility of comparing the results against experimentally generated clusters, for properties such as cluster shape and size, orientation between the cluster and the support, or melting temperature.

Chapter 7

Study of Large Copper-Zinc Oxide Coincident Surfaces

7.1 Motivation

An interesting feature of the Cu-ZnO system is that like many other materials, the pair presents a lattice mismatch. That is, the vectors that describe the periodic surface structures of both materials are not exact integer multiples of each other. They cannot be arranged in such a way that common supercell can be found, without distorting one or both materials. This is of course also the case with most materials, and this effect is of theoretical and experimental interest in many fields such as the study of thin films [274–277] and epitaxial layer growth [278], of Moiré patterns on mismatched materials [279, 280], 2D heterostructures [281–283], among many others. The effect of lattice mismatch is of more interest than just a technicality, since it can decide the electronic structure and stability of catalytic heterolayers [284, 285], or form structures that can capture metallic clusters or other molecules, such as in graphene [255, 286] which buckles under the induced mismatch strain, or hBN [256] which forms pores when deposited on certain metals.

This mismatch has consequences for any interface between the two materials. In real systems this is solved by either the appearance of non-continuous interfaces (cluster-vacuum interfaces where the strain between a cluster and its mismatched support can be relieved) or defects (strain/stress, missing atoms, atomic plane shifts). This induced strain and related defects can lead to interesting or enhanced catalytic properties [287–290], with particular examples for Cu/ZnO presented in refs. 17 and 27. But even in those cases, the mismatch restricts the possible energetically favorable interfaces and configurations. The mismatch also has consequences *in silico*, since if we want to simulate the interface of the two materials we need to find a common periodic system that encompasses both structures.

In this section the mismatched Cu and ZnO interface is studied, for (initially) flat and large slabs with multiple layers and hundreds of atoms. First all the low strain matches up to a given size are generated utilizing the coincidence lattice match algorithm [161] (CLM), which results in a library of thousands of configurations. Then the NNP is employed to quickly evaluate and minimize all of these configurations, and to reach larger configuration sizes than possible with *ab-initio* methods.

In previous chapters, the CLM algorithm is explained (section 2.5.1), and the necessary concepts from strain theory are presented (section 2.5.2). The search parameters for the CLM algorithm are provided in sec. 3.6, and the NNP sampling methodology is described in section 3.3.6. This chapter contains section 7.2.1, which shows the behavior of the chosen lattice match finding algorithm for our particular Cu and ZnO surfaces; and section 7.2.2, where the relative stability and reconstructions of the generated structures are investigated.

The goal of this study is to observe the general trends in the coincident surfaces and their corresponding energies, find the best matching surfaces, and deduce consequences for nano-scale clusters. This will also show the utility of such an analysis for any type of material, and how a NNP eases the investigation of such a system.

7.2 Results

7.2.1 Behavior of the Coincidence Lattice Match Algorithm

Figure 7.1 shows the results of a CLM search for the three low index copper surfaces on ZnO($10\bar{1}0$). Subfigure a) shows the behavior of the geometrical distortion parameter or degree of lattice distortion ($\varepsilon_{\mathrm{DOLD}}$) as a function of the number of atoms in a 1 layer per material matching cell [3], for the three Cu surfaces. In this subfigure a crescent moon shaped distribution can be observed, with larger coincident cells tending to correspond with lower distortion and strain values (although there are small cells that also have low distortion values). The lowest distortion values are only reached starting at 200 atoms per 1 layer per material.

Subfigure b) shows the distribution of sizes with respect to the absolute value of the maximum allowed integer in equation 2.31. As expected, allowing a larger value for the possible integer combinations results in larger coincident lattices.

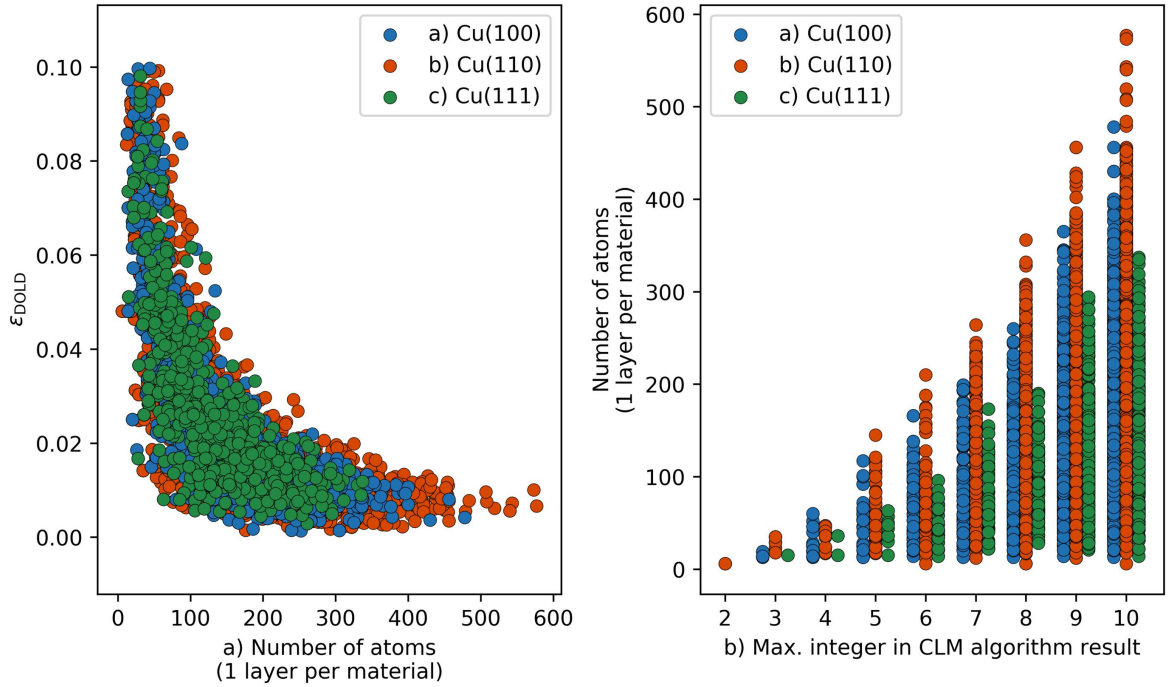

Figure 7.1: a) Geometrical distortion/degree of lattice distortion (δ_{DOLD}) vs. number of atoms (at 1 layer per material) for the CLM algorithm results. Notice the crescent-shaped distribution. b) Size of the matched surfaces (1 layer per material) as a function of maximum allowed integer in the CLM algorithm.

Figure 7.2 shows histogram distributions of various properties across the obtained matches. a) presents a histogram of the sizes of the resultant matches, where it can be seen that most matches have at most 200 atoms (remember that this is only for 1 layer per material), with very few beyond the 300 atom mark. In b) an angle distribution histogram is plotted, which shows that the matches are roughly equally distributed across the swept angles. The surfaces cover different angle ranges due to the symmetries involved: the results for Cu(100) start repeating after 45 degrees, 90 degrees for Cu(110), and 30 degrees for Cu(111). This is related to the fact that ZnO($10\bar{1}0$) has a rectangular geometry (see sec. 3.6.2), while for Cu, depending on the surface, exhibits a square, rectangular, and

[3] There is a one-to-one relation between the size of the supercells in a match, as expressed in terms of supercell with respect to the original lattice or with area, and the number of atoms contained in the match. They both have the same meaning. The two descriptions will be used interchangeably in the following discussions when referring to system size.

hexagonal geometries respectively. Finally, in c), the distribution of the strain values seems to behave like a Poisson distribution (i.e., it is not symmetrically distributed around it's maximum and has a long tail towards higher distortions). No matches appear beyond $\varepsilon_{\text{DOLD}} = 0.10$ since this was used as a limit to filter the generated matches. Most matches have a distortion of only 0.01-0.03.

In total, 2169 matches have been found for Cu(100), 3797 for Cu(110), and 1128 for Cu(111). Some of these matches are redundant, since sometimes very similar supercells are found when matching with angles that are only one degree apart. These similar structures have not been filtered out. Under the current settings, the average amount of matches per angle covered is not uniform, with 48.2 matches/degree for Cu(100), 42.2 matches/degree for Cu(110), and only 37.6 matches/degree for Cu(111).

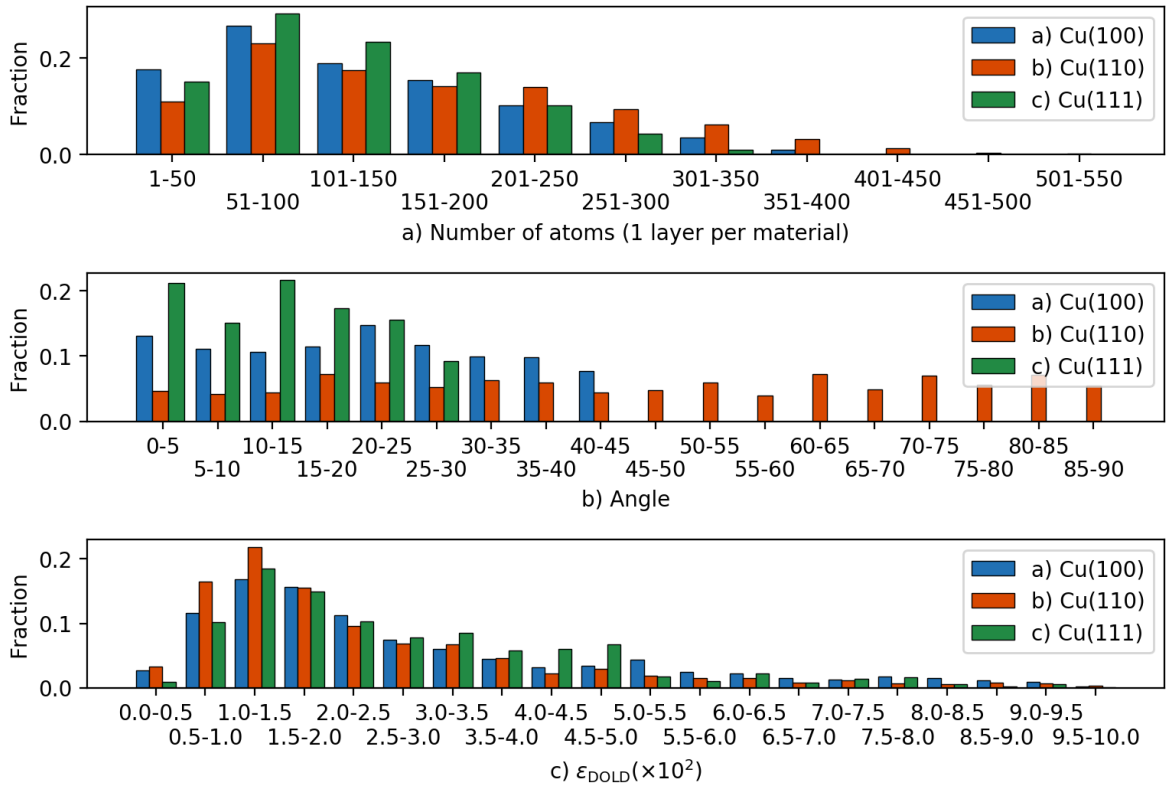

Figure 7.2: Bar plot distribution of various properties for the matched surfaces a) Number of atoms of each coincident surface (one layer per material). b) Angle of the match. c) Geometrical strain distortion of the resulting match.

Finally, figure 7.3 shows a more detailed per-angle view. In this case, for clarity, only results with up to 200 atoms (1 layer per material) have been included. The plots show for each copper surface, the number of atoms in each coincident lattice as a function of the angle between the original lattices, colored by the average distortion $\varepsilon_{\text{DOLD}}$ (eq. 2.33) of each match.

As a general trend, what was already shown in figure 7.1 is observed here: matches with low strain (lighter color) are present higher in the graph because they contain more atoms/require larger cells. For the (100) copper surface in a), the smallest cells are achieved in the angle ranges of 2-6, 19-24, 30-35 and 42-45 degrees; but these cells unfortunately have a rather high strain. For the (110) surface in b), an area with only single matches at 2 and 3 degrees can be noticed (presumably any other matches present in this region are above our strain cutoff of 0.10 or require larger supercells), which also coincides with the region with the smallest matching cells at between 1-8 degrees. Other areas with small coincident matches are present around 60 and 73 degrees. Once again, these small cells correspond to rather high strains. This trend changes somewhat for the (111) surface in c): the smallest cells with medium strain

are present at around 20 degrees, but at 29 and 30 degrees only one solution is present, which has only twice the number of atoms of the previous solutions but an extremely low strain.

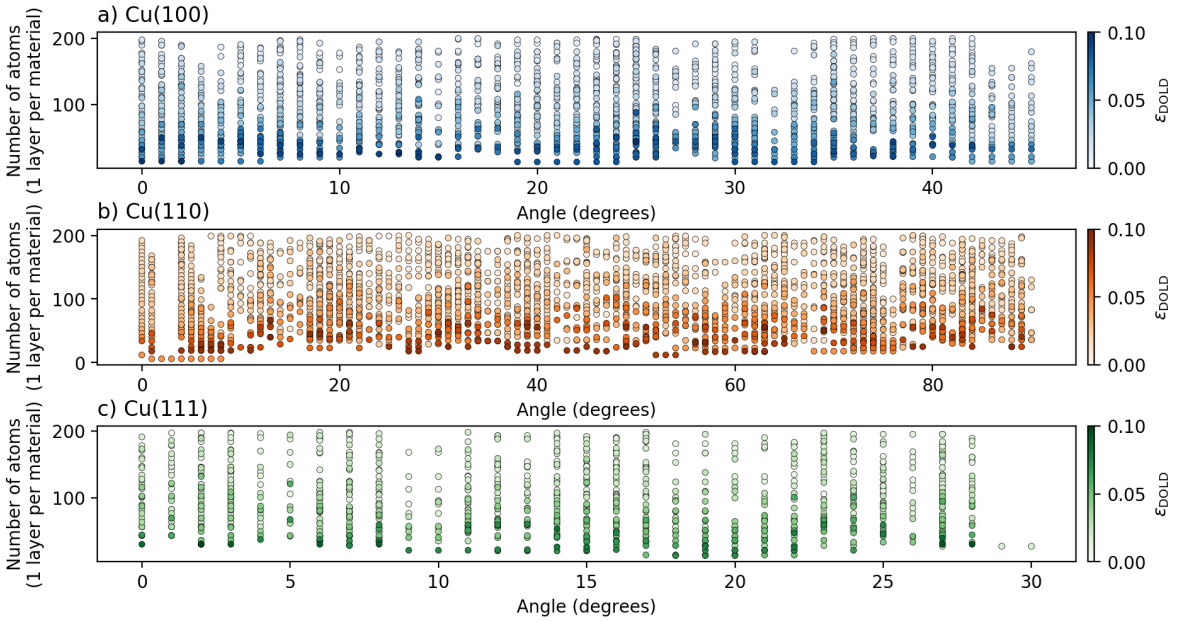

Figure 7.3: Size of the matched cell (one layer per material) as a function of the angle between the matched lattices, colored by the geometric distortion of the match, for the three low index Cu surfaces. a) Cu(100) b) Cu(110) c) Cu(111)

7.2.2 Results from Geometry Minimizations

After allowing the matched cells to relax together utilizing the NNP as a force field, an even more complicated picture emerges. In the following the energy of the relaxed cells is defined as E_{relaxed}, which is given by

$$\begin{aligned} E_{\text{relaxed}}(N_{\text{layers}},\ \text{Miller Index}) = \frac{1}{N_{\text{atoms}}}[&E_{\text{NNP}}(N_{\text{layers}},\ \text{Miller Index}) \\ &- N_{\text{Cu}} \cdot E_{\text{NNP}}(\text{Cu},\ N_{\text{layers}},\ \text{Miller Index}) \\ &- (N_{\text{Zn}} + N_{\text{O}}) \cdot E_{\text{NNP}}(\text{ZnO},\ N_{\text{layers}},\ (10\bar{1}0))], \end{aligned} \tag{7.1}$$

where N_{XX} is the number of atoms of a given species in the matched cell, $E_{\text{NNP}}(N_{\text{layers}},\ \text{Miller Index})$ is the absolute energy calculated by the NNP for a matched cell after relaxation, $E_{\text{NNP}}(\text{Cu},\ N_{\text{layers}})$ is the energy per atom as calculated with the NNP of a relaxed Cu slab with a given Miller index and number of layers , and $E_{\text{NNP}}(\text{ZnO},\ N_{\text{layers}},\ (10\bar{1}0))$ is the equivalent for ZnO. Thus each match is referenced to the relaxed, isolated and not distorted/not strained slabs, that correspond to its surface cut and number of layers.

Figures 7.4, 7.5, and 7.6 show the first series of results from the matched slab relaxations. In these and following figures, care should be taken when analyzing the points with lowest E_{relaxed}, since they usually correspond to reconstructed interfaces (see sec. 7.2.4), and thus do not follow the trend of the unreconstructed surfaces that still correspond to the original Cu slabs.

In these figures, a) shows a view of E_{relaxed} as a function of the matching angle, colored by $\varepsilon_{\text{DOLD}}$. There does not appear to be a pattern between energy and angle of the matched interfaces, except for

Cu(111), where the two solitary matches observed at 29 and 30 degrees in 7.3 also end up corresponding to rather stable configurations.

Subfigure b) presents a plot of E_{relaxed} vs. δ_{CD}, the distance metric between initial and final configurations explained in section 2.6.2. The larger this value is, the more interatomic distances have shifted in the relaxation process, which can be associated with a larger reconstruction of the Cu slab. Each dot is also colored by the initial $\varepsilon_{\text{DOLD}}$, the degree of lattice distortion. Points further to the right (larger reconstruction) usually correspond to darker colors, that is, larger values of $\varepsilon_{\text{DOLD}}$: not surprisingly, matches with a larger initial strain tend to reconstruct more. Across the three surfaces we observe clustering of points to the left (small reconstruction) with lighter colors (smaller $\varepsilon_{\text{DOLD}}$), which likely correspond to structures that do not suffer any reconstruction beyond the shifting of some layers in the direction perpendicular to the slabs. Going to the right we observe other point separated from these main clusters with large reconstruction values: these two facts seem to imply there is not a continuum between the reconstructions that the slabs undergo, but rather a break between slabs that reconstruct little (i.e.: a shift in layer height) and those that reconstruct more (i.e.: a change in Miller index). Not surprisingly, many reconstructed structures also achieve a more stable configuration than the unreconstructed slabs, except for the case of Cu(111).

In subfigure c), we now color by E_{relaxed} and use $\varepsilon_{\text{DOLD}}$ as the y-axis. As a complement to subfigure b), we once again notice a cluster of structures at low values of both $\varepsilon_{\text{DOLD}}$ and δ_{CD}, and also a somewhat linear relationship between two properties that breaks at large values of $\varepsilon_{\text{DOLD}}$. This is the rough break in reconstruction modes mentioned for subfigure b). The $\varepsilon_{\text{DOLD}}$ limit can be estimated visually from the figure to be between 0.04 and 0.06, where the linear clustering between both properties starts to break. From this, we can also estimate that a δ_{CD} of between 0.025-0.050 is the limit for small reconstructions. All three Cu interfaces seem to show the same tendency.

Finally, subfigures d) and e) show a pair of "butterfly plots", where the eigenvalues of the strain tensors are plotted, and colored by either E_{relaxed} or δ_{CD}. The plots are in principle mirrored along the x-y axis since ε_1 and ε_2 are interchangeable. In general the matches combine positive and negative values of the eigenvalues, with few or no matches in the regions where both are positive (upper right quadrant of the plots) or negative (lower left quadrant). This plus the mirroring gives the plots their butterfly shape. If we remember that the eigenvalues of the strain tensor are the stretches experienced along the directions where no shear is present (the eigenvectors, the principal strain directions), we can try to explain why this is. As mentioned in sec. 3.6.2, attempting to match Cu slabs and ZnO($10\bar{1}0$) directly usually results in one direction being too short and another too long. This would result in a stretch and a compression being required to generate good matches between the lattices. Additionally, the butterfly plots seem to present a group of structures arranged in a line, of uncertain origin. This is most notable for Cu(110). A possible explanation is that these correspond to a specific geometric matching pattern that only appears in a small angle range.

Looking now into the colored properties of each butterfly plot, from subfigure d) we can observe more stable matches (more negative relaxation energy, lighter color) towards the center of the plot (both eigenvalues close to 0.0) and towards the bottom and left of the plots (one eigenvalue close to 0.0, the other one negative), while worse matches (less stable) are present towards the top and right (particularly noticeable for Cu(110)). Subfigure e) suggests that larger reconstructions occur at negative eigenvalues (darker coloring towards the left and bottom of the plots). This also explains the behavior of the relaxed energy in d): compressive repulsion usually destabilizes a system more than an equivalent expansion in interatomic distance (energy vs. volume plots for a solid are not symmetrical), so we would expect the structures at high compression to be more unstable. This is indeed the fact, and they are so unstable that they end up reconstructing into different, lower energy configurations (see sec. 7.2.4).

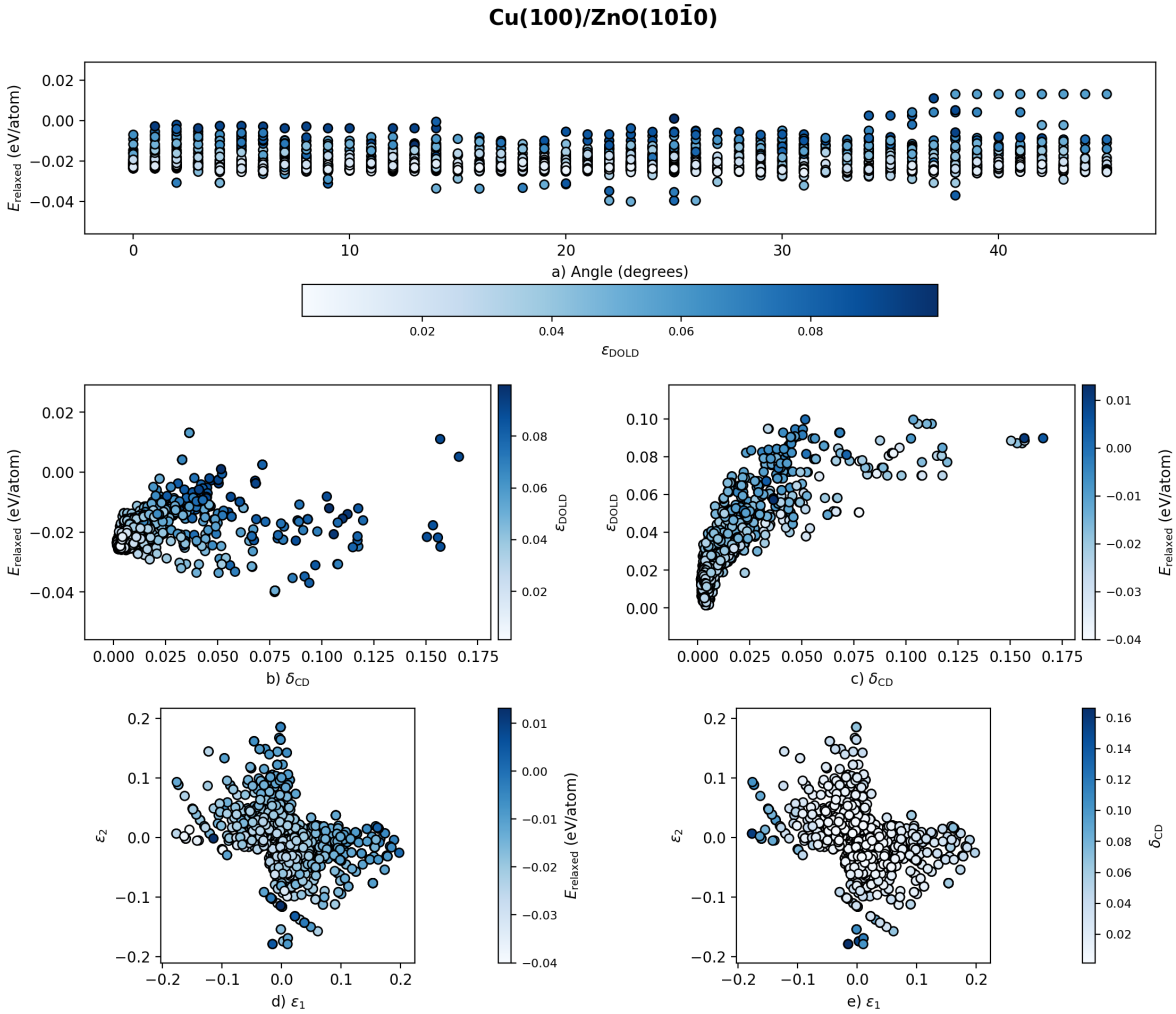

Figure 7.4: For the Cu(100) and ZnO(10$\bar{1}$0) matches: a) E_{relaxed} vs. angle colored by $\varepsilon_{\text{DOLD}}$, the degree of lattice distortion, larger values correspond to initially more strained lattices. b) E_{relaxed} vs. δ_{CD}, a measurement of the amount of reconstruction of the slab, where a higher value indicates a larger reconstruction, colored by $\varepsilon_{\text{DOLD}}$. c) $\varepsilon_{\text{DOLD}}$ vs. δ_{CD}, colored by E_{relaxed}. d) and e) Eigenvalues of the strain tensor associated with a particular match, $\varepsilon_{1,2}$, plotted against each other, and colored by E_{relaxed} or δ_{CD} (similarity between initial and final configuration).

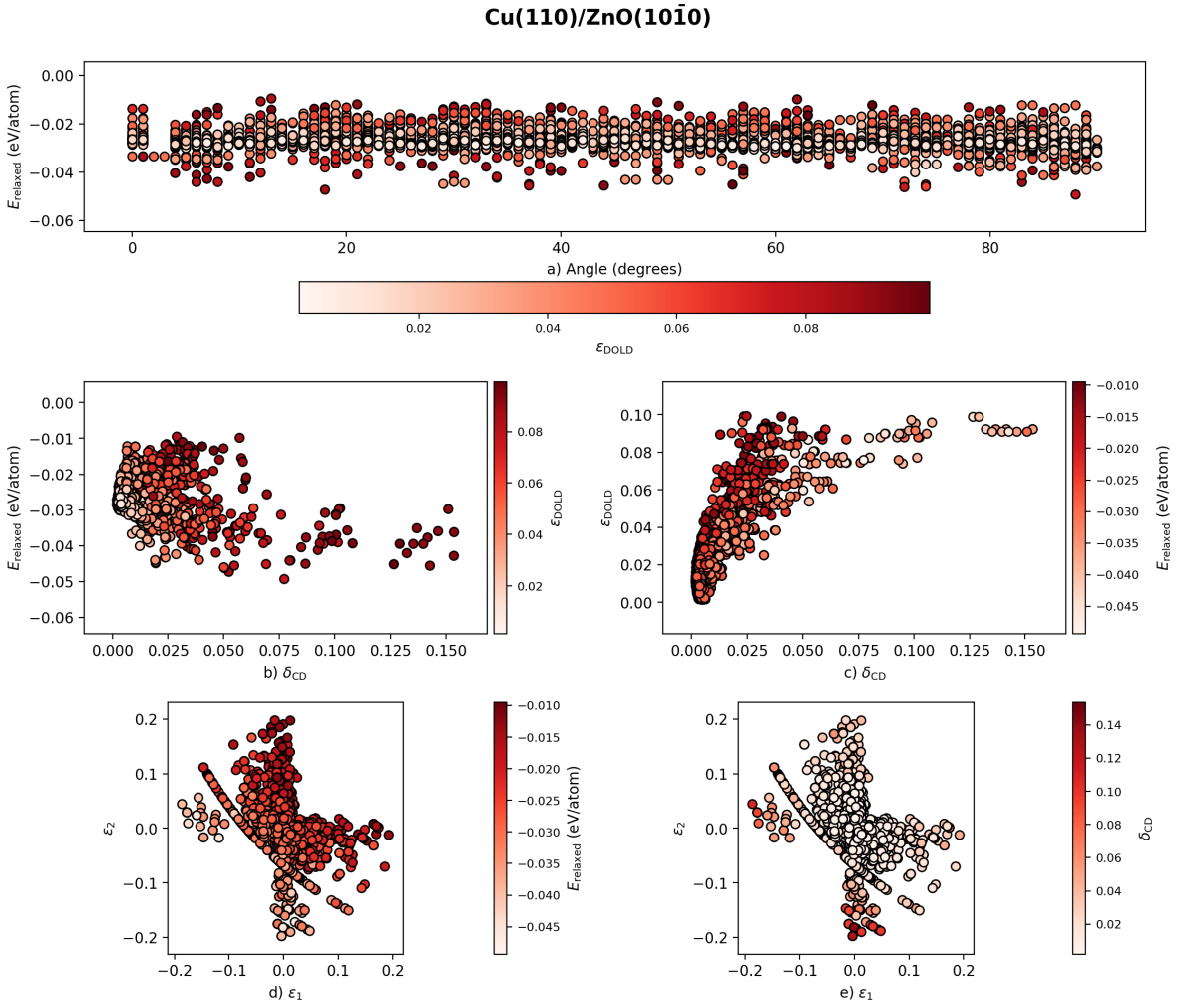

Figure 7.5: For the Cu(110) and ZnO(10$\bar{1}$0) matches. For description of panels, see fig. 7.4.

