# Untersuchung oxidierter Kupferoberflächen mithilfe von Gaußschen Approximationspotenzialen

## Investigation of oxidized Cu surfaces using Gaussian Approximation Potentials

**Wissenschaftliche Arbeit zur Erlangung des Grades**

**Master of Science**

**an der Fakultät für Chemie der Technischen Universität München**


**Betreut von** Professor Dr. Karsten Reuter

und Dr. Nicolas Georg Hörmann

Lehrstuhl für Theoretische Chemie

**Eingereicht von** Nicolas Bergmann

Berlin


**Eingereicht am** Garching, den 30.04.2021

# Zusammenfassung

Kupfer kann ein effizienter CO-Oxidationskatalysator sein. Zuvor muss die Kupferoberfläche jedoch aktiviert werden, was durch morphologische Veränderungen geschieht.[1] Die detaillierte Struktur und Zusammensetzung der veränderten Oberflächenoxidschicht ist bisher unbekannt, was mechanistische Analysen erschwert. Um die Oberflächenstruktur von Cu (111) mit unterschiedlichen Konzentrationen von adsorbiertem Sauerstoff systematisch zu untersuchen, verwenden wir ein durch machine learning trainiertes "Gaussian Approximation Potential" (GAP).[2] In dieser Arbeit diskutieren wir, wie das Cu – O GAP iterativ trainiert wurde, wie sich die verschiedenen Hyperparameter auf die Leistung des GAP auswirken und wie der Trainingsprozess verbessert werden kann. Das GAP basiert auf Daten verschiedener Kupferoxide, errechnet durch Dichtefunktionaltheorie. Des Weiteren verwenden wir Oberflächenstrukturen, die wir durch Molekulardynamik erzeugt haben.

In einer ersten Anwendung des Cu – O GAP benutzen wir Molekulardynamik-Berechnungen bei 300 K, um Cu (111) Oberflächen mit unterschiedlichen Sauerstoffbedeckungen zu analysieren. Wir beobachten, dass auf Cu (111) Oberflächen eine Sauerstoffbedeckung von über 0.25 Monolagen zur Inselbildung und zur Sauerstoffdiffusion unter die Oberfläche führt. Zusätzlich führen wir dieselben Berechnungen auf Oberflächen mit einer niedrigeren Cu-Dichte durch. In diesen Oberflächen fehlen 25 % der Cu-Atome in der obersten Schicht. Ähnliche Dichten wurden in Experimenten bei Anlegen von Oxidationspotenzialen beobachtet, wobei diese niedrigere Cu-Dichte höchstwahrscheinlich mit dem Einbau von Sauerstoff in das Gitter zusammenhängt.[1]

Um die lokalen Oberflächenmotive zu analysieren, klassifizieren wir lokale atomare Umgebungen mithilfe des Deskriptors "Smooth Overlap of Atomic Positions" (SOAP).[3] Wir vergleichen die oxidierten Cu (111) Oberflächen mit einer Auswahl an Kupferoxidoberflächen mit niedrigen Millerschen Indizes. Bei hohen Sauerstoffbedeckungen finden wir eine erhöhte Ähnlichkeit mit $Cu_2O$ (111), unabhängig davon, ob es sich um eine Cu (111) Oberfläche mit hoher oder niedriger Cu-Dichte handelt.

# Abstract

Copper was recently shown to exhibit promising capabilities toward electrochemical CO oxidation, yet only after undergoing activating surface morphological changes.[1] The detailed structure and composition of the formed surface oxidic layer is hitherto unknown, preventing further mechanistic analyses. We use a machine-learned Gaussian Approximation Potential (GAP)[2] to systematically investigate the Cu(111) surface structure with varying concentrations of adsorbed oxygen. In this thesis we discuss the iterative method of training the Cu – O GAP, how the different hyperparameters impact the GAP's performance, and how this process can be improved. The GAP is trained on density-functional-theory data of different bulk copper oxides and molecular-dynamics-generated slab structures.

In a first application of the Cu – O GAP we use molecular dynamics calculations at 300 K on Cu(111) slabs with different oxygen coverages. We find that on Cu (111) surfaces an oxygen coverage above 0.25 monolayers (ML) leads to island formation and subsurface oxygen. Additionally we run the same calculations on low-density Cu (111) surfaces, where 25 % of the Cu atoms are missing in the top layer. Similar Cu densities were observed in experiments upon application of oxidizing potentials, where this lower density is most likely related to the incorporation of O in the lattice rather than surface charging.[1]

To further analyze local surface motifs we create a classifier using the "Smooth Overlap of Atomic Positions" (SOAP) local atomic environment descriptor.[3] We compare the oxidized Cu (111) surfaces to low-index surfaces of common-bulk copper oxides. As oxygen coverage increases, we find a high similarity to $Cu_2O$ (111), regardless of high- or low-density Cu (111) surface.

# Contents

# List of Tables

# List of Figures

# 1 Introduction

Copper is a promising electrocatalyst for a variety of carbon-oxygen reactions.[4] Among these reactions is the carbon dioxide reduction reaction ($CO_2$RR) and the electro-oxidation of CO, both critical reactions to the climate crisis, as $CO_2$ is an infamous greenhouse gas and CO is the primary poison to Pt electrodes in fuel cell reactions.[5,6] A common observation is that changing the Cu surface morphology increases the catalytic abilities, common approaches being alloying Cu with other metals or metal-derivatives such as ZnO, or introducing nanostructures on the Cu surface.[7,8] A classic example of the latter is "oxide-derived copper" (OD-Cu), a nanocrystalline Cu surface created by reduction of $Cu_2O$-derivatives.[9,10]
The exact reason behind this increased activity is not yet fully understood. One belief expressed by Eilert *et al.* in 2017 is that during oxidation of a clean Cu surface, some O atoms are pushed into subsurface layers, remaining there even after reduction of the copper oxide.[10] The authors propose that this causes changes to the electronic structure of the catalyst, resulting in higher activity and selectivity, a proposal refuted the following year by Garza *et al.*.[11]
Another explanation was proposed by Auer *et al.* in 2020.[1] In their paper, they argued that a Cu (111) surface electrooxidizes CO more efficiently after undergoing an oxidation-reduction cycle due to expulsion of Cu atoms out of the surface during oxidation. These low-coordinated Cu adatoms at the surface favor CO adsorption, increasing the activity in an electrochemical environment. The restructuring pathway of clean copper surfaces to this low-density structure is not known, and although density-functional theory (DFT) calculations on model "restructured" surfaces support the energetic observations made by Auer *et al.*, the larger structure of the oxidation-induced restructuring is not yet resolved.

Solving these unknowns could be aided by molecular simulation techniques. However the high computational cost of "first principle" (*ab initio*) methods such as DFT prevent analysis of large-scale copper structures, while the accuracy of classical force fields is not sufficient.[12] In this thesis we make use of Gaussian approximation potentials (GAPs), a machine-learned force field trained on *ab initio* calculated data.[13] Introduced in 2010 by Bartók *et al.*, GAPs approximate the potential energy surface (PES) of a system by comparing local atomic environments of unknown structures to structures in the training set using Gaussian process regression.[2] To do this GAPs use local descriptors such as the

"Smooth overlap of atomic positions" (SOAP) descriptor.[14] The result is a force field with near DFT-accuracy at a fraction of the computational cost.

The thesis is structured as follows: First, we discuss the pros and cons of PES modeling methods, including DFT, classical and machine-learned force fields. Next, we describe how we trained and tested the performance of a GAP for copper oxide structures. Finally, we present a first application of the CuO GAP on different terminations of oxidized Cu (111) surfaces, using SOAP descriptors to resolve local structural motives.

# 2 Theoretical Background

Molecular simulation techniques are limited by the computational cost of the calculations, prohibiting the accurate modeling of complex molecular problems. Currently, the workhorse for "first principles" (*ab initio*) methods in materials simulation is density-functional theory (DFT). However, if we want to look at complex systems and their time evolution, the high computational cost becomes an unsurmountable problem. Typically, there are two methods of dealing with this issue, both at the cost of accuracy. Either we coarse grain our system (e.g. instead of including an entire surface in our model, we only model trial surface terminations in small cells), or we use computationally cheaper methods, e.g. force fields (FF). In the past years, Machine learning (ML) has emerged as a promising candidate to obtain interaction potentials orders of magnitude faster than *ab initio* methods, while still maintaining their level of accuracy. The main reason for this is that the ML-FF is fitted to a database typically generated at DFT level. In this chapter, we will discuss the differences between FFs and DFT, before explaining the approach of the ML-FF used in this thesis: the so-called Gaussian Approximation Potential (GAP).

## 2.1 Density Functional Theory

Fundamentally, the total energy of any system can be computed from the Schrödinger equation, via

$$\hat{H}\Psi = E\Psi \tag{2.1}$$

If the Born-Oppenheimer approximation is valid, then the ground state energy of the entire system is only dependent on the geometry of the atoms $\mathbf{R}$. The potential energy $E(\mathbf{R})$ is often called the potential energy surface (PES).[15] The electronic wave function $\psi_e(\mathbf{r})$, with $\mathbf{r}$ denoting the position variables of the electrons, solves the corresponding stationary Schrödinger equation:[16]

$$\hat{H}(\mathbf{r},\mathbf{R})\psi_e(\mathbf{r}) = E(\mathbf{R})\psi_e(\mathbf{r}). \tag{2.2}$$

The complexity in (2.2) arises from solving the eigenvalue problem for the electronic wave function $\psi_e(\mathbf{r})$. Typical eigenvalue solving algorithms, such as those in the ELPA library, scale as $O(N_{\text{elec.}}^3)$.[17] This is one reason why the Hartree-Fock method, the computationally cheapest technique which treats electrons explicitly, scales to $O(N_{\text{elec.}}^4)$.[18]

In a system with $n$ electrons, the dimension of the wave function would be $3n$ ($4n$ if we additionally consider the spin of the electrons). A system with 22 electrons, such as the $CO_2$ molecule, hypothetically would thus have a computational cost of $O\left(10^5\right)$. In this work, we typically deal with systems on the scale of 1800 electrons, the computational cost would be a factor of $\approx (100)^4 = 10^8$ higher, and thus practically not solvable.
DFT addresses the dimensionality problem by describing the PES as a functional of the electron density $\rho(\mathbf{r})$, a three dimensional observable.[19]

$$\rho(r) = N \int \mathrm{d}\mathbf{1}\mathrm{d}\mathbf{2}\ldots\mathrm{d}\mathbf{N}\,|\Psi(\mathbf{1},\mathbf{2}\ldots\mathbf{N})|^2 \tag{2.3}$$

In 1964, Hohenberg and Kohn stated that all properties of the system (such as the Hamiltonian $\hat{H}$, wave function $\Psi_0$ and the ground state energy $E_0$) are uniquely determined by the ground state electron density $\rho_0(r)$.[20] This is known as the Hohenberg-Kohn existence theorem. As it includes the wave function $\Psi_0$, we are permitted to write the expectation values of observables as functionals of the ground state electron density.[20] The Hohenberg-Kohn variational theorem states that the energy functional of any trial electron density $E_0\left[\rho_{\text{trial}}(r)\right]$ will be greater or equal than the energy at ground state density.[19]

$$E_0\left[\rho_{\text{trial}}(r)\right] \geq E_0[\rho_0(r)] \tag{2.4}$$

Through (2.4) our goal becomes clear: We need to find the ground state electron density $\rho_0(r)$. The most common method of doing so is Kohn-Sham DFT.[21] Presented in 1965, the idea is based on writing the electron density $\rho$ as from a Slater determinant of non-interacting electrons (in this case we neglect the spin) in an external potential $v_{\text{ext}}$:

$$\rho\left(\mathbf{r}\right) = \sum_i \left|\phi_{\text{KS},i}\left(\mathbf{r}\right)\right|^2 \tag{2.5}$$

In (2.5) we sum over all occupied Kohn-Sham orbitals $\phi_{\text{KS},i}$. The resulting integral over the entire space $\mathbf{r}$ equals $N$, the total number of electrons in our system. The total energy of our system $E[\rho]$ can be separated into the following terms:[22]

$$E[\rho] = \underbrace{T_{ni}[\rho] + V_{ne}[\rho] + V_{ee}[\rho]}_{\text{non-interacting system}} + \underbrace{\Delta T[\rho] + \Delta V_{ee}[\rho]}_{\text{"correction terms"}=E_{xc}[\rho]} \tag{2.6}$$

The kinetic energy of the non-interacting electrons $T_{ni}$, the nuclei-electron Coulomb interaction $V_{ne}$, and the electron-electron Coulomb interaction $V_{ee}$ are all easy to determine analytically.[22] However as our model neglects interacting electrons, we need to include the correcting terms $\Delta T$ and $\Delta V_{ee}$. Basically what we have done in (2.6) is shift all the unknown components of the energy into two functionals. The sum of these is called the exchange-correlation energy functional $E_{xc}[\rho]$.[15]

(2.6) is an exact expression of the total energy, allowing us to calculate $\phi_{\text{KS}}$ (and thus $\rho_0(\mathbf{r})$) via variational minimization of the Kohn-Sham orbital equation.[21,23]

$$H_{KS}\left[\rho\right]\phi_{\text{KS},i} = -\frac{1}{2}\nabla^2\phi_{\text{KS},i} + v_{\text{KS}}\left[\rho\right]\phi_{\text{KS},i} = \varepsilon_i\phi_{\text{KS},i} \tag{2.7}$$

Where $v_{\mathrm{KS}}$ is the sum of the external potential $v_{\mathrm{ext}}$, the classical electron-electron Coulomb potential $v_{el}$ and the functional derivative of the exchange-correlation energy to the electron density.[23]

$$v_{\mathrm{KS}}[\rho] = v_{\mathrm{ext}} + v_{\mathrm{el}}[\rho] + \frac{\delta E_{xc}[\rho]}{\delta \rho} \tag{2.8}$$

In theory all calculations until now have been exact. However, there is no known exact expression for $E_{xc}$, here we need to work with approximations. In this study, we used the exchange-correlation functional described by Perdew, Burke and Ernzerhof.[24] This functional uses the Generalized Gradient Approximation (GGA) with a constraint-driven approach.[15] This means that in addition to the local value of $\rho(r)$, GGA also takes the inhomogeneity of $\rho(r)$ into account by evaluating its gradient. These exchange correlation functionals are considered as a standard for solids.[25]

Note that the Kohn-Sham Hamiltonian is dependent on an input density $\rho_{\mathrm{in}}$, which is why (2.7) has to be solved iteratively by a self-consistent field approach.[26] More details can be found in [15, 19].

In this work we are mainly concerned with periodic systems. Based on Bloch's theorem, periodic Kohn-Sham orbitals are created by expanding the orbitals to a finite number of plane waves with wave vectors $\mathbf{G}$.[27] This approach is called plane-wave DFT.[28] Plane wave basis sets are characterized by a kinetic energy cutoff $\mathrm{E_{cut}}$:

$$\mathrm{E_{cut}} = \frac{\hbar^2 \mathbf{G}^2}{2m}. \tag{2.9}$$

Larger cutoffs lead to a bigger basis and thus a more detailed representation. To limit computational cost, the core electrons and their effects on valence eletrons are replaced by pseudopotentials.[29] The pseudopotentials used in this thesis are the ultrasoft pseudopotentials from the GBRV database.[30]

DFT is computationally much faster than explicit wave function methods, however it is still prohibitingly expensive to perform verly long molecular dynamics calculations, or calculate the energy of large systems (e.g. $>$ 1000 atoms).

## 2.2 Force Fields

As stated in section 2.1, the potential energy surface (PES) of a given system is dependent on the molecular geometry. Force fields (FF) use the positions and species of the atoms as inputs and use simple mathematical expressions to map the structure to the PES. This is computationally orders of magnitudes quicker than *ab initio* calculations, which require calculating the electronic structure. For instance, there are reports from 1997 on force fields which scale with $O(N_{\mathrm{atoms}})$.[31]

There are multiple classes of such a mapping, classical non-reactive potentials use classical

atomic interactions, such as the Lennard-Jones potential, while newer reactive potentials such as the ReaxFF force field rely on more complex interaction models. The following section briefly introduces typical approaches and assumptions made by force fields.

### 2.2.1 Non-Reactive Potentials

Typically, non-reactive force fields express the total energy as the sum of different energy contributions. Neglecting cross terms, the dependencies of these singular contributions to one another, the expression for the total energy is[32]

$$E_{\text{total}} = \underbrace{E_{\text{bond}} + E_{\text{angles}} + E_{\text{torsion}}}_{\text{Attractive Interactions}} + \underbrace{E_{\text{electrostatic}} + E_{\text{LJ}}}_{\text{Repulsive Interactions}}. \tag{2.10}$$

The terms in (2.10) represent the bond stretching, angle bending, dihedral torsion, electrostatic, and Lennard-Jones energetic contributions, respectively. These can all be computed individually.
For instance, if we wish to calculate $E_{\text{bond}}$, we will need to define all individual bonds $b$ that contribute to the systems energy. The computationally simplest way to simulate the energetic difference to the system's ground state is by using the harmonic approximation. In this case, the two parameters required per bond type is the bond length at the ground state $d_0(b)$ and the force constant of the bond $K_b$.[32]

$$E_{\text{bond}} = \sum_b K_b (d(b) - d_0(b))^2 \tag{2.11}$$

We proceed in a similar fashion for the other contributions to the total energy. What becomes clear is that a high number of parameters are required to gain an accurate PES. Even though there are strategies in order to reduce this process, parametrization remains a delicate task.[33]
A further problem of classical force fields is that the equations describing the system are typically only valid in areas close to the energetic minimum of the system. Staying with the example of the bond stretching energy: If the bond distance $d(b)$ is nowhere near the ground state bond distance $d_0(b)$, the harmonic approximation is invalid.[34] This results in a PES with limited applicability.

Typical applications of these types of force fields are biomolecular simulations, as they are ideal in simulating the bending and rotation of subgroups in lipids, proteins, and polymers.[35]

### 2.2.2 Reactive Potentials

One shortcoming of non-reactive force fields is that they are unable to model reactions, i.e. bond-breaking events. The user is able to define a set of bonded atoms, from which the

force field will calculate bond stretching energies, bond bending energies, etc.. This set of bonds will remain as the basis on which all other properties will be derived.
Reactive force fields have different methods as to how to define bonded and non-bonded interactions. The embedded-atom method (EAM) addresses this problem by defining the energetic contributions of atoms depending on their local environments.[36] In EAM, the potential for a system with $N_{\text{atoms}}$ is defined as:[36]

$$E_{tot} = \sum_{i}^{N_{\text{atoms}}} \left[ F_i(\rho_{h,i}) + \frac{1}{2} \sum_{j,j\neq i} \phi_{ij}(r_{ij}) \right] \tag{2.12}$$

Here, $F_i$ is the energetic cost of embedding the atom $i$ at its location, $\rho_{h,i}$ is the local electron density from all atoms (except $i$) within in a cutoff radius, and $\phi_{i,j}$ is a repulsion term between the atoms $i$ and $j$.[32]

This serves an example as to how reactive force fields principally function: The force field determines which atoms are bonded and treats local interactions and long-distance interactions separately. Reactive force fields such as such as Bond-Order Potentials (BOP)[37], REBO[38], and ReaxFF[39] use similar methods.