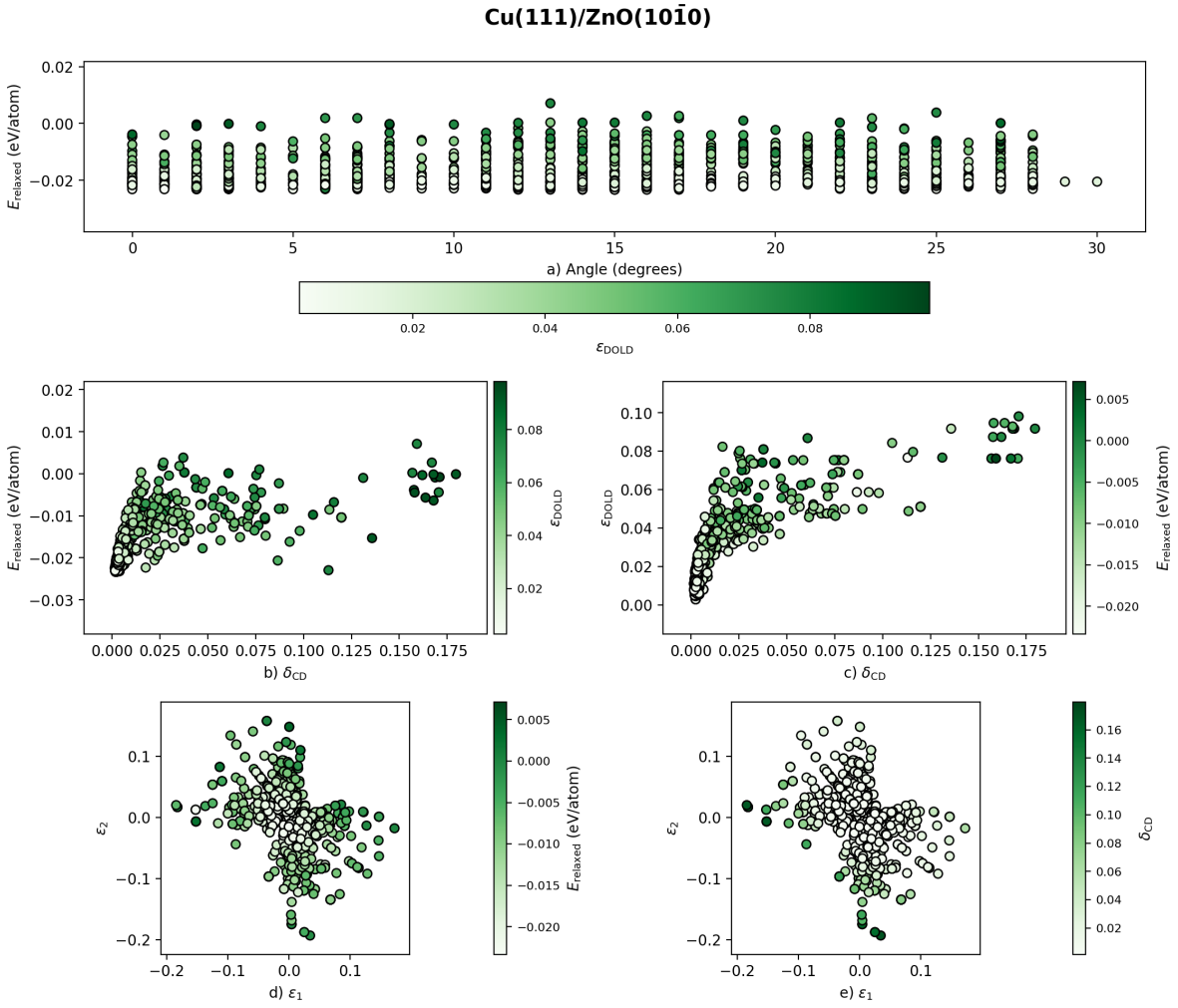

Figure 7.6: For the Cu(111) and ZnO(10$\bar{1}$0) matches. For description of panels, see fig. 7.4.

Further energy and structural relationships are presented in figures 7.7, 7.8, and 7.9. Subfigure a) plots E_{relaxed} vs. $\varepsilon_{\text{DOLD}}$, coloring by number of atoms in the match. As shown in fig. 7.1, lower strains correspond to larger number of atoms. This subplot also shows that for Cu(100) and Cu(110), the lowest energy matches correspond to structures with small number of atoms and rather large distortions. This implies that these structures have probably rearranged under the influence of the large distortion to lower their energy. This is not the case with Cu(111), where the low energy configurations are also large and with low $\varepsilon_{\text{DOLD}}$, implying only minor rearrangements.

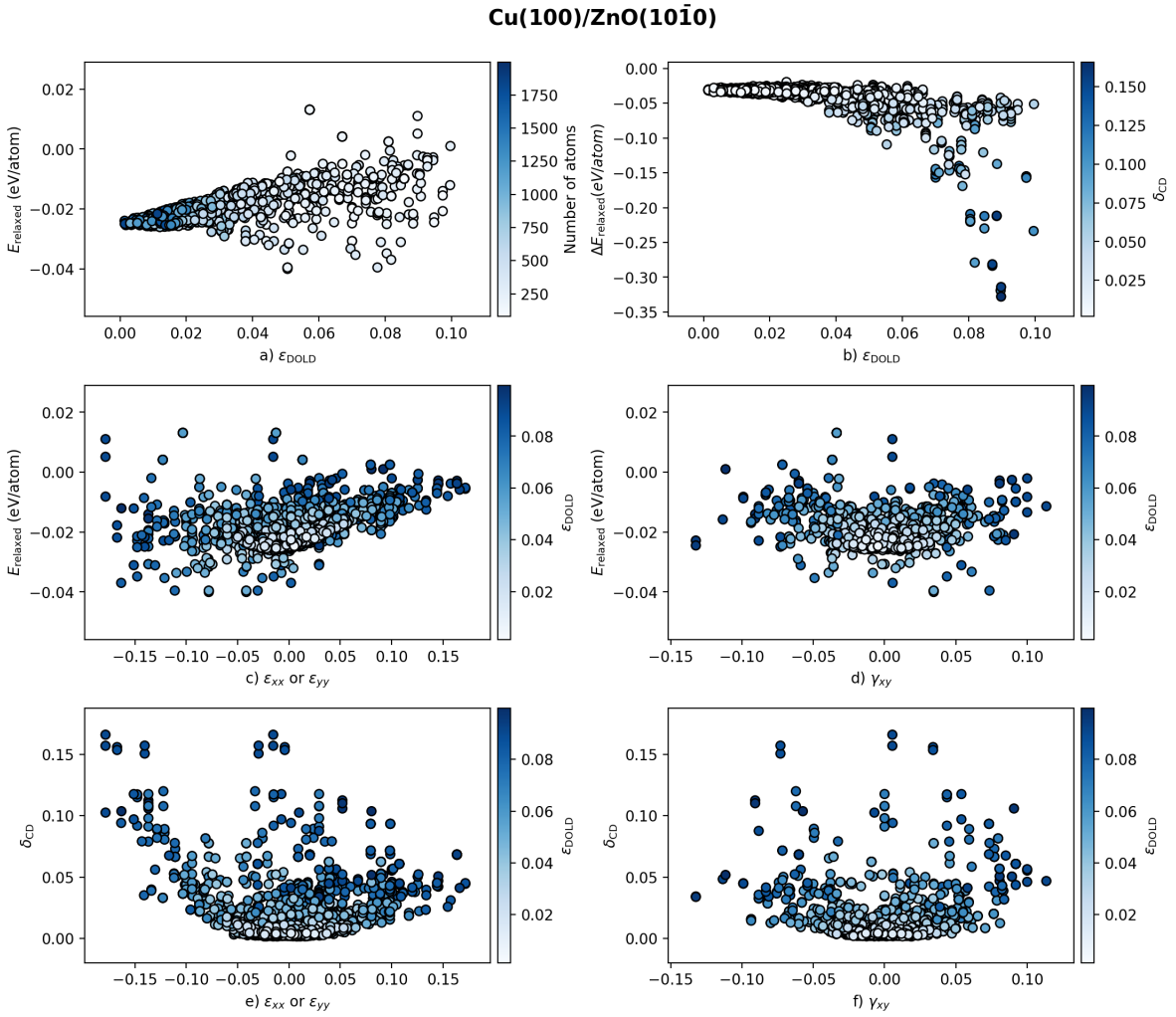

Figure 7.7: For the Cu(100) and ZnO(10$\bar{1}$0) matches: E_{relaxed} vs $\varepsilon_{\text{DOLD}}$, colored by the total number of atoms in each cell. b) $\Delta E_{\text{relaxed}}$ vs $\varepsilon_{\text{DOLD}}$, colored by δ_{CD}. $\Delta E_{\text{relaxed}}$ shows how much energy was gained by the match in the relaxation. c) and d) E_{relaxed} vs elements of the strain tensor (uniaxial strain ε_{xx} and ε_{yy}, shear γ_{xy}), colored by $\varepsilon_{\text{DOLD}}$. For c), each match is represented twice, once for ε_{xx} and once for ε_{yy}. e) and f) are similar to c) and d), but plotting now δ_{CD} vs. the strain tensor components, and coloring again by $\varepsilon_{\text{DOLD}}$.

In subplot b), instead of the previously defined E_{relaxed}, $\Delta E_{\text{relaxed}}$ is plotted, which is given by

$$\Delta E_{\text{relaxed}} = E_{\text{relaxed}} - E_{\text{init}}, \tag{7.2}$$

where E_{relaxed} is as previously defined, and E_{init} is the same but with the initial, unrelaxed energy of the matched slabs, and referenced to the unrelaxed, isolated, non-deformed slabs. In effect, $\Delta E_{\text{relaxed}}$ is the energy gain due to the geometry minimization performed with the NNP.

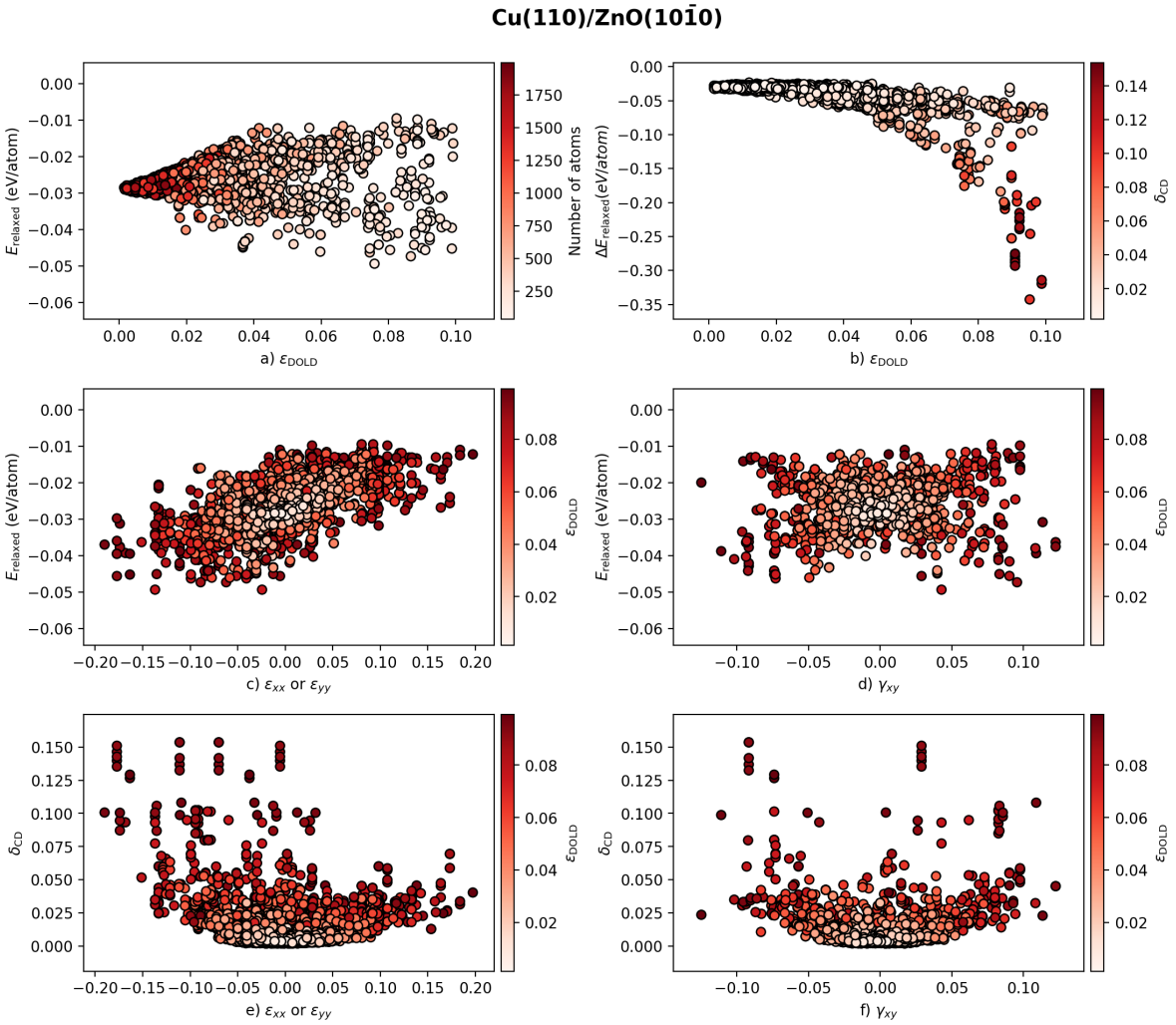

Figure 7.8: For the Cu(110) and ZnO(10$\bar{1}$0) matches. For description of panels, see fig. 7.7.

The points in b) are also colored by δ_{CD}. Unsurprisingly, high values of δ_{CD} tend to correspond with large relaxation energy gains: these are structures that have undergone reconstruction. This subplot reinforces the break in reconstruction levels noticed in the previous figures: between an ε_{DOLD} of 0.04 and 0.06 we observe a rapid increase of bot $\Delta E_{relaxed}$ and δ_{CD}.

The few next subplots (c) through f)) show the behavior of $E_{relaxed}$ and δ_{CD} as a function of the components of the strain tensor. Since care has been taken to rotate the strain tensor into the original lattices, these can be understood as acting on the lattice cells shown in 3.5. In the case of the plots involving uniaxial strains (ε_{xx} and ε_{yy}), since there are two values per strain tensor and they are not interchangeable (compared to the eigenvectors), each match has been plotted twice, once for each direction.

In c) and d), each dot is colored by ε_{DOLD}. As expected from the relationship between ε_{DOLD} and the components of the strain tensor, darker colors and larger ε_{DOLD} are observed as we go away from 0.0 in each plot. Negative ε_{ii} values result in lower energies for Cu(100) and Cu(110), once again probably due to the induced reconstructions. Otherwise, as seen from the plots for Cu(111), deviations from strain component 0.0 result in higher energies.

In e) and f), δ_{CD} is plotted vs. the strain tensor components. As before, negative values of uniaxial strain ε_{ii} result in the larger reconstructions.

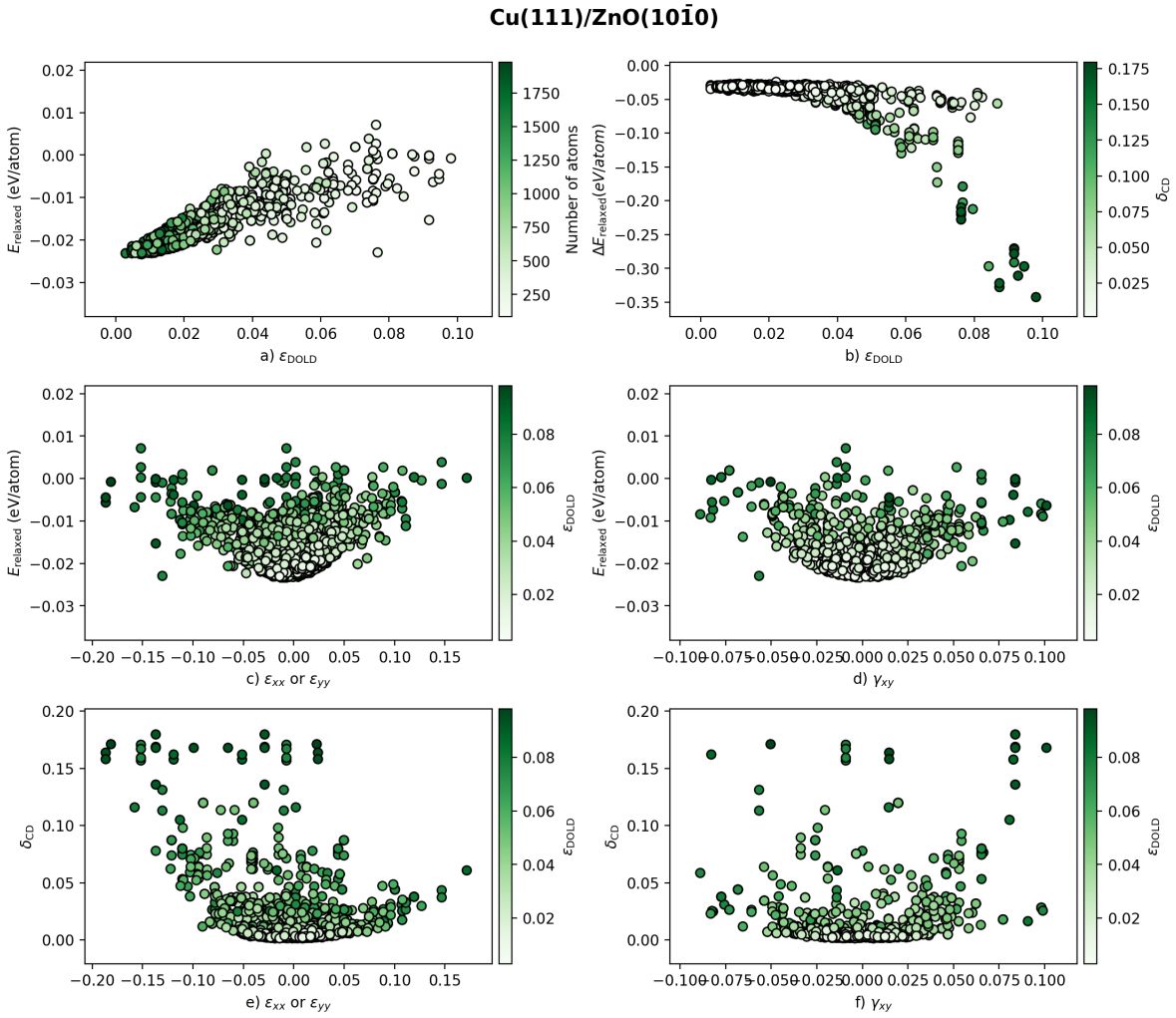

Figure 7.9: For the Cu(111) and ZnO(10$\bar{1}$0) matches. For description of panels, see fig. 7.7.

Finally, figures 7.10, 7.11, and 7.12 show different views of E_{relaxed}. Subplots a) and b) explore the relationship between E_{relaxed} and the number of atoms in the system. For Cu(100) and Cu(110), smaller cells *appear* to be more stable, but this is once again due to these cells reconstructing, as can be deduced from the high values of δ_{CD} and $\varepsilon_{\text{DOLD}}$ for these matches, as well as their absence from the plots for Cu(111).

In these subplots we can clearly see the most stable, lightly reconstructed matches for each Cu surface, by looking at the lowest energy achieved by the largest structures. The line lies at -0.025 eV/atom for Cu(100), -0.03 eV/atom for Cu(110) and -0.023 for Cu(111). That Cu(110) and Cu(111) are respectively the most and least stable when compared to the undeformed, separated slabs is not surprising. Cu(110) is the most unstable of the three surfaces (highest surface energy) to begin with, which means it can gain extra energy by interacting with the ZnO layers. Cu(111) is the most stable surface already, so not as much energy is gained by interacting with the ZnO. If we instead reference E_{relaxed} to bulk Cu and Zn, in which case the three Cu surfaces have the same reference energy, the order for $E_{\text{relaxed}}(\text{bulk})$ becomes: Cu(111)=0.087 eV/atom $<$ Cu(100)=0.102 eV/atom $<$ Cu(110)=0.117 eV/atom.

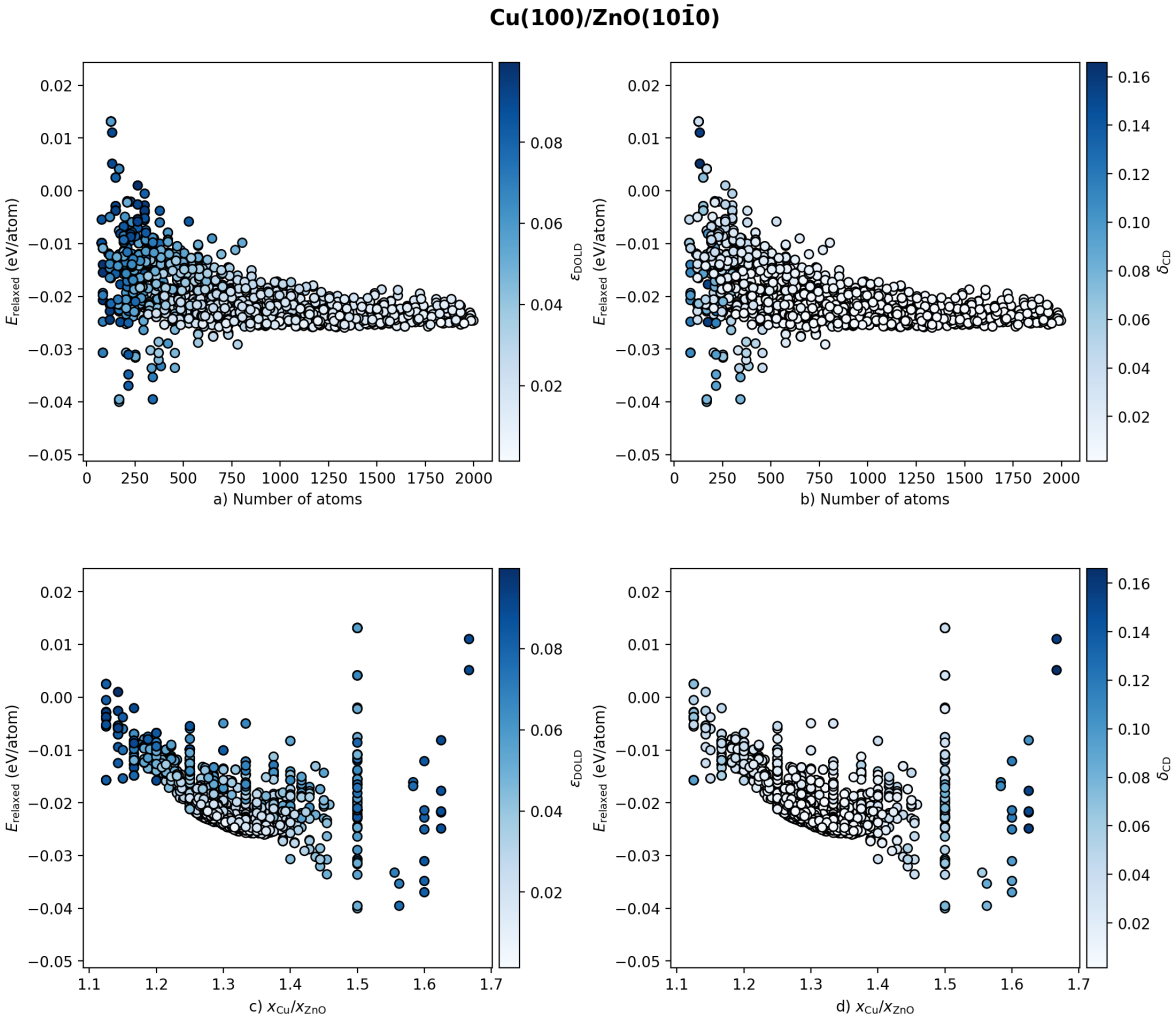

Figure 7.10: For the Cu(100) and ZnO(10$\bar{1}$0) matches: a) and b) E_{relaxed} vs. number of atoms in the match, colored by $\varepsilon_{\text{DOLD}}$ and δ_{CD}. c) and d) E_{relaxed} vs. the proportion of Cu and ZnO in the match ($x_{Cu}/x_{ZnO} = 1$ means one Cu atom per ZnO formula unit in the system), colored by $\varepsilon_{\text{DOLD}}$ and δ_{CD}.

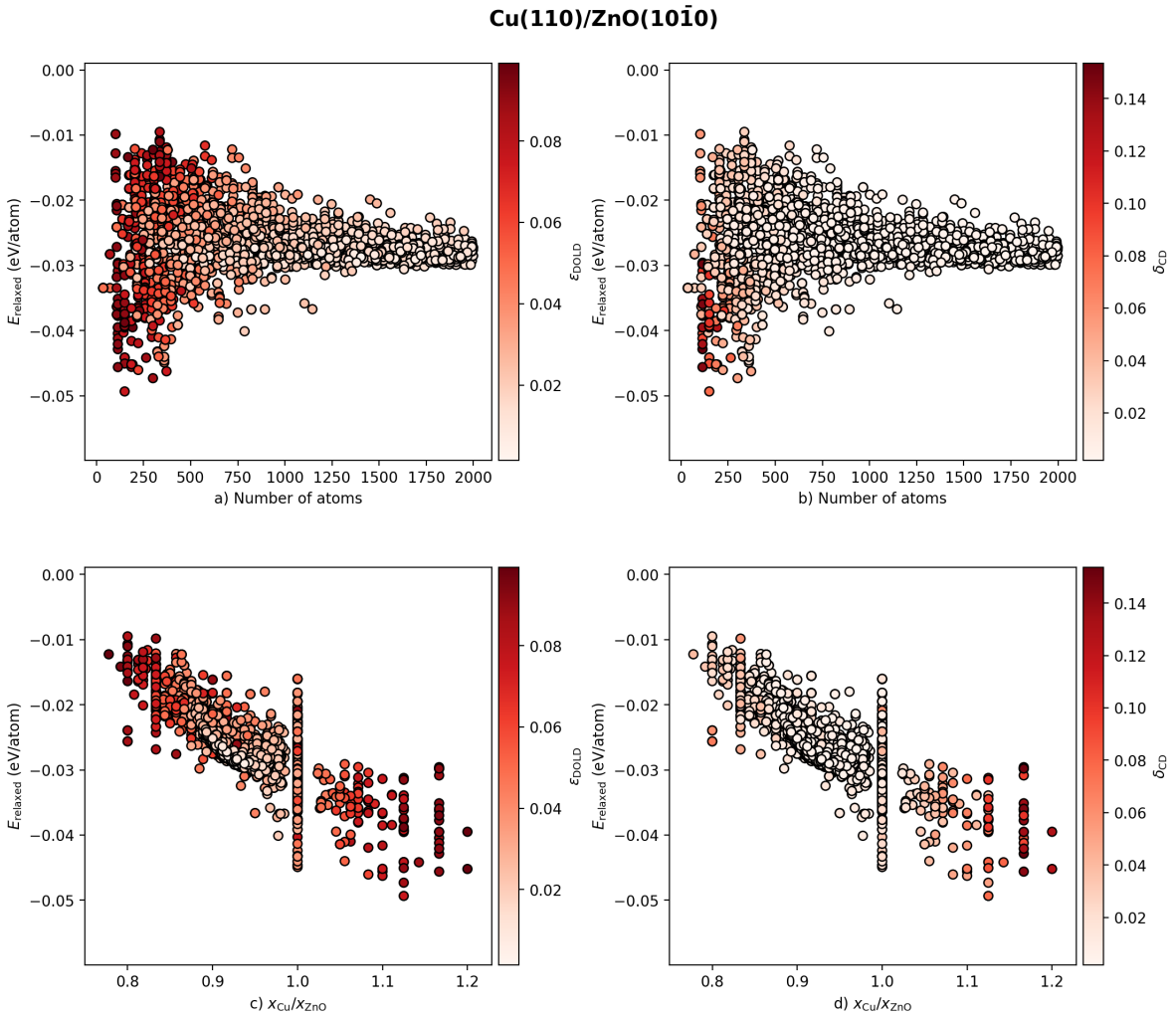

Figure 7.11: For the Cu(110) and ZnO(10$\bar{1}$0) matches. For description of panels, see fig. 7.10.

Subplots c) and d) present another interesting difference between the matches: due to the different contractions and expansions suffered by the Cu layer, its density gets modified, which means that also the proportion between Cu and ZnO in the system changes. Due to the different initial lateral densities of the Cu lattices the matches can reach different proportions of Cu and ZnO. For the original unstrained materials, the x_{Cu}/x_{ZnO} ratio is: Cu(100) 1.32 Cu(110) 0.93 Cu(111) 1.52. Clearly the plots in c) and d) are centered around these values, but the energy minimum for unreconstructed slabs seems to be shifted to slightly larger ratios. For Cu(100) and Cu(111), we once again observe structures with lower energies at higher Cu/ZnO ratios. These correspond as before to compressed structures that tend to reconstruct in the relaxation.

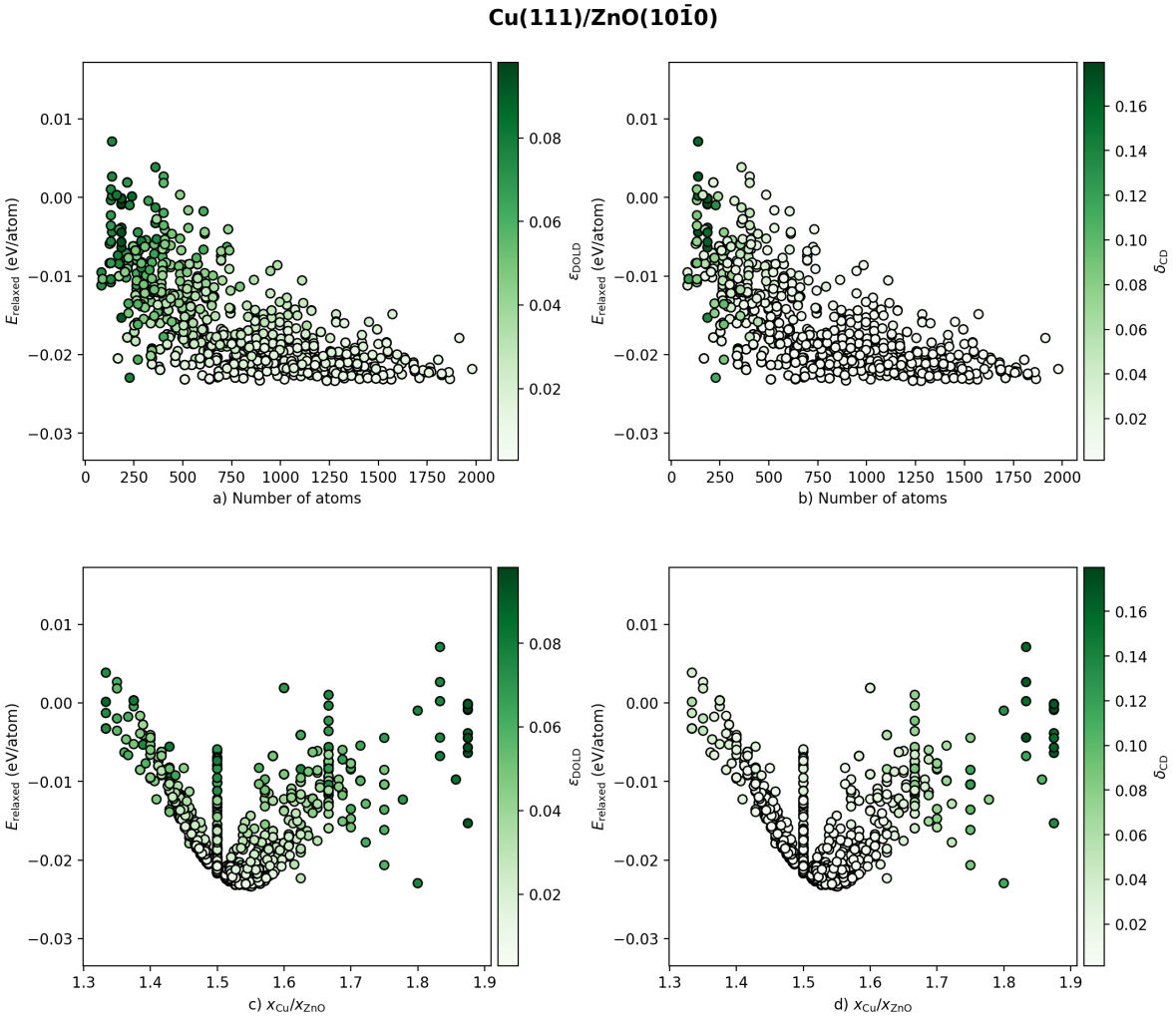

Figure 7.12: For the Cu(111) and ZnO(10$\bar{1}$0) matches. For description of panels, see fig. 7.10.

7.2.3 Translations in the XY Plane

As mentioned in the introduction, one degree of freedom remaining to the matches is the translation between the atomic basis of both materials. In the relaxations described above atoms can also move laterally, but the question now is what would happen with larger shifts that can result in moving from one adsorption position to another. It would be easy to assume that this has a large influence in the final energy of the slab, since atoms can lie on top of more or less favorable positions. In our tests, the result is that the effect is not that large, and this is something that was also found in the original CLM study [161]. Although there is an energy difference when shifting the Cu layer around parallel to the ZnO surface, this difference is usually in the order of 0.1 meV/atom or below, far below the NNP accuracy, and much smaller than the E_{relaxed} energies calculated in the previous section. We do not expect this level of energy gain to alter the general trends observed in our analysis, and since shifting every single match around to find the best position would be an expensive process (although possible with the NNP), it has been decided to not perform shifts in the XY plane. For the unstable structures that reconstruct this might also change the reconstruction path, but once again this would not alter the general trend.

As to the reason for this behavior, the interactions between atoms at the interface might be "saturated". Since the slabs are matched leaving no gaps, all atoms of one material present more or less the same environment. Shifting one atom out of a preferred position just means that the neighboring atoms are now shifting *towards* that same position, which in the end means that the energy gains and losses balance out. This behavior might be different if the layers have more features that break the 2D planar symmetry, such as vacancies right at the interface, or higher Miller index surfaces that present kinks and steps, which then would need to match correctly like building blocks to attain a minimum energy level.

7.2.4 Selected Structures

For illustrative purposes, here a number of relaxed structures are presented, consisting of both reconstructed and not reconstructed surfaces, for each of the three Cu slabs. Table 7.1 shows a summary of the properties of the selected structures.

Cu	Number	N_{atoms}	E_{relaxed} (eV/atom)	$\varepsilon_{\text{DOLD}}$	δ_{CD}	$x_{\text{Cu}}/x_{\text{ZnO}}$
(100)	1	168	-0.040	0.051	0.077	1.500
(100)	41	726	-0.026	0.012	0.006	1.366
(110)	1	150	-0.049	0.076	0.077	1.125
(110)	239	714	-0.031	0.019	0.011	0.975
(111)	1	1278	-0.023	0.007	0.002	1.550
(111)	37	228	-0.022	0.077	0.113	1.800

Table 7.1: Summary of the properties of the selected structures in figures 7.13, 7.14 and 7.15. Number is the order in the structure list for that particular material sorted by E_{relaxed}.

Cu(100)

Reconstructed (a and b) and stable (c and d) Cu(100) slabs are presented in fig. 7.13. Interestingly, some Cu(100) surfaces convert to Cu(111) under the influence of stress and the interaction with the ZnO support. Notice that for the structure in a), the proportion of Cu to ZnO approaches that of Cu(111) (see table 7.1 and figures 7.11 and 7.12), which might explain why this conversion takes place. Cu(111) is the more stable configuration at this density, while the induced strain and the presence of ZnO allow the minimization to find a path between both structures.

Cu(110)

A reconstructed (a) and b)) and stable (c) and d)) Cu(110) slabs are presented in fig. 7.14. As opposed to the Cu(100) case, the reconstructed slab has transformed into a surface with kinks and steps instead of devolving into a Cu(111) surface. When compared to Cu(100), Cu(110) is farther away from Cu(111) in terms of lateral density and Cu to ZnO ratio. Additionally, the structure of Cu(110) is more "open", with atoms in the first layer farther apart from each other and atoms in the first subsurface layer (at the center of the rectangle that defines the surface) almost uncovered. Moreover, as explained in sec. 3.6.2 comparing the different candidate structures, Cu(110) already presents a geometry very similar to that of ZnO($10\bar{1}0$). All these factors together could explain why Cu(100) reconstructs into Cu(111), while Cu(110) instead tends to develop kinks and steps.

Cu(111)

A stable (a) and b)) and reconstructed (c) and d)) Cu(111) slabs are presented in fig. 7.15. As opposed to the previous two cases, the configuration that remains as Cu(111) is the more stable one (see table 7.1). The reconstructed surface once again appears when the slab surpasses a certain density/Cu to ZnO ratio, beyond the apparent stability range of Cu(111). Cu(111) is already the most stable surface of isolated copper, and as such appears to behave as an "attractor" point in the PES, with Cu(100) easily interconverting into Cu(111) when compressed enough.

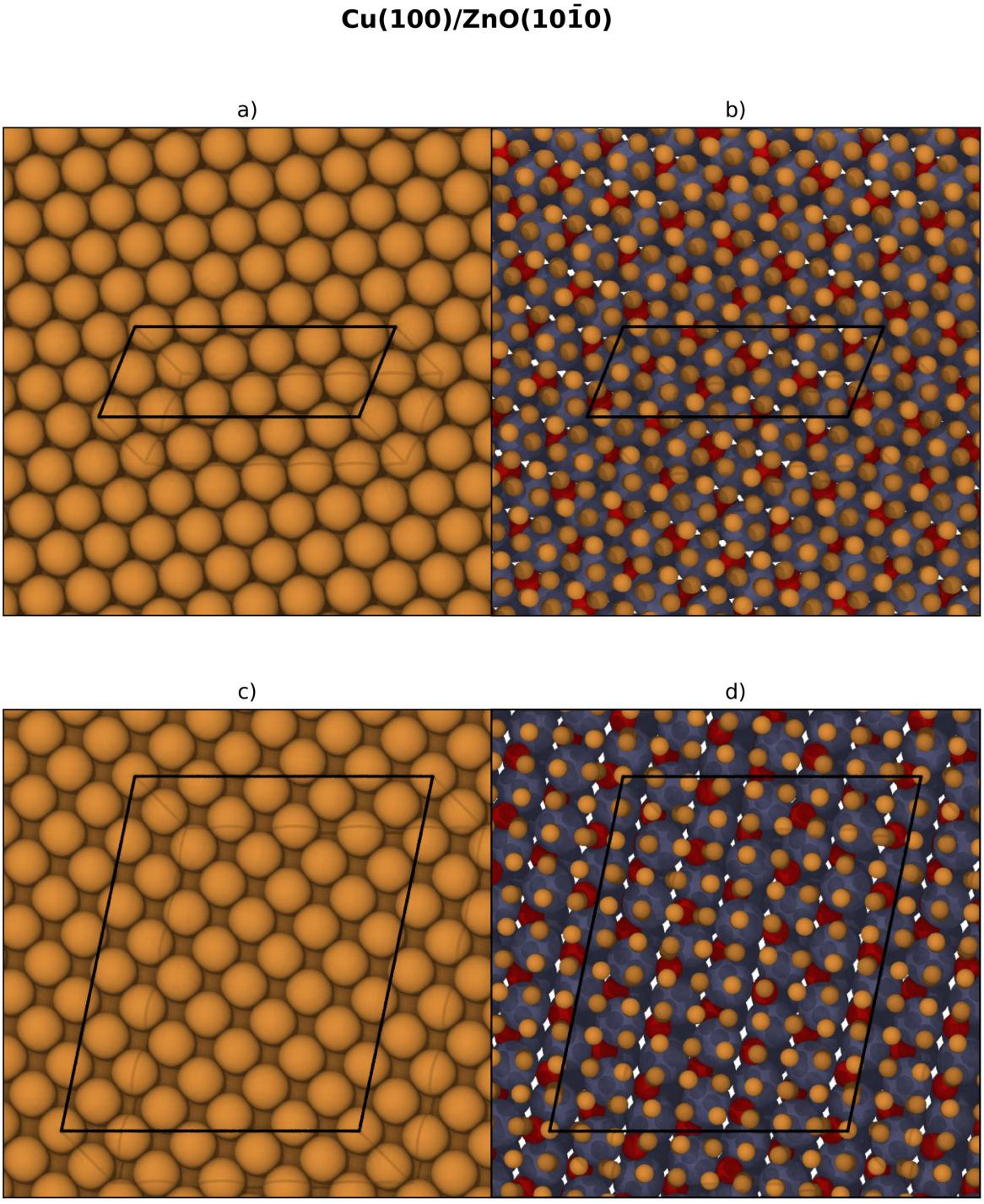

Figure 7.13: Two matches for Cu(100) and ZnO(10$\bar{1}$0), corresponding to structures 1 and 41 from the energetically ordered relaxed interfaces for the two materials, as detailed in table 7.1. a) The Cu surface reconstructs from (100) to (111) b) Presents a view of a) with smaller Cu atoms, which allows to observe the Moiré pattern between the materials. c) A stable Cu(100) configuration. d) Same as b) but for the structure in c).

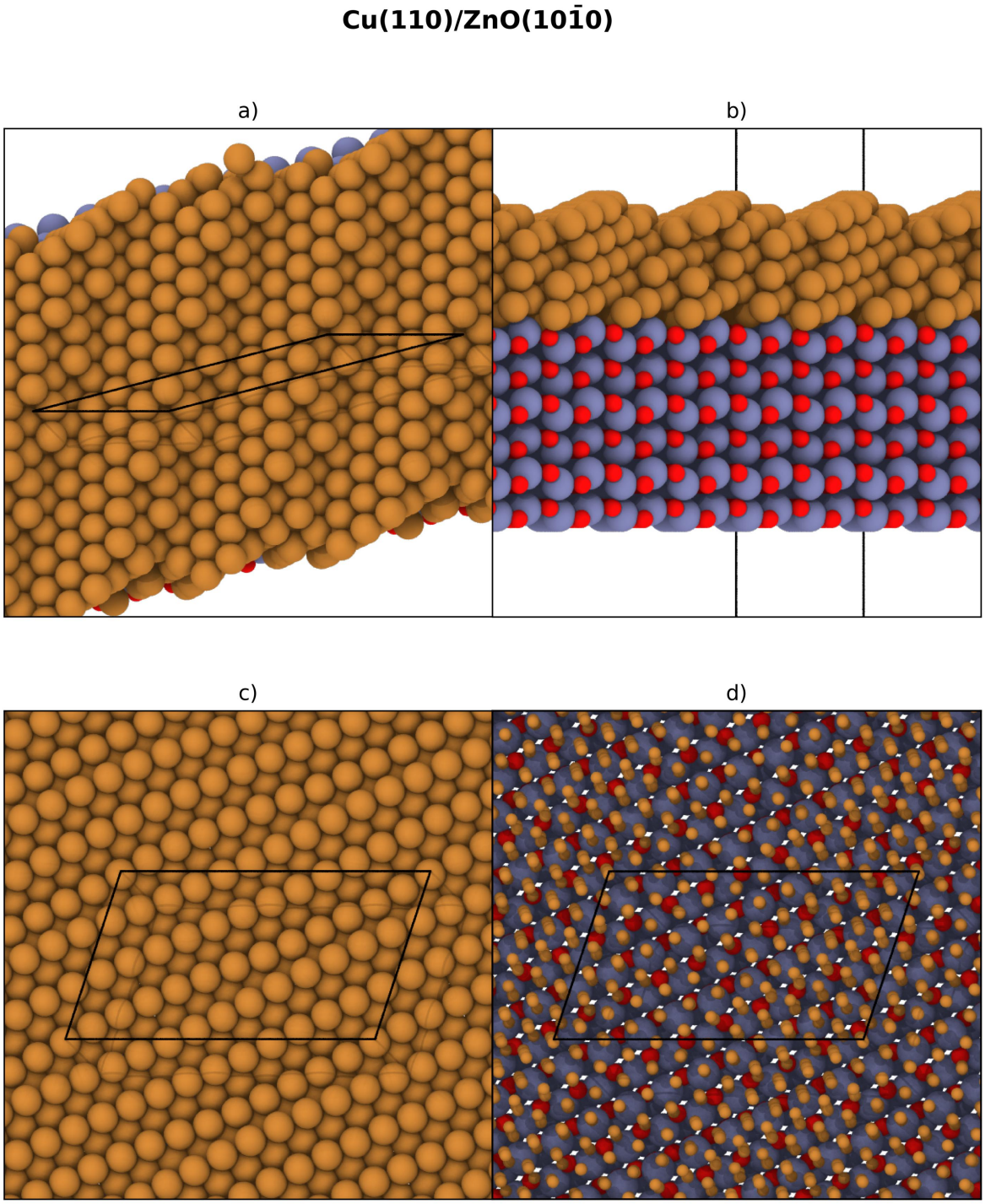

Figure 7.14: Two matches for Cu(110) and ZnO(10$\bar{1}$0) corresponding to structures 1 and 239 from the energetically ordered relaxed interfaces for the two materials, as detailed in table 7.1. a) The Cu surface reconstructs by atoms shifting in the Z-direction, forming a new interface with steps and kinks. b) Side view of a). c) A seemingly unreconstructed Cu(110) surface, but notice that the distances between the edges of the Cu(110) rectangles do not remain constant, forming a sort of zipper effect, as a way to alleviate strain. d) Presents a view of a) with smaller Cu atoms, which allows to observe the Moiré pattern between the materials.

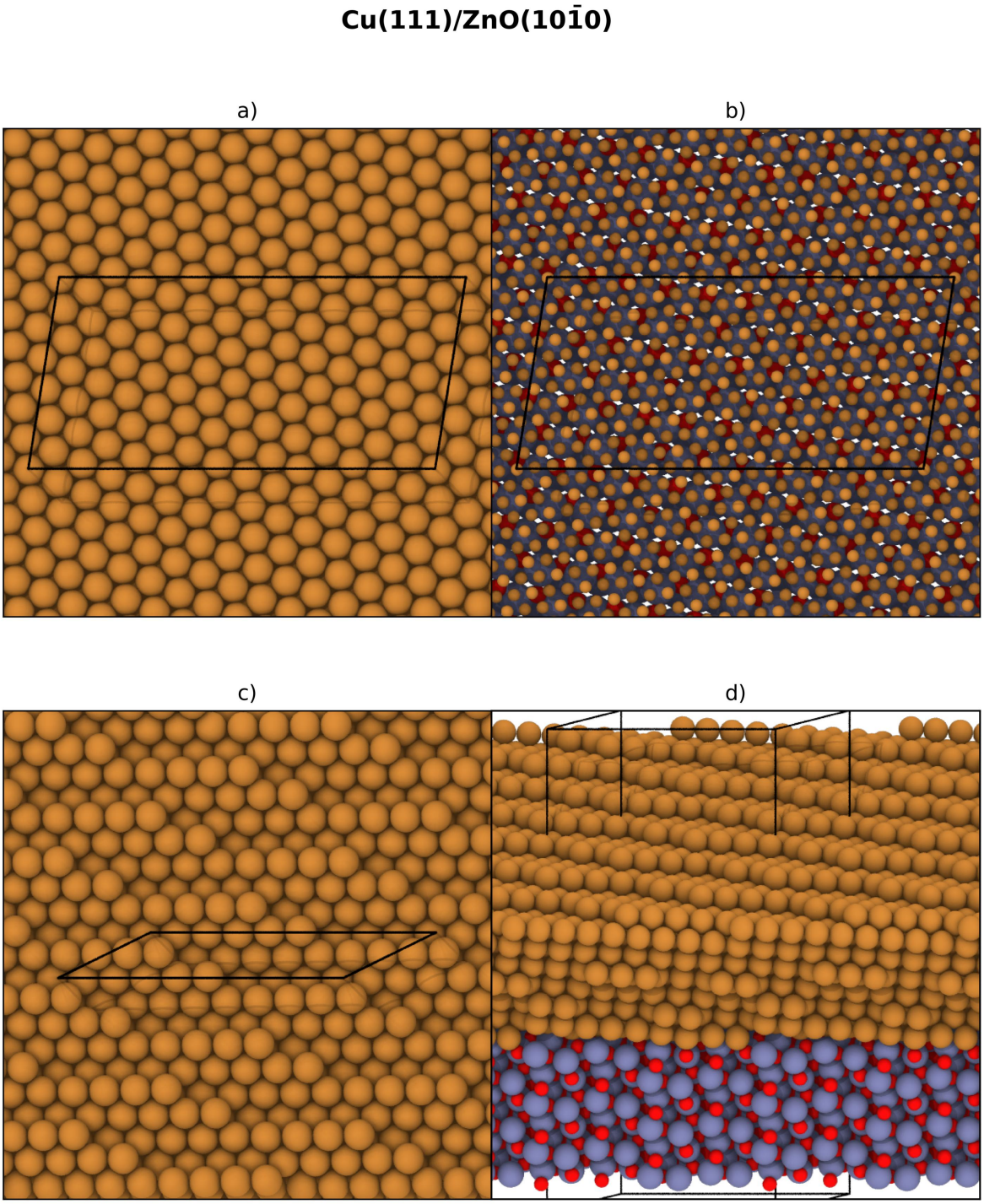

Figure 7.15: Two matches for Cu(111) and ZnO(10$\bar{1}$0) corresponding to structures 1 and 37 from the energetically ordered relaxed interfaces for the two materials, as detailed in table 7.1. a) A stable Cu (111) configuration. b) Presents a view of a) with smaller Cu atoms, which allows to observe the Moiré pattern between the materials. c) A reconstructed slab, with atoms shifting in the Z-direction to form a surface with steps and kinks. d) Side view of c).

7.3 Conclusions

An exhaustive, high-throughput search for the coincident interfaces between ZnO($10\bar{1}0$) and the three low Miller index surfaces of Cu has been performed. Analysis of the behavior of the matching algorithm shows that low levels of strain in the matched surfaces can only be achieved starting from a given slab size. This is a disadvantage since if simulations of low strain configurations are desired, structures with a large number of atoms are required.

Relaxing the matched structures leads to a number of interesting results. The described $\varepsilon_{\mathrm{DOLD}}$ and δ_{CD} parameters appear to be highly helpful in summarizing the strain suffered by each slab as well as the degree of reconstruction each one goes through. What at first seems to be the expected result, that structures with larger strain will result in higher energy configurations, is not actually observed. Low strain configurations are indeed stable, but large strains (in particular, large negative strains, which correspond to a reduced lattice size or equivalently a higher atom density than in normal conditions) tend to induce large reconstructions that stabilize the Cu structure into other configurations. This is particularly the case for Cu(100) and Cu(110), while Cu(111) appears to be more stable with fewer reconstructions. In fact, some Cu(100) structures with density overlapping that of Cu(111) transform into that surface. Two theories can be proposed as to why Cu(111) seems to be so resistant to reconstructions: it is well known that this Miller cut is already the most energetically stable Miller cut of copper (in isolation); and Cu(111) is also the densest Cu slab (in terms of surface density, that is, atoms per unit of surface area of the slab), which makes it harder for atoms to move past each other in a relaxation to reconstruct. That is, although the structure could represent an unstable maximum in the PES, there is no simple path to another local minimum that does not require atoms overlapping each other.

From these results a question that might arise: How can the "best" match for a given simulation be generated? Ideally, without the need to actually perform a relaxation on all the possible candidates. The results presented give us some rules of thumb that can be used to choose such a structure from the results of the CLM procedure. If we want our structure to not reconstruct (because we are trying to simulate a specific interface), one needs to pick a match:

1. with a low $\varepsilon_{\mathrm{DOLD}}$, ideally close to 0.0 but values below 0.03-0.04 seem to be safe in general

2. with expansive rather than compressive strain

3. with the largest amount of atoms that can be simulated with the method of choice

How do these interfaces compare with results for supported Cu clusters on ZnO? Global optimization of small Cu clusters show that they exhibit a preference for forming Cu(111) and (110) interfaces (actually a continuum between both surfaces) with the supporting oxide. Large coincident Cu(111) surfaces also exhibit in these results a large degree of stability, by usually not reconstructing, and are the most stable energetically if we refer the energy of the matches to the bulk configuration of the materials. Although the coincident Cu(110) matches tend to reconstruct, the unreconstructed configurations are the most stable of the three under study when compared to the free-standing slabs. Of course the comparison to clusters is not that straightforward. Small cluster have extra relaxation paths, since the interatomic distances in the X-Y direction are allowed to change when compared with the periodic matching lattices where this is not possible, and clusters present more interfaces to the vacuum, whereas the coincident interfaces only have one such interface. Even large clusters, where atoms far away from the edge of the cluster-support interface have a similar environment to that of the coincident cells, have other way of releasing strain. Results from simulated annealing runs of such large clusters show that every so often positions at the interface are "skipped", resulting in holes in the pattern of Cu atoms. This releases strain by relaxing the condition of having a uniformly strained material.

It would be interesting to study the behavior of different surfaces, such as the other stable low Miller index cut of ZnO($11\bar{2}0$), or stepped surfaces of Cu. These might also exhibit different results in terms of X-Y lateral scans.