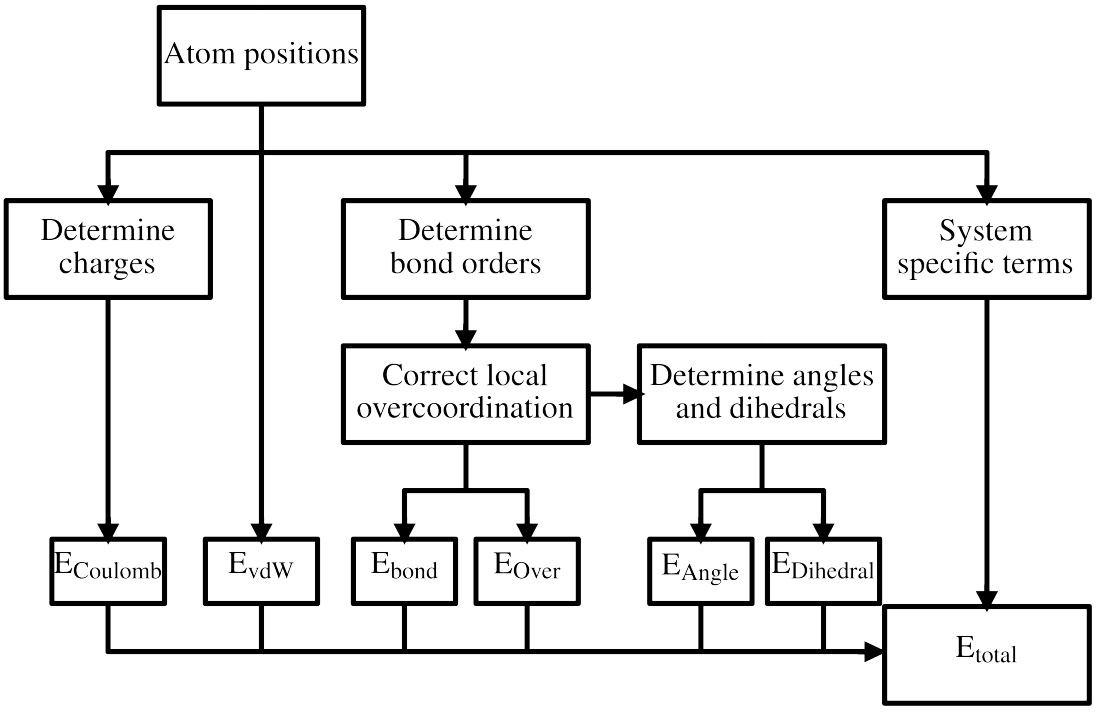



**Figure 2.1** Schematic workflow of the ReaxFF Force Field. Figure adapted from Senftle *et al.*.[39]

## 2.3 Machine Learned Force Fields

Machine learned Force Fields (ML-FF) are currently in the focus of research.[32] Machine Learning is based on algorithms that use data to learn behaviors. After a learning/training phase the respective algorithms are applied to an unknown problem which ultimately allows validation.[40] In ML-FFs, the data are *ab initio* calculations and the task is predicting the PES for structural configurations not in the training set. The success metric is usually the root mean square error (RMSE) of the total energy, or of the individual forces on the atoms.[41]

There are two sets of ML problems: classification and regression. As we are interested in predicting the continuous values of the PES, the problem at hand is a regression problem. The main difference between different ML-FFs is how they solve this regression problem, e.g. first approaches used neural networks (NN) as regressors.[42] The FF used in this thesis uses Gaussian process regression (GPR).

### 2.3.1 Neural Network Potentials

This section serves as a quick introduction of neural network potentials (NNPs). For a more detailed view on the subject we refer to the appendix and [41]. The first class of FFs to implement elements of machine learning were feed-forward neural network potentials (FFNNPs).[43] In principle, these consist of neural networks with three different types of layers. The input layer $G$ contains the geometric information of the input structure. $G$ is then processed by the hidden layer $y$, or more commonly by multiple hidden layers $y_i$. The hidden layer then maps the physical information in $G$ to the output layer $E$ (see Figure 2.2).[41]

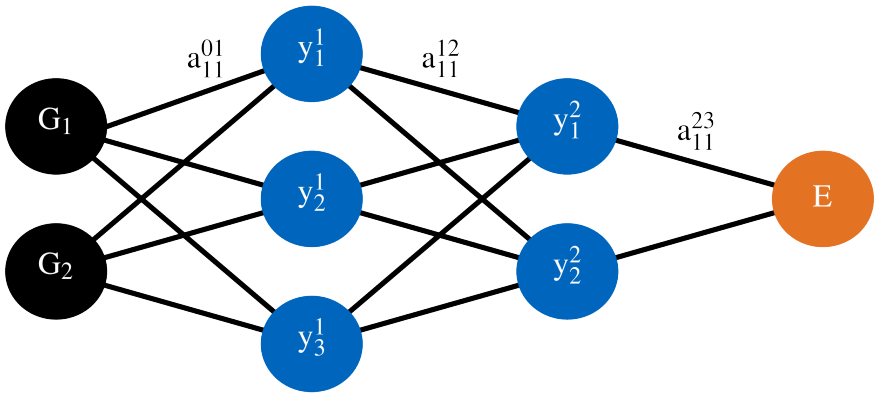



**Figure 2.2** Theoretical structure of a FFNNP with two hidden layers. The figure is a simplified version from [41].

During training, the weighting of the connections between nodes is optimized with the help of algorithms such as back propagation, or the Kalman Filter.[44]

FFNNPs are limited by the rigidity of the input vector. For all applications, the input vectors must have the same dimension as during training. High Dimensional NNPs (HDNNPs), first presented by Behler and Parrinello in 2007[42], solve this by treating the total energy of a system as the sum of many local energetic contributions. The first step of such an approach is translating the atomic information around a central atom to a cutoff radius $R_c$ into a vector with a pre-defined number of dimensions. This translation introduces the concept of a local atomic environment "descriptor" functions, which we discuss in section 2.3.2. Popular descriptor functions for NNP applications are the Behler-Parrinello symmetry functions[45] and the "Smooth Overlap of Atomic Positions" (SOAP) descriptor (see section 2.3.2). These local contributions are then predicted using "atomic" NNs, which are structured identically to FFNNPs.[41]

## 2.3.2 Descriptors

Before we discuss Gaussian Approximation Potentials (GAPs), we will introduce the concept of a local atomic environment descriptor. Potentials such as the HDNNPs and GAPs predict the PES by summing over local energies. This requires a definition of local atomic environments, as the potentials must be able to accurately differentiate between structures.[14,41]

### Properties of a Descriptor

The most classical way to represent the coordinates of atoms is by using Cartesian coordinates. While this representation uniquely describes the location of these atoms, comparing two structures using this method is not suitable. There are multiple cases where Structure $A$ is identical to Structure $B$, however their Cartesian coordinates differ e.g. rotating $A$ or $B$. To solve this problem, a suitable descriptor is "*invariant* with respect to permutational, rotational, reflectional and translational symmetries".[14] Additionally, the descriptor should be able to compare two environments with different numbers of atoms.[41] The first example of descriptors that fulfill these conditions was introduced by Behler and Parrinello in 2007.[42]

### Smooth Overlap of Atomic Positions (SOAP)

The smooth overlap of atomic positions (SOAP) descriptor is constructed by considering the local atomic density $\rho(\mathbf{r})$ within a cutoff $R_c$ around the central atom.[14]

$$\rho(\mathbf{r}) = \sum_{i}^{\text{neighbors}} w_{Z_i}\delta(\mathbf{r}-\mathbf{r_i}) \approx \sum_{i}^{\text{neighbors}} \exp\left(-\frac{|\mathbf{r}-\mathbf{r_i}|^2}{2\sigma_{Z_i}^2}\right) \tag{2.13}$$

To ensure that similar local environments return similar $\rho(\mathbf{r})$, each "neighbor" atom $i$ is represented by a Gaussian with its maximum at $\mathbf{r_i}$. The variance of the Gaussian $\sigma_{Z_i}^2$ is

specific to the atomic species $Z_i$.[14]
A vector-like representation of this descriptor can be readily obtained when (2.13) is expanded in spherical harmonics as[3]

$$\rho(\mathbf{r}) = \sum_{n=1}^{n_{max}} \sum_{l=0}^{l_{max}} \sum_{m=-l}^{l} c_{nlm} g_n(r) Y_{lm}(\hat{\mathbf{r}}) . \tag{2.14}$$

In (2.14), $c_{nlm}$ are the spherical harmonic expansion coefficients,[2] $g_n(r)$ are orthonormal radial basis functions with which we incorporate the distance between the central atom to its neighbors,[14] and $Y_{lm}(\hat{r})$ are the spherical harmonic functions.[14] This representation of the local atomic density is the SOAP descriptor, sometimes called the SOAP fingerprint.[3]

The benefit of the SOAP approach is best explained by comparing two atomic densities $\rho$ and $\rho'$. The similarity is defined as $S(\rho, \rho')$:[14]

$$S(\rho, \rho') = \int \rho(\mathbf{r}) \rho'(\mathbf{r}) \mathrm{d}\mathbf{r} \tag{2.15}$$

Although (2.15) fulfills the permutational invariance and the translational criterion, the value of $S$ changes when either $\rho$ or $\rho'$ is rotated by $\hat{R}$. The rotational invariance criterion is satisfied by integration of (2.15) over all possible rotations.[3,14] Therefore Bartók *et al.* introduce the following similarity measure, with the exponent $n > 1$.[3,14]

$$k(\rho, \rho') = \int \mathrm{d}\hat{R} \left| \int \rho(\mathbf{r}) \rho'(\hat{R}\mathbf{r}) \mathrm{d}\mathbf{r} \right|^n \tag{2.16}$$

This similarity kernel $k(\rho, \rho')$ can be computed analytically using the spherical harmonics expansion (2.14). Additionally, if the set of radial basis functions is orthonormal and $n = 2$, then (2.16) corresponds to the inner product of the power spectra vectors $\mathbf{d}$ of the local densities .[14]

$$k(\rho, \rho') = \sum_{n,m} \sum_{n',m'} \sum_{l} c_{nlm} \left(c'_{nlm'}\right)^* \left(c'_{nlm}\right)^* c_{n'lm'} = \sum_{n,n',l} d_{nn'l} d'_{nn'l} \tag{2.17}$$

This means that the only data that we require in order to measure similarities is the power spectrum vector $\mathbf{d}$ of each local environment. The elements of this vector are computed by:[3]

$$d(\mathbf{r})^{Z_1 Z_2}_{nn',l} = \pi \sqrt{\frac{8}{2l+1}} \sum_m \left(c^{Z_1}_{nlm}\right)^\dagger c^{Z_2}_{n'lm} \tag{2.18}$$

In summary, the SOAP descriptor vector $\mathbf{d}$ is the power spectrum vector of the local atomic density $\rho$[46] and the regular dot product between two descriptor vectors (2.17) corresponds to the rotationally invariant similarity kernel (2.16).[14]

## 2.4 Gaussian Approximation Potentials

The Gaussian Approximation Potential (GAP) method was introduced by Bartók *et al.* in 2010.[13] As with HDNNPs, the total energy of the system is calculated as the sum of the local energies of each atom, the local environment of which is also described via atom-centered descriptors. However, instead of using neural networks for predictions, GAPs use Gaussian process regression (GPR).[2,47] The following paragraphs only touch on the basic procedure of GAPs. For more in depth analysis and description, we refer to the available literature [2, 13, 47].

### 2.4.1 Gaussian Process Regression

In GAPS we deal with two vectors: The atomic configuration vectors $\mathbf{t}$ and the vector of physical observables $\varepsilon$. In $\mathbf{t}$ the element $i$ corresponds to the descriptor vector $\mathbf{d_i}$ of the local environment around the atom $i$. In $\varepsilon$ the $i$-element is the atomic energy $\varepsilon_i$.
Additionally, the data is split into training data and unknown data, which we will connote by the subscript $*$. The goal of the GAP is to predict an unknown energy vector $\varepsilon_*$.
Considering the atom $i$ in the training data, there exists a mapping of the local atomic environment $\mathbf{t_i}$ to the local atomic energy $\varepsilon_i$ that we write as:[46]

$$\varepsilon_i = \varepsilon\left(t_i, \mathbf{w}\right) = \sum_h w_h \phi_h\left(\mathbf{d}_i\right). \tag{2.19}$$

In (2.19) $\varepsilon_i$ is defined by the inner product of the descriptor vectors projected into the feature space by a set of basis functions $\phi$ and a weight vector $\mathbf{w}$.[2] The choice of basis functions is dependent on the chosen covariance function.[47]

As we have no prior information on $\mathbf{w}$, we assume that the prior probability distribution of $\mathbf{w}$ to be a multivariate normal Gaussian distribution.[47]

$$p\left(\mathbf{w}\right) = \mathcal{N}\left(0, \sigma_w \mathbf{I}\right) \propto \exp\left(-\frac{1}{2}\mathbf{w}^T \sigma_w \mathbf{I}\mathbf{w}\right), \tag{2.20}$$

This allows us to simplify the covariance of two different local energies

$$\left\langle \varepsilon_i \varepsilon_j \right\rangle = \sum_{hh'} \left\langle w_h w_{h'} \right\rangle \phi_h\left(\mathbf{d_i}\right) \phi_{h'}\left(\mathbf{d_j}\right) = \sum_{hh'} \sigma_w^2 \delta_{hh'} \phi_h\left(\mathbf{d_i}\right) \phi_{h'}\left(\mathbf{d_j}\right) = \sigma_w^2 \sum_h \phi_h\left(\mathbf{d_i}\right) \phi_h\left(\mathbf{d_j}\right), \tag{2.21}$$

since the covariance of a vector with itself is its variance.

$$\left\langle \mathbf{w}, \mathbf{w} \right\rangle = \sigma_w^2 \tag{2.22}$$

In their 2015 paper, Bartók and Csyányi introduce the kernel function:[2]

$$c(\mathbf{d_i}, \mathbf{d_j}) = \sum_h \phi_h(\mathbf{d_i})\, \phi_h\left(\mathbf{d_j}\right). \tag{2.23}$$

This similarity measure between two local atomic environments allows us to apply the "kernel trick".[47] As (2.21) "is defined solely in terms of inner products in input space"[47], we can proceed considering only the kernels.[2,47] This substantially reduces the required computational memory.
If we calculate the covariance of each element in $\varepsilon$, this leads to the covariance matrix $\mathbf{C}$:

$$\left\langle \varepsilon\varepsilon^T \right\rangle = \sigma_w^2 \begin{pmatrix} c(\mathbf{d_i}, \mathbf{d_i}) & c(\mathbf{d_i}, \mathbf{d_{i+1}}) & \dots & c(\mathbf{d_i}, \mathbf{d_{max}}) \\ c(\mathbf{d_{i+1}}, \mathbf{d_i}) & c(\mathbf{d_{i+1}}, \mathbf{d_{i+1}}) & \dots & c(\mathbf{d_{i+1}}, \mathbf{d_{max}}) \\ \vdots & \vdots & \ddots & \vdots \\ c(\mathbf{d_{max}}, \mathbf{d_i}) & c(\mathbf{d_{max}}, \mathbf{d_{i+1}}) & \dots & c(\mathbf{d_{max}}, \mathbf{d_{max}}) \end{pmatrix} = \mathbf{C}(\mathbf{t}, \mathbf{t}) \tag{2.24}$$

Again, with no prior knowledge we define the prior probability of observing $\varepsilon$ as a multivariate Gaussian function with mean zero and a variance of the covariance matrix $\mathbf{C}$. This is a common assumption of GPR.[47]

$$p(\varepsilon) = \mathcal{N}(0, \mathbf{C}(\mathbf{t}, \mathbf{t})) \propto \exp\left(-\frac{1}{2}\mathbf{t}^T \mathbf{C}(\mathbf{t}, \mathbf{t})^{-1}\mathbf{t}\right). \tag{2.25}$$

How does this help to predict the unknown values of $\varepsilon_*$? If we add these to the vector of known values $\varepsilon$, the prior probability of this vector is:[47]

$$p(\varepsilon \cap \varepsilon_*) = \begin{pmatrix} \varepsilon \\ \varepsilon_* \end{pmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{C}(\mathbf{t}, \mathbf{t}) & \mathbf{C}(\mathbf{t}, \mathbf{t}_*) \\ \mathbf{C}(\mathbf{t}_*, \mathbf{t}) & \mathbf{C}(\mathbf{t}_*, \mathbf{t}_*) \end{pmatrix}\right) = \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{C} & \mathbf{F} \\ \mathbf{F}^T & \mathbf{D} \end{pmatrix}\right) \tag{2.26}$$

We are interested in the posterior probability of $\varepsilon_*$ under the condition that $\varepsilon$ is known. According to Bayes' theorem, the conditional probability is defined as:

$$p(\varepsilon_* | \varepsilon) = \frac{p(\varepsilon \cap \varepsilon_*)}{p(\varepsilon)} = \mathcal{N}\left(\mathbf{F}^T\mathbf{C}^{-1}\mathbf{t}, \quad \mathbf{D} - \mathbf{F}^T\mathbf{C}^{-1}\mathbf{F}\right) \tag{2.27}$$

This is a Gaussian distribution with mean $\mathbf{F}^T\mathbf{C}^{-1}\mathbf{t}$, and a covariance $\mathbf{D} - \mathbf{F}^T\mathbf{C}^{-1}\mathbf{F}$.[47] The derivation of the right side of (2.27) is given in the cited literature.[47,48]
The prediction $\bar{\varepsilon}_*$ is the mean of the probability distribution $p(\varepsilon_*|\varepsilon)$, a common approach in GPR, and thus[47]

$$\boxed{\bar{\varepsilon}_* = \mathbf{C}(\mathbf{t}_*, \mathbf{t})\mathbf{C}(\mathbf{t}, \mathbf{t})^{-1}\varepsilon.} \tag{2.28}$$

This means the required information to make a prediction is the covariance matrix $\mathbf{C}(\mathbf{t}, \mathbf{t})$ and vectorized inputs $\mathbf{t}$ of the training set. Additionally, we must calculate the covariance matrices $\mathbf{C}(\mathbf{t}_*, \mathbf{t})$ of the new test set $\mathbf{t}_*$ and old training sets $\mathbf{t}$.[2,47]

In comparison to regression via neural networks, GPR's definition is more precise mathematically. This allows the user to calculate confidence intervals of the regression process, which can speed up hyperparameter optimization. Additionally, multiple different kernel functions can be used, provided they fulfill the mathematical requirements (see [47], chapter 4).

### 2.4.2 Sparsification

GPR uses the entire training set to accurately create the covariance matrix. This is an advantage as we include all available information in our fitting process, however this can lead to large matrices that slow down computation. For instance, a training set of 300 Structures with 80 atoms per structure, would have to store a matrix of $24000^2$ elements.

To counteract this, an approximation is introduced via sparsifying the training set.[13] The assumption Bartók *et al.* make is that there are redundancies in the input local chemical environments.[13] If one environment is slightly perturbed, the resulting energy will also be similar. By introducing this approximation we change the mathematically exact GPR-regressed potential to a Gaussian approximation potential (GAP).

The sparsification is introduced by taking "all observations in the dataset" under account and finding "representative atomic neighborhood environments".[2] The number of these representative neighborhoods depends on the number of sparse points, a hyperparameter that can be set by the user.

# 3 Computational Methods

## 3.1 Density Functional Theory Calculations

All electronic structure calculations were set up using the AIIDA-Quantum Espresso package and were run using a plane wave basis set.[28,49] The exchange correlation functional used was PBE,[50] and ultrasoft pseudopotentials from the GBRV database (GBRV 1.5).[30]

### Single Point Calculations

To evaluate the atomic forces and the total energy of a specific geometry (e.g. a snapshot within a molecular dynamics trajectory), we performed self-consistent field calculations, also referred to as "single point calculations". We set the electron convergence threshold for self-consistency to $10^{-8}$ Ry and used local-density-dependent Thomas-Fermi screening as the electronic mixing mode, recommended for highly inhomogeneous systems. The mixing factor was set to 0.3. As we use a plane wave basis set, kinetic energy cutoffs of wave functions and charge density have to be set. These were respectively: 45 Ry and 360 Ry. This proportion is recommended for ultra-soft pseudo potentials. For metals, we included a smearing with a value of $2.0 \times 10^{-2}$ Ry, based on Marzari-Vanderbilt-DeVita-Payne cold smearing.[51]

Evaluating periodic systems requires $k$-point sampling of the Brillouin zone.[52] To ensure similar accuracy between structures, we used an approximately constant $k$-point density in the reciprocal space. This means that if the size of the cell increases, the number of $k$-points decreases. We determine the number of $k$-points for a cell vector $a$ according to:

$$k_a = \frac{3.62\,\text{Å} \cdot 10}{\text{length}_a[\text{Å}]} \tag{3.1}$$

With (3.1), the resulting $k$-point sampling is equivalent to a $10 \times 10 \times 10$ $k$-point grid for bulk cubic copper in the conventional unit-cell with cell length 3.62 Å.[53]

### Geometric Relaxation Calculations

Throughout our study, we performed geometric relaxations. For instance, this was the case for the bulk copper oxides that were used for the first training data generation. The electronic convergence parameters were kept identical to that of the single point

calculations. Additionally, the convergence thresholds for total energy and forces were set to $10^{-4}$ Ry and $10^{-4}$ Ry/bohr, respectively for structural relaxations.

### Slab Calculations

The input geometries were derived from previously optimized slabs provided by Nicolas Hörmann. Originally, the structures were four layers of a Cu (111) surface, in a 4x4 cell in *xy* plane. The lattice **M** (3.2) ensures a vacuum layer of ∼16 Å. The slabs are placed in the center of the cell in *z* direction.

$$\mathbf{M} = \begin{pmatrix} 10.248 & 0.000 & 0.000 \\ -5.124 & 8.875 & 0.000 \\ 0.000 & 0.000 & 26.414 \end{pmatrix} \tag{3.2}$$

For slab calculations, the k-point density parallel to the surface is determined with (3.1), whilst the number of k-points in the direction perpendicular to the surface can be kept low due to the loss of periodicity in that direction. Applied to (3.1), **M** results in a *k*-grid of $4 \times 4 \times 1$.

For geometric relaxation calculations on slabs, we first relaxed the structure with a *k*-grid of $1 \times 1 \times 1$, before moving on to the higher density *k*-grid. During these calculations the unit cell vectors and positions of the bottom two Cu layers were fixed, only permitting the top atoms in the slab to move.