The NNP is a good tool for treating this kind of system. Neural networks are also heavily and successfully used as classifiers [291, 292], where tags are assigned to a collection of objects. We can think of the structures generated by the CLM algorithm as a collection of closely related slabs that can be completely determined (classified) by a finite and unique number of parameters (initial materials, Miller indices, number of layers, distances between slabs, angle between materials, inflicted strain, atomic basis, etc.). These properties can be contained in a vector, and each match thus represents one point in this multi-dimensional configuration space. This space is much simpler than the one that could be created for a liquid or a cluster, for example. As such, this space is simple to sample and predict with a NNP based approach, due to its limited degrees of freedom. Of course the NNP approach does not have direct access to these tags, but they are implicitly encoded in the atomic structure of the slabs. In effect the generated structures are close in configuration space, but still easily distinguishable, an ideal case for a NNP fit.

Given the recent abundance of computational power and cloud storage, and the development of automated potential building tools such as NNPs, it is now possible to move towards hands-off construction of material databases. Previous examples include generating cluster collections with GA [199, 207, 208], evolutionary algorithms for solid structures [293] or automatic potential energy surface exploration combined with relevant structure selection [294]. For such a database an algorithm that is capable of generating physically relevant and "correct" structures is required. We propose that CLM in combination with a HDNNP (or other ML algorithms) is another strong candidate for such databases. The structures generated by CLM are usually "well behaved" (if the maximum allowed strained is limited), and such a database could help with the discovery of new 2D materials or interesting interfaces or new reconstructions. The MLP would be trained on the information from small matches (plus other known distortion methods as illustrated in 3.3), and then utilized to predict properties for larger matches.

Part III

Summary and Bibliography

Chapter 8

Summary

A neural network potential for the ternary copper-zinc-oxygen system has been constructed. This potential has been utilized to perform a variety of different simulations, from global optimization searches with a genetic algorithm (sec. 5), to partial optimization with simulated annealing molecular dynamics (sec. 6), and high-throughput analysis of matching Cu-ZnO surfaces (sec. 7). This covers the range from small clusters, via large macroscopic nanoparticles, to continuous interfaces, with the analysis of the results focusing on the interface between both materials. With this the goals set out by the project have been accomplished: to perform simulations beyond the simple small models that can fit within the computational limitations of an *ab-initio* calculation, and with this, to surpass what has been available until now in the literature for Cu/ZnO and better understand this catalyst.

The first set of results (sec. 5) corresponds to the global optimization of small copper clusters on the ZnO($10\bar{1}0$) surface. Previous studies on deposited copper clusters relied either on pre-designed clusters [68], clusters optimized in the vacuum [71], or parametrized force fields [67]. As such this is likely the first study generating unbiased cluster structures due to the utilized genetic algorithm, and based on reference DFT data. For this to be possible a number of methodological developments were necessary. GA optimization was not the first chosen tool, and previous attempts with basin hopping Monte Carlo [149] proved too inefficient to obtain converged results. For this purpose, the GA library for the ASE Python library has been extended, and linked to the N2P2 library for LAMMPS that provides force and energy calculations for the NNP. Two new algorithms related to configurational sampling were also required. The first one was the cube cut algorithm (sec. 2.6.1), designed to help with extracting smaller configurations that can fit into affordable electronic structure calculations, from the larger configurations required for properly periodic simulations. The second one was the bin and hash algorithm (sec. 4), intended to facilitate atomic environment selection and filtering. The BAH algorithm is complementary to the query by committee [88] approach of comparing the results of two NNPs trained on the same data, when presented with new data. BAH takes advantage of the hash table data structure to quickly find repeated (or unique) atomic environments in a dataset, as described by a given set of ACSF. This may seem trivial at first glance, but it is a subset of the well known problem of efficiently finding distances in multi-dimensional spaces. Despite the simplicity of the algorithm, it has been shown to be capable of effectively curating a structural database without loss of fit accuracy (sec. 4.3.4), estimating the quality of sets of atomic descriptors without requiring a costly fit (sec. 4.3.3), and finding potentially contradictory information in a dataset (sec. 4.3.6).

In spite of the small size of the optimized clusters, the first hints of trends in the interaction between Cu and the ZnO support can already be extracted. Besides revealing that small clusters prefer to interact with the support at oxygen positions through the Cu(110) and (111) surfaces (sec. 5.2.2), many structural growth patterns and families can be identified for the clusters across the different sizes (sec. 5.2.1). Being able to perform unbiased searches for supported clusters addresses the first question posed in the introduction to this thesis. For such small clusters, the cluster-interface interactions dominate over the cluster-cluster interactions. It is thus risky to extrapolate the observed trends to larger systems with only these results. For this reason, simulations with larger systems were also required, leading to the CLM and SA results. As shown in this work and previous research [207, 208], organized PES exploration algorithms such as GA are good candidates as tools for the generation of automated NNP datasets. The algorithms and examples developed in this work should help to advance this goal.

The second results section in this work presents the simulated annealing partial optimization of large supported Cu clusters on ZnO($10\bar{1}0$) (sec. 6), which expands on the results obtained from small clusters. Replicating the trends from the previous and following sections, large clusters are also found to interact mainly through forming intermediate Cu(111) and (110) surfaces (sec. 6.2.7), while exhibiting mostly Cu(111) and (100) facets on the cluster-vacuum interface (sec. 6.2.8). This agrees with previously reported experimental results such as those in ref. 60. Structural analysis reveals how much the ZnO support strains the clusters (sec. 6.2.5), by showing changes in the inter-atomic distances that reach the uppermost layers of the clusters far from the interface even for the largest sizes. The straining of the clusters could have consequences on the behavior of the supported clusters in experimental and reaction conditions, since many experimental studies of the industrial catalyst highlight the correlation between strain and reactivity [16, 17, 27]. The simulations also reveal the formation of multiple fcc domains separated by hcp regions (sec. 6.2.4), which was also observed in simulations of freestanding copper and brass nanoparticles utilizing the same potential [61, 92]. These partially optimized clusters finally reach the nanometer size range, which would make them good candidates for comparison against atomic deposition experiments such as those presented in refs. 18 and 60, in regards to shape, size, orientation, faceting, etc. For such a comparison it is necessary to look at the behavior of an ensemble of clusters and not just singular examples. Generation of multiple representative clusters has been facilitated in this case by the NNP, and the SA setup. This completes the second question proposed in the introduction.

To extend the analysis of the interface between Cu and ZnO to a continuous scale, high-throughput coincidence lattice match simulations were performed (sec. 7). Through an elaborate algorithm based on the geometry of the involved materials known as the coincidence lattice match [161], it was possible to generate thousands of candidate matching structures. This algorithm needed to be implemented from scratch in Python, in close association with the structure generation and database capabilities of the ASE library. In the future these scripts will be made available in a public code repository, with a goal of turning the code into an extra module for ASE.

From the trends across the three low Miller index surfaces of Cu matched with ZnO($10\bar{1}0$), it can be concluded that the (111) surface remains the most stable when put into contact with ZnO($10\bar{1}0$) (sec. 7.2.2), but also that Cu(110) is stabilized in combination with the ZnO support when compared to the freestanding surface. This is in accordance with the results observed from small and large clusters, where the Cu(110) and Cu(111) interfaces were preferred when interacting with the support. Additionally, this study has revealed interesting trends in how Cu reconstructs under high strain levels and deformations of the ideal Cu lattice (sec. 7.2.4). Here, Cu(111) is still the most stable configuration, rarely reconstructing except for high levels of strain, probably due to its relative stability and high atom density per surface area which prevents the Cu atoms from moving away from their lattice positions. Cu(100) often reconstructs into Cu(111) when subjected to high levels of strain, probably due to its relative instability after being deposited on ZnO when compared with the other two interfaces, and due to its proximity in terms of atom surface density to Cu(111). In this way, Cu(111) seems to act as an "attractor" in the Cu strain "phase diagram". Cu(110) also tends to reconstruct, but more often exhibits out-of-plane shifts of the Cu layers, instead of turning into Cu(111). A possible explanation for this is how exposed atoms in the first subsurface layer of Cu(110) are, which can lead to them easily slipping past neighboring lattice positions. Also once again looking at densities per surface area, Cu(110) is far away from Cu(111). This analysis of the interfaces rounds up the third question of the introduction, and is backed by results from the previous two simulation approaches.

Analyzing closely related slab configurations is another simulation *ansatz* ideal for NNPs, since the generated structures are all rather similar but still energetically distinct, and easy to generate programatically. Once again it is possible to draft a simulation approach that could automatically cycle through pairs of materials, attempting to discover new interfaces with interesting properties while automatically constructing a NNP for these materials.

To summarize, the more concrete results from the simulations presented in this work are:

- The interface between Cu clusters and ZnO($10\bar{1}0$) has a structure between that of Cu(111) and Cu(110), as proposed by some experimental results [60]. This is evident for both small clusters, large clusters, and initially flat interfaces.
- Related to the previous point, from the coincident surfaces results it can be derived that ZnO($10\bar{1}0$) stabilizes the Cu(111) and Cu(110) surfaces the most, to the point that Cu(110) becomes more stable than Cu(100). This could also be a factor for other metal/metal oxide interfaces.
- Strain and deviations from the equilibrium neighbor distance of Cu is present even for large supported nanoparticles, and for multiple layers. This is important since the experimental literature often reports a relationship between strain and catalytic activity [16, 17, 27].

Two other projects were developed in the course of this thesis, but are not detailed here. The first project is a new flexible grid based global optimization algorithm, shortly described in the introduction to sec. 5 and further explained in ref. 199. It would be interesting to compare in the future the performance of these grids when compared to the GA approach for the optimization of small supported clusters, particularly for sizes beyond 10 atoms. The second project is the study of the structure and composition of freestanding brass nanoparticles developed in the context of student practicals and a Master's thesis, and detailed in refs. 61 and 92. Results from this project showed how to successfully apply SA to partially optimized large clusters, and also evidenced that for brass alloys Zn tends to accumulate at the vacuum interface (the surface of the system) and at defects. This analysis could be contrasted in the future with the behavior of small supported globally optimized brass clusters, and large supported clusters simulated with a semi-grand canonical ensemble, for which some preliminary simulations are already available. The behavior of brass in these circumstances is important, since there are many experimental reports that brass may form in reaction conditions and thus be important for catalytic activity [33–35].

Much work can still be performed with the currently developed tools by extending the NNP dataset to different degrees. Some results are available for the global optimization of clusters beyond 10 atoms, up into the 20 to 30 atom range, where GA optimization appears to stop being as effective as could be desired. Preliminary simulations for global optimization and coincident lattice match analysis on the ZnO($11\bar{2}0$) surface have already been performed. Given the different geometries between this surface and the ZnO($10\bar{1}0$) surface analyzed in this work, these results could lead to a deeper understanding of the role of cluster and support interaction as mediated by support geometry and lattice mismatch. Similarly, some very early data is available for the global optimization of small ternary clusters, which could provide information on the formation of amorphous ZnO overlayers over Cu clusters [19]. Finally, some simulations are also available studying the formation of brass on large supported copper clusters utilizing the semi-grand canonical ensemble, as has already been tested in adjacent works with free-standing clusters [61, 92].

Chapter 9

Acknowledgments

First of all, I am particularly grateful to my supervisor Prof. Dr. Jörg Behler for giving me the opportunity to complete this work, and guiding me in its completion. Thanks also to Prof. Dr. Ricardo Mata in his role as second supervisor of this thesis.

I am particularly grateful for the many office and scientific conversations with Sebastian Wille, Tsz Wai Ko, and former group guest Dr. Nathan Daelman. I am also grateful to the students who I have collaborated with, and who have helped this and other projects with their work and knowledge: Jan Weinreich, Anton Römer, and Sebastian Thurm. I extend my thanks to the rest of the Behler and Mata groups at the Georg-August University Göttingen, as well as former members of the Behler group from the Ruhr University Bochum. I am beholden to Dr. Rainer Oswald for his technical support, and to the administrative staff of the university for multiple instances of help.

I would like to thank the Leibniz Rechenzentrum (LRZ) during the project Summer of Simulation 2016 and the Norddeutscher Verbund für Hoch- und Höchstleistungsrechnen (HLRN) under the project number nic00046 for providing computation time to generate much of the reference calculations contained in the NNP dataset. I am also grateful to the Deutsche Forschungsgemeinschaft (DFG) for funding as part of projects Be3264/10-1, project number 289217282 and INST186/1294-1 FUGG, project number 405832858.

Finally, I also thank Dr. Thorsten Teuteberg for providing the LaTeX template that forms the basis for the current document.

Bibliography

[1] F. Zaera, "New Challenges in Heterogeneous Catalysis for the 21st Century," Catalysis Letters **142**, 501–516 (2012).

[2] I. Fechete, Y. Wang, and J. C. Védrine, "The past, present and future of heterogeneous catalysis," Catalysis Today Catalytic Materials for Energy: Past, Present and Future, **189**, 2–27 (2012).

[3] J. M. Thomas and W. J. Thomas, *Principles and Practice of Heterogeneous Catalysis* (John Wiley & Sons, 2014).

[4] J. K. Nørskov, F. Studt, F. Abild-Pedersen, and T. Bligaard, *Fundamental Concepts in Heterogeneous Catalysis* (John Wiley & Sons, 2014).

[5] R. Schlögl, "Heterogeneous Catalysis," Angewandte Chemie International Edition **54**, 3465–3520 (2015).

[6] Z. Ma and F. Zaera, "Heterogeneous Catalysis by Metals," in *Encyclopedia of Inorganic Chemistry* (American Cancer Society, 2006).

[7] K. Oura, V. G. Lifshits, A. A. Saranin, A. V. Zotov, and M. Katayama, *Surface Science: An Introduction* (Springer Science & Business Media, 2013).

[8] L. Grajciar, C. J. Heard, A. A. Bondarenko, M. V. Polynski, J. Meeprasert, E. A. Pidko, and P. Nachtigall, "Towards operando computational modeling in heterogeneous catalysis," Chemical Society Reviews **47**, 8307–8348 (2018).

[9] K. C. Waugh, "Methanol Synthesis," Catalysis Letters **142**, 1153–1166 (2012).

[10] M. Behrens, F. Studt, I. Kasatkin, S. Kühl, M. Hävecker, F. Abild-Pedersen, S. Zander, F. Girgsdies, P. Kurr, B.-L. Kniep, M. Tovar, R. W. Fischer, J. K. Nørskov, and R. Schlögl, "The Active Site of Methanol Synthesis over $Cu/ZnO/Al_2O_3$ Industrial Catalysts," Science **336**, 893–897 (2012).

[11] J. Hu, W.-P. Guo, X.-R. Shi, B.-R. Li, and J. Wang, "Copper Deposition and Growth over ZnO Nonpolar $(10\bar{1}0)$ and $(11\bar{2}0)$ Surfaces: A Density Functional Theory Study," The Journal of Physical Chemistry C **113**, 7227–7235 (2009).

[12] E. B. M. Doesburg, R. H. Höppener, B. de Koning, X. Xiaoding, and J. J. F. Scholten, "Preparation and Characterization of Copper/Zinc Oxide/Alumina Catalysts for Methanol Synthesis," in *Studies in Surface Science and Catalysis*, Preparation of Catalysts IV, Vol. 31, edited by B. Delmon, P. Grange, P. A. Jacobs, and G. Poncelet (Elsevier, 1987) pp. 767–783.

[13] J. Słoczyński, R. Grabowski, A. Kozłowska, P. K. Olszewski, and J. Stoch, "Reduction kinetics of CuO in $CuO/ZnO/ZrO_2$ systems," Physical Chemistry Chemical Physics **5**, 4631–4640 (2003).

[14] M. Kurtz, N. Bauer, C. Büscher, H. Wilmer, O. Hinrichsen, R. Becker, S. Rabe, K. Merz, M. Driess, R. A. Fischer, and M. Muhler, "New Synthetic Routes to More Active Cu/ZnO Catalysts Used for Methanol Synthesis," Catalysis Letters **92**, 49–52 (2004).

[15] T. Ressler, B. L. Kniep, I. Kasatkin, and R. Schlögl, "The Microstructure of Copper Zinc Oxide Catalysts: Bridging the Materials Gap," Angewandte Chemie International Edition **44**, 4704–4707 (2005).

[16] I. Kasatkin, B. Kniep, and T. Ressler, "Cu/ZnO and Cu/ZrO_2 interactions studied by contact angle measurement with TEM," Physical Chemistry Chemical Physics **9**, 878–883 (2007).

[17] I. Kasatkin, P. Kurr, B. Kniep, A. Trunschke, and R. Schlögl, "Role of Lattice Strain and Defects in Copper Particles on the Activity of $Cu/ZnO/Al_2O_3$ Catalysts for Methanol Synthesis," Angewandte Chemie International Edition **46**, 7324–7327 (2007).

[18] M. Kroll, T. Löber, V. Schott, C. Wöll, and U. Köhler, "Thermal behavior of MOCVD-grown Cu-clusters on ZnO(100)," Physical Chemistry Chemical Physics **14**, 1654–1659 (2012).

[19] T. Lunkenbein, J. Schumann, M. Behrens, R. Schlögl, and M. G. Willinger, "Formation of a ZnO Overlayer in Industrial Cu/ZnO/Al_2O_3 Catalysts Induced by Strong Metal–Support Interactions," Angewandte Chemie **127**, 4627–4631 (2015).

[20] S. Sá, H. Silva, L. Brandão, J. M. Sousa, and A. Mendes, "Catalysts for methanol steam reforming—A review," Applied Catalysis B: Environmental **99**, 43–57 (2010).

[21] R. Burch, S. E. Golunski, and M. S. Spencer, "The role of copper and zinc oxide in methanol synthesis catalysts," Journal of the Chemical Society, Faraday Transactions **86**, 2683–2691 (1990).

[22] T. Fujitani, M. Saito, Y. Kanai, T. Kakumoto, T. Watanabe, J. Nakamura, and T. Uchijima, "The role of metal oxides in promoting a copper catalyst for methanol synthesis," Catalysis Letters **25**, 271–276 (1994).

[23] M. Spencer, "The role of zinc oxide in Cu/ZnO catalysts for methanol synthesis and the water–gas shift reaction," Topics in Catalysis **8**, 259 (1999).

[24] L. Martínez-Suárez, J. Frenzel, D. Marx, and B. Meyer, "Tuning the Reactivity of a Cu/ZnO Nanocatalyst via Gas Phase Pressure," Physical Review Letters **110**, 086108 (2013).

[25] M. Behrens, "Heterogeneous Catalysis of CO_2 Conversion to Methanol on Copper Surfaces," Angewandte Chemie International Edition **53**, 12022–12024 (2014).

[26] M. Behrens, "Promoting the Synthesis of Methanol: Understanding the Requirements for an Industrial Catalyst for the Conversion of CO_2," Angewandte Chemie International Edition **55**, 14906–14908 (2016).

[27] P. Kurr, I. Kasatkin, F. Girgsdies, A. Trunschke, R. Schlögl, and T. Ressler, "Microstructural characterization of Cu/ZnO/Al_2O_3 catalysts for methanol steam reforming - A comparative study," Applied Catalysis A: General **348**, 153–164 (2008).

[28] B. S. Clausen, B. Lengeler, and B. S. Rasmussen, "X-ray absorption spectroscopy study of Cu-based methanol catalysts. 1. Calcined state," J. Phys. Chem. **89:11** (1985), https://doi.org/10.1021/j100257a035.

[29] K. R. Harikumar and C. N. R. Rao, "Interaction of CO with CuZnO catalyst surfaces prepared in situ in the electron spectrometer: evidence for CO_2- and related species relevant to methanol synthesis," Applied Surface Science **125**, 245–249 (1998).

[30] J. D. Grunwaldt, A. M. Molenbroek, N. Y. Topsøe, H. Topsøe, and B. S. Clausen, "In Situ Investigations of Structural Changes in Cu/ZnO Catalysts," Journal of Catalysis **194**, 452–460 (2000).

[31] P. L. Hansen, J. B. Wagner, S. Helveg, J. R. Rostrup-Nielsen, B. S. Clausen, and H. Topsøe, "Atom-Resolved Imaging of Dynamic Shape Changes in Supported Copper Nanocrystals," Science **295**, 2053–2055 (2002).

[32] M. Duan, J. Yu, J. Meng, B. Zhu, Y. Wang, and Y. Gao, "Reconstruction of Supported Metal Nanoparticles in Reaction Conditions," Angewandte Chemie **130**, 6574–6579 (2018).

[33] T. Fujitani and J. Nakamura, "The chemical modification seen in the Cu/ZnO methanol synthesis catalysts," Applied Catalysis A: General **191**, 111–129 (2000).

[34] Y. Choi, K. Futagami, T. Fujitani, and J. Nakamura, "The role of ZnO in Cu/ZnO methanol synthesis catalysts - morphology effect or active site model?" Applied Catalysis A: General **208**, 163–167 (2001).

[35] S. Kuld, M. Thorhauge, H. Falsig, C. F. Elkjær, S. Helveg, I. Chorkendorff, and J. Sehested, "Quantifying the promotion of Cu catalysts by ZnO for methanol synthesis," Science **352**, 969–974 (2016).

[36] S. Kattel, P. J. Ramírez, J. G. Chen, J. A. Rodriguez, and P. Liu, "Active sites for CO_2 hydrogenation to methanol on Cu/ZnO catalysts," Science **355**, 1296–1299 (2017).

[37] S. Kattel, P. Liu, and J. G. Chen, "Tuning Selectivity of CO_2 Hydrogenation Reactions at the

Metal/Oxide Interface," Journal of the American Chemical Society **139**, 9739–9754 (2017).
[38] L. Martínez-Suárez, N. Siemer, J. Frenzel, and D. Marx, "Reaction Network of Methanol Synthesis over Cu/ZnO Nanocatalysts," ACS Catalysis **5**, 4201–4218 (2015).
[39] A. Janotti and C. G. V. d. Walle, "Fundamentals of zinc oxide as a semiconductor," Reports on Progress in Physics **72**, 126501 (2009).
[40] C. Wöll, "The chemistry and physics of zinc oxide surfaces," Progress in Surface Science **82**, 55–120 (2007).
[41] B. Meyer and D. Marx, "Density-functional study of the structure and stability of ZnO surfaces," Physical Review B **67**, 035403 (2003).
[42] O. Dulub, L. A. Boatner, and U. Diebold, "STM study of Cu growth on the ZnO($10\bar{1}0$) surface," Surface Science **504**, 271–281 (2002).
[43] Z. L. Wang, "Zinc oxide nanostructures: growth, properties and applications," Journal of Physics: Condensed Matter **16**, R829–R858 (2004).
[44] A. Moezzi, A. M. McDonagh, and M. B. Cortie, "Zinc oxide particles: Synthesis, properties and applications," Chemical Engineering Journal **185-186**, 1–22 (2012).
[45] B. Wang, S. Nagase, J. Zhao, and G. Wang, "Structural Growth Sequences and Electronic Properties of Zinc Oxide Clusters (ZnO)n (n=2-18)," The Journal of Physical Chemistry C **111**, 4956–4963 (2007).
[46] A. Kołodziejczak-Radzimska and T. Jesionowski, "Zinc Oxide - From Synthesis to Application: A Review," Materials **7**, 2833–2881 (2014).
[47] V. E. Henrich and P. A. Cox, *The Surface Science of Metal Oxides* (Cambridge University Press, 1996).
[48] J. R. Davis, *Copper and Copper Alloys* (ASM International, 2001).
[49] N. A. Dhas, C. P. Raj, and A. Gedanken, "Synthesis, Characterization, and Properties of Metallic Copper Nanoparticles," Chemistry of Materials **10**, 1446–1452 (1998).
[50] P. K. Khanna, S. Gaikwad, P. V. Adhyapak, N. Singh, and R. Marimuthu, "Synthesis and characterization of copper nanoparticles," Materials Letters **61**, 4711–4714 (2007).
[51] B. Khodashenas and H. R. Ghorbani, "Synthesis of copper nanoparticles : An overview of the various methods," Korean Journal of Chemical Engineering **31**, 1105–1109 (2014).
[52] M. I. Din and R. Rehan, "Synthesis, Characterization, and Applications of Copper Nanoparticles," Analytical Letters **50**, 50–62 (2017).
[53] L. V. Koplitz, O. Dulub, and U. Diebold, "STM Study of Copper Growth on ZnO(0001)-Zn and ZnO($000\bar{1}$)-O Surfaces," The Journal of Physical Chemistry B **107**, 10583–10590 (2003).
[54] O. Dulub, M. Batzill, and U. Diebold, "Growth of Copper on Single Crystalline ZnO: Surface Study of a Model Catalyst," Topics in Catalysis **36**, 65–76 (2005).
[55] K. Ozawa, T. Sato, Y. Oba, and K. Edamoto, "Electronic Structure of Cu on ZnO($10\bar{1}0$): Angle-Resolved Photoemission Spectroscopy Study," The Journal of Physical Chemistry C **111**, 4256–4263 (2007).
[56] M. Kroll and U. Köhler, "Small Cu-clusters on ZnO(0001)–Zn: Nucleation and annealing behavior," Surface Science **601**, 2182–2188 (2007).
[57] M. Ay, A. Nefedov, A. Remhof, and H. Zabel, "Structural properties of Cu clusters on the O-terminated ZnO ($000\bar{1}$) surface," Applied Surface Science **226**, 405–411 (2004).
[58] H. Qiu, F. Gallino, C. Di Valentin, and Y. Wang, "Shallow Donor States Induced by In-Diffused Cu in ZnO: A Combined HREELS and Hybrid DFT Study," Physical Review Letters **106**, 066401 (2011).
[59] I. Beinik, M. Hellström, T. N. Jensen, P. Broqvist, and J. V. Lauritsen, "Enhanced wetting of Cu on ZnO by migration of subsurface oxygen vacancies," Nature Communications **6**, 8845 (2015).
[60] U. Köhler, M. Kroll, T. Löber, A. Birkner, V. Schott, and C. Wöll, "The thermally induced interaction of Cu and Au with ZnO single crystal surfaces," physica status solidi (b) **250**, 1222–1234 (2013).

[61] J. Weinreich, A. Römer, M. L. Paleico, and J. Behler, "Properties of alpha-Brass Nanoparticles. 1. Neural Network Potential Energy Surface," The Journal of Physical Chemistry C **124**, 12682–12695 (2020).

[62] B. Meyer and D. Marx, "Density-functional study of Cu atoms, monolayers, films, and coadsorbates on polar ZnO surfaces," Physical Review B **69**, 235420 (2004).

[63] S. A. French, A. A. Sokol, C. R. A. Catlow, and P. Sherwood, "The Growth of Copper Clusters over ZnO: the Competition between Planar and Polyhedral Clusters," The Journal of Physical Chemistry C **112**, 7420–7430 (2008).

[64] K. Reuter and M. Scheffler, "First-Principles Atomistic Thermodynamics for Oxidation Catalysis: Surface Phase Diagrams and Catalytically Interesting Regions," Physical Review Letters **90**, 046103 (2003).

[65] B. Meyer, "First-principles study of the polar O-terminated ZnO surface in thermodynamic equilibrium with oxygen and hydrogen," Physical Review B **69**, 045416 (2004).

[66] Y.-T. Cheng, T. Liang, X. Nie, K. Choudhary, S. R. Phillpot, A. Asthagiri, and S. B. Sinnott, "Cu cluster deposition on ZnO $(10\bar{1}0)$: Morphology and growth mode predicted from molecular dynamics simulations," Surface Science **621**, 109–116 (2014).

[67] D. Mora-Fonz, T. Lazauskas, S. M. Woodley, S. T. Bromley, C. R. A. Catlow, and A. A. Sokol, "Development of Interatomic Potentials for Supported Nanoparticles: The Cu/ZnO Case," The Journal of Physical Chemistry C **121**, 16831–16844 (2017).

[68] M. Hellström, D. Spångberg, K. Hermansson, and P. Broqvist, "Small Cu Clusters Adsorbed on ZnO(1010) Show Even-Odd Alternations in Stability and Charge Transfer," The Journal of Physical Chemistry C **118**, 6480–6490 (2014).

[69] J. Martinez, T. Liang, S. B. Sinnott, and S. R. Phillpot, "A third-generation charge optimized many body (COMB3) potential for nitrogen-containing organic molecules," Computational Materials Science **139**, 153–161 (2017).

[70] M. Hellström, D. Spångberg, P. Broqvist, and K. Hermansson, "Water-Induced Oxidation and Dissociation of Small Cu Clusters on ZnO$(10\bar{1}0)$," The Journal of Physical Chemistry C **119**, 1382–1390 (2015).

[71] Q. Wan, F. Wei, Y. Wang, F. Wang, L. Zhou, S. Lin, D. Xie, and H. Guo, "Single atom detachment from Cu clusters, and diffusion and trapping on $CeO_2(111)$: implications in Ostwald ripening and atomic redispersion," Nanoscale **10**, 17893–17901 (2018).

[72] M. Higham, D. Mora-Fonz, A. A. Sokol, S. M. Woodley, and C. R. A. Catlow, "Morphology of Cu clusters supported on reconstructed polar ZnO (0001) and $(000\bar{1})$ surfaces," Journal of Materials Chemistry A **8**, 22840–22857 (2020).

[73] X. W. Zhou, H. N. G. Wadley, J.-S. Filhol, and M. N. Neurock, "Modified charge transfer–embedded atom method potential for metal/metal oxide systems," Physical Review B **69**, 035402 (2004).

[74] A. C. T. van Duin, V. S. Bryantsev, M. S. Diallo, W. A. Goddard, O. Rahaman, D. J. Doren, D. Raymand, and K. Hermansson, "Development and Validation of a ReaxFF Reactive Force Field for Cu Cation/Water Interactions and Copper Metal/Metal Oxide/Metal Hydroxide Condensed Phases," The Journal of Physical Chemistry A **114**, 9507–9514 (2010).

[75] F. H. Stillinger and T. A. Weber, "Computer simulation of local order in condensed phases of silicon," Physical Review B **31**, 5262–5271 (1985).

[76] M. S. Daw and M. I. Baskes, "Semiempirical, Quantum Mechanical Calculation of Hydrogen Embrittlement in Metals," Physical Review Letters **50**, 1285–1288 (1983).

[77] K. Chenoweth, A. C. T. van Duin, and W. A. Goddard, "ReaxFF Reactive Force Field for Molecular Dynamics Simulations of Hydrocarbon Oxidation," The Journal of Physical Chemistry A **112**, 1040–1053 (2008).

[78] I. G. Shuttleworth, "Development of the ReaxFF Reactive Force-Field Description of Gold Oxides," The Journal of Physical Chemistry C **121**, 25255–25270 (2017).

[79] M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, T. Frauenheim, S. Suhai, and G. Seifert, "Self-consistent-charge density-functional tight-binding method for simulations of complex materials properties," Physical Review B **58**, 7260–7268 (1998).

[80] T. Frauenheim, G. Seifert, M. Elsterner, Z. Hajnal, G. Jungnickel, D. Porezag, S. Suhai, and R. Scholz, "A Self-Consistent Charge Density-Functional Based Tight-Binding Method for Predictive Materials Simulations in Physics, Chemistry and Biology," physica status solidi (b) **217**, 41–62 (2000).

[81] N. H. Moreira, G. Dolgonos, B. Aradi, A. L. da Rosa, and T. Frauenheim, "Toward an Accurate Density-Functional Tight-Binding Description of Zinc-Containing Compounds," Journal of Chemical Theory and Computation **5**, 605–614 (2009).

[82] M. Gaus, H. Jin, D. Demapan, A. S. Christensen, P. Goyal, M. Elstner, and Q. Cui, "DFTB3 Parametrization for Copper: The Importance of Orbital Angular Momentum Dependence of Hubbard Parameters," Journal of Chemical Theory and Computation **11**, 4205–4219 (2015).

[83] M. Van den Bossche, "DFTB-Assisted Global Structure Optimization of 13- and 55-Atom Late Transition Metal Clusters," The Journal of Physical Chemistry A **123**, 3038–3045 (2019).

[84] G. Zheng, H. A. Witek, P. Bobadova-Parvanova, S. Irle, D. G. Musaev, R. Prabhakar, K. Morokuma, M. Lundberg, M. Elstner, C. Köhler, and T. Frauenheim, "Parameter Calibration of Transition-Metal Elements for the Spin-Polarized Self-Consistent-Charge Density-Functional Tight-Binding (DFTB) Method: Sc, Ti, Fe, Co, and Ni," Journal of Chemical Theory and Computation **3**, 1349–1367 (2007).

[85] C. M. Handley and P. L. A. Popelier, "Potential Energy Surfaces Fitted by Artificial Neural Networks," The Journal of Physical Chemistry A **114**, 3371–3383 (2010).

[86] J. Behler, "Perspective: Machine learning potentials for atomistic simulations," The Journal of Chemical Physics **145**, 170901 (2016).

[87] V. Botu, R. Batra, J. Chapman, and R. Ramprasad, "Machine Learning Force Fields: Construction, Validation, and Outlook," The Journal of Physical Chemistry C **121**, 511–522 (2017).

[88] N. Artrith and J. Behler, "High-dimensional neural network potentials for metal surfaces: A prototype study for copper," Physical Review B **85**, 045439 (2012).

[89] N. Artrith, T. Morawietz, and J. Behler, "High-dimensional neural-network potentials for multicomponent systems: Applications to zinc oxide," Physical Review B **83**, 153101 (2011).

[90] S. Kondati Natarajan and J. Behler, "Self-Diffusion of Surface Defects at Copper–Water Interfaces," The Journal of Physical Chemistry C **121**, 4368–4383 (2017).

[91] V. Quaranta, M. Hellström, and J. Behler, "Proton-Transfer Mechanisms at the Water–ZnO Interface: The Role of Presolvation," The Journal of Physical Chemistry Letters **8**, 1476–1483 (2017).

[92] J. Weinreich, M. L. Paleico, and J. Behler, "Properties of alpha-Brass Nanoparticles. 2: Structure and Composition," The Journal of Physical Chemistry C **125**, 14897–14909 (2021).

[93] N. Artrith, B. Hiller, and J. Behler, "Neural network potentials for metals and oxides – First applications to copper clusters at zinc oxide," physica status solidi (b) **250**, 1191–1203 (2013).

[94] M. Born and R. Oppenheimer, "Zur Quantentheorie der Molekeln," Annalen der Physik **389**, 457–484 (1927).

[95] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory* (Courier Corporation, 1996).

[96] P. Hohenberg and W. Kohn, "Inhomogeneous Electron Gas," Physical Review **136**, B864–B871 (1964).

[97] W. Kohn and L. J. Sham, "Self-Consistent Equations Including Exchange and Correlation Effects," Physical Review **140**, A1133–A1138 (1965).

[98] W. Koch and M. C. Holthausen, *A Chemist's Guide to Density Functional Theory* (John Wiley & Sons, 2015).

[99] J. P. Perdew and K. Schmidt, "Jacob's ladder of density functional approximations for the

exchange-correlation energy," AIP Conference Proceedings **577**, 1–20 (2001).
[100] T. M. Mitchell, *Machine Learning* (McGraw-Hill, 1997).
[101] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," Neural Computation **29**, 2352–2449 (2017).
[102] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," Nature **550**, 354–359 (2017).
[103] K.-H. Ott, N. Aranibar, B. Singh, and G. W. Stockton, "Metabonomics classifies pathways affected by bioactive compounds. Artificial neural network classification of NMR spectra of plant extracts," Phytochemistry Plant Metabolomics, **62**, 971–985 (2003).
[104] J. Zupan and J. Gasteiger, *Neural Networks in Chemistry and Drug Design: An Introduction* (Wiley, 1999).
[105] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," The bulletin of mathematical biophysics **5**, 115–133 (1943).
[106] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review **65**, 386–408 (1958).
[107] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry* (MIT Press, 2017).
[108] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. thesis, Harvard University (1975).
[109] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," Neural Networks **3**, 551–560 (1990).
[110] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," The Journal of Chemical Physics **134**, 074106 (2011).
[111] J. Behler, "Representing potential energy surfaces by high-dimensional neural network potentials," Journal of Physics: Condensed Matter **26**, 183001 (2014).
[112] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, "Neural network models of potential energy surfaces," The Journal of Chemical Physics **103**, 4129–4137 (1995).
[113] T. W. Ko, J. A. Finkler, S. Goedecker, and J. Behler, "General-Purpose Machine Learning Potentials Capturing Non-local Charge Transfer," Submitted (2020).
[114] J. Behler and M. Parrinello, "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces," Physical Review Letters **98**, 146401 (2007).
[115] J. Behler, "Constructing high-dimensional neural network potentials: A tutorial review," International Journal of Quantum Chemistry **115**, 1032–1050 (2015).
[116] J. Behler, "First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems," Angewandte Chemie International Edition **56**, 12828–12840 (2017).
[117] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "SchNet – A deep learning architecture for molecules and materials," The Journal of Chemical Physics **148**, 241722 (2018).
[118] O. T. Unke and M. Meuwly, "PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges," Journal of Chemical Theory and Computation **15**, 3678–3693 (2019).
[119] J. S. Smith, O. Isayev, and A. E. Roitberg, "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost," Chemical Science **8**, 3192–3203 (2017).
[120] B. Jiang and H. Guo, "Permutation invariant polynomial neural network approach to fitting potential energy surfaces," The Journal of Chemical Physics **139**, 054112 (2013).
[121] A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," Physical Review B **87**, 184115 (2013).

[122] S. Jindal, S. Chiriki, and S. S. Bulusu, "Spherical harmonics based descriptor for neural network potentials: Structure and dynamics of Au_{147} nanocluster," The Journal of Chemical Physics **146**, 204301 (2017).

[123] J. Jenke, A. P. A. Subramanyam, M. Densow, T. Hammerschmidt, D. G. Pettifor, and R. Drautz, "Electronic structure based descriptor for characterizing local atomic environments," Physical Review B **98**, 144102 (2018).

[124] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, and P. Marquetand, "wACSF—Weighted atom-centered symmetry functions as descriptors in machine learning potentials," The Journal of Chemical Physics **148**, 241709 (2018).

[125] W. Pronobis, A. Tkatchenko, and K.-R. Müller, "Many-Body Descriptors for Predicting Molecular Properties with Machine Learning: Analysis of Pairwise and Three-Body Interactions in Molecules," Journal of Chemical Theory and Computation **14**, 2991–3003 (2018).

[126] F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld, "Alchemical and structural distribution based representation for universal quantum machine learning," The Journal of Chemical Physics **148**, 241717 (2018).

[127] E. Kocer, J. K. Mason, and H. Erturk, "A novel approach to describe chemical environments in high-dimensional neural network potentials," The Journal of Chemical Physics **150**, 154102 (2019).

[128] B. Parsaeifard, D. S. De, A. S. Christensen, F. A. Faber, E. Kocer, S. De, J. Behler, A. von Lilienfeld, and S. Goedecker, "An assessment of the structural resolution of various fingerprints commonly used in machine learning," arXiv:2008.03189 [cond-mat, physics:physics] (2020).

[129] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, "Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons," Physical Review Letters **104**, 136403 (2010).

[130] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker, "Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials," Journal of Computational Physics **285**, 316–330 (2015).

[131] A. V. Shapeev, "Moment Tensor Potentials: A Class of Systematically Improvable Interatomic Potentials," Multiscale Modeling & Simulation **14**, 1153–1173 (2016).

[132] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," IEEE Control Systems Magazine **10**, 18–23 (1990).

[133] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering **82**, 35–45 (1960).

[134] S. K. Natarajan, T. Morawietz, and J. Behler, "Representing the potential-energy surface of protonated water clusters by high-dimensional neural network potentials," Physical Chemistry Chemical Physics **17**, 8356–8371 (2015).

[135] S. K. Natarajan and J. Behler, "Neural network molecular dynamics simulations of solid–liquid interfaces: water at low-index copper surfaces," Physical Chemistry Chemical Physics **18**, 28704–28725 (2016).

[136] V. Quaranta, J. Behler, and M. Hellström, "Structure and Dynamics of the Liquid–Water/Zinc-Oxide Interface from Machine Learning Potential Simulations," The Journal of Physical Chemistry C **123**, 1293–1304 (2019).

[137] M. Eckhoff and J. Behler, "From Molecular Fragments to the Bulk: Development of a Neural Network Potential for MOF-5," Journal of Chemical Theory and Computation **15**, 3793–3809 (2019).

[138] M. L. Paleico and J. Behler, "Global optimization of copper clusters at the $ZnO(10\bar{1}0)$ surface using a DFT-based neural network potential and genetic algorithms," The Journal of Chemical Physics **153**, 054704 (2020).

[139] Z. Michalewicz and C. Z. Janikow, "Genetic algorithms for numerical optimization," Statistics and Computing **1**, 75–91 (1991).

[140] D. M. Deaven and K. M. Ho, "Molecular Geometry Optimization with a Genetic Algorithm," Physical Review Letters **75**, 288–291 (1995).

[141] L. B. Vilhelmsen and B. Hammer, "Systematic Study of Au_6 to Au_{12} Gold Clusters on MgO(100) F Centers Using Density-Functional Theory," Physical Review Letters **108**, 126101 (2012).

[142] L. B. Vilhelmsen and B. Hammer, "Identification of the Catalytic Site at the Interface Perimeter of Au Clusters on Rutile TiO_2(110)," ACS Catalysis **4**, 1626–1631 (2014).

[143] P. Huang, Y. Jiang, T. Liang, E. Wu, J. Li, and J. Hou, "Structural exploration of Au_xM- (M = Si, Ge, Sn; x = 9–12) clusters with a revised genetic algorithm," RSC Advances **9**, 7432–7439 (2019).

[144] F. Buendía, J. A. Vargas, R. L. Johnston, and M. R. Beltrán, "Study of the stability of small AuRh clusters found by a Genetic Algorithm methodology," Computational and Theoretical Chemistry **1119**, 51–58 (2017).

[145] S. Heydariyan, M. R. Nouri, M. Alaei, Z. Allahyari, and T. A. Niehaus, "New candidates for the global minimum of medium-sized silicon clusters: A hybrid DFTB/DFT genetic algorithm applied to Si_n, n = 8-80," The Journal of Chemical Physics **149**, 074313 (2018).

[146] E. Bozkurt, M. A. S. Perez, R. Hovius, N. J. Browning, and U. Rothlisberger, "Genetic Algorithm Based Design and Experimental Characterization of a Highly Thermostable Metalloprotein," Journal of the American Chemical Society **140**, 4517–4521 (2018).

[147] P. B. Jensen, S. Lysgaard, U. J. Quaade, and T. Vegge, "Designing mixed metal halide ammines for ammonia storage using density functional theory and genetic algorithms," Physical Chemistry Chemical Physics **16**, 19732–19740 (2014).

[148] M. S. Jørgensen, U. F. Larsen, K. W. Jacobsen, and B. Hammer, "Exploration versus Exploitation in Global Atomistic Structure Optimization," The Journal of Physical Chemistry A **122**, 1504–1509 (2018).

[149] D. J. Wales and J. P. K. Doye, "Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms," The Journal of Physical Chemistry A **101**, 5111–5116 (1997).

[150] L. B. Vilhelmsen and B. Hammer, "A genetic algorithm for first principles global structure optimization of supported nano structures," The Journal of Chemical Physics **141**, 044711 (2014).

[151] G. G. Rondina and J. L. F. Da Silva, "Revised Basin-Hopping Monte Carlo Algorithm for Structure Optimization of Clusters and Nanoparticles," Journal of Chemical Information and Modeling **53**, 2282–2298 (2013).

[152] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," Science **220**, 671–680 (1983).

[153] S. Goedecker, "Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems," The Journal of Chemical Physics **120**, 9911–9917 (2004).

[154] L. B. Vilhelmsen, K. S. Walton, and D. S. Sholl, "Structure and Mobility of Metal Clusters in MOFs: Au, Pd, and AuPd Clusters in MOF-74," Journal of the American Chemical Society **134**, 12807–12816 (2012).

[155] R. L. Johnston, "Evolving better nanoparticles: Genetic algorithms for optimising cluster geometries," Dalton Transactions , 4193–4207 (2003).

[156] D. Frenkel and B. Smit, *Understanding Molecular Simulations* (Academic Press, 2002).

[157] R. A. Andrievski, "Size-dependent effects in properties of nanostructured materials," Reviews on Advanced Materials Science **21**, 107–133 (2009).

[158] R. L. Johnston, "Chapter 1 - Metal Nanoparticles and Nanoalloys," in *Frontiers of Nanoscience*, Metal Nanoparticles and Nanoalloys, Vol. 3, edited by R. L. Johnston and J. P. Wilcoxon (Elsevier, 2012) pp. 1–42.

[159] J. P. Wilcoxon, "Chapter 2 - Nanoparticles—Preparation, Characterization and Physical Proper-

ties," in *Frontiers of Nanoscience*, Metal Nanoparticles and Nanoalloys, Vol. 3, edited by R. L. Johnston and J. P. Wilcoxon (Elsevier, 2012) pp. 43–127.

[160] F. Ding, A. Rosén, S. Curtarolo, and K. Bolton, "Modeling the melting of supported clusters," Applied Physics Letters **88**, 133110 (2006).

[161] D. S. Koda, F. Bechstedt, M. Marques, and L. K. Teles, "Coincidence Lattices of 2D Crystals: Heterostructure Predictions and Applications," The Journal of Physical Chemistry C **120**, 10895–10908 (2016).

[162] M. Hazewinkel, ed., *Encyclopaedia of Mathematics: Volume 3*, Encyclopaedia of Mathematics (Springer Netherlands, 1989).

[163] E. Tasci, G. de la Flor, D. Orobengoa, C. Capillas, J. Perez-Mato, and M. Aroyo, "An introduction to the tools hosted in the Bilbao Crystallographic Server," EPJ Web of Conferences **22**, 00009 (2012).

[164] G. d. l. Flor, D. Orobengoa, E. Tasci, J. M. Perez-Mato, and M. I. Aroyo, "Comparison of structures applying the tools available at the Bilbao Crystallographic Server," Journal of Applied Crystallography **49**, 653–664 (2016).

[165] "Strain Tensor Calculation," (2020), `https://www.cryst.ehu.es/cryst/strain.html`.

[166] J. L. Schlenker, G. V. Gibbs, and M. B. Boisen, "Strain-tensor components expressed in terms of lattice parameters," Acta Crystallographica Section A **34**, 52–54 (1978).

[167] M. Catti, "Calculation of elastic constants by the method of crystal static deformation," Acta Crystallographica Section A **41**, 494–500 (1985).

[168] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, and G. Ceder, "Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis," Computational Materials Science **68**, 314–319 (2013).

[169] "pymatgen.analysis.elasticity.strain module — pymatgen 2020.9.14 documentation," (2020), `https://pymatgen.org/pymatgen.analysis.elasticity.strain.html?highlight=strain#module-pymatgen.analysis.elasticity.strain`.

[170] X. Peng and J. Cao, "A continuum mechanics-based non-orthogonal constitutive model for woven composite fabrics," Composites Part A: Applied Science and Manufacturing **36**, 859–874 (2005).

[171] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dulak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—a Python library for working with atoms," Journal of Physics: Condensed Matter **29**, 273002 (2017).

[172] Y. Zhou, M. Karplus, K. D. Ball, and R. S. Berry, "The distance fluctuation criterion for melting: Comparison of square-well and Morse potential models for clusters and homopolymers," The Journal of Chemical Physics **116**, 2323–2329 (2002).

[173] M. Zhang, M. Y. Efremov, F. Schiettekatte, E. A. Olson, A. T. Kwan, S. L. Lai, T. Wisleder, J. E. Greene, and L. H. Allen, "Size-dependent melting point depression of nanostructures: Nanocalorimetric measurements," Physical Review B **62**, 10548–10557 (2000).

[174] H. Jiang, K.-s. Moon, H. Dong, F. Hua, and C. P. Wong, "Size-dependent melting properties of tin nanoparticles," Chemical Physics Letters **429**, 492–496 (2006).

[175] J. Sun and S. L. Simon, "The melting behavior of aluminum nanoparticles," Thermochimica Acta Chemical Thermodynamics and Thermal Analysis, **463**, 32–40 (2007).

[176] G. Höhne, W. F. Hemminger, and H.-J. Flammersheim, *Differential Scanning Calorimetry* (Springer Science & Business Media, 2013).

[177] P. M. Larsen, S. Schmidt, and J. Schiøtz, "Robust Structural Identification via Polyhedral Template Matching," Modelling and Simulation in Materials Science and Engineering **24**, 055007

(2016).
[178] A. Stukowski, "Structure identification methods for atomistic simulations of crystalline materials," Modelling and Simulation in Materials Science and Engineering **20**, 045021 (2012).
[179] A. Stukowski, "Computational Analysis Methods in Atomistic Modeling of Crystals," JOM **66**, 399–407 (2014).
[180] A. Stukowski, "Visualization and analysis of atomistic simulation data with OVITO–the Open Visualization Tool," Modelling and Simulation in Materials Science and Engineering **18**, 015012 (2009).
[181] G. Kresse and J. Furthmüller, "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set," Physical Review B **54**, 11169–11186 (1996).
[182] G. Kresse and D. Joubert, "From ultrasoft pseudopotentials to the projector augmented-wave method," Physical Review B **59**, 1758–1775 (1999).
[183] P. E. Blöchl, "Projector augmented-wave method," Physical Review B **50**, 17953–17979 (1994).
[184] G. Kresse and J. Hafner, "Norm-conserving and ultrasoft pseudopotentials for first-row and transition elements," Journal of Physics: Condensed Matter **6**, 8245–8257 (1994).
[185] J. P. Perdew, K. Burke, and M. Ernzerhof, "Generalized Gradient Approximation Made Simple," Physical Review Letters **77**, 3865–3868 (1996).
[186] A. Tkatchenko and M. Scheffler, "Accurate Molecular Van Der Waals Interactions from Ground-State Electron Density and Free-Atom Reference Data," Physical Review Letters **102**, 073005 (2009).
[187] 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," The Journal of Chemical Physics **132**, 154104 (2010).
[188] A. Tkatchenko, R. A. DiStasio, R. Car, and M. Scheffler, "Accurate and Efficient Method for Many-Body van der Waals Interactions," Physical Review Letters **108**, 236402 (2012).
[189] T. Bučko, S. Lebègue, J. G. Ángyán, and J. Hafner, "Extending the applicability of the Tkatchenko-Scheffler dispersion correction via iterative Hirshfeld partitioning," The Journal of Chemical Physics **141**, 034114 (2014).
[190] J. Antony and S. Grimme, "Density functional theory including dispersion corrections for intermolecular interactions in a large benchmark set of biologically relevant molecules," Physical Chemistry Chemical Physics **8**, 5287–5293 (2006).
[191] 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**, 8368–8373 (2016).
[192] P. Wisesa, K. A. McGill, and T. Mueller, "Efficient generation of generalized Monkhorst-Pack grids through the use of informatics," Physical Review B **93**, 155109 (2016).
[193] W. S. Morgan, J. J. Jorgensen, B. C. Hess, and G. L. W. Hart, "Efficiency of Generalized Regular k-point grids," Computational Materials Science **153**, 424–430 (2018).
[194] W. S. Morgan, J. E. Christensen, P. K. Hamilton, J. J. Jorgensen, B. J. Campbell, G. L. W. Hart, and R. W. Forcade, "Generalized regular k-point grid generation on the fly," Computational Materials Science **173**, 109340 (2020).
[195] G. Wulff, "Zur Frage der Geschwindigkeit des Wachsthums und der Auflösung der Krystallflächen," Zeitschrift für Kristallographie - Crystalline Materials **34**, 449–530 (1901).
[196] D. A. Kofke and E. D. Glandt, "Monte Carlo simulation of multicomponent equilibria in a semigrand canonical ensemble," Molecular Physics **64**, 1105–1131 (1988).
[197] C. W. Glass, A. R. Oganov, and N. Hansen, "USPEX-Evolutionary crystal structure prediction," Computer Physics Communications **175**, 713–720 (2006).
[198] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," Mathematical Programming **45**, 503–528 (1989).
[199] M. L. Paleico and J. Behler, "A flexible and adaptive grid algorithm for global optimization

utilizing basin hopping Monte Carlo," The Journal of Chemical Physics **152**, 094109 (2020).

[200] A. Singraber, J. Behler, and C. Dellago, "Library-Based LAMMPS Implementation of High-Dimensional Neural Network Potentials," Journal of Chemical Theory and Computation **15**, 1827–1840 (2019).

[201] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," Journal of Computational Physics **117**, 1–19 (1995).

[202] D. R. Lide, *Handbook of Chemistry and Physics*, 89th ed. (CRC Press, 2009).

[203] S. Nosé, "A unified formulation of the constant temperature molecular dynamics methods," The Journal of Chemical Physics **81**, 511–519 (1984).

[204] W. G. Hoover, "Canonical dynamics: Equilibrium phase-space distributions," Physical Review A **31**, 1695–1697 (1985).

[205] M. L. Paleico and J. Behler, "A Bin and Hash Method for Analyzing Reference Data and Descriptors in Machine Learning Potentials," Machine Learning: Science and Technology **2**, 037001 (2021).

[206] L. B. Pártay, A. P. Bartók, and G. Csányi, "Efficient Sampling of Atomic Configurational Spaces," The Journal of Physical Chemistry B **114**, 10502–10512 (2010).

[207] E. L. Kolsbjerg, A. A. Peterson, and B. Hammer, "Neural-network-enhanced evolutionary algorithm applied to supported metal nanoparticles," Physical Review B **97**, 195424 (2018).

[208] P. C. Jennings, S. Lysgaard, J. S. Hummelshøj, T. Vegge, and T. Bligaard, "Genetic algorithms for computational materials discovery accelerated by machine learning," npj Computational Materials **5**, 1–6 (2019).

[209] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, 2009).

[210] M. Ceriotti, G. A. Tribello, and M. Parrinello, "Simplifying the representation of complex free-energy landscapes using sketch-map," Proceedings of the National Academy of Sciences **108**, 13023–13028 (2011).

[211] S. De, F. Musil, T. Ingram, C. Baldauf, and M. Ceriotti, "Mapping and classifying molecules from a high-throughput structural database," Journal of Cheminformatics **9**, 6 (2017).

[212] A. Sadeghi, S. A. Ghasemi, B. Schaefer, S. Mohr, M. A. Lill, and S. Goedecker, "Metrics for measuring distances in configuration spaces," The Journal of Chemical Physics **139**, 184118 (2013).

[213] L. Zhu, M. Amsler, T. Fuhrer, B. Schaefer, S. Faraji, S. Rostami, S. A. Ghasemi, A. Sadeghi, M. Grauzinyte, C. Wolverton, and S. Goedecker, "A fingerprint based metric for measuring similarities of crystalline structures," The Journal of Chemical Physics **144**, 034203 (2016).

[214] S. De, A. P. Bartók, G. Csányi, and M. Ceriotti, "Comparing molecules and solids across structural and alchemical space," Physical Chemistry Chemical Physics **18**, 13754–13769 (2016).

[215] J. L. Bentley, "Multidimensional binary search trees used for associative searching," Communications of the ACM **18**, 509–517 (1975).