### Bulk Calculations

In order to gain reference information on Cu and copper oxides, we performed calculations on the respective bulk structures. These structures were obtained from Materialsproject[54] and from Nicolas Hörmann. The calculation parameters are mostly identical to the slab calculations, with following exceptions: Instead of fixing the unit cell vectors the cell itself is relaxed along with the atomic positions of all atoms.

## 3.2 Molecular Dynamics Calculations

We performed molecular dynamics (MD) calculations with the LAMMPS package.[55] Unless stated otherwise, the calculation parameters for all MD calculations are the following: We used the "full" atom style implemented by the MOLECULE LAMMPS plugin, and defined neighbors up to 2.0 Å above the force cutoff radius, the default setting for LAMMPS. The time step was set to 1 fs. Velocities are sampled via the canonical ensemble using the Nosé-Hoover thermostat. The $T_{\text{damp}}$ parameter, which determines in what time frame the temperature is relaxed, is set to 0.1 ps. To prevent the "flying ice cube effect"[56], the total momentum of the atoms in the cell was set to 0 every 2000 time steps.

Additionally for slab calculations atoms attempting to move through the cell boundaries in $z$ directions are reflected back to the cell. Also atoms in the bottom atomic layer are fixed and excluded from temperature calculations.

## 3.3 Training of Copper Oxide GAP

We created Gaussian approximation potentials (GAPs) with the QUIP Code by Bartók *et al.* and its GAP plugin.[2,13]

The QUIP Code permits training the GAP on up to four physical observables: the total energy, atomic forces, virial stress, and hessians. We chose to train our potential on total energy and atomic forces. This requires setting the `default_sigma` hyperparameter of the QUIP Code, representing the relative weight of each physical observable in the fit, and with which accuracy the data is treated. For instance a value of `default_sigma` $= 0.001\,\mathrm{eV}$ means that the standard deviation of the training error on the observable energy is 0.001 eV. It follows that smaller values offer a more precise treatment of the respective observable, however this simultaneously increases the risk of overfitting. Following an evaluation of Cu data, we set `default_sigma = {`$0.001\,\mathrm{eV}\ \ 0.001\,\mathrm{eV\,\AA^{-1}}$`}`.

**Table 3.1** Table describing `default_sigma` hyperparameter. "default value" refers to the values we use at the beginning of our training. They do not necessarily need to refer to default values of the QUIP package.

| Hyperparameter | Description | "default value" |
|---|---|---|
| `default_sigma` | Accuracy of energy, forces(, stress, and hessians). Smaller values correspond to more importance in the fitting process | `{0.001eV 0.001eV/Å}` |

As explained in section 2.3.2, GAPs require local atomic descriptors. For our GAP we use the two-body pairwise distance descriptor and the "Smooth Overlap of Atomic Positions" (SOAP) descriptor.[14]

### Two-Body Pairwise Distance Descriptor

**Table 3.2** Table describing the hyperparameters of the Two-Body Distance Descriptor.

| Hyperparameter | Description | "default value" |
|---|---|---|
| `cutoff` | Maximum distance of local interactions $r_{\text{cut}}$ | 8.0 Å |
| `add_species` | GAP considers different atomic species | `true` |
| `covariance_type` | Kernel function to determine covariance matrix | `ard_se` |
| `theta_uniform` | Gaussian width of kernel function | 1. |
| `n_sparse` | Number of sparse points | 20 |
| `sparse_method` | Method to determine sparse points | `uniform` |
| `delta` | Relative weight of the descriptor | 0.1 |

The two-body pairwise distance descriptor maps the local energy and the atomic forces to the bond distance between two atoms within a cutoff distance $r_{\text{cut}}$.
By setting `add_species = true`, the descriptor takes the atomic species into account that are participating in the bonding. In our case this means that separate descriptors are created for Cu – Cu, Cu – O, and O – O bonds. To calculate the covariance matrix for each descriptor, we set `covariance_type = ard_se`. This means that the kernel function is the squared exponential kernel, the Gaussian width of which we set to: `theta_uniform=1`. To create GAPs from this descriptor, the information must be sparsified, which requires two inputs: the number of sparse points, and the sparsification method. In this case we use the `uniform` sparsification method. This creates equally large "bins" between bond distance 0 and $r_{\text{cut}}$ in which the data is sorted. The number of sparse points varied throughout the training process. We started with 20 sparse points and a cutoff of $r_{\text{cut}} = 8.0\,\text{Å}$.

### Smooth Overlap of Atomic Positions Descriptor

**Table 3.3** Table describing the hyperparameters of the SOAP Descriptor.

| Hyperparameter | Description | "default value" |
|---|---|---|
| `cutoff` | Maximum distance of local interactions $r_{\text{cut}}$ | 8.0 Å |
| `add_species` | GAP considers different atomic species | `true` |
| `covariance_type` | Kernel function to determine covariance matrix | `linear` |
| `n_sparse` | Number of sparse points | 1000 |
| `sparse_method` | Method to determine sparse points | `cur_points` |
| `n_max` | Maximum number of radial basis functions | 10 |
| `l_max` | Maximum number of angular basis functions | 3 |
| `zeta` | Kernel enhancement power factor | 2 |
| `atom_sigma` | Width of the atomic Gaussians | 0.5 |
| `delta` | Relative weight of the descriptor | 0.1 |

As explained more precisely in section 2.3.2, the soap vector maps a local atomic density around the center atom, with each atom in the vicinity of the central atom being represented by a Gaussian.[14]
For our original generation, the cutoff radius was set to 8 Å, identical to the two-body pairwise distance descriptor. We also distinguish between different atomic species by setting `add_species=True`. The width of the atomic gaussians was set to `atom_sigma = 0.5`. To translate the Gaussians into spherical harmonics, the number of radial basis functions was set to `n_max = 10` and the angular indices to `l_max = 3`.
For our covariance function we chose the dot product kernel, as is suggested for the SOAP descriptor in the literature (see section 2.3.2).[2,14] The kernel returns a value between 0 (no similarity) and 1 (identical environment). To improve the sensitivity the kernel is squared (`zeta` hyperparameter), thus increasing the difference between similar environments.[14,46]
The sparsification process is more complex than for the two-body descriptor. Ideally, we want the sparse points to contain every unique local atomic environment, while disregarding redundancies in the training set. In our case, we used `sparse_method = CUR_points`, which applies the leverage-score CUR algorithm to the training set of SOAP descriptors to get the most diverse set of sparse points possible.[57] We set `n_sparse = 1000`.

As we use two types of descriptors to describe our GAP, we must attribute the weight of each descriptor to the GAP. This can be understood as determining to what extent two-body interactions and local many-body interactions contribute to the PES. Mathematically, these relative weights are dictated by the standard deviations of the probabilistic Gaussian

process.[2] In the QUIP code, this is the `delta` parameter, which has to be set for each descriptor. Our first generations were created using `delta=0.1` for the two-body distance descriptor and `delta=1.0` for the SOAP descriptor.

# 4 Optimizing the Cu – O GAP

Our construction scheme of a copper oxide GAP can be split into two parts: First we built a structural dataset and trained the GAP with the default hyperparameters described in 3.3, iteratively adding structural data. After accumulating enough data, we optimized a selection of hyperparameters which lead to significant performance improvements.

The following sections are structured as follows: First, we explain our method of accumulating structure data, making sure that we sample as much as possible of the known CuO structural space. Then we present the results of our hyperparameter optimizations. We hope that this analysis will help accelerate future GAP creations, as we found multiple pathways to lighten the computational load, without compromising the GAP accuracy.

## 4.1 Generating the Structure Dataset

The data with which we trained the first iteration, or generation 0, of the Cu – O-GAP, consisted of DFT SCF calculations on eight bulk Copper Oxide structures from Materials Project.[58] To increase the number of local environments for generation 0, we used supercells of 60 to 80 atoms for each polymorph. For each structure, we applied lattice expansions and compressions by 10 %, as well as random atomic displacements with a standard deviation of 0.01 Å.[1] Additionally, SCF calculations were performed on single O and Cu atoms in 10 Å × 10 Å × 10 Å cells to give the GAP a reference energy for the isolated atom.

We created the next 15 generations by adding new structures to the training data, without changing the hyperparameters. To test the accuracy of generation $n$, we run MD calculations on a CuO structure, this structure does not have to be included in the training set of generation $n$. The general logic of this iterative GAP training process is illustrated in Figure 4.1.

[1] All the alterations to atomic structures were made using the Atomic Simulation Environment.[59]

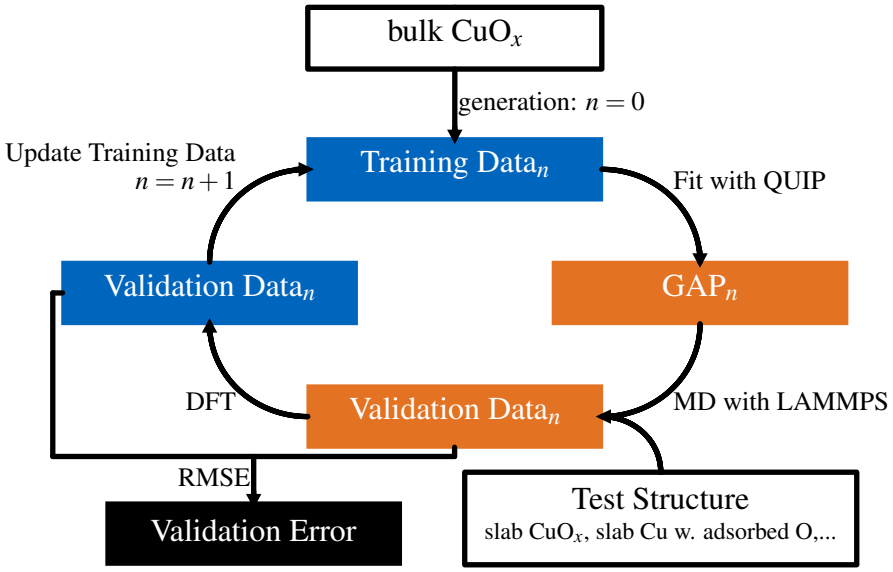



**Figure 4.1** Overview of the iterative GAP generating process. Starting with distorted bulk Copper Oxides, we train the 0th GAP generation. By generating structures unknown to the GAP via MD, we gain an accuracy measure by comparing the predicted energy and forces to the DFT-calculated values. We then include these values into the next generation.

The validation data of generation $n$ consists of 10-20 sample structures from the resulting MD trajectory. Comparing the results of *ab initio* single point calculations on the validation data with the results by generation $n$ GAP gives us the validation error generation $n$. GAP generation $n+1$ is then trained on the training and validation data of generation $n$. Training and validation errors for generations 0 to 15 are shown in Figure 4.2.

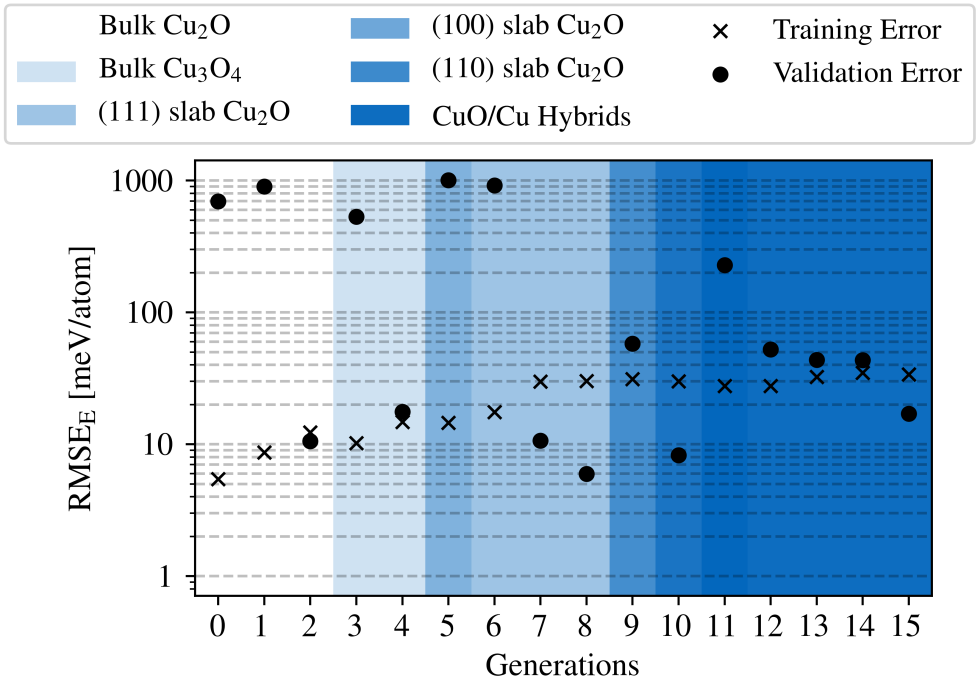



**Figure 4.2** Root mean square error (RMSE) of the atomic energy for structures in the training and validation data for each generation up to Gen. 15. The shading background illustrates the structure type used to create the validation set of generation *n*.

The shading of Figure 4.2 refers to the structure type used for the MD calculation. As soon as the validation error converges to the training error, we change the structure type for the next generation's validation data. For generations 0 to 4 the MD was performed on bulk structures included in the training data. For the first structure type we used, three GAP generations were required until the validation errors matched the training error.
From generation 5 onward the MD was generated with slab structures, in order to slowly introduce surface effects to the GAP. This was done step wise, starting with the (111) surface of 4 layers cubic $Cu_2O$. Once the MD calculations converged and returned reasonable energies and forces, we moved on to (100) and (110) surfaces. To speed up the learning process we also added single-point calculated distorted slab structures to the training data in generation 9. This led to generations 9 and 10 converging immediately.
From generation 11 onwards the GAP was trained with oxidized copper surfaces (labeled "CuO/Cu Hybrids" in the legend of figure 4.2). We begin introducing bulk copper structures in generation 11 by running the MD on a slab of two layers $Cu_2O$ atop a layer of bulk Cu. The atomic positions of the Cu layer were fixed throughout the MD calculation. In subsequent generations we increase the number of Cu layers, while simultaneously decreasing the number of fixed Cu layers. This concludes in generation 15 where the MDs are run on structures with three Cu layers, only the lowest of which was fixed.

### Cu – Cu interactions

Another method of visualizing the impact of adding training data to the GAP is by investigating the Lennard-Jones plots of the different bond types in the GAP: Cu – Cu, O – O, Cu – O.

To generate the Lennard-Jones potential of the Cu – Cu bond, we expand a cubic fcc copper cell. This energy vs bond distance is closely related to a Birch-Murnaghan type equation of state,[60] which we use to test the the quality of the GAP potential in regions far outside of what can be achieved with typical MD runs. The energy shown in in Figure 4.3 is normed to the isolated atoms, the bond distance refers to the minimum bond distance between two atoms in the periodic cell.

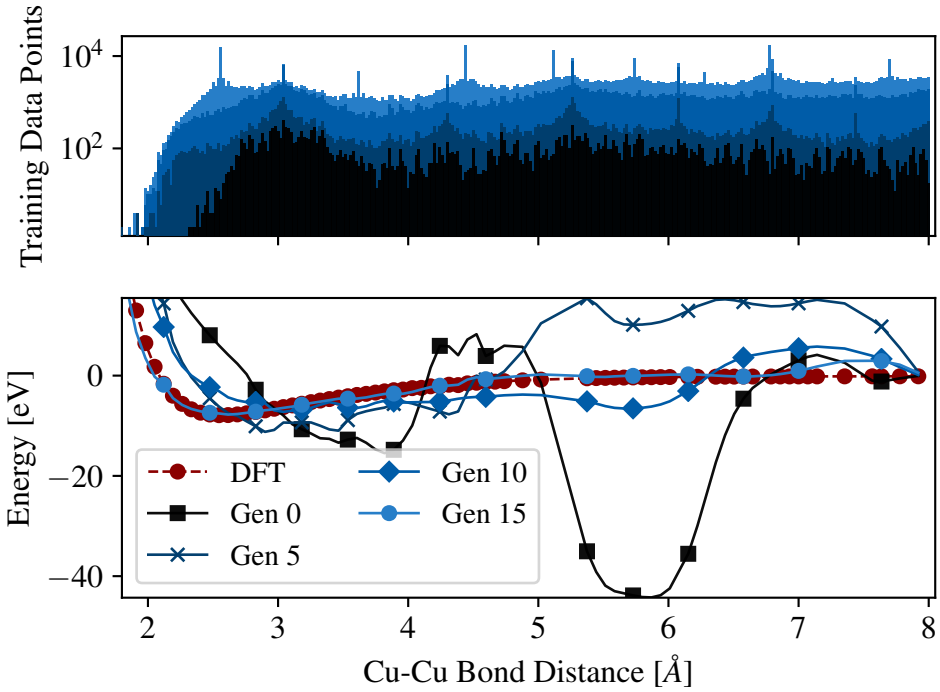



**Figure 4.3** Top panel: Histogram of the training data at that specific bond length. Bottom panel: The Cu – Cu bonding energy predicted by different generations of the Cu – O-GAP. (Obtained by isotropic rescaling of the conventional fcc bulk Cu cell.)

The top panel of Figure 4.3 shows that our method of adding new data elements is successful in increasing the sampling in regions not previously covered, specifically in regions below 2.4 Å. Although the maxima and minima in the training data histogram are at the same bond distances for generation 10 as for generation 15, the differences between minima and maxima are less extreme. This increased sampling leads to the GAP-predicted energy being progressively in agreement with the DFT data. For generation 15, we achieve an energy minimum at a Cu – Cu bond distance of 2.62 Å, roughly equaling the DFT bond

minima of 2.55 Å.
However, Figure 4.3 also indicates a problem with our chosen parameters. In early generations we observe sizeable fluctuations at large bond distances, although physical knowledge, as well as DFT data suggests that there are virtually no interactions at bond distances above 4 Å. Even in late generations the GAP predicts repulsive interactions at bond distances between 7 and 8 Å. In fact, generation 15 proves to be more of an outlier, as can be seen in Figure 4.4. For the 16th generation mentioned in Figure 4.4, we added pure copper slabs, as well as copper slabs with adsorbed oxygen atoms on the surface. We observe that both generations are in agreement with DFT data for bond distances up to 3 Å.

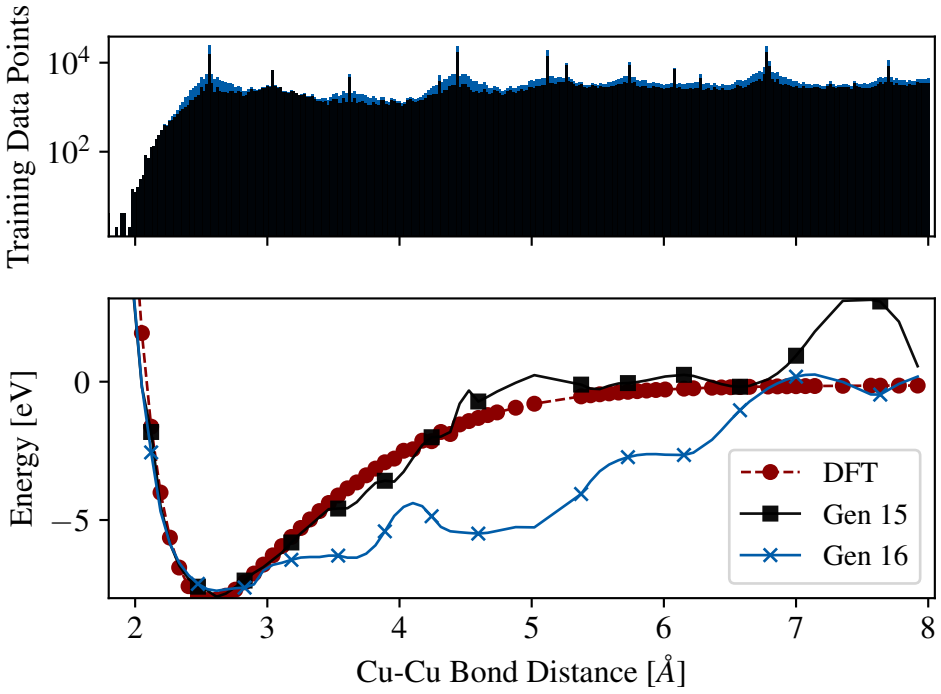



**Figure 4.4** Comparison of GAP generation 15 and 16. Generation 16 has additional copper slabs in the training data. A similar plot with the final GAP can be found in Figure 4.10.

As we will show below, decreasing the descriptor cutoff radius during training leads to a reduction of the predictive error. We believe this is due to the omission of such erroneous long-distance effects.

### O – O interactions

To improve the O – O bond modeling we added $O_2$ dimers to the training data in generation 14. As shown in Figure 4.5, generations 13 and lower all were unable to recreate the attractive interaction at oxygen's gaseous bond distance of 1.1 Å. This is mainly attributed to insufficient sampling at these regions, as is exhibited by the training data histogram of Figure 4.5. The few structures in the region below 2.0 Å are typically structures were

the closeness of two Oxygen atoms causes enormous forces in the structure, causing the structure's total energy to be very large, which in turn leads to the energetic prediction at these bond distances to be very unfavorable. By adding only three dimer structures (below/at/slightly over the "ideal" bond distance of gaseous $O_2$), we were able to significantly improve the energetic structure.