[216] K. Pearson, "On lines and planes of closest fit to systems of points in space," The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**, 559–572 (1901).

[217] H. Hotelling, "Analysis of a complex of statistical variables into principal components," Journal of Educational Psychology **24**, 417–441 (1933).

[218] F. Hutter, J. Lücke, and L. Schmidt-Thieme, "Beyond Manual Tuning of Hyperparameters," KI - Künstliche Intelligenz **29**, 329–337 (2015).

[219] G. Luo, "A review of automatic selection methods for machine learning algorithms and hyper-parameter values," Network Modeling Analysis in Health Informatics and Bioinformatics **5**, 18 (2016).

[220] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets," in *Artificial Intelligence and Statistics* (2017) pp. 528–536.

[221] N. J. Browning, R. Ramakrishnan, O. A. von Lilienfeld, and U. Roethlisberger, "Genetic Optimization of Training Sets for Improved Machine Learning Models of Molecular Properties," The Journal of Physical Chemistry Letters **8**, 1351–1359 (2017).

[222] G. Imbalzano, A. Anelli, D. Giofré, S. Klees, J. Behler, and M. Ceriotti, "Automatic selection of atomic fingerprints and reference configurations for machine-learning potentials," The Journal of Chemical Physics **148**, 241730 (2018).

[223] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92 (Association for Computing Machinery, New York, NY, USA, 1992) pp. 287–294.

[224] E. V. Podryabinkin and A. V. Shapeev, "Active learning of linearly parametrized interatomic potentials," Computational Materials Science **140**, 171–180 (2017).

[225] L. Zhang, D.-Y. Lin, H. Wang, R. Car, and W. E, "Active learning of uniformly accurate interatomic potentials for materials simulation," Physical Review Materials **3**, 023804 (2019).

[226] C. Schran, J. Behler, and D. Marx, "Automated Fitting of Neural Network Potentials at Coupled Cluster Accuracy: Protonated Water Clusters as Testing Ground," Journal of Chemical Theory and Computation **16**, 88–99 (2020).

[227] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space," in *Database Theory — ICDT 2001*, Lecture Notes in Computer Science, edited by J. Van den Bussche and V. Vianu (Springer, Berlin, Heidelberg, 2001) pp. 420–434.

[228] "Python 3.8.5 documentation - 5. Data Structures," (2020), `docs.python.org/3/tutorial/datastructures.html`.

[229] B. C. Gates, "Supported Metal Clusters: Synthesis, Structure, and Catalysis," Chemical Reviews **95**, 511–522 (1995).

[230] A. K. Santra and D. W. Goodman, "Oxide-supported metal clusters: models for heterogeneous catalysts," Journal of Physics: Condensed Matter **15**, R31–R62 (2002).

[231] U. Heiz and E. L. Bullock, "Fundamental aspects of catalysis on supported metal clusters," Journal of Materials Chemistry **14**, 564–577 (2004).

[232] W. Yu, M. D. Porosoff, and J. G. Chen, "Review of Pt-Based Bimetallic Catalysis: From Model Surfaces to Supported Catalysts," Chemical Reviews **112**, 5780–5817 (2012).

[233] I. Cabria, M. J. López, and J. A. Alonso, "Theoretical study of the transition from planar to three-dimensional structures of palladium clusters supported on graphene," Physical Review B **81**, 035403 (2010).

[234] R. Robles and S. N. Khanna, "Oxidation of Pd_n (n=1-7, 10) Clusters Supported on Alumina/-NiAl(110)," Physical Review B **82**, 085428 (2010).

[235] Y. Tang, Z. Yang, and X. Dai, "A theoretical simulation on the catalytic oxidation of CO on Pt/graphene," Physical Chemistry Chemical Physics **14**, 16566–16572 (2012).

[236] U. Heiz, F. Vanolli, L. Trento, and W.-D. Schneider, "Chemical reactivity of size-selected supported clusters: An experimental setup," Review of Scientific Instruments **68**, 1986–1994 (1997).

[237] G. Haas, A. Menck, H. Brune, J. V. Barth, J. A. Venables, and K. Kern, "Nucleation and growth of supported clusters at defect sites: Pd/MgO(001)," Physical Review B **61**, 11105–11108 (2000).

[238] A. Yamaguchi and E. Iglesia, "Catalytic activation and reforming of methane on supported palladium clusters," Journal of Catalysis **274**, 52–63 (2010).

[239] S. Peters, S. Peredkov, M. Neeb, W. Eberhardt, and M. Al-Hada, "Size-dependent XPS spectra of small supported Au-clusters," Surface Science **608**, 129–134 (2013).

[240] J. M. Rahm and P. Erhart, "Beyond Magic Numbers: Atomic Scale Equilibrium Nanoparticle Shapes for Any Size," Nano Letters **17**, 5775–5781 (2017).

[241] C. R. Henry, "Morphology of supported nanoparticles," Progress in Surface Science **80**, 92–116

(2005).

[242] E. Ringe, R. P. Van Duyne, and L. D. Marks, "Wulff Construction for Alloy Nanoparticles," Nano Letters **11**, 3399–3403 (2011).

[243] S. J. Tauster, "Strong metal-support interactions," Accounts of Chemical Research **20**, 389–394 (1987).

[244] C.-J. Pan, M.-C. Tsai, W.-N. Su, J. Rick, N. G. Akalework, A. K. Agegnehu, S.-Y. Cheng, and B.-J. Hwang, "Tuning and exploiting Strong Metal-Support Interaction (SMSI) in Heterogeneous Catalysis," Journal of the Taiwan Institute of Chemical Engineers **74**, 154–186 (2017).

[245] W. T. Figueiredo, G. B. Della Mea, M. Segala, D. L. Baptista, C. Escudero, V. Pérez-Dieste, and F. Bernardi, "Understanding the Strong Metal–Support Interaction (SMSI) Effect in Cu_xNi_{1-x}/CeO_2 ($0 < x < 1$) Nanoparticles for Enhanced Catalysis," ACS Applied Nano Materials **2**, 2559–2573 (2019).

[246] M. Eckhoff, D. Schebarchov, and D. J. Wales, "Structure and Thermodynamics of Metal Clusters on Atomically Smooth Substrates," The Journal of Physical Chemistry Letters **8**, 5402–5407 (2017).

[247] M. Mavrikakis, B. Hammer, and J. K. Nørskov, "Effect of Strain on the Reactivity of Metal Surfaces," Physical Review Letters **81**, 2819–2822 (1998).

[248] K. Amakawa, L. Sun, C. Guo, M. Hävecker, P. Kube, I. E. Wachs, S. Lwin, A. I. Frenkel, A. Patlolla, K. Hermann, R. Schlögl, and A. Trunschke, "How Strain Affects the Reactivity of Surface Metal Oxide Catalysts," Angewandte Chemie International Edition **52**, 13553–13557 (2013).

[249] M. A. Bissett, S. Konabe, S. Okada, M. Tsuji, and H. Ago, "Enhanced Chemical Reactivity of Graphene Induced by Mechanical Strain," ACS Nano **7**, 10335–10343 (2013).

[250] W. R. Tyson and W. A. Miller, "Surface free energies of solid metals: Estimation from liquid surface tension measurements," Surface Science **62**, 267–276 (1977).

[251] R. Tran, Z. Xu, B. Radhakrishnan, D. Winston, W. Sun, K. A. Persson, and S. P. Ong, "Surface energies of elemental crystals," Scientific Data **3**, 1–13 (2016).

[252] M. Iannuzzi, A. Laio, and M. Parrinello, "Efficient Exploration of Reactive Potential Energy Surfaces Using Car-Parrinello Molecular Dynamics," Physical Review Letters **90**, 238302 (2003).

[253] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," ACM Transactions on Mathematical Software **22**, 469–483 (1996).

[254] A. Halder, L. A. Curtiss, A. Fortunelli, and S. Vajda, "Perspective: Size selected clusters for catalysis and electrochemistry," The Journal of Chemical Physics **148**, 110901 (2018).

[255] F. Düll, U. Bauer, F. Späth, P. Bachmann, J. Steinhauer, H.-P. Steinrück, and C. Papp, "Bimetallic Pd–Pt alloy nanocluster arrays on graphene/Rh(111): formation, stability, and dynamics," Physical Chemistry Chemical Physics **20**, 21294–21301 (2018).

[256] F. Düll, M. Meusel, F. Späth, S. Schötz, U. Bauer, P. Bachmann, J. Steinhauer, H.-P. Steinrück, A. Bayer, and C. Papp, "Growth and stability of Pt nanoclusters from 1 to 50 atoms on h-BN/Rh(111)," Physical Chemistry Chemical Physics **21**, 21287–21295 (2019).

[257] M. W. Thompson, J. S. Colligon, R. Smith, C. Xirouchaki, and R. E. Palmer, "Deposition of size-selected metal clusters generated by magnetron sputtering and gas condensation: a progress review," Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **362**, 117–124 (2004).

[258] S. Kunz, K. Hartl, M. Nesselberger, F. F. Schweinberger, G. Kwon, M. Hanzlik, K. J. J. Mayrhofer, U. Heiz, and M. Arenz, "Size-selected clusters as heterogeneous model catalysts under applied reaction conditions," Physical Chemistry Chemical Physics **12**, 10288–10291 (2010).

[259] S. Lucas and P. Moskovkin, "Simulation at high temperature of atomic deposition, islands coalescence, Ostwald and inverse Ostwald ripening with a general simple kinetic Monte Carlo

code," Thin Solid Films **518**, 5355–5361 (2010).
[260] S. G. Kim, "Large-scale three-dimensional simulation of Ostwald ripening," Acta Materialia **55**, 6513–6525 (2007).
[261] M. C. Gimenez, A. J. Ramirez-Pastor, and E. P. M. Leiva, "Monte Carlo simulation of metal deposition on foreign substrates," Surface Science **600**, 4741–4751 (2006).
[262] G.-F. Shao, N.-N. Tu, T.-D. Liu, L.-Y. Xu, and Y.-H. Wen, "Structural studies of Au–Pd bimetallic nanoparticles by a genetic algorithm method," Physica E: Low-dimensional Systems and Nanostructures **70**, 11–20 (2015).
[263] M. Chen, T. P. Straatsma, Z. Fang, and D. A. Dixon, "Structural and Electronic Property Study of (ZnO)n, n $\leq$ 168: Transition from Zinc Oxide Molecular Clusters to Ultrasmall Nanoparticles," The Journal of Physical Chemistry C **120**, 20400–20418 (2016).
[264] G. Shao, Y. Shangguan, J. Tao, J. Zheng, T. Liu, and Y. Wen, "An improved genetic algorithm for structural optimization of Au–Ag bimetallic nanoparticles," Applied Soft Computing **73**, 39–49 (2018).
[265] X. Shao, L. Cheng, and W. Cai, "A dynamic lattice searching method for fast optimization of Lennard–Jones clusters," Journal of Computational Chemistry **25**, 1693–1698 (2004).
[266] K. Yu, X. Wang, L. Chen, and L. Wang, "Unbiased fuzzy global optimization of Lennard-Jones clusters for N $\leq$ 1000," The Journal of Chemical Physics **151**, 214105 (2019).
[267] I. Pickering, M. Paleico, Y. A. P. Sirkin, D. A. Scherlis, and M. H. Factorovich, "Grand Canonical Investigation of the Quasi Liquid Layer of Ice: Is It Liquid?" The Journal of Physical Chemistry B **122**, 4880–4890 (2018).
[268] M. H. Factorovich, V. Molinero, and D. A. Scherlis, "A simple grand canonical approach to compute the vapor pressure of bulk and finite size systems," The Journal of Chemical Physics **140**, 064111 (2014).
[269] C. L. Kelchner, S. J. Plimpton, and J. C. Hamilton, "Dislocation nucleation and defect structure during surface indentation," Physical Review B **58**, 11085–11088 (1998).
[270] P. J. Steinhardt, D. R. Nelson, and M. Ronchetti, "Bond-orientational order in liquids and glasses," Physical Review B **28**, 784–805 (1983).
[271] M. Hou, "Solid–liquid and liquid–solid transitions in metal nanoparticles," Physical Chemistry Chemical Physics **19**, 5994–6005 (2017).
[272] C. Mottet and J. Goniakowski, "Melting and freezing of Pd nanoclusters: effect of the MgO(100) substrate," Surface Science Proceedings of the 22nd European Conference on Surface Science, **566-568**, 443–450 (2004).
[273] W. Luo, K. Su, K. Li, G. Liao, N. Hu, and M. Jia, "Substrate effect on the melting temperature of gold nanoparticles," The Journal of Chemical Physics **136**, 234704 (2012).
[274] V. Srikant, J. S. Speck, and D. R. Clarke, "Mosaic structure in epitaxial thin films having large lattice mismatch," Journal of Applied Physics **82**, 4286–4295 (1997).
[275] R. A. Rao, D. Lavric, T. K. Nath, C. B. Eom, L. Wu, and F. Tsui, "Effects of film thickness and lattice mismatch on strain states and magnetic properties of $La_{0.8}Ca_{0.2}MnO_3$ thin films," Journal of Applied Physics **85**, 4794–4796 (1999).
[276] X. Chen and J. Yin, "Buckling patterns of thin films on curved compliant substrates with applications to morphogenesis and three-dimensional micro-fabrication," Soft Matter **6**, 5667–5680 (2010).
[277] J. Gazquez, S. Bose, M. Sharma, M. A. Torija, S. J. Pennycook, C. Leighton, and M. Varela, "Lattice mismatch accommodation via oxygen vacancy ordering in epitaxial $La_{0.5}Sr_{0.5}CoO_3$-delta thin films," APL Materials **1**, 012105 (2013).
[278] A. Koma, "Van der Waals epitaxy—a new epitaxial growth method for a highly lattice-mismatched system," Thin Solid Films Papers presented at the International Workshop on Science and Technology of Thin Films for the 21st Century, Evanston,IL, USA, July 28-August 2, 1991, **216**, 72–76 (1992).

[279] S. Tiefenbacher, C. Pettenkofer, and W. Jaegermann, "Moiré pattern in LEED obtained by van der Waals epitaxy of lattice mismatched WS_2/$MoTe_2$(0001) heterointerfaces," Surface Science **450**, 181–190 (2000).

[280] C. Zhang, C.-P. Chuu, X. Ren, M.-Y. Li, L.-J. Li, C. Jin, M.-Y. Chou, and C.-K. Shih, "Interlayer couplings, Moiré patterns, and 2D electronic superlattices in MoS_2/WSe_2 hetero-bilayers," Science Advances **3**, e1601459 (2017).

[281] A. K. Geim and I. V. Grigorieva, "Van der Waals heterostructures," Nature **499**, 419–425 (2013).

[282] K. S. Novoselov, A. Mishchenko, A. Carvalho, and A. H. C. Neto, "2D materials and van der Waals heterostructures," Science **353** (2016), 10.1126/science.aac9439.

[283] D. Jariwala, T. J. Marks, and M. C. Hersam, "Mixed-dimensional van der Waals heterostructures," Nature Materials **16**, 170–181 (2017).

[284] E. Fako, A. S. Dobrota, I. A. Pašti, N. López, S. V. Mentus, and N. V. Skorodumova, "Lattice mismatch as the descriptor of segregation, stability and reactivity of supported thin catalyst films," Physical Chemistry Chemical Physics **20**, 1524–1530 (2018).

[285] L. Ge, H. Yuan, Y. Min, L. Li, S. Chen, L. Xu, and W. A. Goddard, "Predicted Optimal Bifunctional Electrocatalysts for the Hydrogen Evolution Reaction and the Oxygen Evolution Reaction Using Chalcogenide Heterostructures Based on Machine Learning Analysis of in Silico Quantum Mechanics Based High Throughput Screening," The Journal of Physical Chemistry Letters **11**, 869–876 (2020).

[286] J. Wintterlin and M. L. Bocquet, "Graphene on metal surfaces," Surface Science Special Issue of Surface Science dedicated to Prof. Dr. Dr. h.c. mult. Gerhard Ertl, Nobel-Laureate in Chemistry 2007, **603**, 1841–1852 (2009).

[287] M. Huang, Y. Jiang, C. Jin, J. Ren, Z. Zhou, and L. Guan, "Pt–Cu alloy with high density of surface Pt defects for efficient catalysis of breaking C–C bond in ethanol," Electrochimica Acta **125**, 29–37 (2014).

[288] S. Zhang, X. Zhang, G. Jiang, H. Zhu, S. Guo, D. Su, G. Lu, and S. Sun, "Tuning Nanoparticle Structure and Surface Strain for Catalysis Optimization," Journal of the American Chemical Society **136**, 7734–7739 (2014).

[289] A. Khorshidi, J. Violet, J. Hashemi, and A. A. Peterson, "How strain can break the scaling relations of catalysis," Nature Catalysis **1**, 263–268 (2018).

[290] D. Kopač, B. Likozar, and M. Huš, "Catalysis of material surface defects: Multiscale modeling of methanol synthesis by CO_2 reduction on copper," Applied Surface Science **497**, 143783 (2019).

[291] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," Journal of Biomedical Informatics **35**, 352–359 (2002).

[292] A. A. M. Al-Saffar, H. Tao, and M. A. Talab, "Review of deep convolution neural network in image classification," in *2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)* (2017) pp. 26–31.

[293] E. V. Podryabinkin, E. V. Tikhonov, A. V. Shapeev, and A. R. Oganov, "Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning," Physical Review B **99**, 064114 (2019).

[294] N. Bernstein, G. Csányi, and V. L. Deringer, "De novo exploration and self-guided learning of potential-energy surfaces," npj Computational Materials **5**, 1–9 (2019).

[295] V. Fiorentini and M. Methfessel, "Extracting convergent surface energies from slab calculations," Journal of Physics: Condensed Matter **8**, 6525–6529 (1996).

Part IV

Appendices

Appendix A

Notes on the Coincidence Lattice Match Algorithm and Strain Theory

A.1 Some notes on Linear Algebra

At the core of the CLM method, we need to find at least two solutions to the equation

$$\mathbf{AM}\vec{m} = \mathbf{B}\vec{n}, \tag{A.1}$$

where **A** and **B** are 2×2 matrices containing the lattice vectors of the 2D lattices for materials A and B in their columns; **M** is the usual 2D rotation matrix, and $\vec{m}$ and $\vec{n}$ are column vectors (although in the text we will express them also as row vectors when this notation is more comfortable). Equivalent expressions are available if we want to express **A** and **B** as containing the lattice vectors in their rows instead. A "solution" here is a pair of vector $\vec{n}$ and $\vec{m}$ that satisfy the conditions of the equation.

This equation always has solutions in the real domain. To see this, we can think of a fixed, given vector $\vec{n}$, and a given fixed rotation matrix **M**. In this case, the right hand side of the equation is constant, and we are presented with a regular inhomogeneous linear equation system. The solution to such a system exists if any of the following equivalent statements is true:

1. the columns of **AM** are linearly independent
2. $\det\Big(\mathbf{AM}\Big) \neq 0$
3. $\Big(\mathbf{AM}\Big)^{-1}$ exists

The first statement is the easiest one to prove: since the columns of **A** correspond to the lattice vectors of 2D lattice, they are always linearly independent (LI). This is required, otherwise the two vector cannot tile and cover the whole 2D plane. Additionally, in 2D, vectors are only linearly dependent (LD) if they are parallel, that is, one is a multiple of the other. Two parallel vectors clearly do not form any good lattice vector set. Since pure rotations do not affect the LI relationship of the columns of **A**, **AM** is still a matrix with LI columns. Thus eq. A.1 has an exact real-valued solution that is trivial to find by inverting **AM** for given values of $\vec{n}$. Since it has a solution for arbitrary $\vec{n}$, it has solution for every possible value of $\vec{n}$.

Another property is that given a vector $\vec{n}_1$, and a multiple of it $\vec{n}_2$ such that

$$\vec{n}_2 = k \cdot \vec{n}_1, \;\; k \in \mathbb{R}, \tag{A.2}$$

the solution corresponding to $\vec{n}_2$ is also the same multiple of the solution for $\vec{n}_1$

$$\vec{m}_2 = k \cdot \vec{m}_1. \tag{A.3}$$

This is trivial to prove. If $\vec{m}_1$ and $\vec{m}_2$ are solutions to eq. A.1, then

$$\vec{m}_1 = \left(\mathbf{AM}\right)^{-1}\mathbf{B}\vec{n}_1, \tag{A.4}$$

$$\vec{m}_2 = \left(\mathbf{AM}\right)^{-1}\mathbf{B}\vec{n}_2. \tag{A.5}$$

$$\tag{A.6}$$

Using these equivalences, eq. A.2, and the commutative property for scalars in matrix and vector multiplication, we obtain

$$\vec{m}_2 = \left(\mathbf{AM}\right)^{-1}\mathbf{B}(k.\vec{n}_1), \tag{A.7}$$

$$\vec{m}_2 = k\cdot\left[\left(\mathbf{AM}\right)^{-1}\mathbf{B}\vec{n}_1\right], \tag{A.8}$$

$$\vec{m}_2 = k\cdot\vec{m}_1. \tag{A.9}$$

This is related to the fact that if we find a coincident supercell or vector, any other supercells (i.e. multiples or linear combinations) of this supercell will also be coincident. Equivalently, sums of solutions are also solutions, for the same reason.

The problem then arises in the integer domain, when we attempt to find an integer solution $\vec{m}$ to a given pair of integers $\vec{n}$. The treatment of linear Diophantine equations like this is rather complex, so it is not trivial to determine under which conditions this equation has exact integer solutions for the general case.

One specific case where an exact integer solution is always present (if not necessarily desirable) is the case where $\vec{m}$ and $\vec{n}$ are composed of rational numbers. Since as mentioned before any multiples of the initial $\vec{n}$ and $\vec{m}$ are also solutions, we can multiply both solutions by their common integer denominators until the decimal parts of both $\vec{m}$ and $\vec{n}$ are gone and thus obtain an exact integer solution. The problem in this case is that it might require a large supercell, too large for any practical computations.

One such case is if we have any pair of orthogonal lattices, with the rotation matrix at 0°. For example, for ZnO$(10\bar{1}0)$ and Cu(100), our equation A.1 looks like the following, with $\vec{n} = (1,0)$,

$$\begin{bmatrix} d_{\mathrm{Cu}} & 0 \\ 0 & d_{\mathrm{Cu}} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\vec{m} = \begin{bmatrix} a_{\mathrm{ZnO}} & 0 \\ 0 & c_{\mathrm{ZnO}} \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{A.10}$$

$$\vec{m} = \begin{bmatrix} 1/d_{\mathrm{Cu}} & 0 \\ 0 & 1/d_{\mathrm{Cu}} \end{bmatrix}\begin{bmatrix} a_{\mathrm{ZnO}} & 0 \\ 0 & c_{\mathrm{ZnO}} \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{A.11}$$

$$\vec{m} = (a_{\mathrm{ZnO}}/d_{\mathrm{Cu}}, 0) = (3.29\ \text{Å}/2.55\ \text{Å}, 0) = 329/255 \approx 1.290..., \tag{A.12}$$

where d_{Cu} is the nearest neighbor distance in bulk fcc Cu, and $a_{\mathrm{ZnO}}, c_{\mathrm{ZnO}}$ are the lattice constants for bulk ZnO. Since the solution for $\vec{m}$ is a vector composed of rational numbers, we can multiply both it and the original proposed $\vec{n}$ by the denominator and obtain a new, exactly valid, integer solution

$$\vec{m} = 255\cdot(329/255, 0) = (329, 0), \tag{A.13}$$

$$\vec{n} = 255\cdot(1,0) = (255, 0). \tag{A.14}$$

$$\tag{A.15}$$

It can be confirmed that this solution is correct

$$\begin{bmatrix} d_{\mathrm{Cu}} & 0 \\ 0 & d_{\mathrm{Cu}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 329 \\ 0 \end{bmatrix} - \begin{bmatrix} a_{\mathrm{ZnO}} & 0 \\ 0 & c_{\mathrm{ZnO}} \end{bmatrix} \begin{bmatrix} 255 \\ 0 \end{bmatrix} = 0 \tag{A.16}$$

$$329 d_{\mathrm{Cu}} - 255 a_{\mathrm{ZnO}} = 0 \tag{A.17}$$

$$329 \cdot 2.55 \text{ Å} = 255 \cdot 3.29 \text{ Å} \tag{A.18}$$

$$838.95 = 838.95 \quad . \tag{A.19}$$

Now we can apply the same procedure for $\vec{n}' = (0,1)$ to obtain the other point of the new supercell, with a result of $\vec{m}' = c_{\mathrm{ZnO}}/d_{\mathrm{Cu}} = 5.30 \text{ Å}/2.55 \text{ Å} = 529/255 \approx 2.075....$ Scaling both to obtain an integer, our result is $\vec{m}' = (0,529), \vec{n}' = (0,255)$. This creates a coincident supercell equivalent to $\vec{n} \times \vec{n}' = 255 \cdot 255 = 65.025$ cells of ZnO(10$\bar{1}$0) and $\vec{m} \times \vec{m}' = 329 \cdot 529 = 174.041$ cells of Cu(100), which is a huge system

In the general case this huge exact integer solution might not even be available, since the rotation matrix itself can easily introduce irrational square roots (or other irrational numbers from sine and cosine operations) into the equation, or one of the components of the lattice vectors might be also irrational if obtained geometrically. We just need *approximate* integer solutions, that is, any value can be proposed, and then we need to grade the solutions with a given error.

A.2 Worked CLM Example for Cu(111) and ZnO(10$\bar{1}$0)

Here we present a worked example for a more complicated case, involving Cu(111) and ZnO(10$\bar{1}$0) at a 20° rotation. Here, only one of the lattices is orthogonal, and we also include a rotation. Eq. A.1 for this pair looks like

$$\begin{bmatrix} d_{\mathrm{Cu}} & 0.5\, d_{\mathrm{Cu}} \\ 0 & \sqrt{3}/2\, d_{\mathrm{Cu}} \end{bmatrix} \begin{bmatrix} \cos 20^\circ & -\sin 20^\circ \\ \sin 20^\circ & \cos 20^\circ \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} - \begin{bmatrix} a_{\mathrm{ZnO}} & 0 \\ 0 & c_{\mathrm{ZnO}} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} = 0 \quad . \tag{A.20}$$

$$\tag{A.21}$$

We then need to propose solutions in the form of 4 different integers (including zero, but not all zero at the same time) m_1, m_2, n_1, n_2. These solutions are evaluated and a δ error value is obtained, equal to the left hand side of eq. A.20. Of course, the closer to zero this δ is, the better the solution is. The results for integer values between -5 and 5 (to limit the amount of results) and their errors are presented in table A.1.

These solutions represent single points where linear combinations of the lattice vectors of both lattices coincide. From these solutions, we now need to select at least two that are linearly independent (otherwise, we cannot form a proper superlattice). Testing for linear independence in a computer program is simple, by calculating the determinant of a matrix containing $\vec{m}, \vec{m}'$ or $\vec{n}, \vec{n}'$ as its columns. If this determinant equals zero, the vectors that form the matrix are linearly dependent and thus one of them needs to be discarded. Since the numbers involved correspond to integers, the determinant can be calculated exactly with integer variables and compared exactly to zero, so no float comparison is required. A growing list of unique solutions is kept, and all further solutions are compared against this list, with solutions that have successfully tested as LI against all previous unique solutions being added to the list.