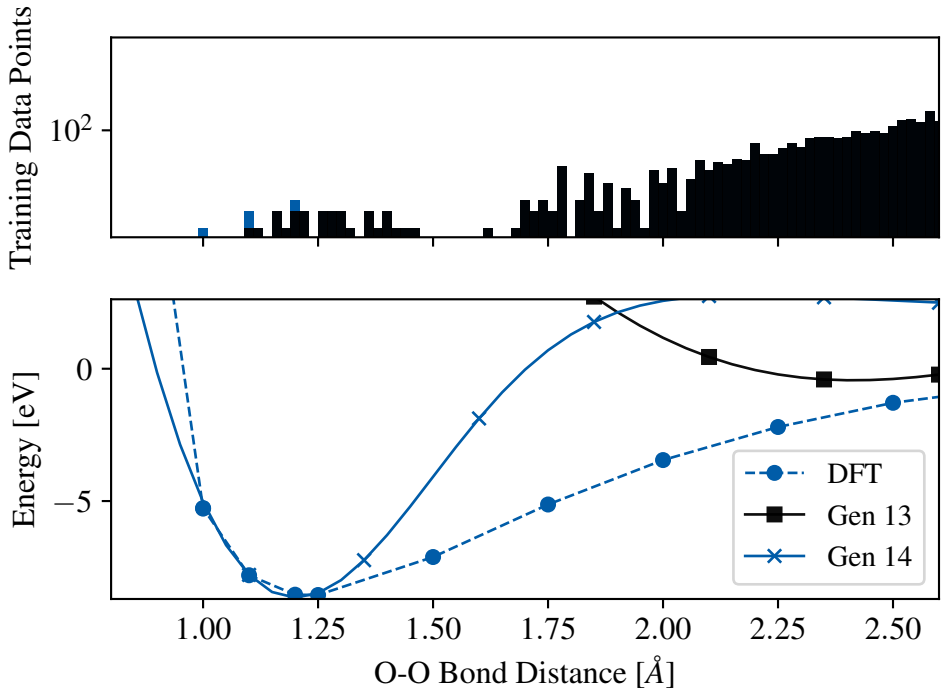



**Figure 4.5** Comparison of gas dimer O-O Lennard Jones plot after adding gas dimers to the training data.

### 4.1.1 GAP Hyperparameter Optimization

Figure 4.2 shows that with the given parameters we are able to model oxidized copper surfaces with an error of approximately 30 meV per atom. In literature, the error is often on the scale of 3-5 meV per atom.[61] To improve the capabilities of the GAP we turn our attention to the hyperparameters.

To optimize the hyperparameters we perform cross-validations on the training data of generation 0, as this is the smallest dataset and therefore is the cheapest to calculate. This is illustrated in Figure 4.6.

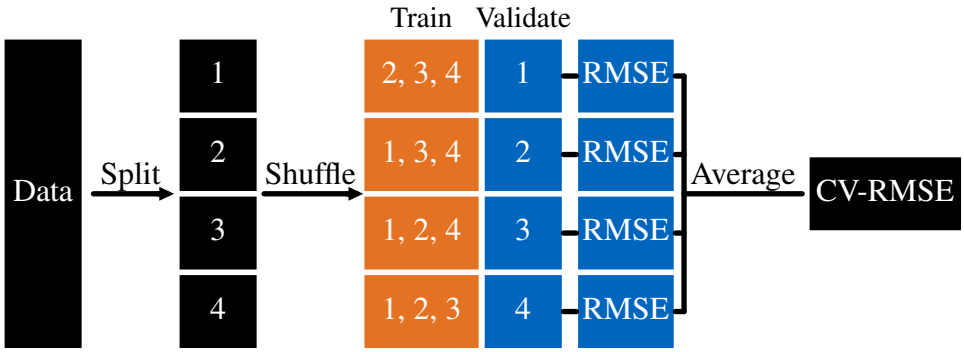



**Figure 4.6** Illustration of our cross-validation process. We divided the training set of generation 0 into four folds. For each hyperparameter setup we train a GAP with a training set of three of the four folds (orange). Our primary success metrics are the root-mean-square error (RMSE) of total energy and atomic forces, which we calculate for the remaining fold (blue). This limits the risk of overfitting. Finally we average the four RMSEs for a given set of hyperparameters to obtain the cross-validation RMSE (CV-RMSE). Depending on the hyperparameters we choose either a random variable search, or a grid search to find the optimal hyperparameters.

### Hyperparameter: `default_sigma`

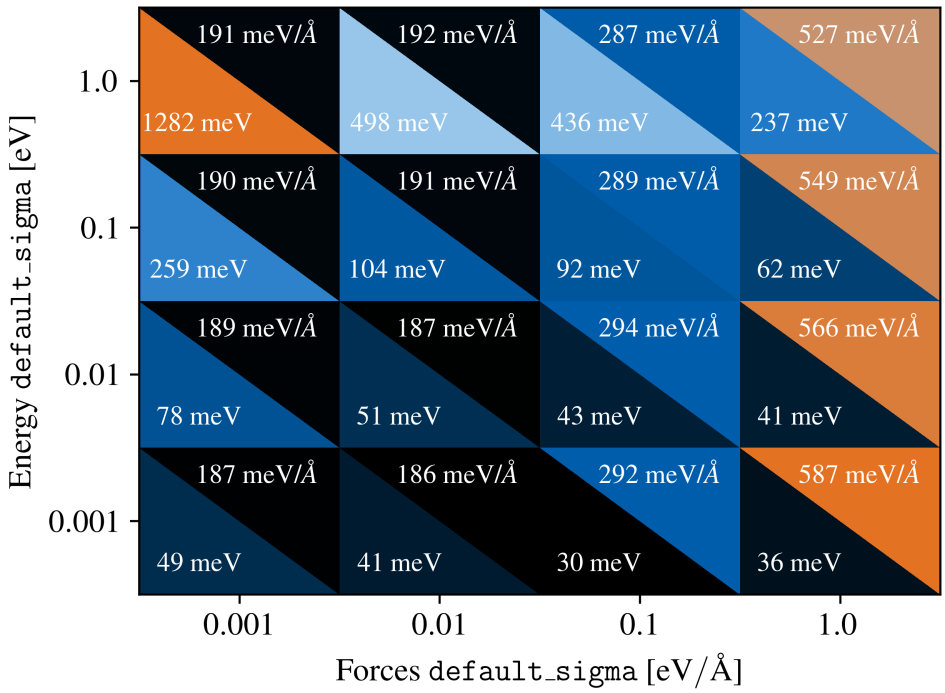



**Figure 4.7** Cross-validation results for the `default_sigma` parameter. The squares represent the CV-RMSE of the atomic energy (bottom triangle), and the CV-RMSE of the atomic forces (upper triangle).

Figure 4.7 shows the results of the grid varied `default_sigma` parameter. We observe that root mean square energy of the energy and atomic forces correlates directly to the respective sigma value. This is expected, as the sigma value can be viewed as a metric quantifying how accurately the GAP tries to match the physical observable in the training data. In the case of $\texttt{default\_sigma}_{\text{energy}} = 0.001\,\text{eV}$ and $\texttt{default\_sigma}_{\text{forces}} = 1.0\,\text{eV}\,\text{Å}^{-1}$ the energy error will therefore be small, whereas the forces error will be large. As we are interested in minimizing both, we find an optimal setup at $\texttt{default\_sigma}_{\text{energy}} = 0.001\,\text{eV}$ and $\texttt{default\_sigma}_{\text{forces}} = 0.01\,\text{eV}\,\text{Å}^{-1}$. By introducing this change we improve our energetic error by 8 meV per atom while keeping the force error constant.

### Hyperparameter: `delta`

The next hyperparameter we consider is the relative weighting of the descriptors `delta`. The results of the cross-validation, shown in table 4.1, indicate that setting `delta` to 1.0 for both the two-body pairwise distance descriptor and the SOAP descriptor yields the best results. The improvement is on the scale of 9 meV per atom.

**Table 4.1** Results of the cross-validation on the optimal `delta` parameters for each GAP.

| $\texttt{delta}_{\text{Two-body}}$ | $\texttt{delta}_{\text{SOAP}}$ | CV-RMSE $_{E_{\text{total}}}$ [meV per atom] | CV-RMSE $_{f_{\text{atomic}}}$ [meV Å$^{-1}$] |
|---|---|---|---|
| 0.1 | 0.1 | 34 | 188 |
| | 1.0 | 26 | 183 |
| 1.0 | 0.1 | 30 | 181 |
| | 1.0 | 17 | 168 |

### Hyperparameter: `r_cut` (Cutoff radius)

As discussed in section 4.1, we assume that the GAP overestimates long range interactions. We therefore perform a cross-validation on the descriptor cutoff radii. In a range of 3.5 to 6.5 Å we randomly assign cutoff radii for each descriptor. We choose a random search on this hyperparameter, believing that the two-body pairwise distance descriptor is potentially less important to the success of the GAP, thus allowing us to sample more values of the SOAP cutoff radius. In total we evaluated 500 different configurations, resulting in Figure 4.8, which shows the energy error as a function of SOAP $r_{\text{cut}}$ and Two-body $r_{\text{cut}}$.

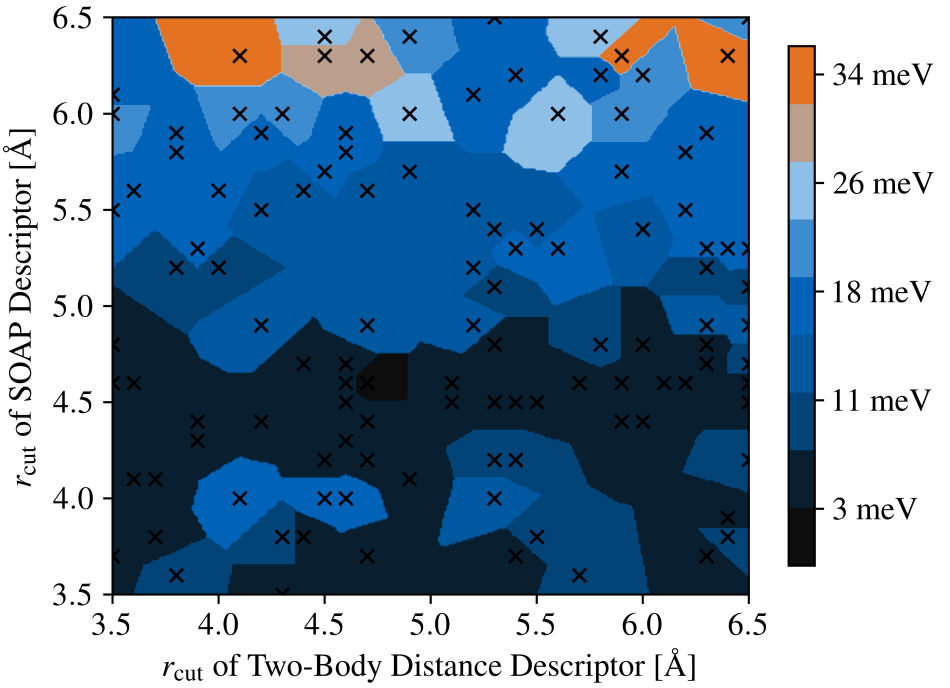



**Figure 4.8** Cross-validation RMSE for the atomic energy as a function of cutoff radii. The markers represent the values of the random parameter search. Note, we use equivalent cutoffs for all atom types.

Evidently, the energy error strongly correlates with the cutoff radius of the SOAP descriptor, while the two-body distance descriptor has little impact. We observe that increasing the cutoff above 5 Å causes the CV-RMSE to increase by over 20 meV, possibly due to the SOAP descriptor including atoms in the second shell of the central atom, and the algorithm not being able to natively lead to an appropriate separation into short and long range interactions.

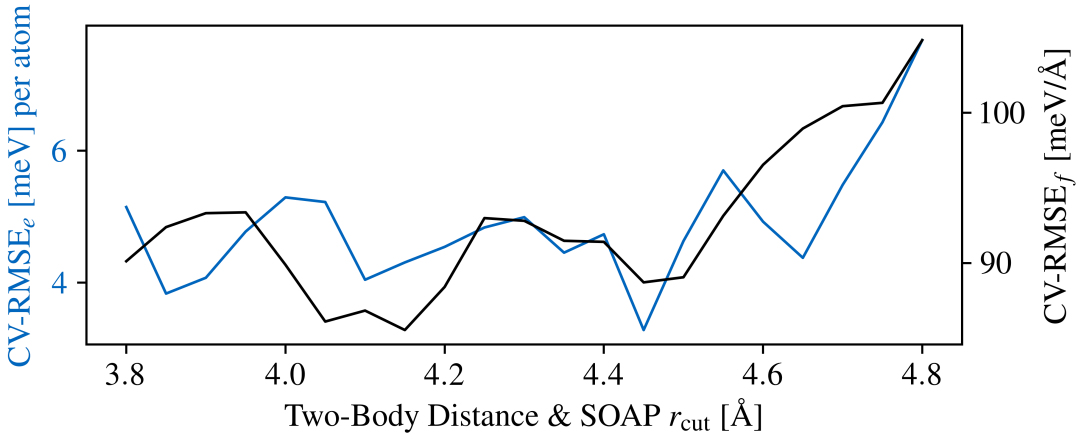



**Figure 4.9** CV-RMSE for the atomic energy (left $y$-axis) and forces (right $y$-axis) as a function of $r_{cut}$.

As a result the in principle more general cutoffs of ≈6 Å seem to lead to worse cross-validation errors than the less flexible setups with similar cutoffs. The reduction of $r_{\text{cut}}$ from 6 Å to a cutoff of 4.2 Å leads to an error reduction by 80 %. If we wish to keep the cutoff radii consistent for each descriptor, we observe the ideal cutoff at 4.2 Å (Figure 4.9).

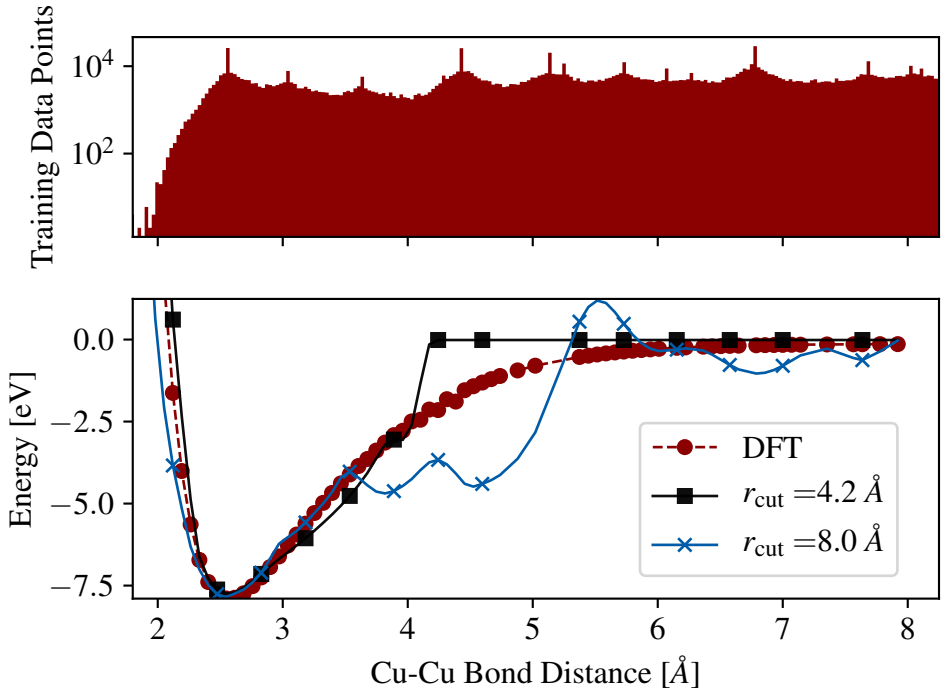



**Figure 4.10** Atomic energy as a function of Cu – Cu bond distance for two GAPS with identical training data and hyperparameter setups save for the descriptor cutoff radii.

As shown in Figure 4.10, reducing the cutoffs to 4.2 Å resolves the unphysical energy oscillations at higher bond distances previously observed in Figure 4.4. However, as a result of the smaller cutoff, the energy variations for bond distances between 4.2 and 6 Å that are still significant in the DFT calculations can not be described with respective model. The GAP energy immediately goes to 0 eV above its cutoff.

Figure 4.11 compares the performance of different GAPs on a MD trajectory of a Cu (111) slab with a surface termination of $Cu_{16}O_{10}$. The GAPs were all trained on the training data of generation 16.

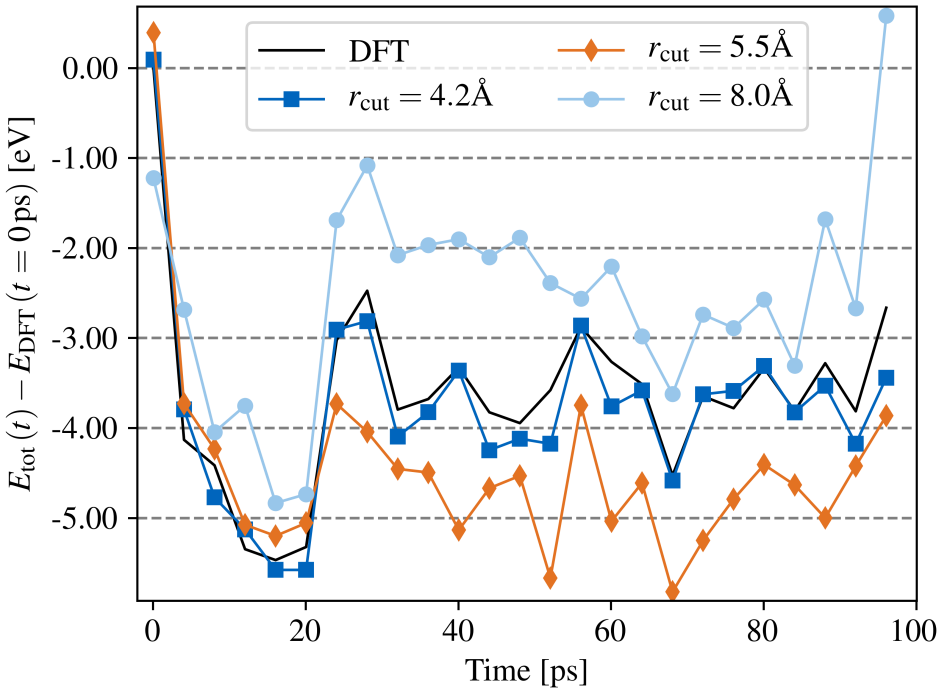



**Figure 4.11** Comparison of different GAPs to a MD trajectory of a $4 \times 4$ slab of Cu with 10 adsorbed O. (note: This system consists of 64 Cu and 10 O atoms.)

Evidently our observation from the cross-validation on the 0th generation structural dataset prevails: While all models in Figure 4.11 seem to qualitaviley recover the PES, only the small cutoff of 4.2 Å can lead to a highly accurate quantitative description.

### Hyperparameter: `n_sparse`

During the GAP fitting process the covariance matrix is sparsified to reduce computational costs. A sparse point can be understood as the replacement of $n$ structures in a single average.[2] For instance, if the training set consisted of 10 diatomic molecules with bond distances of 1.1, 1.2 ... 2.0 and the `n_sparse` parameter of the two-body descriptor is set to 1. The energetic values of all the structures would be averaged into this one point. We therefore assume that an increase of `n_sparse` increases the nuances of the covariance matrix, reducing the validation error, however at additional computational cost during the fitting process.

Varying the sparse points for the training data in generation 0 did not result in any substantial changes in GAP accuracy. Figure 4.12 shows the results of a grid search crossvalidation on the data of generation 16.

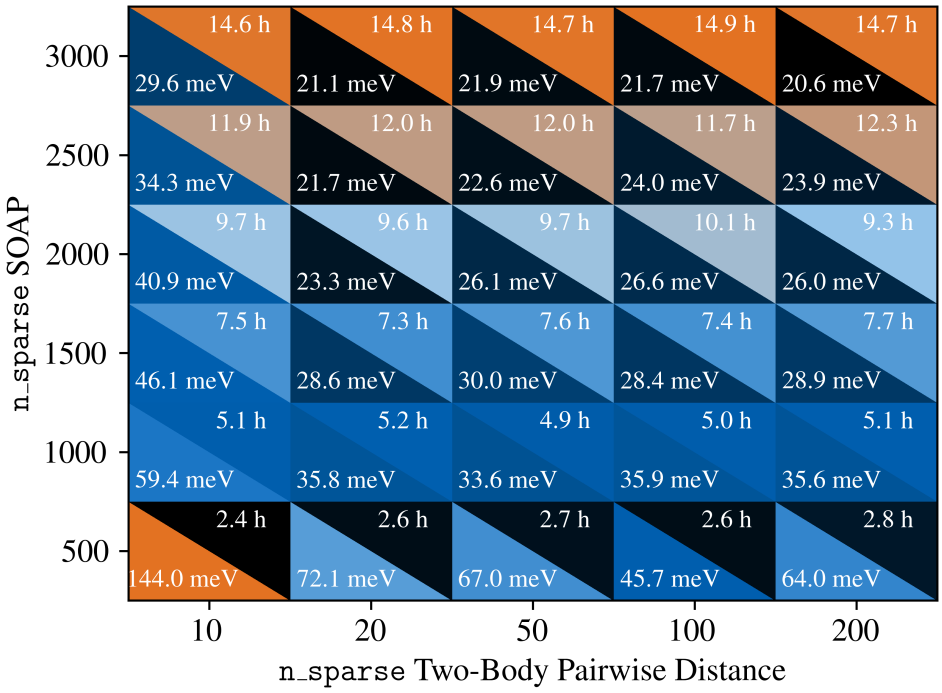


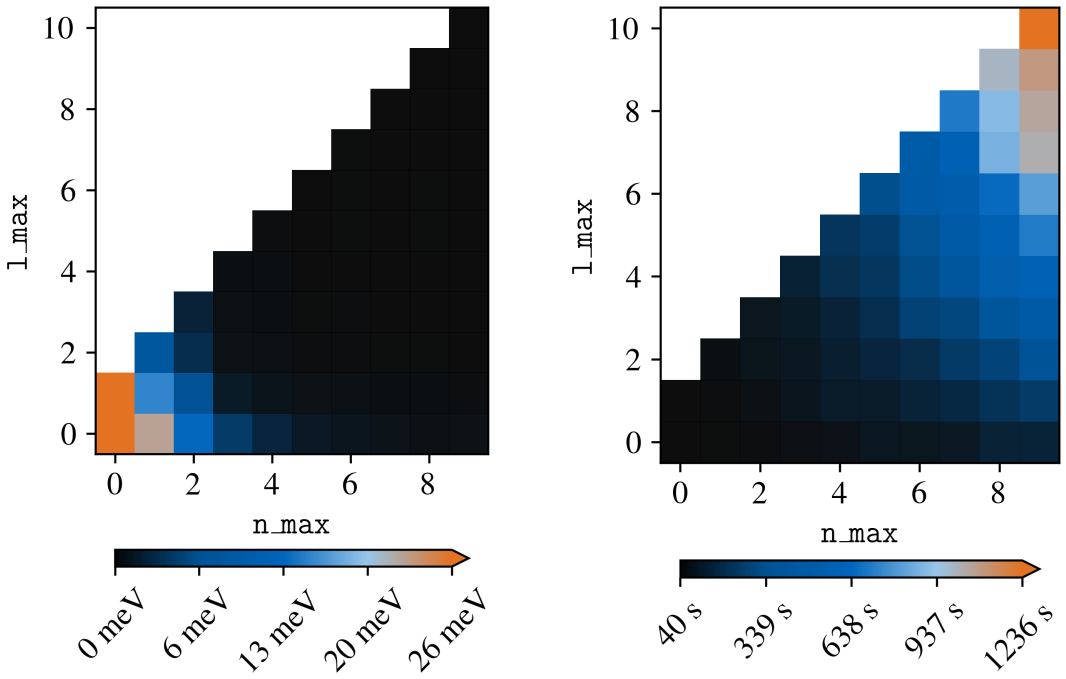

**Figure 4.12** Atomic Energy (lower left triangles) CV-RMSE and average GAP fitting runtime (upper right triangles) depending on the `n_sparse` for each descriptor. The CV-RMSE of atomic forces shows almost identical behaviour to the atomic energy.

At the beginning of the GAP fitting process we chose default values of `n_sparse`(Two-Body) = 20 and `n_sparse`(SOAP) = 1000. This hyperparameter setup applied to the structural dataset in Figure 4.12 in an atomic energy CV-RMSE of 35.8 meV per atom. On average the GAP fitting process took five hours. Similar to the variation of the `r_cut` hyperparameter, we observe that altering the hyperparameter for the SOAP contribution has a larger impact than for the two-body contribution, with the CV-RMSE mainly unaffected by increasing the number of two-body sparse points. At variance, doubling the number of SOAP sparse points can lead to an error reduction by approximately 12 meV per atom. However, an approximate doubling of the runtime is observed.

### Hyperparameter: `n_max, l_max` (SOAP Dimensionality)

We performed a grid search on the maximum number of radial `n_max` and angular `l_max` basis functions of the SOAP vectors. During the fitting process, SOAP vectors are calculated for each atom. For the 0th generation our chosen hyperparameters were `n_max` = 10 and `l_max` = 3. For systems with two atomic species, this results in SOAP vectors of 840 dimensions. The goal of this cross-validation was to see if there is a benefit to reducing the dimensionality of these vectors.

To analyze the energetic performance of lower dimensional SOAP vectors, we iterated the GAP fitting process through each possible combination of parameters up to `n_max`=10. The errors of the training data are shown in Figure 4.13.




**(a)** Average atomic energy training RMSE.

**(b)** Average GAP fitting runtime.

**Figure 4.13** Partial results of the cross-validation on `n_max` and `l_max`.

Unsurprisingly, the average training error gets smaller with increasing SOAP dimensionality. The minimum being 0.39 meV per atom at the (10, 10) SOAP vector, vectors with 2300 dimensions. These also have the largest computational cost, with a GAP-fit taking on average 1237 seconds. We see similar results for SOAP vectors above (3,2), at greatly reduced computational cost (average GAP fitting runtime: 88 seconds). This is potentially very valuable information, as a reduced GAP fitting runtime removes a major bottleneck in a thorough cross-validation of other hyperparameters.

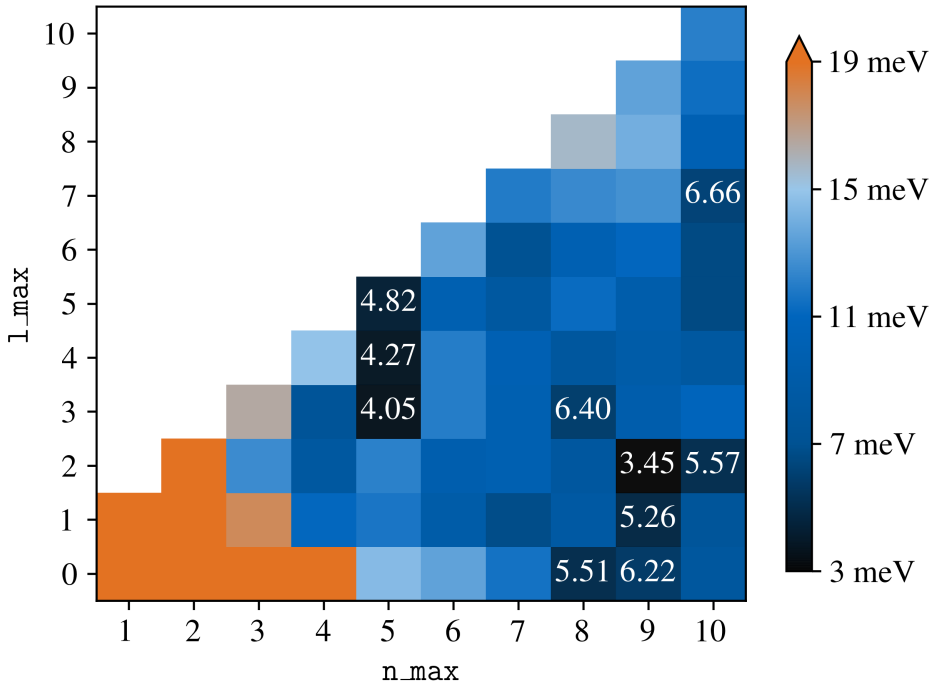



**Figure 4.14** CV-RMSE of atomic energy with varying number of radial basis functions $n$ and angular basis functions $l$. The data is acquired using the training data of generation 0.

If we look at the atomic energy CV-RMSE in Figure 4.14, we find no clear correlation between `n_max`, and `l_max` and the CV-RMSE. The minimum is at lower dimensional SOAP vectors, particularly the values at `n_max`=5 stand out for routinely returning good results. For example, the GAP fit on a (5, 3) SOAP vector has a CV-RMSE of 4 meV per atom, while the (10, 10) SOAP vector has a CV-RMSE above 12 meV.
Apart from overfitting, one reason why we observe better energetic results for less precise vectors could be that the training data in the 0th generation is too homogeneous. The training set consists of different bulk copper oxide structures and does not include copper structures, or surface/vacuum interfaces.

**Table 4.2** Different SOAP vector setups for GAPs trained on the structure dataset of generation 16. The RMSE not in parentheses is the validation RMSE of different surfaces, the value in parentheses is the RMSE of the training data.

| GAP Parameters | RMSE $E_{\text{total}}$ [meVper atom] | RMSE $f_{\text{atomic}}$ [meV Å$^{-1}$] |
|---|---|---|
| $r_{\text{cut}} = 4.2$ Å, $n_{\text{max}} = 10$, $l_{\text{max}} = 3$ | 3.4 (2.7) | 75.5 (88.7) |
| $r_{\text{cut}} = 4.2$ Å, $n_{\text{max}} = 5$, $l_{\text{max}} = 3$ | 5.0 (3.3) | 88.7 (111.4) |
| $r_{\text{cut}} = 8.0$ Å, $n_{\text{max}} = 10$, $l_{\text{max}} = 3$[a)] | 15.8 (4.1) | 124.3 (133.9) |
| $r_{\text{cut}} = 8.0$ Å, $n_{\text{max}} = 5$, $l_{\text{max}} = 3$ | 27.0 (23.5) | 217.8 (210.2) |

a) Original set of parameters

Comparing GAPs trained on the data of generation 16, one with the previous (10, 3) SOAP vectors and one with (5, 3) SOAP vectors, we find that, judging by the RMSE values, the (10, 3) GAP outperforms the (5, 3) GAP by 1.6 meV per atom.
However, a more significant difference for the two hyperparameter sets is observed for the prediction of the $O_2$ bonding curve. As shown in Figure 4.15, while the (10, 3) vector stays close to the DFT values, the (5, 3) vectors completely change the shape of the curve and introduce a minimum at 1.75 Å.

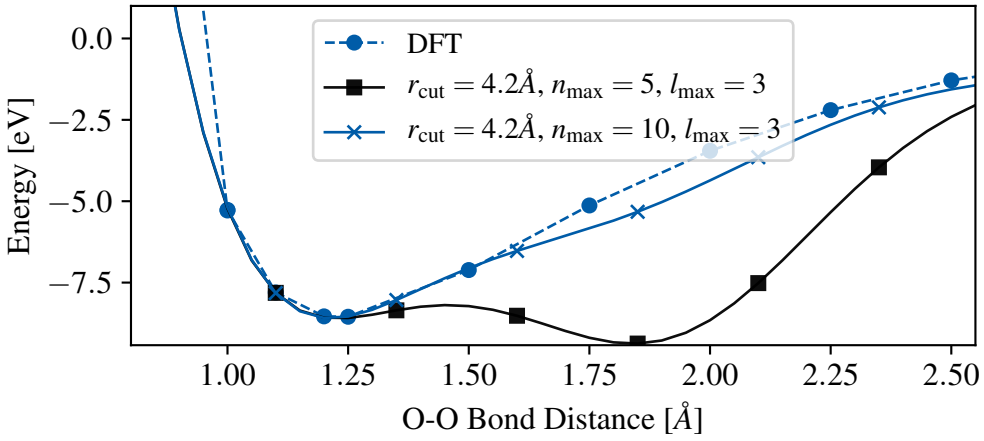



**Figure 4.15** Impact of different SOAP dimensionality on the prediction of $O_2$-dimers for generation 16.

### Final hyperparameter choice

Our choice for the optimal GAP hyperparameters are listed in table 4.3.

**Table 4.3** Final GAP Hyperparameter choice.

| | Hyperparameter | Input |
|---|---|---|
| | `default_sigma` | $\{0.001\,\text{eV}\ 0.01\,\text{eV}\,\text{Å}^{-1}\}$ |
| Descriptor | Hyperparameter | Input |
| `distance_2b` | `cutoff` | 4.2 Å |
| | `add_species` | True |
| | `n_sparse` | 200 |
| | `sparse_method` | uniform |
| | `covariance_type` | ard_se |
| | `theta_uniform` | 1.0 |
| | `delta` | 1.0 |
| `soap` | `cutoff` | 4.2 Å |
| | `add_species` | True |
| | `n_sparse` | 2000 |
| | `sparse_method` | cur_points |
| | `covariance_type` | dot_product |
| | `zeta` | 2.0 |
| | `n_max` | 10 |
| | `l_max` | 3 |
| | `atom_sigma` | 0.5 |
| | `delta` | 1.0 |

# 5 Analysis of Cu (111) Surface Morphologies

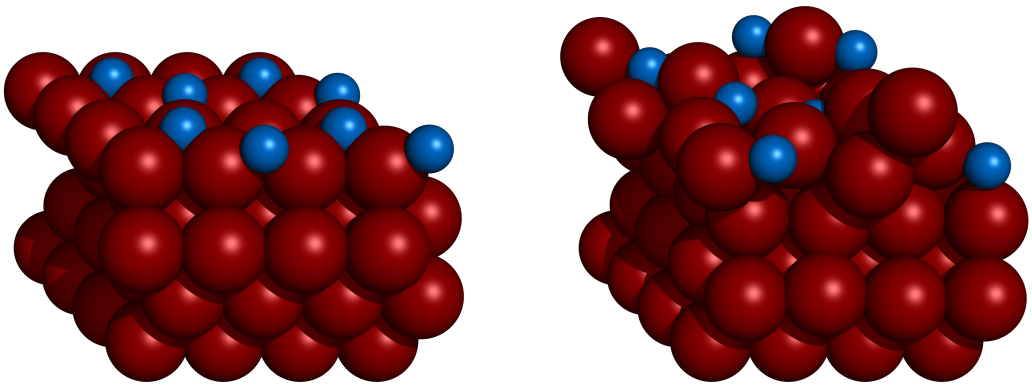
In the following chapter we will use the finalized Cu – O GAP from chapter 4 to study partially oxidized Cu (111) surfaces. More specifically we investigate the surface structure of Cu (111) at different stages of oxidation. Typically the correct representation of such an oxidation process requires grand-canonical sampling. This means investigating slabs with varying amounts of Cu and O with sophisticated sampling methods such as Monte-Carlo simulations. Such an investigation would be beyond the scope of this Master's Thesis, which is why we present this simplified analysis for a rough understanding of the processes at the oxidized Cu (111) surface. Instead of sampling the Cu and O degrees of freedom, we run canonical molecular dynamics (MD) calculations with varying O content, representing different stages of the oxidation of the first layer. Varying Cu densities that have been observed in experiments are tested by introducing a custom-made "low-density" Cu (LD-Cu) (111) surface. A natural consequence of the extrusion of Cu by O incorporation in the surface and subsurface layers is a reduced Cu density in the surface region. Several experiments indicate that the Cu density of the oxidized surface corresponds approximately to the density of $Cu_2O$.[1] Therefore we reverse-engineered the oxidation, namely starting with a low density $Cu_2O_x$-like surface termination, where the Cu density is $3/4$ the normal density.

We observe that already a very limited amount of oxygen adsorbed at the surface can lead to complex morphological changes e.g. the partial embedding of oxygen and extrusion of copper from the topmost layer. This added level of complexity renders concepts such as surface coverage ambiguous and requires a more exact definition of surface atoms, and the surface itself, which we discuss in section 5.1. We subsequently use this definition to analyze surface corrugation as a function of O content.

Also we present methods of utilizing the "Smooth Overlap of Atomic Positions" (SOAP) descriptor to distinguish local surface atomic environments. This entails an approach with kernel principal component analysis (kernel-PCA, section 5.3) method, where we look at the differences between surfaces at different stages of oxidation. In addition, we compare the similarity of MD-generated oxidized Cu (111) surface structures with those of low-index terminations of simple copper oxides.

## 5.1 Surface and Surface Coverage

Classically, the surface coverage is defined by the number of adsorbates normed either by surface area, or by the number of surface atoms.[62] For instance, a $4 \times 4$ cell with 4 adsorbed oxygen atoms is considered to have an oxygen coverage of $\Theta_O = 25\,\%$ monolayer (ML). On surfaces with virtually no reconstruction this definition makes sense, as the adsorbate sites are well defined and each oxygen atom is easily and equally accessible, sitting more or less above the substrate.
However, in the case that the surface undergoes structural changes severe enough that the periodicity of the metal at the surface becomes unrecognizable, this definition is no longer accurate. In Figure 5.1 we see a Cu (111) slab with oxygen atoms sitting on the threefold hollow adsorption sites ($\Theta_O = 50\,\%$), which are favored for low oxygen coverages. Starting an MD run from such a high-symmetry oxygen configuration leads quickly to a destruction of the planar Cu (111) surface (cf. the structure reported in Figure 5.1b is obtained after $t = 100\,\text{ps}$ at 300 K).[1] While a certain number of Cu atoms are pushed out of the surface, several oxygen atoms move to subsurface sites.
Clearly our initial definition of oxygen coverage of 50 % becomes problematic, as not all oxygen atoms sit on the surface and are thus not accessible e.g. for surface reactions.



**(a)** $Cu_{16}O_8$ at $t = 0\,\text{ps}$. **(b)** $Cu_{16}O_8$ at $t = 100\,\text{ps}$

**Figure 5.1** $Cu_{16}O_8$ surface at the beginning (**a**) and end (**b**) of MD run at 300 K. In (**b**) we observe that a few oxygen atoms (blue) have moved subsurface and that some copper atoms (red) have formed adlayers on top of the surface.

As a result, we resort to a structure-specific definition of the surface and thus define the

[1] The nomenclature of all partially oxidized slab structures is as follows: $Cu_xO_y$, where $x$ refers to the number of Cu atoms in the top layer at the begin of the MD run, and $y$ refers to the number of O atoms in the structure.

"surface accessible coverage" $\theta_i$ of an atomic species $i$. The general ideas are borrowed from the definition of interfaces in implicit solvent models and most of the framework python routines were provided by Nicolas Hörmann.[63]

### 5.1.1 Definition of surface, surface atoms and surface accessible coverage

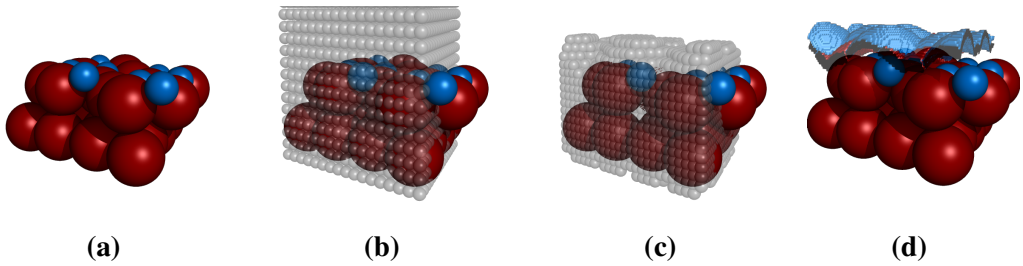
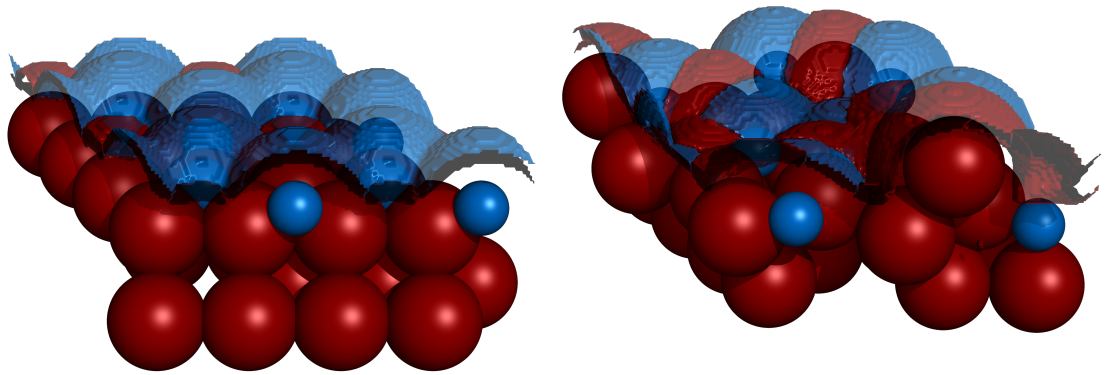



**Figure 5.2** Schematic overview of the method behind our surface coverage definition. (a) shows the top half of a typical Cu (111) slab. In (b), the smaller gray semitransparent spheres represent the grid on which we evaluate the surface. The volume of occupied space is created by summing 3D Gaussian error functions that are positioned at the atomic sites (c). Finally, the magnitude of the gradient of the occupied space is taken in (d) to obtain the surface.