This process leaves us with 4 solutions in this case, which we can combine in unique pairs to form 6 candidate coincident lattices. These lattices are shown in tables A.2 and A.3. The pairs of solutions are indicated as $\vec{m}, \vec{n}$ and $\vec{m}', \vec{n}'$. From these lattices, some are still linear combinations of other solutions, so they need to be discarded. For example, notice that for lattice b

Index	m_1	m_2	n_1	n_2	δ
a	**2**	**1**	**2**	**0**	**0.2524**
b	-2	-1	-2	0	0.2524
c	4	2	4	0	0.5048
d	-4	-2	-4	0	0.5048
e	**-1**	**3**	**1**	**1**	**0.5844**
f	1	-3	-1	-1	0.5844
g	**2**	**4**	**4**	**1**	**0.6192**
h	-2	-4	-4	-1	0.6192
i	**1**	**4**	**3**	**1**	**0.6195**
j	-1	-4	-3	-1	0.6195

Table A.1: Proposed integer solutions to eq. A.20 for integers between -5 and 5, organized by quality of the solution δ_{QOS}. Bold text indicates the selected unique, linearly independent solutions (a, e, g and i). Notice that for example b, c, and d are multiples of a.

$$\vec{n}(b) = 1 \cdot \vec{n}'(a) - 1 \cdot \vec{n}(a), \tag{A.22}$$
$$(2,0) = (3,1) - (1,1)$$
$$\vec{n}'(b) = 1 \cdot \vec{n}(a) + 0 \cdot \vec{n}'(a), \tag{A.23}$$

and the same applies to the $\vec{m}$'s, which means that b and a are actually equivalent solutions. This arises when the two vectors of a solution can be expressed as integer linear combinations of the vectors of another solution (for the shown example, 1, -1 and 0 are the integer factors). As a counter example, lattices c and d are *not* equivalent since although $\vec{n}(c) = \vec{n}(d)$,

$$\vec{n}'(d) = 0.5 \cdot \vec{n}(c) + 1 \cdot \vec{n}'(c), \tag{A.24}$$

which has non integer factors. This can be tested for in a program by calculating the equation

$$x = b \cdot \mathbf{A}^{-1}, \tag{A.25}$$

where x are the components of vector b ($\vec{n}$ or $\vec{n}'$ of the lattice being tested) in the basis A (a matrix containing $\vec{n}$ and $\vec{n}'$ of the reference lattice as columns), and testing if the obtained components are integer valued. This is done for both vectors of a given candidate superlattice against all known unique superlattices. If at least one coordinate is non-integer, then a new unique lattice has been detected and is added to the unique lattice list.

Noticed that the calculated strain $\varepsilon_{\text{DOLD}}$ (explained in the next paragraphs) is also identical for the non-unique configurations, as well as some other properties such as the number of atoms or the multiplicity from the original lattice. This procedure leaves us with only 4 unique lattices: a, d, e and f.

N_{atoms} is the number of atoms in the resulting superlattice. This is calculated from the base number of atoms for one layer of each material (4 for ZnO, 1 for Cu), multiplied by the area of the superlattice compared to the original lattice, which is obtained from $|m \times m'|$ and $|n \times n'|$.

Table A.3 shows the new proposed vectors for the resulting superlattices, which are obtained by multiplying the original lattice vectors of each material with the corresponding $\vec{n}$ or $\vec{m}$. As was desired from the algorithm, $\vec{\vec{a}}_{1,2} \approx \vec{\vec{b}}_{1,2}$, that is, the vectors for the new superlattices for each material almost coincide.

We can visualize the coincidence between the lattices when utilizing these solutions in figure A.1.

Index	m_1	m_2	m'_1	m'_2	n_1	n_2	n'_1	n'_2	$\lvert m \times m' \rvert$	$\lvert n \times n' \rvert$	N_{atoms}	$\delta(m,n)$	$\delta(m',n')$	$\varepsilon_{\text{DOLD}}$
a	**-1**	**3**	**1**	**4**	**1**	**1**	**3**	**1**	**7**	**2**	**15**	**0.5844**	**0.6195**	**0.0511**
b	2	1	-1	3	2	0	1	1	7	2	15	0.2524	0.5844	0.0511
c	2	1	1	4	2	0	3	1	7	2	15	0.2524	0.6195	0.0511
d	**2**	**1**	**2**	**4**	**2**	**0**	**4**	**1**	**6**	**2**	**14**	**0.2524**	**0.6192**	**0.0735**
e	**-1**	**3**	**2**	**4**	**1**	**1**	**4**	**1**	**10**	**3**	**22**	**0.5844**	**0.6192**	**0.0753**
f	**2**	**4**	**1**	**4**	**4**	**1**	**3**	**1**	**4**	**1**	**8**	**0.6192**	**0.6195**	**0.2265**

Table A.2: Resulting superlattice candidates, from combining candidates in table A.1, ordered by strain (ε). $|m \times m|$ shows which multiple of the original lattice for material A the superlattice is. This is useful for calculating the area or number of atoms (N_{atoms}) of the new superlattice. In bold, those candidates that are completely LI.

Index	m_1	m_2	m'_1	m'_2	n_1	n_2	n'_1	n'_2	$\widetilde{a}_{1x}$	$\widetilde{a}_{1y}$	$\widetilde{a}_{2x}$	$\widetilde{a}_{2y}$	$\widetilde{b}_{1x}$	$\widetilde{b}_{1y}$	$\widetilde{b}_{2x}$	$\widetilde{b}_{2y}$
a	**-1**	**3**	**1**	**4**	**1**	**1**	**3**	**1**	**3.496**	**5.844**	**10.305**	**5.737**	**3.290**	**5.297**	**9.870**	**5.297**
b	2	1	-1	3	2	0	1	1	6.809	-0.106	3.496	5.844	6.580	0.000	3.290	5.297
c	2	1	1	4	2	0	3	1	6.809	-0.106	10.305	5.737	6.580	0.000	9.870	5.297
d	**2**	**1**	**2**	**4**	**2**	**0**	**4**	**1**	**6.809**	**-0.106**	**12.724**	**4.857**	**6.580**	**0.000**	**13.160**	**5.297**
e	**-1**	**3**	**2**	**4**	**1**	**1**	**4**	**1**	**3.496**	**5.844**	**12.724**	**4.857**	**3.290**	**5.297**	**13.160**	**5.297**
f	**2**	**4**	**1**	**4**	**4**	**1**	**3**	**1**	**12.724**	**4.857**	**10.305**	**5.737**	**13.160**	**5.297**	**9.870**	**5.297**

Table A.3: Continuation of table A.2, showing the resulting new superlattice vectors for each material. $\vec{\widetilde{a}}_1 = (\widetilde{a}_{1x}, \widetilde{a}_{1y})$ is one lattice vector of the new superlattice for material A, and so on for the other components.

Here, we represent both lattices with their lattice points, that is, all possible linear integer combinations of the lattice vectors. These lattice points tile 2D space continuously, filling it in. Where lattice points of one lattice are close to the ones from the other lattice, we have found a match. For example, coincident lattice d from table A.2, has $\vec{n} = (2,0)$. This means one side of the new superlattice is found if we move 2 times along the first lattice vector of ZnO(10$\bar{1}$0), and 0 times along the second lattice vector. This corresponds to starting at the origin where the red and blue lattices meet, and displacing two blue points to the right, which in effect is shown as one of the sides of the black coincident superlattice parallelogram. The other side of this parallelogram is given by $\vec{n}' = (4,1)$. once again, if we move 4 blue dots horizontally from the origin and 1 vertically, we will find the other coincident point. A similar approach is followed for the Cu lattice, but it is slightly more difficult to follow given that the Cu(111) lattice is rotated, and not orthogonal.

As can be seen, since the original solutions are not exact, the generated superlattices are close but not exactly on top of each other. At each edge of the black parallelograms, we have red crosses and blue dots that almost but not quite overlap (except at the origin, where the overlap is exact). We need a way of evaluating how good each of these matches is, that is, how large the deviation between coincident dots really is.

To evaluate this, we proceed to the last step in the algorithm: deforming one of the materials. To make the solution exact, one of the materials needs to be deformed, by contraction, expansion, or change of the angle between the lattice vectors. Alternatively, both materials can be deformed to some sort of intermediate solution. We choose to expand only the Cu layer, by setting its lattice vectors to those of the ZnO superlattice and displacing atoms into the new lattice by maintaining their fractional coordinates in the new reference frame, maintaining the structure given by the ZnO layer. This is shown in fig. A.2. Notice that now the superlattices match exactly, at the cost of having altered the Cu(111) lattice. Notice that now the red cross patterns are not the same across the images, and that the red cross lattice is highly deformed in the case of f. Figure fig. A.3 shows the resulting atomic system corresponding to each of these coincident lattices.

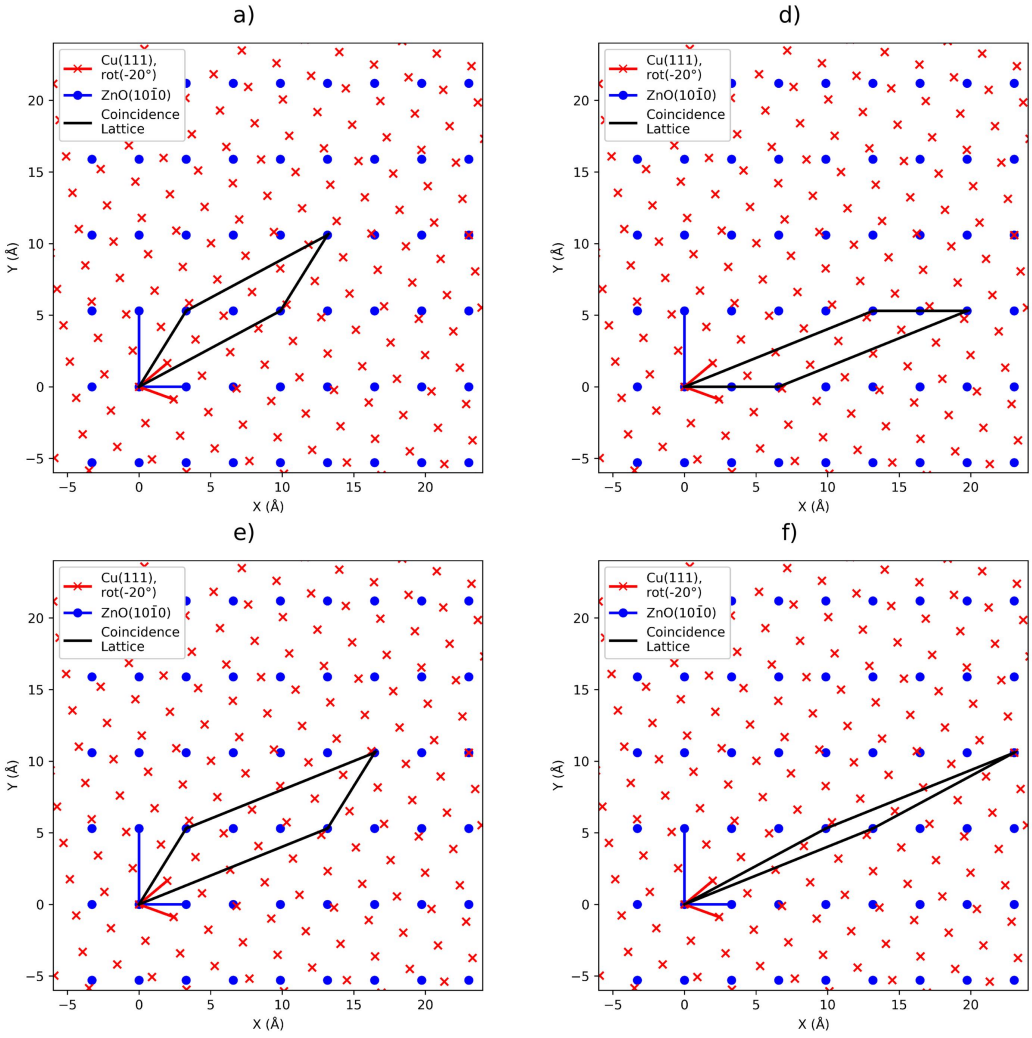

Figure A.1: Dot plots of the unique lattices from table A.2. Red crosses represent the Cu(111) lattice, rotated 20° clockwise, filling in all of 2D space. Blue dots represent ZnO(10$\bar{1}$0). Both lattices coincide at the origin. The coincident superlattice for ZnO is marked with black lines.

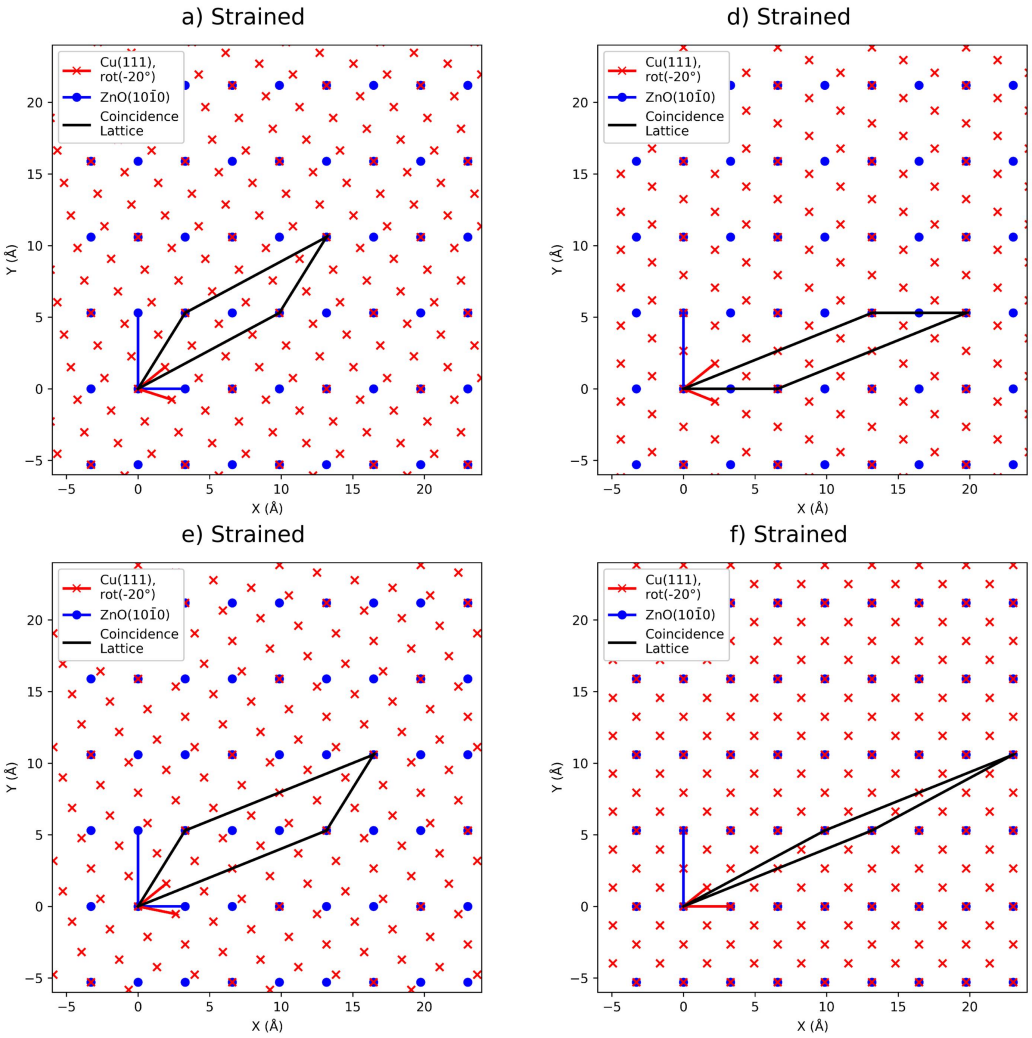

Figure A.2: As in fig. A.1, but with the Cu lattice strained to match the ZnO lattice exactly.

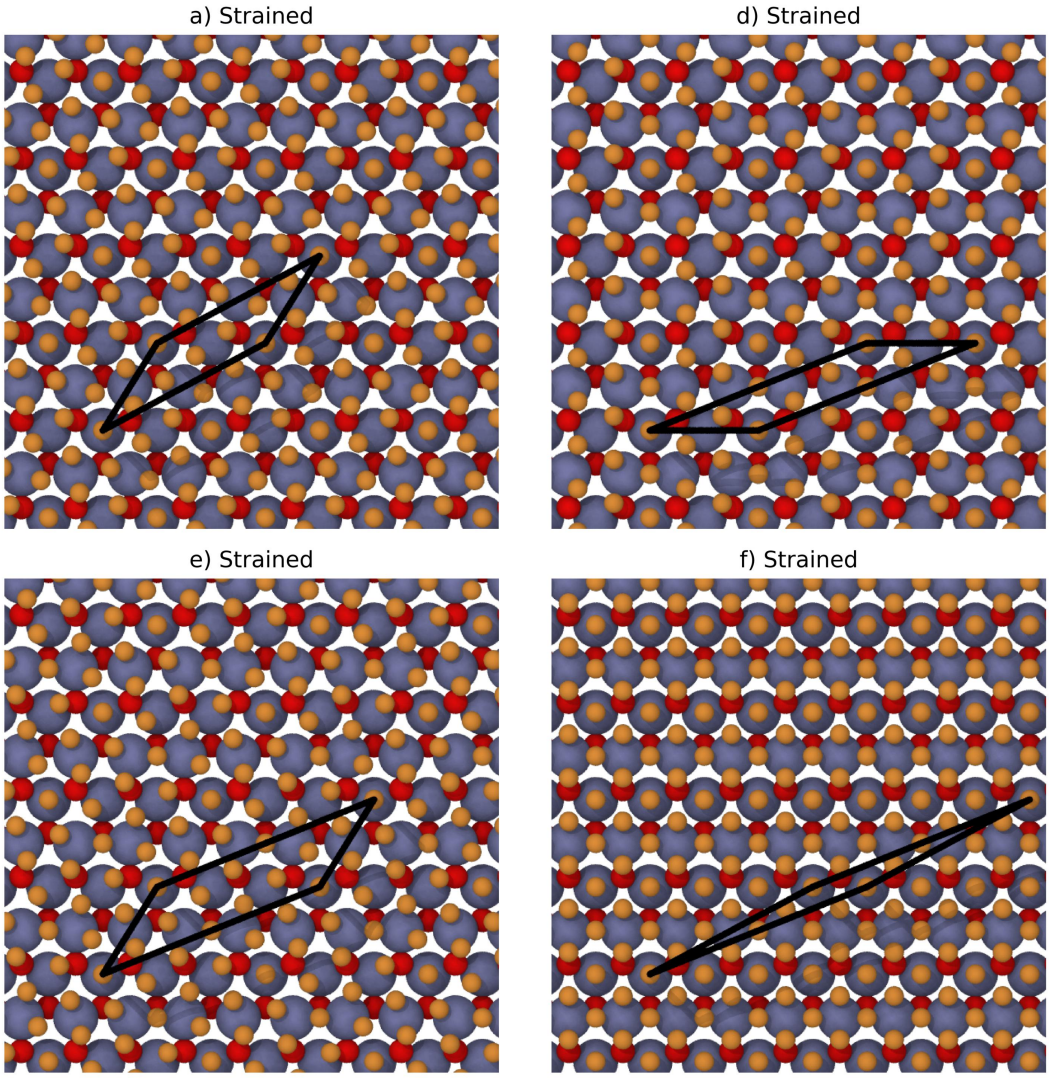

Figure A.3: As in fig. A.2, but now adding the corresponding atoms to 1 layer of each material.

A.3 Strain Theory

The deformation inflicted in the previous section can be interpreted through the lens of the theory of geometric strain, to obtain a numeric quality indicator in the form of the average strain $\varepsilon_{\mathrm{DOLD}}$. Another advantage of utilizing the strain is that if the finite version is used, any rigid body rotations (that is, any pure rotations that would not affect interatomic distances) between the initial and final lattices are removed.

The topic of strain (and associated stress and elastic constants) is treated extensively in many mechanics and engineering textbooks, but this treatment is usually too complex, or based on derivatives or time evolution of a system. This appendix is intended to clarify and simplify the available literature. Here we develop the theory from the point of view of two lattice cells, one initial and one deformed, which are described by their lattice vectors. For simplicity we will work in 2D, but the extension to 3D is trivial.

Many derivations [166, 167] are available for the case of a lattice cell described in the crystallographic convention of 3 lengths and 3 angles, but this description is incomplete since it does not take into account possible translations or rotations of the lattices with respect to each other. In the crystallographic convention, the lattices are always centered on the origin, with the first lattice vector pointing in the direction of the x-axis (that is, only its x-component is different form zero and positive), the second in the x-y plane (no z component) with positive y component, and the third vector not restricted to a plane but with positive z component. In this case the only allowed lattice vector matrices can be expressed as

$$(\mathbf{a},\mathbf{b},\mathbf{c}) = \begin{bmatrix} a_x & b_x & c_x \\ 0.0 & b_y & c_y \\ 0.0 & 0.0 & c_z \end{bmatrix}, \tag{A.26}$$

where each lattice vector is a column of the matrix, a_i, b_i, c_i are the components of each vector, and a_x, b_y, c_z are strictly positive (not 0.0).

This convention limits the range of possible cells and is more inconvenient to work with, but in the end the results with the vector matrix utilized here are exactly the same as for the crystallographic definition and conversions are possible if cumbersome.

Some of the available tools for calculating strain, such as the Bilbao Crystallographic Server's STRAIN tool [165] are based on the crystallographic definition, and can be used as a comparison tool. The Pymatgen [169] library for Python offers tools for calculating a deformed cell given an initial cell and a strain/deformation tensor, but what is usually lacking is the reverse: calculating a strain from a given set of lattice vectors. For this reason, to aid in understanding the behavior of strain, this section explains how certain modifications to the initial lattice vectors result in specific strain modes (pure normal strain, pure shear strain, etc.).

For an arbitrary set of lattice vectors, we can calculate two other matrices/tensors. The metric tensor G provides all the projections (dot products) between the vectors describing the lattice cell

$$\mathbf{G} = \begin{bmatrix} \vec{a}_1 \cdot \vec{a}_1 & \vec{a}_1 \cdot \vec{a}_2 \\ \vec{a}_2 \cdot \vec{a}_2 & \vec{a}_2 \cdot \vec{a}_2 \end{bmatrix}, \tag{A.27}$$

$$= \begin{bmatrix} |\vec{a}_1|^2 & |\vec{a}_1| \cdot |\vec{a}_2| \cdot \cos(\theta) \\ |\vec{a}_1| \cdot |\vec{a}_2| \cdot \cos(\theta) \cdot & |\vec{a}_1|^2 \end{bmatrix}, \tag{A.28}$$

$$= \mathbf{A}^T\mathbf{A}, \tag{A.29}$$

$$\tag{A.30}$$

where $\mathbf{A}$ is a matrix containing the lattice vectors as columns, $\vec{a}_1, \vec{a}_2$ are the lattice vectors of our cell

and θ the angle between them. $\mathbf{G}$ is symmetric, that is, equal to its own transpose

$$\mathbf{G}^T = (\mathbf{A}^T\mathbf{A})^T = \mathbf{A}^T(\mathbf{A}^T)^T = \mathbf{A}^T\mathbf{A} = \mathbf{G}, \tag{A.31}$$

where we have used the following property of matrix multiplication and transpose

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T. \tag{A.32}$$

This tensor has the property that it is not sensitive to rotations of the original matrix describing the cell, a property which we will make use of later. Since it is symmetric and also positive definite (a property of any $\mathbf{A}^T\mathbf{A}$ matrix), it can also be decomposed into a root matrix. If we use the Cholesky decomposition algorithm for this, we obtain a lower triangular tensor $\mathbf{R}$ such that:

$$\mathbf{G} = \mathbf{R}\mathbf{R}^T. \tag{A.33}$$

Notice that here the matrix multiplication is the other way around from the metric tensor. This tensor is utilized by the reference papers [166, 167] for the STRAIN tool [165], where they are known as the "standard root tensor", and we will also make use of it. It provides a convenient triangular description of any given cell, that is also independent of rotations of the original matrix.

A.3.1 Deformation Tensor

Now, to calculate the strain, we first build the matrices corresponding to the original and deformed lattice vectors, with the vectors as columns

$$\mathbf{A} = \begin{bmatrix} a_{1x} & a_{2x} \\ a_{1y} & a_{2y} \end{bmatrix} \quad \widetilde{\mathbf{A}} = \begin{bmatrix} \widetilde{a_{1x}} & \widetilde{a_{2x}} \\ \widetilde{a_{1y}} & \widetilde{a_{2y}} \end{bmatrix}. \tag{A.34}$$

From these we can calculate a deformation (sometimes also known as distortion or displacement) tensor, by first proposing that $\widetilde{\mathbf{A}}$ is equal to $\mathbf{A}$ plus the distortion

$$\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{e}\mathbf{A} = (\mathbf{I} + \mathbf{e})\mathbf{A}, \tag{A.35}$$

where $\mathbf{I}$ is the identity matrix. Multiplying both sides to the right by the inverse of $\mathbf{A}$ (which as described before, always exists) yields

$$\widetilde{\mathbf{A}}\mathbf{A}^{-1} = (\mathbf{I} + \mathbf{e})\mathbf{A}\mathbf{A}^{-1} \tag{A.36}$$

$$\mathbf{e} = \widetilde{\mathbf{A}}\mathbf{A}^{-1} - \mathbf{I}. \tag{A.37}$$

A similar tensor can be constructed if we switch around the initial and final matrices (initial vs. deformed coordinate reference frame), or if we change the order of the right hand side in A.35. As the magnitude of the distortion to the initial and final lattices decreases these tensors all tend to be the same, and the results are in any case equivalent as long as it is clear which version of the tensor we are working with.

A.3.2 Infinitesimal Strain Tensor

To obtain a useful strain tensor that fulfills the expected properties of strain (more on this later), we build a symmetric tensor out of $\mathbf{e}$, which is called (under many other names) the infinitesimal, Cauchy,

small or linear strain tensor ε, as

$$\varepsilon = 0.5(\mathbf{e}+\mathbf{e}^T) = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} \\ \gamma_{yx} & \varepsilon_{yy} \end{bmatrix}, \tag{A.38}$$

where ε_{xx} and ε_{yy} are the linear strains in the direction of the x- and y-axis (that is, how much a line that was originally lying on the x- or y-axis gets stretched or contracted by the deformation), and $\gamma_{xy} = \gamma_{yx} = \gamma$ is the shear strain (related to the change in angle for a pair of lines that were originally orthogonal and parallel to the Cartesian axis). That this is symmetric is easily proven for any arbitrary matrix added to its transpose (replace with a,b,c,d coefficients and calculate). Notice that sometimes γ is taken to be 2 times the γ in this matrix, depending on conventions.

Another related useful tensor measures the degree of rigid body rotation in a given deformation

$$\omega = 0.5(\mathbf{e}-\mathbf{e}^T) = \begin{bmatrix} 0.0 & \omega \\ -\omega & 0.0 \end{bmatrix}. \tag{A.39}$$

ω is always antisymmetric, that is, $\omega_{ij} = -\omega_{ji}$. Once again, this is easily proven by proposing an arbitrary matrix with given coefficients a,b,c,d and calculating.

Notice that $\mathbf{e}$ is actually the sum of the other two tensors

$$\mathbf{e} = (0.5+0.5)\mathbf{e}+0.5(\mathbf{e}^T-\mathbf{e}^T) = 0.5(\mathbf{e}+\mathbf{e}^T)+0.5(\mathbf{e}+\mathbf{e}^T) = \varepsilon+\omega. \tag{A.40}$$

The typical examples of basic deformation are i) normal strain, ii) pure shear and iii) a small rigid body rotation. The following figure A.4 shows all the possible basic strains, wit subfigure a) containing an example of uniaxial strain for a perpendicular set of lattice vectors (the case is more complicated if they are not perpendicular).