In order to quantify the morphology changes of the oxidized copper surfaces we define surface areas occupied by specific atom types.
For this we accumulate three dimensional Gaussian error functions that are centered on the atomic positions in the periodic cell. Via normalization and numerical evaluation of the gradient on a grid, it is possible to define a surface on the 3D grid which is essentially equivalent to the interface definition in implicit solvent models coupled to DFT codes (see Figure 5.2).[63] Having defined this surface on a grid, we can then search for the closest atom and thus relate every numerical surface point with an atom in the computational cell. Knowledge of the atom types thus allows us to unambiguously define the *surface accessible coverage* $\theta_i$ as the number of surface grid points closest to atom type $i$ divided by total number of surface grid points. We chose a uniform radius of 2.0 Å for the size of the error function (approx. size of an atom) and a real space grid with a resolution of 0.1 Å (possible inaccuracy of 5 % w.r.t. to higher grid densities) that allows for an efficient numerical evaluation of the respective quantities. In addition, we removed atoms that are definitely not part of the surface (sorting the atoms in the lattice by $z$-coordinate and removing the bottom half) prior to the detailed analysis to lower the computational cost.
We acknowledge that the detailed results of this analysis depend on the chosen radius, and the grid density. However, the qualitative trends remain unaffected by our choice of parameters, as do the determined surface atoms.

Surface, real space gid, accumulated (normalized) error functions, and finally the normalized gradient representing the numerical surface are plotted in Figure 5.2a, 5.2b, 5.2c, and 5.2d, respectively.
Note that this procedure allows us to determine the relative exposure of each surface atom individually, which is e.g. rather high for atoms that stick out of the surface, rather low for atoms that are embedded in the surface, and vanishing if atoms become subsurface. As a side remark, similar to the coverage and surface definitions, the question whether an atom is subsurface is a question of perspective. However, the present algorithm allows an unambiguous decision (at least for given error function sizes).



**(a)** $Cu_{16}O_8$ at $t = 0\,\text{ps}$. **(b)** $Cu_{16}O_8$ at $t = 100\,\text{ps}$

**Figure 5.3** Example of our surface definition in action. At $t = 0\,\text{ps}$ (**a**) we see 8 distinct isosurface domes attributed to the 8 surface oxygen atoms of $Cu_{16}O_8$ adsorbed on the fcc and hcp sites. At $t = 100\,\text{ps}$ (**b**) the copper adatoms (red) dominate the surface structure, while the oxygen atoms (blue) forced subsurface do not contribute to the isosurface anymore.

In the next sections we will denote the typically used surface coverage definition with a capital $\Theta$, and the surface accessible coverage with $\theta$.

$$\text{"Classical" Surface Coverage:} \quad \Theta_i = \frac{\text{Number of adsorbate atoms}}{\text{Number of surface atoms}}$$

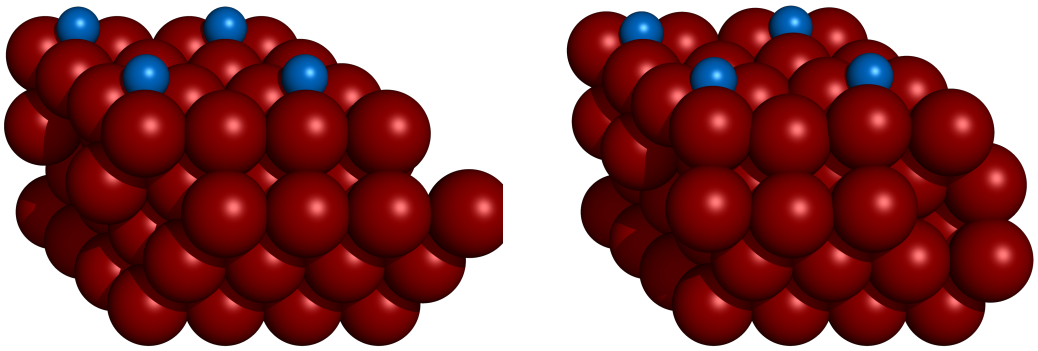
$$\text{Surface Accessible Coverage:} \quad \theta_i = \frac{\text{Surface grid points attributed to atom type } i}{\text{Number of surface grid points}}$$

## 5.2 Surface Corrugation

To investigate the behavior of Cu surfaces with different oxygen coverages, we conduct MD calculations. These run for 100 ps at 300 K with time steps of 1 fs, we store every 50th structure.
We investigate two classes of (111) surfaces: the normal unreconstructed Cu (111) surface with 16 top layer copper atoms in a $4 \times 4$ cell, and a low-density Cu (LD-Cu) (111) surface with 12 top layer copper atoms in a $4 \times 4$ cell.
As explained before, the lower Cu density (12 instead of 16 Cu surface atoms) corresponds nearly identically to the Cu density in bulk Cu oxide. While ideally this "reconstruction" should arise naturally in our simulations it was unclear if its formation might be kinetically limited, which is why we included these structures as well.

### Cu(111) Surfaces

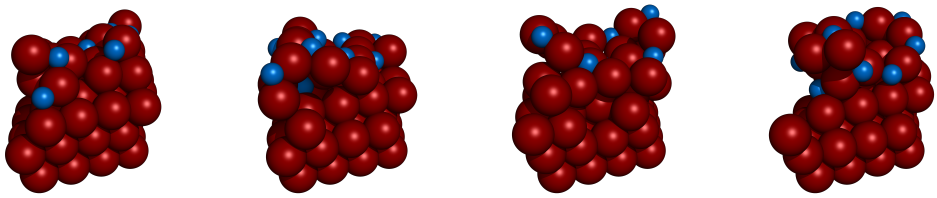
The MD runs of $Cu_{16}O_4$ show that this composition does not undergo any sort of reconstruction. Even during an MD run at 600 K, the only movement apart from atoms rattling is the occasional shift of an adsorbate from the fcc to the hcp site.



**(a)** $Cu_{16}O_4$ at $t = 0$ ps. **(b)** $Cu_{16}O_4$ at $t = 100$ ps

**Figure 5.4** $Cu_{16}O_4$ surface at the beginning (**a**) and end (**b**) of MD run at 300 K. We observe little change throughout the MD run.

Increasing O coverage above 25 % ML immediately leads to surface restructuring. In all MD runs from $Cu_{16}O_5$ to $Cu_{16}O_{11}$ an increasing number of Cu atoms are pushed out of the surface and create islands atop of it. Simultaneously O atoms go subsurface.



**(a)** $Cu_{16}O_8$ **(b)** $Cu_{16}O_9$ **(c)** $Cu_{16}O_{10}$ **(d)** $Cu_{16}O_{11}$

**Figure 5.5** Comparison of different unreconstructed surfaces with high oxygen content at the end of their MD runs. As shown in Figure 5.7, increasing the oxygen content does not correspond to more oxygen atoms at the surface, instead these atoms are found subsurface.

This also implies that the DFT optimized, well-defined fcc and hcp $Cu_{16}O_8$ hexagonal superstructure is not likely to occur, instead transforming to a lower energy structure, as shown in Figure 5.6 (cf. also Figure 5.1).[2] This transformation reduces the O surface accessible coverage $\theta_O$ from close to 100% to only $\approx 55\%$.

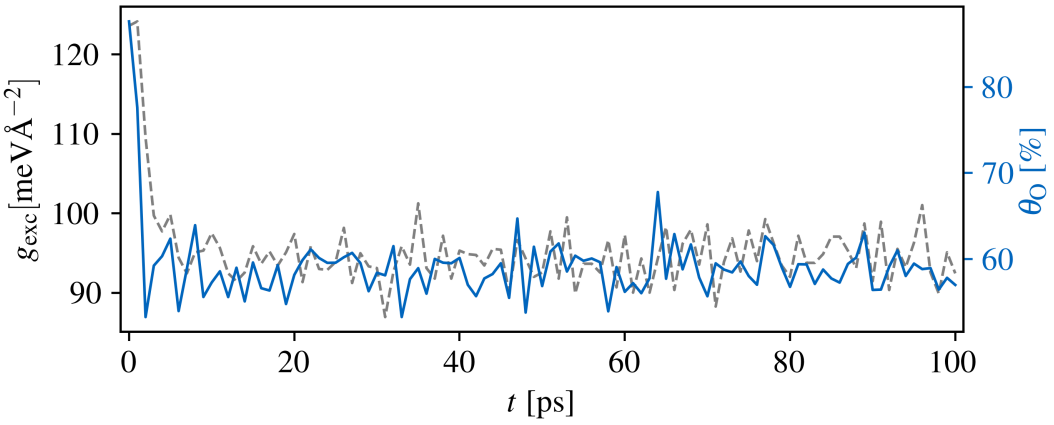



**Figure 5.6** $Cu_{16}O_8$'s excess surface free energy $g_{exc}$ and surface accessible oxygen coverage $\theta_O$ during the course of an MD run at 300 K. We observe an immediate drop of both properties after beginning the MD, consistent with the dramatic surface restructuring as visualized in Figure 5.1.

The observed reconstruction is also reflected in the average O surface accessible coverage $\overline{\theta}_O$ (Figure 5.7 which is consistently between 45 and 65 % for high amounts of oxygen.

[2] The derivation of the excess surface energy $g_{exc}$ is given in the appendix.

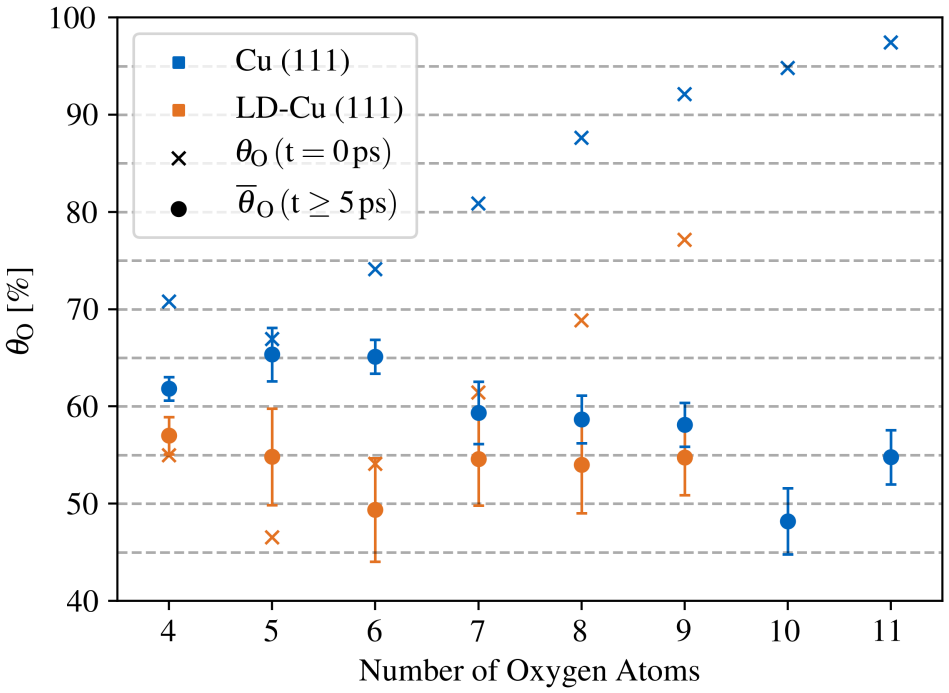



**Figure 5.7** The surface accessible oxygen coverage $\theta_O$ for both Cu (111) and LD-Cu (111) surfaces. We compare the coverage at the beginning of the MD run ($t = 0$ ps) and the average $\theta_O$ after 5 ps. We observe that $\theta_O(t = 0\,\text{ps})$ decreases until $N_O = 6$. For LD-Cu (111) this is due to a switch in O adsorption site from the three-fold hollow site in the top atomic layer (see Figure 5.9a), to sites embedded in the top Cu layer (see Figure 5.9b).

Next, we chose to investigate the movement of surface accessible Cu atoms perpendicular to the surface, leading to a metric of surface Cu deformation $\Delta z$.
For each frame in a MD trajectory, we collect the surface accessible Cu atoms. Only taking these into account, we calculate the difference between the "lowest" (with the smallest $z$-coordinate) and the "highest" (with the largest $z$-coordinate) Cu atom to create $\Delta z$. A small $\Delta z$ value would mean that the surface is virtually uncorrugated, while a large value suggests island formation, vacancy formation, or both. In Figure 5.8 we plot the relative distribution of $\Delta z$ for MD runs of the Cu (111) surfaces.

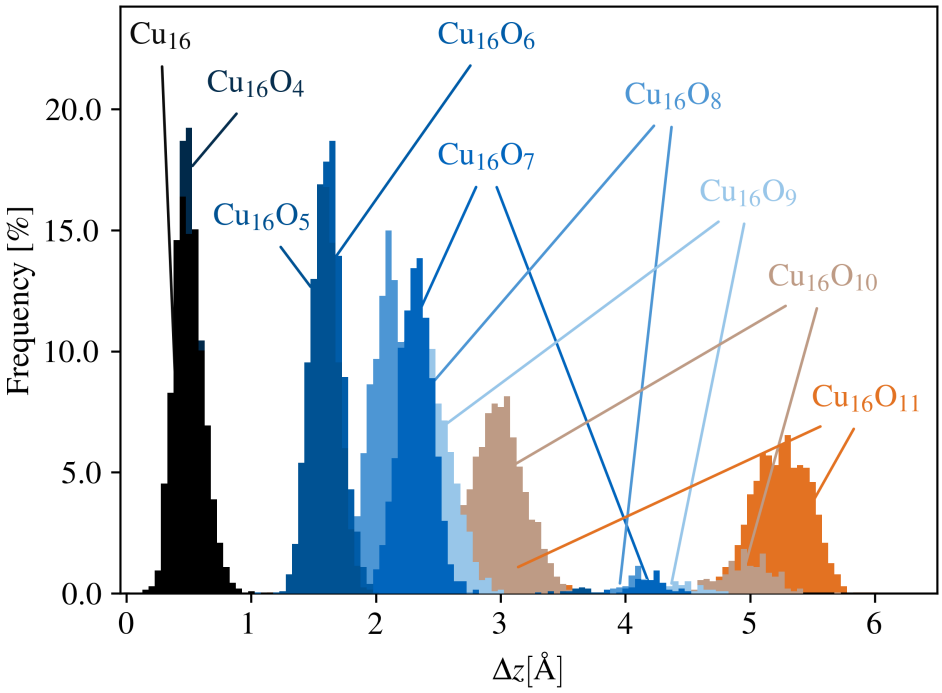
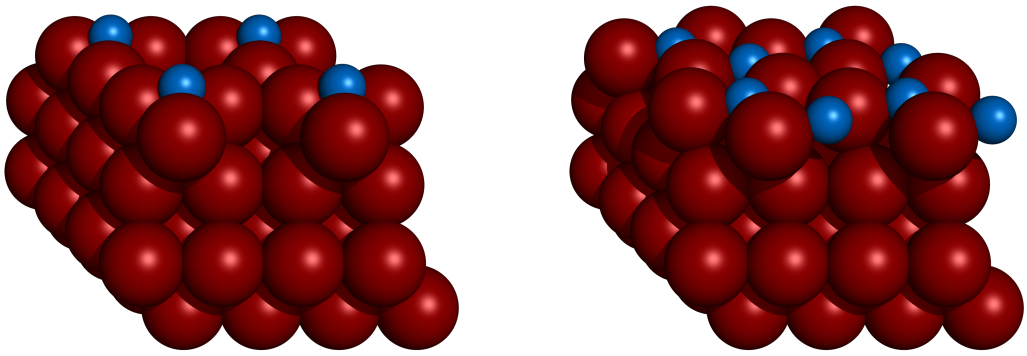



**Figure 5.8** Relative distribution of surface corrugation $\Delta z$ in surface accessible Cu atoms. The data is accumulated from the MD runs of Cu (111) surfaces.

In the cases where we do not observe any major restructuring ($Cu_{16}$ and $Cu_{16}O_4$) Figure 5.8 shows one peak centered around 0.5 Å. This indicates that mostly the atoms vibrate, but that there is no surface corrugation.
Increasing the O coverage up to six oxygen atoms causes the peak in Figure 5.8 to shift to $\Delta z = 1.8$ Å values, indicating first Cu adatoms forming.
Further increases in O coverages lead to double peaks in Figure 5.8. For the respective surface terminations the peak with a smaller $\Delta z$ value is located between 2.0 and 3.0 Å. We interpret this group as structures with one or several adatoms on the original structure's top layer.
The second peak grouping is in the range: $4.0\,\text{Å} \leq \Delta z \geq 5.3\,\text{Å}$. Occuring sporadically throughout the $Cu_{16}O_8$ MD, a majority of $Cu_{16}O_{11}$ frames belong to this grouping. We interpret this peak as stuctures with either more than one adlayer, or structures where surface reconstruction exposes subsurface Cu.

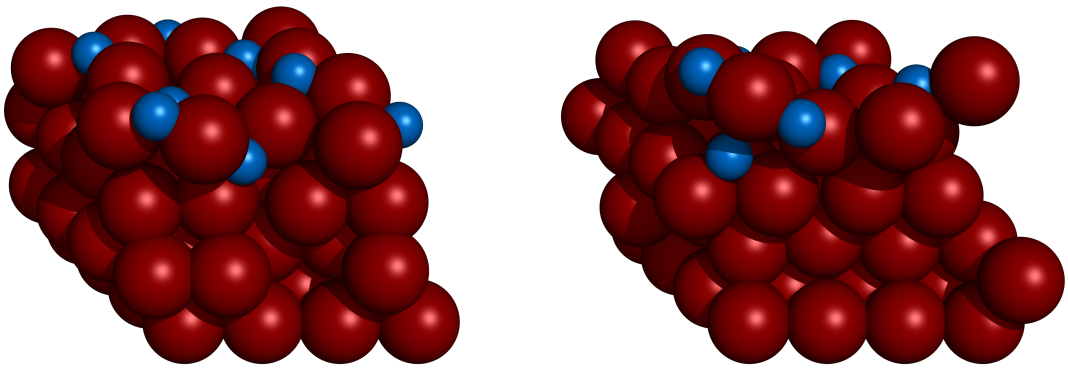
### Low-Density Cu (111) surfaces

All starting configurations of the LD-Cu (111) were derived from $Cu_{12}O_4$ and $Cu_{12}O_8$, two structures that we had previously investigated using DFT.



**(a)** $Cu_{12}O_4$ **(b)** $Cu_{12}O_8$

**Figure 5.9** $Cu_{12}O_4$ and $Cu_{12}O_8$ DFT optimized surfaces. We note the different absorption sites. Whilst in $Cu_{12}O_4$ the oxygen atoms adsorb in the threefold hollow site above the top layer (similar to the oxygen adsorption in Cu (111)), in $Cu_{12}O_8$ O adsorbs between the gaps of the top Cu layer.



**(a)** $Cu_{12}O_8$ **(b)** $Cu_{12}O_9$

**Figure 5.10** $Cu_{12}O_8$ and $Cu_{12}O_9$ structures at the end of their respective MD runs. In $Cu_{12}O_8$ we observe buckling, as well as reorientation of the surface oxygen atoms. In $Cu_{12}O_9$ we see oxygen atoms going completely subsurface, pushing copper atoms above the surface. This is a similar observation as in Figure 5.1b.

We find that the LD-Cu (111) surfaces with coverages between 4 and 8 oxygen atoms behave similarly during their respective MD runs, not undergoing any remarkable structural changes. With $Cu_{12}O_9$ we observe adlayer formation.
The experimentally indicated $Cu_2O$-like termination with a reduced Cu density of 75 % can incorporate up to 8 oxygen in the $4 \times 4$ cell without substantial structural distortions.

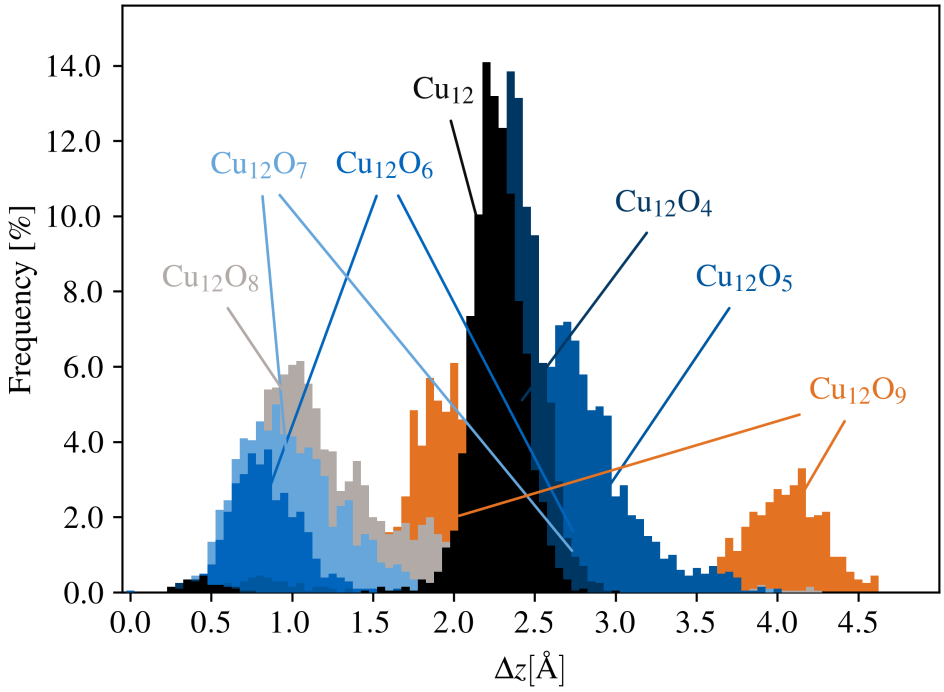



**Figure 5.11** Histogram of the $z$-coordinate difference in surface atoms throughout the MD runs.

In contrast to Figure 5.8, the pristine LD-Cu ($Cu_{12}$) surfaces shown in Figure 5.11 show peaks at 2.1 Å. This is caused by our surface definition, which identifies several Cu atoms from the lower copper layers as surface atoms. It is noteworthy that the as-constructed pristince LD-Cu surfaces undergo a direct reconstruction to form an island with normal Cu – Cu distances as on the (111) Cu surface, and a cluster of vacancies, which exposes the subsurface atoms.
With increasing oxygen coverage, we observe a growing peak around 1.0 Å, which we attribute to oxygen atoms covering the aforementioned subsurface copper atoms.
Finally, $Cu_{12}O_9$ shows a substantial peak at 4.0 Å, indicating adlayer formation in this structure.

## 5.3 Local Surface Environment Analysis with Principal Component Analysis

Our observations in section 5.2 are somewhat vague and not quantitative. In the following two sections we make use of our surface accessible coverage definition and "smooth overlap of atomic positions" (SOAP) local atomic environment descriptors in an attempt to mathematically analyze the morphology changes that the surfaces undergo with varying oxygen content.

### 5.3.1 Kernel Principal Component Analysis (PCA)

We begin with an overview of "Kernel Principal Component Analysis" (PCA).[64] In short, this tool breaks down a set of multidimensional vectors into a set of $n$-dimensional vectors that include the properties which distinguish the vectors as much as possible. Ideally, no information gets lost during this process, e.g. in the case of a set $S$ of vectors that differ in dimension $m$ but are identical for all others, a PCA to one dimension would retain all unique information of $S$. The values would represent the entire variance of dimension $m$. However, in typical applications the dimensionality reduction via PCA results in some information loss.

In our case, we use the sklearn PCA implementation with a linear kernel to reduce the 840-dimensional SOAP descriptors of surface atoms into plottable 2-dimensional PCA vectors, thus simplifying analysis considerably. The first two principle components (PC) retain over 95 % of the variance of the entire data sets. We use these two PCs to investigate how SOAP vectors describe changing chemical environments on Cu surfaces, as well as to better understand the influence of different parameters such as MD temperature and cutoff radius.

One additional note: In the following sections all PCA plots will have an $x$-axis labelled "PC 1" and a $y$-axis labelled "PC 2". These labels refer to the first 2 PC of the PCA conducted on the data for that respective plot and are typically not transferable to other plots.

### 5.3.2 PCA results

We use the surface accessible coverage $\theta_i$ defined in section 5.1.1 to determine the surface Cu atoms for frames of the MD runs mentioned in 5.2. Due to computational limitations we limit the following analysis to the final 100 frames of each MD run. For each identified surface atoms we compute the SOAP vector using the package Dscribe.[65] The parameters of the SOAP vectors are: $r_{\text{cut}} = 4.2\,\text{Å}$, $n_{\text{max}} = 10$, and $l_{\text{max}} = 3$ (for more information see section 3.3).
In total, we collect a dataset of more than 10000 local surface atomic environments, the PCA of which is shown in Figure 5.12.

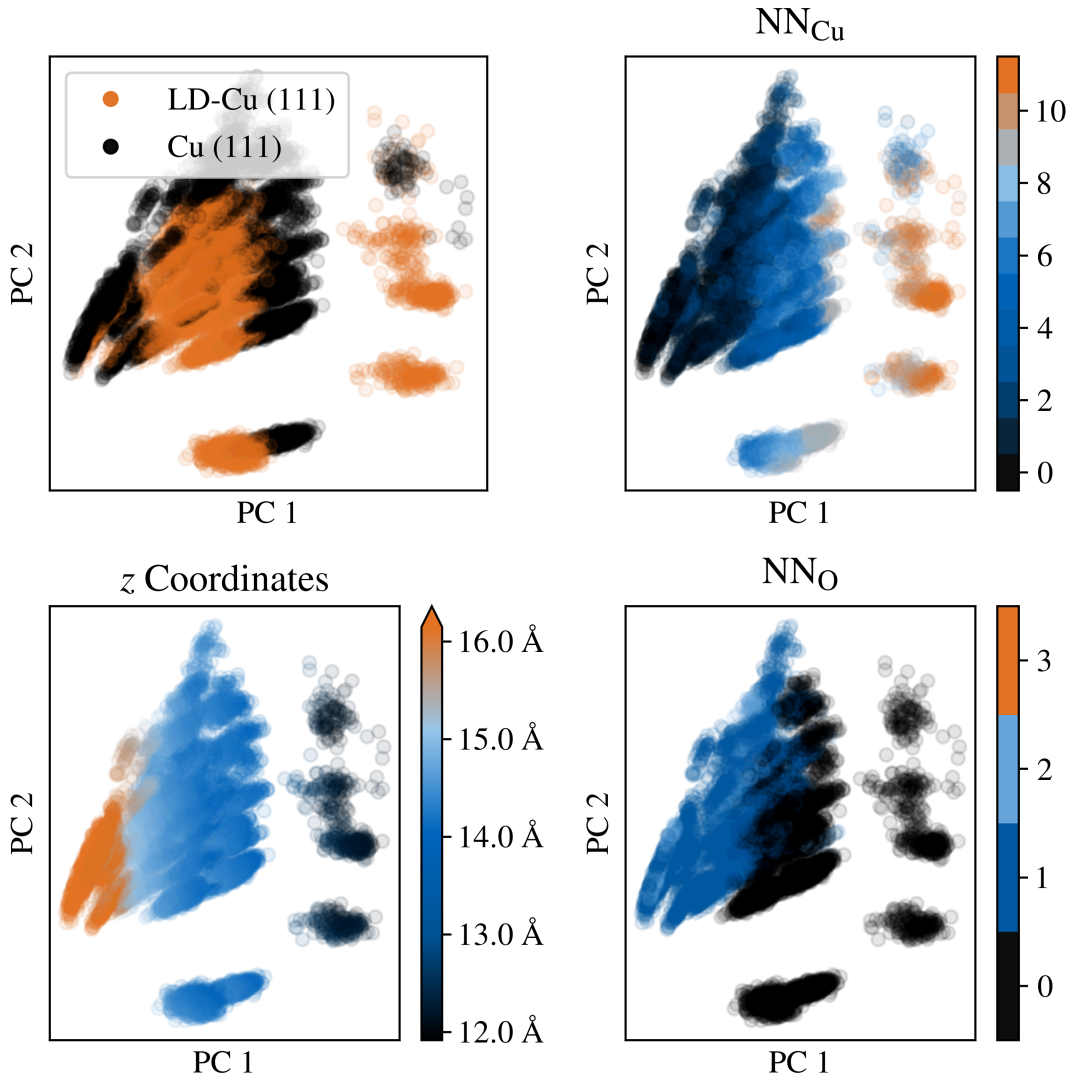



**Figure 5.12** PCA of the last 100 frames of all MD runs. Each data point represents one local Cu surface environment. The plots are color coded by the following parameters: Top left - type of surface, bottom left - the $z$ coordinate of the surface Cu atom, top right - the number of nearest neighbor Cu atoms ($NN_{Cu}$), and bottom right - the number of nearest neighbor O atoms ($NN_O$).

The number of nearest neighbors are determined using a Voronoi algorithm implemented in the VoronoiNN class of pymatgen.[66,67] The required tolerance parameter was set so that neighbor polyhedras with areas above 40% of the largest polyhedra are included in the Voronoi grid.

Evidently, the rightmost clusters for large PC 1 values are related to subsurface Cu atoms that are exposed to the surface via vacancies in the surface. We also observe that the number of nearest neighbor oxygen $NN_O$ decreases and the number of nearest neighbor copper $NN_{Cu}$ increases with increasing values of PC 1, indicating some correlation. Other than the chemical composition of the entire structure (see Figure 5.13 and Figure 5.14), we were unable to find a "basic" local observable with a correlation in the second PC.

### Cu (111) surfaces

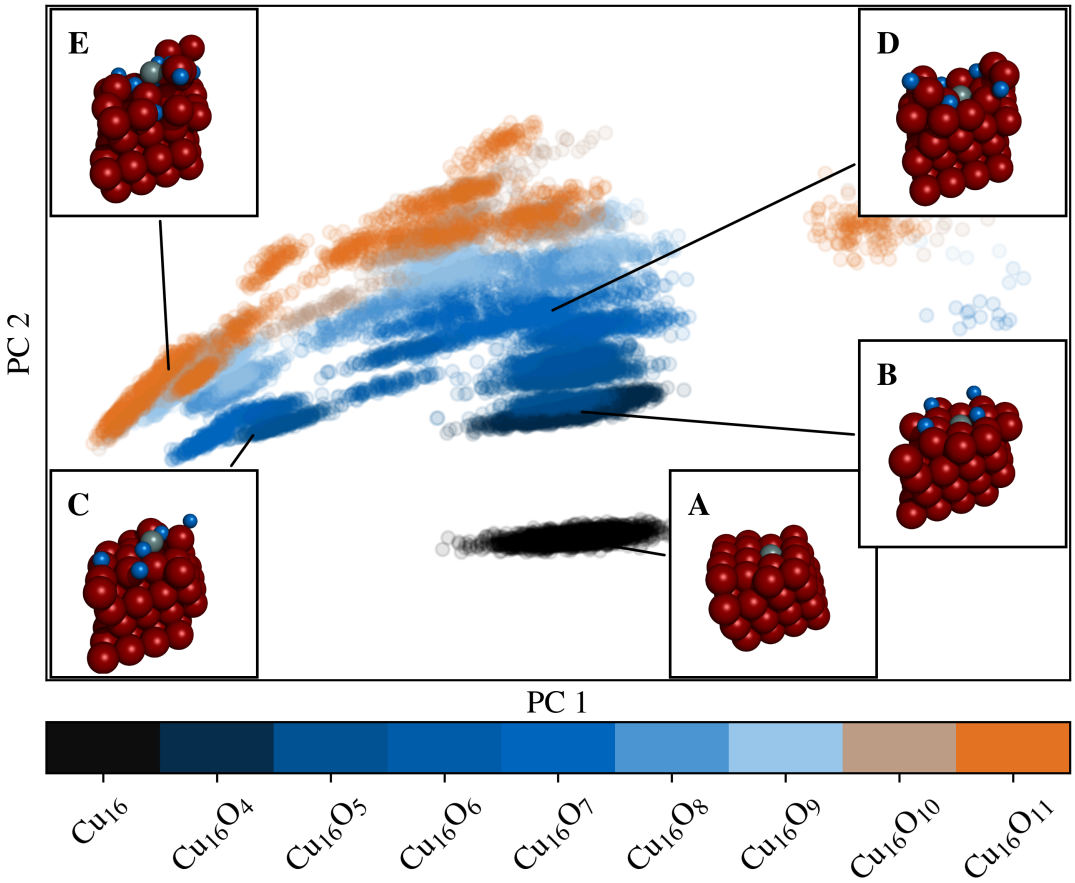



**Figure 5.13** PCA of Cu (111) slabs, color coded by surface composition. Images of representative local Cu surface environments (the center atom marked grey) are also pictured. We observe that $Cu_{16}$ and $Cu_{16}O_4$ have only one cluster, while compositions with higher O content separate into several groups.

Figure 5.13 reports the PCA results for only the Cu (111) surface atoms in Figure 5.12, color coded by the surface termination. Although not apparent from the $NN_O$ (Figure 5.12),

Figure 5.13 implies that the second PC correlates with the O content of the entire surface.

We see that $Cu_{16}$, and $Cu_{16}O_4$ are grouped to one cluster each, meaning that all surface local atomic environments are virtually identical (**A** and **B**). The separation between **A** and **B** can be attributed to the inclusion of oxygen atoms into the SOAP vectors. For surfaces with higher oxygen content than $Cu_{16}O_4$, the data points for each chemical composition of the topmost layer become progressively more scattered and separate clusters form at low values of PC 1 (**C**, **E**). These are the Cu adatoms that are pushed out of the top layer due to O moving subsurface. The number of data points falling within these clusters grows with increasing oxygen content, whilst clusters belonging to surface Cu in the original topmost layer (**B**, **D**) decreases. This supports the observation that a higher oxygen content results in more Cu adatoms.

### Low-Density Cu (111) surfaces

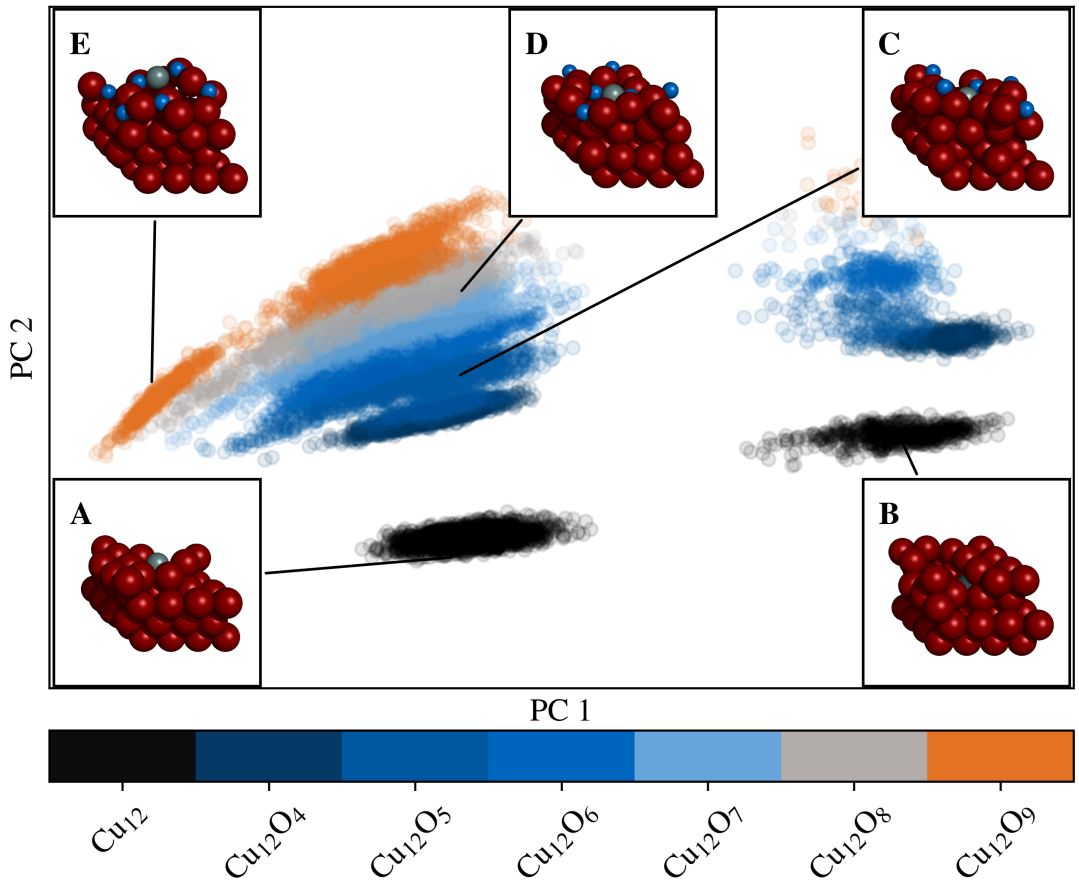



**Figure 5.14** PCA of LD-Cu (111) slabs, color coded by the surface termination. Images of representative local Cu surface environments (the center atom marked grey) are also pictured. Compared to Figure 5.13, we note that the data points for each chemical composition are mostly divided into two main groups: sub- and top-layer Cu.

For the LD-Cu (111) data points in Figure 5.12 we observe that the data set is split into two major clusters, including the oxygen-free surface $Cu_{12}$. Note that the LD-Cu (111) surface typically relaxes into regions with Cu – Cu environments as on Cu (111) surface, and regions with vacancies. The two clusters in Figure 5.14 are attributed to the atoms that are in the top layer (**A**), and the atoms in the sublayers that are exposed to the surface due to vacancies in the top layer (**B**). When looking at the subsurface clusters, we notice that with increasing oxygen content the number of atoms forming these clusters decreases, consistent with the idea of a stable lower-density Cu – O surface phase at higher oxygen coverages.

Another separation of the dataset occurs in the second dimension of the PCA, splitting the data points between $Cu_{12}$ from all oxygen covered surfaces.

Finally we mention the small cluster of $Cu_{12}O_9$ atoms on the left side of Figure 5.14. As shown in Figure 5.14 (**E**), these data points belong to the Cu atoms that are pushed out of the top layer during the MD runs.

## 5.4 Similarity to Low-Index Surfaces of Copper Oxides

In 2016 De et al. introduced a distance metric $d$ for normed SOAP vectors $\mathbf{p}(\chi')$ and $\mathbf{p}(\chi'')$ of two local chemical environments $\chi'$ and $\chi''$, with[3]

$$d = \sqrt{2 - 2\,(\mathbf{p}(\chi') \cdot \mathbf{p}(\chi''))}. \tag{5.1}$$

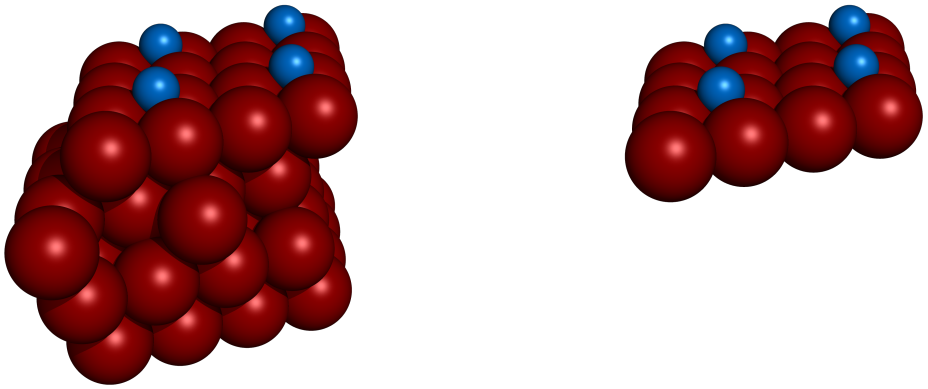
Smaller values for $d$ indicate more similar local environments. Note that this metric sets a "maximum" dissimilarity of two environments of $2 = \sqrt{2 - 2 \cdot (-1)}$, limiting our ability to discern dissimilar local atomic environments from one another.

However, with the distance metric at hand, it is possible to compare local surface environments in our MD runs with environments at low-index surfaces of various Cu oxides.

The approach is as follows: Using the SlabGenerator class from pymatgen, we create (100), (110), and (111) surfaces from various bulk Cu oxides.[3] Next, we randomly displace the atoms around a mean of 0.05 Å. Repeating this process 35 times, we generate approximately 500 local surface Cu environments for each surface.
As both the Cu (111) and LD-Cu (111) structures from our MD runs have sublayers of pure Cu, comparisons to those acquired from low-index bulk oxide surfaces, whose sublayers

[3]The systems taken into account are different polymorphs of $Cu_2O_3$, $Cu_3O_4$, CuO, $Cu_4O_3$, $Cu_2O$, and Cu.

consist of Cu – O bonds, becomes difficult. Therefore, for all structures, we remove all atoms with surface contributions below 1 % (aka subsurface atoms). An example of the resulting structures is shown in Figure 5.15. This step limits the scope of our investigation, however it allows the analysis of local motives in the surface region.



**(a)** Complete $Cu_{16}O_4$ structure

**(b)** $Cu_{16}O_4$ structure with non-surface accessible atoms removed.

**Figure 5.15** Example of a typical Cu (111) structure a. As the system is only partially oxidized, we remove the Cu sublayers to increase the similarity to bulk oxides b.

We evaluate the last 50 ps of MD runs at 300 K, per MD leaving us with approximately 13000 surface atoms to evaluate. We calculate $\min(d)$ between each of these local surface environments and those of the low-index bulk surfaces. If $\min(d) < 0.1$, we classify the MD surface atom as belonging to the respective low-index bulk oxide surface. If the minimum distance is above 0.1, we judge that the dissimilarity is too large and do not classify it as having a low-index bulk oxide counterpart.
The SOAP vectors of surface accessible Cu atoms for all structures were computed with identical parameters as in section 5.3.