In the subfigure, L is the original length in the direction of the first lattice vector (which coincides with the x-axis) and $L+\Delta L$ is the length after straining. This setup results in the matrices

$$\mathbf{A} = \begin{bmatrix} L & 0 \\ 0 & C \end{bmatrix} \quad \widetilde{\mathbf{A}} = \begin{bmatrix} L+dL & 0 \\ 0 & C \end{bmatrix} \mathbf{e} = \begin{bmatrix} \frac{L+dL}{L}-1 & 0 \\ 0 & 0 \end{bmatrix} \tag{A.41}$$

$$\varepsilon = \begin{bmatrix} \frac{1.0dL}{L} & 0 \\ 0 & 0 \end{bmatrix} \omega = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{A.42}$$

As expected, the rotation tensor is zero, and the only component for the strain tensor is in the diagonal. This component, $\Delta L/L$, is the fractional increase (or decrease) in length due to the strain. Notice that this component is linear in L, hence the name of the linear strain tensor (as opposed to the finite tensor we will use later).

For a pure shear, we apply a deformation parallel to one or both of the Cartesian axis. This changes the angle between lattice vectors of our cell. Two types of shear are presented in fig. A.4. The shear in b) includes partly a rigid body rotation which is equal to half of the initial shear. For a pure proper shear with no rigid body rotation, we need to shear the initial square equally from both sides, as in c).

For the shears in fig. A.4, we have the following resulting tensors. For b), shear with rotation, we obtain

$$\mathbf{A} = \begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \quad \widetilde{\mathbf{A}} = \begin{bmatrix} C & D \\ 0 & L \end{bmatrix} \quad \mathbf{e} = \begin{bmatrix} 0 & \frac{D}{L} \\ 0 & 0 \end{bmatrix} \tag{A.43}$$

$$\varepsilon = \begin{bmatrix} 0 & \frac{0.5D}{L} \\ \frac{0.5D}{L} & 0 \end{bmatrix} \omega = \begin{bmatrix} 0 & \frac{0.5D}{L} \\ -\frac{0.5D}{L} & 0 \end{bmatrix}. \tag{A.44}$$

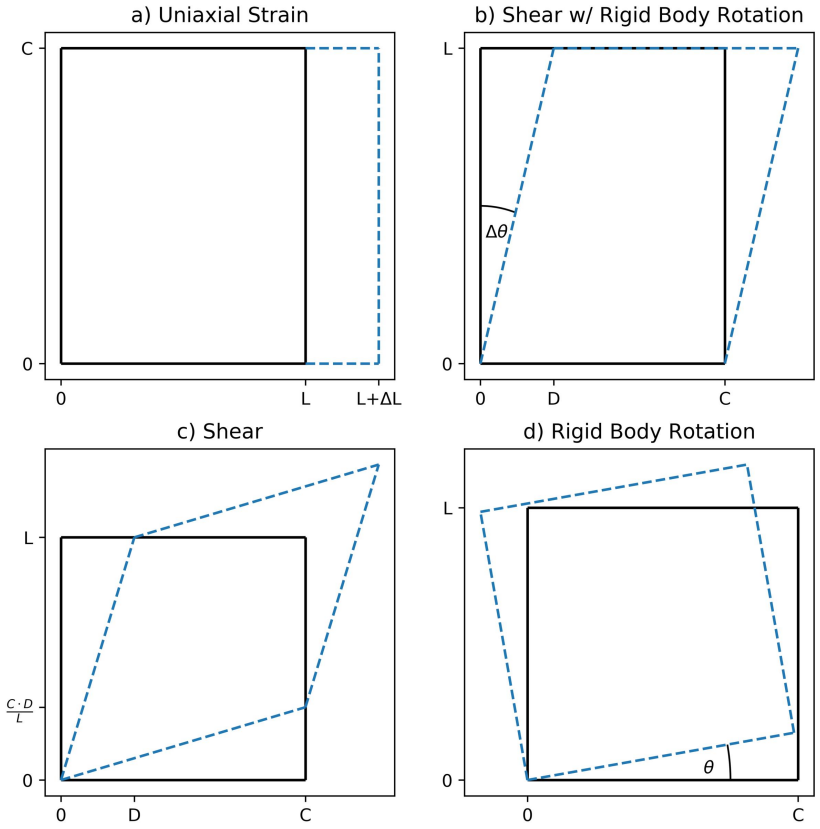

Figure A.4: Example of the basic strains: a) Pure uniaxial strain b) Shear strain, with $\Delta\theta$, the angle change, marked c) Pure shear d) Rigid body rotation, with rotation angle θ marked

Notice that $\gamma = 0.5D/L = 0.5\tan(\Delta\theta)$ in fig. A.4 b), and that for small $\Delta\theta$, the tangent approaches $\Delta\theta$, which means that γ is approximately equal to half the change in angle between the original orthogonal axis, as desired.

For the other shear with no rotation, we obtain

$$\mathbf{A} = \begin{bmatrix} C & 0.0 \\ 0.0 & L \end{bmatrix} \quad \widetilde{\mathbf{A}} = \begin{bmatrix} C & D \\ \frac{CD}{L} & L \end{bmatrix} \mathbf{e} = \begin{bmatrix} 0 & \frac{D}{L} \\ \frac{D}{L} & 0 \end{bmatrix} \tag{A.45}$$

$$\varepsilon = \begin{bmatrix} 0 & \frac{1.0D}{L} \\ \frac{1.0D}{L} & 0 \end{bmatrix} \omega = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{A.46}$$

Notice that if C and L are of different lengths, we need to scale the distortion to the length of each side to obtain a pure shear with no axial strain or rotation.

For a pure rotation as shown in fig. A.4d), we just need to multiply with a rotation matrix. We obtain the following equations

$$\mathbf{A} = \begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{A.47}$$

$$\widetilde{\mathbf{A}} = \mathbf{MA} = \begin{bmatrix} C\cos(\theta) & -L\sin(\theta) \\ C\sin(\theta) & L\cos(\theta) \end{bmatrix} \qquad \mathbf{e} = \begin{bmatrix} \cos(\theta)-1 & -\sin(\theta) \\ \sin(\theta) & \cos(\theta)-1 \end{bmatrix} \tag{A.48}$$

$$\varepsilon = \begin{bmatrix} 1.0\cos(\theta)-1.0 & 0 \\ 0 & 1.0\cos(\theta)-1.0 \end{bmatrix} \omega = \begin{bmatrix} 0 & -1.0\sin(\theta) \\ 1.0\sin(\theta) & 0 \end{bmatrix}. \tag{A.49}$$

Notice that although we expected ε to be just zero, we actually have a component that depends on the cosine of the rotation angle. For small angles, this cosine approaches 1, and ε is exactly zero. This is the disadvantage of using the infinitesimal strain tensor: any rotation between the initial and final lattice affects the measure of the strain. This not a desired behavior since such a rigid rotation does not affect the energy of the system (interatomic distances remain constant), so it is not modifying the system in a relevant way. This is the real limitation of the small strain tensor: it is inaccurate for large rotations. For rotations of 90 or 180 degrees for example, it is completely wrong with strains being the opposite of what they should be, or exchanged in axis.

A.3.3 Finite Strain Tensor

To solve the rotation problems of the linear strain tensor, the finite or Lagrange-Green strain tensor is introduced, defined as

$$\eta = 0.5(\mathbf{e} + \mathbf{e}^T + \mathbf{e}^T\mathbf{e}) = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} \\ \gamma_{yx} & \varepsilon_{yy} \end{bmatrix}, \tag{A.50}$$

where we still identify the components of η wit the linear and shear strains, but as we will see later, they are slightly different from the ones obtained for the infinitesimal strain tensor.

This new tensor is immune to rotation between the initial and deformed conditions, since

$$\eta = 0.5(\mathbf{e} + \mathbf{e}^T + \mathbf{e}^T\mathbf{e}) = 0.5(\mathbf{A}^{-T}\widetilde{\mathbf{A}}^T\widetilde{\mathbf{A}}\mathbf{A}^{-1} - \mathbf{I}) = 0.5(\mathbf{A}^{-T}\mathbf{G}_{\widetilde{\mathbf{A}}}\mathbf{A}^{-1} - \mathbf{I}), \tag{A.51}$$

where $\mathbf{G}_{\widetilde{\mathbf{A}}}$ is the metric tensor of $\widetilde{\mathbf{A}}$, which as explained before does not depend on the orientation of the lattice orientation. In this way, we get rid of rigid rotations between the initial and deformed lattice, but not of rotation between the initial lattice and the Cartesian axis (see next section).

A disadvantage of this tensor is that it is no longer linear with strain, and second order terms start appearing when calculating normal and shear strains. For the strains calculated before, we have, for eq.A.42 (pure linear strain), the result

$$\eta = \begin{bmatrix} \frac{\Delta L}{L} + \frac{(0.5\Delta L)}{L^2} & 0 \\ 0 & 0 \end{bmatrix}. \tag{A.52}$$

Notice the appearance of the additive $0.5\Delta L/L^2$ term that was not present before. This is a consequence of utilizing the finite tensor, the introduction of quadratic higher order terms, and the reason why the infinitesimal tensor is known as "linear". As the magnitude of the strain decreases this term goes to zero and the two strain tensors coincide.

For eqs. A.44 and A.46 (shear with and without rotation), the result is

$$\eta = \begin{bmatrix} 0 & \frac{0.5D}{L} \\ \frac{0.5D}{L} & \frac{0.5D^2}{L^2} \end{bmatrix} \tag{A.53}$$

$$\eta = \begin{bmatrix} \frac{0.5D^2}{L^2} & \frac{1.0D}{L} \\ \frac{1.0D}{L} & \frac{0.5D^2}{L^2} \end{bmatrix}. \tag{A.54}$$

Notice that now quadratic terms have been included in the diagonal where we would expect to see the axial strain. Once again, these terms go to zero as the deformation becomes small. These extra terms of the finite tensor are not a problem, since they are usually small, and in either case as long as the notation being used is clear and explicit they are well defined.

Eq.A.49 (only rigid body rotation) results in

$$\eta = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{A.55}$$

which gives us the expected zero strain tensor.

A.3.4 Strain for Rotated Systems

In the cases inspected until now, the original lattice cell coincided with the Cartesian axis. If we have an initial configuration where the lattice vectors form an angle with the Cartesian axis, the interpretation of the strain tensor becomes more complicated. The strain we have studied until now **is expressed with respect to the Cartesian axis coordinates**, and **NOT** with respect to the lattice matrices provided. That is, ε_{xx} measures the stretch or compression of a line originally parallel to the X vector, and not necessarily parallel to the lattice vectors of our cell.

To recover the strain as expressed in our lattice coordinate system, we need to perform a coordinate basis change on the strain tensor, using a normalized version of the lattice matrix (equivalent to turning our lattice vectors into directional unit versors and multiplying by the Cartesian versors)

$$\varepsilon' \text{ or } \eta' = \mathbf{C}^T (\varepsilon \text{ or } \eta)\mathbf{C} \tag{A.56}$$

$$\mathbf{C} = \begin{bmatrix} \cos\alpha & \cos\beta \\ \sin\alpha & \sin\beta \end{bmatrix} = \begin{bmatrix} \hat{a_1}\cdot\hat{x} & \hat{a_2}\cdot\hat{x} \\ \hat{a_1}\cdot\hat{y} & \hat{a_2}\cdot\hat{y} \end{bmatrix}, \tag{A.57}$$

where $\mathbf{C}$ is the so called cosine matrix, α and β are the angles between the first lattice vector and the x-axis, and the second lattice vector and the x-axis, respectively; $\hat{x}$ and $\hat{y}$ are the unit versors associated with the Cartesian axis; and $\hat{a_1}$ and $\hat{a_2}$ are the versors of the lattice vectors (unitary lattice vectors).

With this cosine matrix we can undo the rotation originated by $\mathbf{M}$, to recover the strain tensor in the original Cartesian coordinates. This makes it easier to interpret the components of the strain tensor ($\varepsilon_{xx}, \varepsilon_{yy}, \gamma$), but derived properties such as the eigenvalues remain the same with or without this rotation.

A.3.5 Principal Stretches and Principal Strain Directions

As the strain tensor is just a 2 by 2 matrix, we can easily calculate its eigenvalues, a.k.a. principal stretches, as

$$\varepsilon_{1,\,2} = \varepsilon_{\text{max, min}} = \frac{\varepsilon_{xx}+\varepsilon_{yy}}{2} \pm \sqrt{\left(\frac{\varepsilon_{xx}-\varepsilon_{yy}}{2}\right)^2 + \gamma_{xy}^2}, \tag{A.58}$$

for either the finite or infinitesimal versions. As with any matrix, the eigenvalues allow us to diagonalize the strain matrix. This gives an interpretation for the associated eigenvectors as the principal strain directions: these are the directions that suffer only pure stretch with no shear strain.

We can calculate the spontaneous strain from these eigenvalues to obtain a measure of general distortion, $\varepsilon_{\mathrm{DOLD}}$, as

$$\varepsilon = 0.5\sqrt{(}\varepsilon_1^2 + \varepsilon_2^2). \tag{A.59}$$

The eigenvalues are associated with the invariants of the strain tensor, properties that remain constant no matter the orientation of the coordinate system

$$I_1 = \Theta = \mathrm{tr}(\eta) = \sum_{\mathrm{i}} \eta_{\mathrm{ii}} = \sum_{\mathrm{i}} \varepsilon_{\mathrm{i}}, \tag{A.60}$$

$$I_2 = \Phi = 0.5(\mathrm{tr}^2(\eta) - \mathrm{tr}(\eta^2)) = \sum_{\mathrm{i}\neq\mathrm{j}} \eta_{\mathrm{ii}}\eta_{\mathrm{jj}} - \sum_{\mathrm{i}\neq\mathrm{j}} \eta_{\mathrm{ij}}\eta_{\mathrm{ji}} = \sum_{\mathrm{i}\neq\mathrm{j}} \varepsilon_{\mathrm{i}}\varepsilon_{\mathrm{j}}, \tag{A.61}$$

$$I_3 = \Psi = \det(\eta) = \eta_{11}\eta_{22} - \eta_{12}\eta_{21} = \prod_{\mathrm{i}} \varepsilon_{\mathrm{i}}, \tag{A.62}$$

where tr is the trace of a matrix, η_{ij} are the components of the η tensor, ε_i are the eigenvalues of said tensor; I_1, the first invariant, is associated with the hydrostatic strain (diagonal elements), I_2, the second strain invariant, is associated with the deviatoric strain, and global strain measures such as the von Mises strain or $\varepsilon_{\mathrm{DOLD}}$, and I_3, the third strain invariant, is not so simple to interpret physically but is just the determinant of the strain tensor. The determinant in I_3 is of course only valid for a 2 by 2 strain tensor. For large tensors the expressions in terms of the tensor components becomes more complicated, for this reason it is usually easier to work directly with the eigenvalues if available. Since they do not depend on the coordinate system, we can calculate them for the diagonalized strain tensor (where the eigenvalues are in the diagonal), or for the original, non diagonalized version, with the results of course being the same.

A.3.6 Worked Example

The following shows the values for all these matrices and eigenvalues for coincidence lattice a of table A.2.

$$\mathbf{A} = \begin{bmatrix} 3.4964 & 10.3054 \\ 5.8437 & 5.7375 \end{bmatrix} \quad \widetilde{\mathbf{A}} = \begin{bmatrix} 3.2900 & 9.8700 \\ 5.2969 & 5.2969 \end{bmatrix} \tag{A.63}$$

$$\mathbf{e} = \begin{bmatrix} -0.0339 & -0.0151 \\ 0.0140 & -0.1019 \end{bmatrix} \quad \varepsilon = \begin{bmatrix} -0.0339 & -0.0005 \\ -0.0005 & -0.1019 \end{bmatrix} \tag{A.64}$$

$$\eta = \begin{bmatrix} -0.0332 & -0.0010 \\ -0.0010 & -0.0966 \end{bmatrix} \eta' = \begin{bmatrix} -0.0400 & 0.0196 \\ 0.01963 & -0.0898 \end{bmatrix} \tag{A.65}$$

$$\omega = \begin{bmatrix} 0.0 & -0.0145 \\ 0.0145 & 0.0 \end{bmatrix} \tag{A.66}$$

From η' (or η, since these are invariant under rotations, save some round-off errors), the eigenvalue related quantities are

$$\varepsilon_1 = -0.0332 \; \varepsilon_2 = -0.0966 \tag{A.67}$$

$$\varepsilon_{\mathrm{DOLD}} = 0.0511 \quad . \tag{A.68}$$

Appendix B

RuNNer Settings

```
#####################################################
### general keywords
#####################################################
#Commented lines have been deactivated, but are kept
#to show that that option has been explicitly disallowed
nn_type_short 1        # 1=Behler-Parrinello
runner_mode 1/2        # 1: SF Generation 2: Fitting
number_of_elements 3
elements O Cu Zn
random_seed 1000       # seed for initial random weight parameters
                       #and train/test splitting
random_number_type 1   # 1=ran0, 2=ran1, 3=ran2, 4=ran3
#remove_atom_energies # remove atomic energies before fitting
energy_threshold 0.00 # energy threshold for fitting data in Ha per atom
bond_threshold 0.5d0  # threshold for the shortest bond in structure
#force_threshold 10.0

#####################################################
### NN structure of the short-range NN
#####################################################
use_short_nn           # use NN for short range interactions
global_hidden_layers_short 2
global_nodes_short 20 20
global_activation_short t t l

#####################################################
### symmetry function generation ( mode 1):
#####################################################
test_fraction 0.20000 # threshold for splitting between fitting and test set

#####################################################
### symmetry function definitions (all modes):
#####################################################
# INFO: symfunction format: reference atom, type, neighbor element 1
#(and neighbor element 2), symfunction parameters
#
# GLOBAL SYMMETRY FUNCTIONS FOR SHORT RANGE NN
# SAMPLE TYPE 2: global_symfunction_short 2 7.14214 0.0 11.338
#                                  ! type eta rshift funccutoff
# SAMPLE TYPE 4: global_symfunction_short 4 7.14214 11.338
#                                     ! type eta funccutoff
#-----------------------------------------------------
global_symfunction_short 2 0.0010   0.0000   12.00000
global_symfunction_short 2 0.0100   0.0000   12.00000
global_symfunction_short 2 0.0200   0.0000   12.00000
global_symfunction_short 2 0.0500   0.0000   12.00000
global_symfunction_short 2 0.1000   0.0000   12.00000
global_symfunction_short 2 0.2000   0.0000   12.00000
global_symfunction_short 2 0.050    3.0000   12.00000
global_symfunction_short 2 0.100    3.0000   12.00000
```

```
global_symfunction_short 2 0.200    3.0000   12.00000
global_symfunction_short 2 0.500    3.0000   12.00000
global_symfunction_short 2 0.900    3.0000   12.00000

global_symfunction_short 3     0.0010   1.0   1.0   12.00000
global_symfunction_short 3     0.0010   1.0   2.0   12.00000
global_symfunction_short 3     0.0010   1.0   4.0   12.00000
global_symfunction_short 3     0.0010   1.0  16.0   12.00000
global_symfunction_short 3     0.0010  -1.0   1.0   12.00000
global_symfunction_short 3     0.0010  -1.0   2.0   12.00000
global_symfunction_short 3     0.0010  -1.0   4.0   12.00000
global_symfunction_short 3     0.0010  -1.0  16.0   12.00000
global_symfunction_short 3     0.0030   1.0   1.0   12.00000
global_symfunction_short 3     0.0030   1.0   4.0   12.00000
global_symfunction_short 3     0.0030  -1.0   1.0   12.00000
global_symfunction_short 3     0.0030  -1.0   4.0   12.00000

####################################################
### fitting (mode 2): general inputs for short range:
####################################################
epochs 15                # number of fitting epochs
points_in_memory 2500    # max number of structures in memory
mix_all_points           # mix order of training points
scale_symmetry_functions # scale symmetry functions
fitting_unit  eV         # unit for error output in mode 2 (eV or Ha)
#normalize_nodes         #normalize incoming values to node to avoid saturation
precondition_weights     # precondition initial weights
print_mad                #print other error measurements
#check that forces in a structure add up to zero/a given threshold
check_input_forces 0.1

####################################################
### fitting options ( mode 2): short range part:
####################################################
# optimization mode short range energies(1=Kalman filter,
# 2=conjugate gradient, 3=steepest descent)
optmode_short_energy 1
optmode_short_force 1        # optimization mode short range forces (same)
short_energy_error_threshold 0.1  # threshold of adaptive Kalman filter short E
short_force_error_threshold 1.2   # threshold of adaptive Kalman filter short F
kalman_lambda_short 0.98000  # Kalman parameter short E/F
kalman_nue_short 0.99870     # Kalman parameter short E/F

short_energy_group 1         # group energies for update
short_energy_fraction 0.75   # percentage of energies used for fitting 100%=1.0
short_force_group 2          # group forces for update
short_force_fraction 0.03    # percentage of forces used for fitting 100%=1.0
use_short_forces             # use forces for fitting

weights_min -0.5       # minimum value for initial random short range weights
weights_max  0.5       # maximum value for initial random short range weights

max_force 0.25               # do not use larger forces in Ha/Bohr for update
#max_energy -0.09245
scale_min_short 0.0          # minimum value for scaling (scmin_short)
scale_max_short 1.0          # maximum value for scaling (scmax_short)
#force_grouping_by_structure # group all forces within one structure for fitting
```

Listing B.1: Example RuNNer input settings file for a N_{SF}-20-20-1 architecture NNP, with the SF listed in sec. D

Appendix C

VASP Settings

```
SYSTEM=VASP CALCULATION

#INITIALIZATION
ISTART = 0      # restart from scratch
ICHARG = 2      # initial charge: from atomic charge densities
INIWAV = 1      # random initialization for wf
NELM   = 100     # maximum of NELM electronic steps
NELMIN =  2     # minimum of NELMIN convergence steps
NELMDL = -5     # no update of charge for NELMDL steps
EDIFF  = 1.00E-06  # accuracy for electronic minimization
PREC = Accurate # precision level for various other settings
GGA = PE #x-c potential
ALGO = Fast #diagonalization algorithm

ENCUT  = 500   #energy cutoff of planewaves
ISMEAR = 0; SIGMA = 0.1; #Fermi level treatment
#Gaussian(0)

#MISC
LREAL = A #projection in real or reciprocal space
NSIM = 8
LASPH = .TRUE. #aspherical contributions
```

Listing C.1: VASP input settings utilized in this work

Appendix D

Symmetry Functions for Ternary CuZnO Systems

Based on the SF values presented in 93. The SF set described here applies to all element pairs and triplets.

Number	η (eta) (1/Bohr^2)	R_{shift} (Bohr)	Cutoff (Bohr)
1	0.001	0.0	12.0
2	0.010	0.0	12.0
3	0.020	0.0	12.0
4	0.050	0.0	12.0
5	0.100	0.0	12.0
6	0.200	0.0	12.0
7	0.050	3.0	12.0
8	0.100	3.0	12.0
9	0.200	3.0	12.0
10	0.500	3.0	12.0
11	0.900	3.0	12.0

Table D.1: Parameters for radial symmetry functions G^2 for all element combinations.

Number	η (eta) (1/Bohr^2)	ζ (zeta)	λ (lambda)	Cutoff (Bohr)
1	0.001	1.0	1.0	12.0
2	0.001	2.0	1.0	12.0
3	0.001	4.0	1.0	12.0
4	0.001	16.0	1.0	12.0
5	0.001	1.0	-1.0	12.0
6	0.001	2.0	-1.0	12.0
7	0.001	4.0	-1.0	12.0
8	0.001	16.0	-1.0	12.0
9	0.003	1.0	1.0	12.0
10	0.003	4.0	1.0	12.0
11	0.003	1.0	-1.0	12.0
12	0.003	4.0	-1.0	12.0

Table D.2: Parameters for angular symmetry functions G^4 for all element combinations.

Appendix E

Neural Network Dispersion Plots

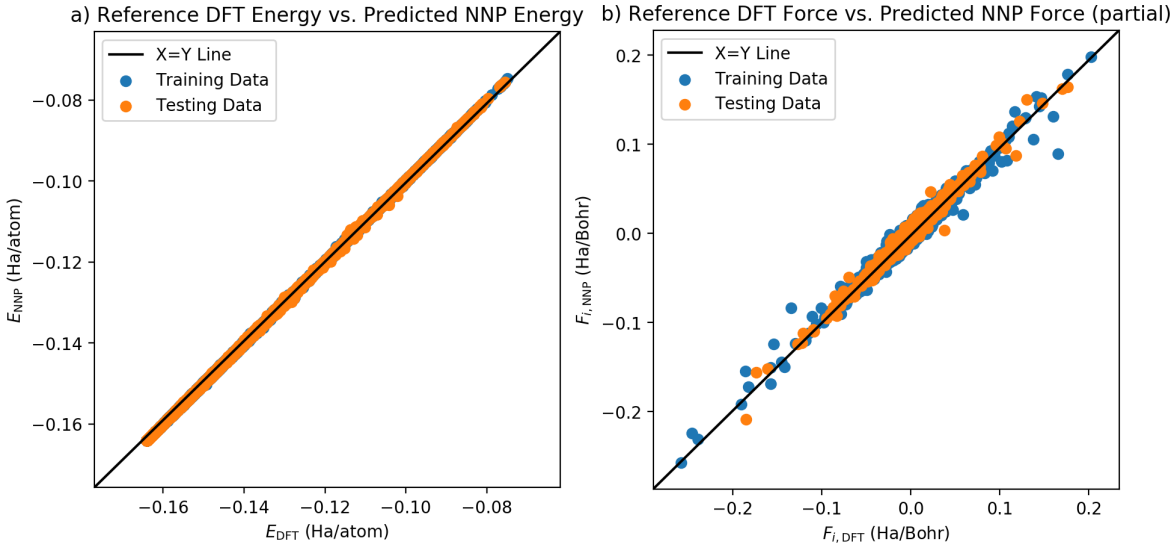

Figure E.1: Comparison of reference (DFT) and predicted (NNP) quantities. a) Energy b) Force component (due to the amount of force components in the dataset, only a portion of the total is shown).

Figure E.1 shows the predicted and reference energies and forces for the selected NNP utilized in this work, for both training and testing data. Both energies and forces are predicted with high accuracy, as can be seen from most of the data lying on the X=Y line. The predicted energy shows slight deviations at very high and very low energy values, while the force prediction shows a somewhat wider spread with some outliers.

Appendix F

Structural and Energetic Parameters for Cu and ZnO

Tables F.1 presents the solid lattice constants of Cu and ZnO, as calculated with the NNP, the reference DFT method, and experimental values. Table F.2 contains the surface energies for the three low-index surfaces of Cu, as calculated with the NNP and the reference DFT method.

Parameter (Unit)	NNP	DFT	Exp. [202]
fcc Cu a (Å)	3.64	3.64	3.61
wurtzite ZnO a (Å)	3.29	3.30	3.25
wurtzite ZnO c (Å)	5.35	5.38	5.21
wurtzite ZnO u (frac. dist. Zn-O)	0.385	0.385	0.382

Table F.1: Equilibrium lattice constants for Cu and ZnO, as obtained with DFT (reference data for the NPP) and the NNP.

Parameter (Unit)	NNP	DFT
$\gamma_{Cu(100)}$ (meV/Å^2)	166.0	166.0
$\gamma_{Cu(110)}$ (meV/Å^2)	164.6	164.8
$\gamma_{Cu(111)}$ (meV/Å^2)	120.9	121.2

Table F.2: Surface energies for the different Cu low index Miller surfaces, as obtained with the usual algorithm by Fiorentini and Methfessel [295] method, with DFT and the NNP.