### Cu (111) Surfaces

The results of our classification approach are shown in Figure 5.16.

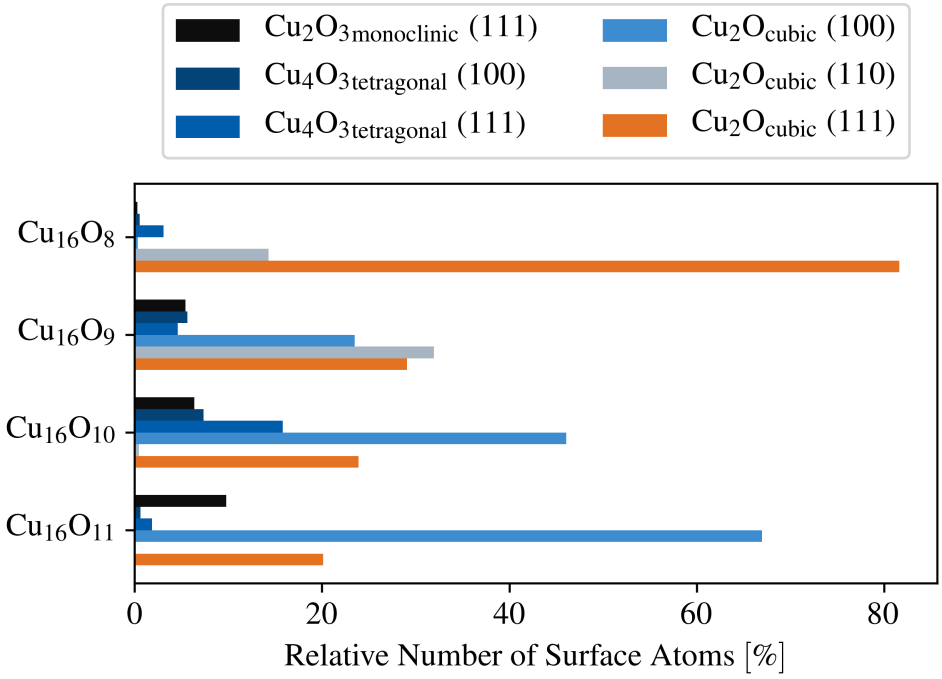
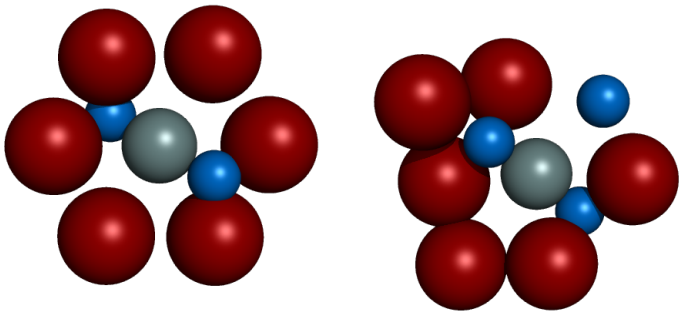



**Figure 5.16** Results of the similarity classification analysis between SOAP vectors of Cu (111) and low-index $Cu_xO_y$ (111) surface accessible Cu atoms. We observe that increasing the O content above $\Theta_O = 50\%$ leads to a reduction in $Cu_2O$ (111) local environments, whilst simultaneously increasing the number of $Cu_2O$ (100) environments.

Figure 5.16 begins at $Cu_{16}O_8$ because $\min(d)$ is above our chosen threshold at lower oxygen coverages. Where surface Cu atoms exhibit only environments that are very different from those at Cu oxide surfaces. As explained previously, for O coverages $\Theta_O > 25\%$ an increasing number of O is incorporated as subsurface oxygen and apparently already at $\Theta_O = 50\%$ local Cu environments are extremely similar to those of an oxide surface.
We observe that with increasing oxygen content in the surface, the $Cu_2O$ (111) character of the surface disappears and is replaced by an ever increasing amount of $Cu_2O$ (100) content.

For $Cu_{16}O_8$, we find that the majority of surface Cu are coordinated to five in-plane Cu atoms, as shown in Figure 5.17.

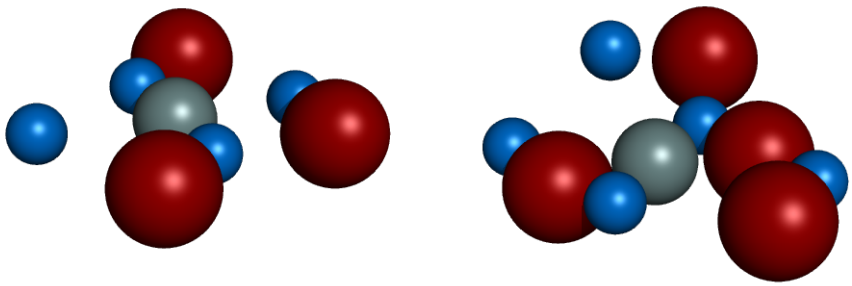


**(a)** $Cu_2O_{cubic}$ (111) **(b)** $Cu_{16}O_8$

**Figure 5.17** Two local environments centered around a Cu atom marked grey which the SOAP classifier deems "most similar".

The distances between the center and in-plane Cu atoms vary between 2.76 Å and 3.60 Å. Occasionally, a further Cu atom slightly out of the plane at a distance of 4 Å is included in the local environment. Presumably since the cutoff was set to 4.2 Å this atom does not always appear. Our classifier connects these local environments to the Cu atoms in the (111) surface of cubic $Cu_2O$. These are coordinated to six neighboring Cu atoms at distances of 3.0 Å, arranged in a hexagonal pattern around the central atom.
Both the $Cu_2O$ (111) and $Cu_{16}O_8$ atoms are bonded to two oxygen atoms at distances between 1.78 and 1.88 Å, with one O on the surface above the central atom, and one coordinated subsurface at 180° to the other O atom. In the $Cu_{16}O_8$ surface, typically a third oxygen atom is located at bond distances of approximately 3.5 Å, where we we would expect the sixth in-plane Cu atom to be in the $Cu_2O$ (111) surface.
Additionally, we observe that the atoms corresponding to the $Cu_2O$ (111) surface are not those that form islands above the original surface layer. Only in 26 of 2700 environments of $Cu_{16}O_8$ attributed to $Cu_2O$ (111), did the central atom possess the largest $z$-coordinate in its local environment. These "island atoms" are also not the main contribution to the sizeable bar of $Cu_2O$ (110) atoms in Figure 5.16. Only 18 % of these atoms possess the largest $z$-coordinate in their local environment. Mainly, these island atoms are not connected to any of the studied Cu oxide terminations.

If we increase the oxygen content to $Cu_{16}O_9$, the classifier greatly reduces the percentage of $Cu_2O$ (111) surface atoms, instead the relative percentage of $Cu_2O$ (110) doubles to 33 %, and we see a sudden appearance of $Cu_2O$ (100) atoms, roughly 25 %.



**(a)** $Cu_2O_{cubic}$ (110) **(b)** $Cu_{16}O_9$

**Figure 5.18** Comparison of local surface environments of $Cu_2O$ (110) and $Cu_{16}O_9$. Similarities include the "oxygen cross" and the three in-plane copper neighbors. Typically we see an additional out of plane copper and oxygen atom, as visible in (b).

We tentatively explain the reduction of the $Cu_2O$ (111) contribution with the observation that the additional oxygen on average displaces one of the neighboring Cu atoms. As shown in table 5.1, the number of surface atoms with 3 or 4 neighboring Copper atoms is much higher for $Cu_{16}O_9$ than for the other compositions. 94 % of these structures are classified as $Cu_2O$ (110).

**Table 5.1** Percentages of neighboring copper atoms in the local environments for all unreconstructed MD runs. $NN_{Cu}$ denotes the number of nearest neighbors Cu neighbors, calculated with the same method as in section 5.3.2.

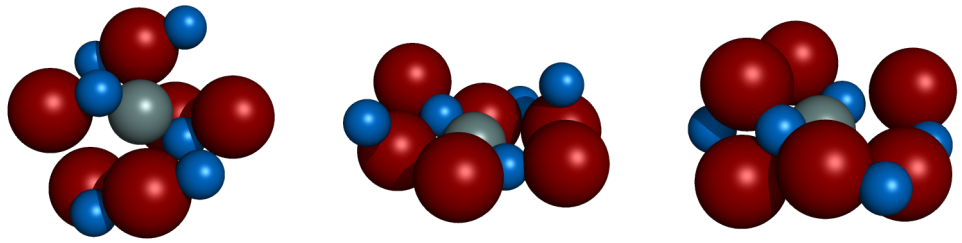
| $NN_{Cu}$ | $Cu_{16}O_8$ [%] | $Cu_{16}O_9$ [%] | $Cu_{16}O_{10}$ [%] | $Cu_{16}O_{11}$ [%] |
|---|---|---|---|---|
| 3 | 4 | 10 | 0 | 0 |
| 4 | 9 | 21 | 3 | 0 |
| 5 | 17 | 16 | 21 | 8 |
| 6 | 41 | 25 | 36 | 31 |
| 7 | 26 | 20 | 25 | 42 |
| 8 | 3 | 6 | 13 | 15 |

Table 5.1 also shows that adding another oxygen atom immediately leads to the almost complete disappearance of local environments containing fewer than 5 neighboring Copper atoms, coinciding with the elimination of the $Cu_2O$ (110) bar in Figure 5.16. Replacing $Cu_2O$ (110), $Cu_2O$ (100) becomes the major class of surface atoms. This shift is not linked to the number of nearest neighbor Copper or oxygen atoms, shown in tables 5.1 and 5.2 respectively, which are similar for $Cu_{16}O_9$ and $Cu_{16}O_{10}$.

**Table 5.2** Percentages of neighboring oxygen atoms in the local environments for all unreconstructed MD runs.

| $NN_O$ | $Cu_{16}O_8$ [%] | $Cu_{16}O_9$ [%] | $Cu_{16}O_{10}$ [%] | $Cu_{16}O_{11}$ [%] |
|---|---|---|---|---|
| 3 | 16 | 0 | 2 | 0 |
| 4 | 30 | 6 | 20 | 4 |
| 5 | 35 | 53 | 46 | 25 |
| 6 | 19 | 34 | 26 | 38 |
| 7 | 0 | 6 | 6 | 27 |
| 8 | 0 | 0 | 1 | 5 |

What then is the cause of this massive shift? Breaking down the observed data of $Cu_{16}O_9$ and $Cu_{16}O_{10}$, we find that the biggest swing occurs in environments with 6 $NN_{Cu}$ and 5 $NN_O$ (cf. tables 5.1, 5.2). A comparison of these structures is shown in Figure 5.19.



**(a)** $Cu_2O_{cubic}$ (100) **(b)** $Cu_{16}O_9$ **(c)** $Cu_{16}O_{10}$

**Figure 5.19** Comparison of a typical $Cu_2O$ (100) surface local environment with 6 $NN_{Cu}$ and 5 $NN_O$ local environments for $Cu_{16}O_9$ and $Cu_{16}O_{10}$.

The main difference between $Cu_{16}O_9$ and $Cu_{16}O_{10}$ is the increased number of Copper adatoms in $Cu_{16}O_{10}$. While in Figure 5.19 (b) only one atom is out-of plane, half the Copper atoms in (c) are distinctively above the central atom. This increased 3D nature of the local structure is most likely the cause of the increased agreement with the $Cu_2O$ (100) substructures, which are more open than $Cu_2O$ (111) terminations. Additionally, we also note the different oxygen coordinations in Figure 5.19 (b) and (c). $Cu_{16}O_{10}$ indicates the development of clear rows of $CuO_2$ units moving along the surface as shown in Figure 5.19 (a).

### Low-Density Cu (111) Surfaces

The results of our classification approach on the reconstructed monolayers are shown in Figure 5.20. Although we also investigated structures with lower oxygen coverages, none of the atoms in these structures returned $d$ values below our classification threshold.

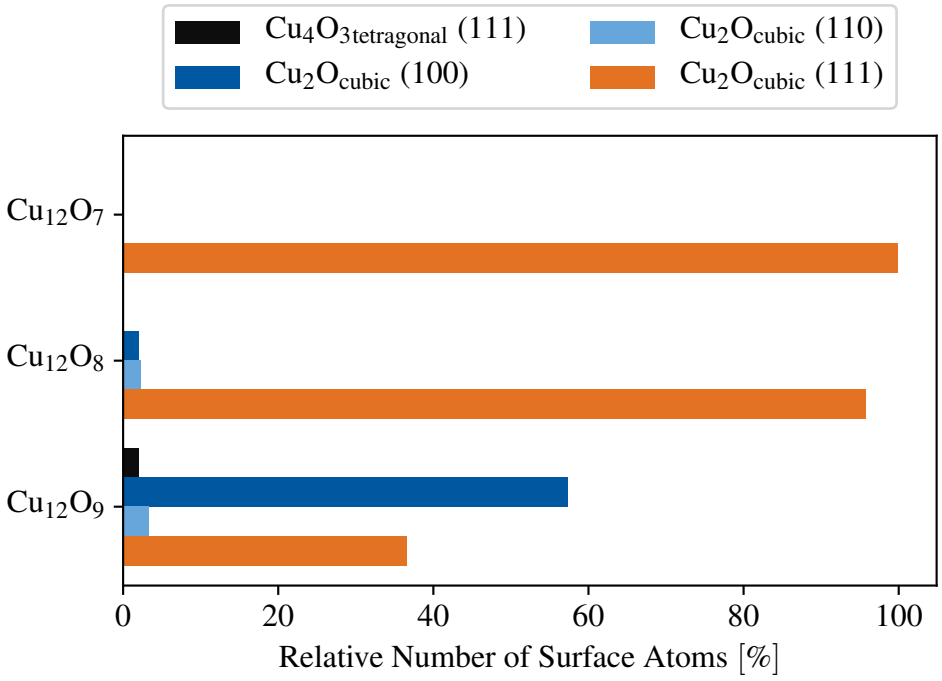
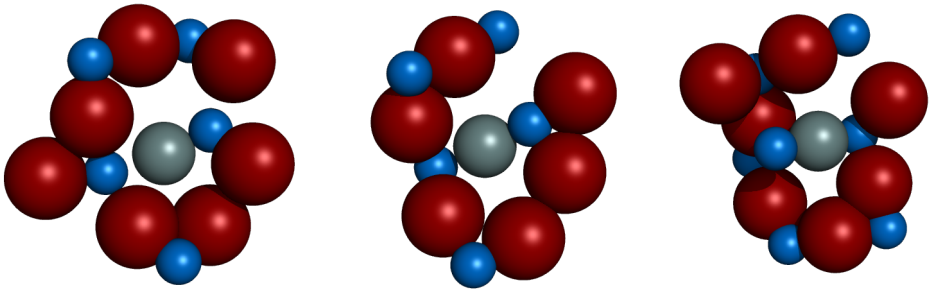



**Figure 5.20** Results of the similarity classification analysis between SOAP vectors of LD-Cu (111) and low-index $Cu_xO_y$ (111) surface accessible Cu atoms. As in Figure 5.16, we observe that for lower coverages the majority of surface environments are classified as $Cu_2O$ (111). At $Cu_{12}O_9$ the $Cu_2O$ (111) content is reduced, replaced by $Cu_2O$ (100) environments.

Our observations are similar to the Cu (111) structures. Again, at low oxygen coverages the surface is almost exclusively classified as $Cu_2O$ (111). This is expected, as our original LD-Cu (111) surface structure does indeed correspond to a $Cu_2O$ (111) termination and the results in Figure 5.20 thus validate our classification algorithm. Increasing the oxygen content leads the classifier to identify $Cu_2O$ (100) substructures. The key difference is that the shift to (100) surfaces starts at much higher oxygen coverages than in the case of Cu (111) surfaces. This is consistent with the observation that Cu is pushed out of the surface at $Cu_{12}O_9$ (section 5.2).



**(a)** $Cu_{12}O_7$ **(b)** $Cu_{12}O_8$ **(c)** $Cu_{12}O_9$

**Figure 5.21** LD-Cu structures with high oxzgen content.

# 6 Conclusion and Outlook

We utilize machine learned Gaussian approximation potentials (GAPs), to better understand the morphology of Cu (111) surfaces undergoing oxidation.

GAPs are trained on *ab initio* data, in our case DFT, and fit to physical observables for a set of training structures via Gaussian process regression. Built around local atomic environment descriptors, GAPS map the atomic species and positions within a cutoff $r_{\text{cut}}$ to the total energy and atomic forces of the structures. The descriptors used in this thesis are the two-body pairwise distance descriptor and the "Smooth Overlap of Atomic Positions" (SOAP) descriptor.
The Cu – O GAP was created via an iterative process. Starting with a small dataset of different bulk copper oxide polymorphs, we created a first GAP. Subsequently, we ran molecular dynamics calculations on new structures, and validated the performance of the GAP on these unknown structures. Structures were added to the GAP's training set until the validation error converged to the training error, after which we moved onto new structural classes. This allowed us to track the progress of the GAP for different structure types.
Once the training set was sufficiently large and diverse, we optimized the GAP hyperparameters via cross-validation, finding multiple paths to improve training efficiency, as well as GAP performance. Among these findings is that the computational time can be reduced by selecting small SOAP basis sets at little accuracy cost, and that the choice of cutoff radius is vital to GAP accuracy.

Using the Cu – O GAP, we ran MD calculations at 300 K on Cu (111) surfaces with different oxygen coverages. We observe that oxygen coverages above 0.25 monolayers lead to surface Cu atoms being pushed out of the surface, forming islands. At the same time, some oxygen atoms go subsurface.
Various experiments suggest that Cu (111) surfaces that have undergone oxidation-reduction cycles have surface Cu densities similar to $Cu_2O$.[1] Therefore, we create low-density (LD) Cu (111) surfaces, with 25 % less top layer Cu than Cu (111) surfaces. The MD runs on these surfaces suggest possible oxygen coverages of 75 % without the creation of adatoms.

To analyze morphology changes during the calculations, we introduce surface acces-

sible coverages $\theta_i$. Instead of calculating the coverage by getting the fraction of adsorbates vs the surface atoms, we calculate a 3-d isosurface and attribute the surface area to the individual atom types.
We find that both Cu and LD-Cu (111) restructure in such a way that the surface accessible oxygen coverage $\theta_O$ is between 50 and 60 %. Furthermore, using the SOAP descriptor, we find that at low surface coverages the local structures at Cu and LD-Cu are similar to those found at $Cu_2O$ (111) surfaces. Increasing oxygen content leads to these local motifs disappearing and being replaced with those in $Cu_2O$ (100). Also, LD-Cu without adsorbates separates into islands with normal (111) surface termination and vacancies.
Taking these considerations into account, we observe no considerable behavioral difference of the LD-Cu and Cu (111) surfaces. This suggests that no significant kinetic barriers exist which makes the study of Cu (111) oxidation straightforward in the future, without the need to introduce "artificial" LD-Cu (111) surfaces.

Using the Cu – O GAP, we can perform MD calculations at a fraction of the computational cost of *ab initio* methods. In the case of oxidized copper surfaces this can be used to investigate a larger surface area of copper, making it possible to examine adsorption gradient effects. Another possibility is the analysis of subsurface oxygen, as some researchers believe the electronic structure and the mobility of these atoms greatly affects the catalytic properties of copper surfaces.[10] Further applications are advanced sampling methods to uncover the reconstruction pathways of Cu (111) to low-density $Cu_2O$ derivatives that have been observed in literature.[1,9]
By adding species such as hydrogen or carbon to the training set, the GAP can be a valuable tool to clarify reaction pathways for relevant catalysis reactions such as the carbon dioxide reduction reaction or the electrochemical carbon monoxide oxidation reaction.

# Bibliography


(1) Auer, A.; Andersen, M.; Wernig, E.-M.; Hörmann, N. G.; Buller, N.; Reuter, K.; Kunze-Liebhäuser, J. *Nature Catalysis* **2020**, *3*, 797–803.

(2) Bartók, A. P.; Csányi, G. *International Journal of Quantum Chemistry* **2015**, *115*, 1051–1057.

(3) De, S.; Bartók, A. P.; Csányi, G.; Ceriotti, M. *Physical Chemistry Chemical Physics* **2016**, *18*, 13754–13769.

(4) Gattrell, M.; Gupta, N.; Co, A. *Journal of Electroanalytical Chemistry* **2006**, *594*, 1–19.

(5) Herzog, T. *World Resources Institute* **2009**.

(6) Dhar, H.; Christner, L.; Kush, A.; Maru, H. *Journal of the Electrochemical Society* **1986**, *133*, 1574.

(7) Jeon, H. S.; Timoshenko, J.; Scholten, F.; Sinev, I.; Herzog, A.; Haase, F. T.; Roldan Cuenya, B. *Journal of the American Chemical Society* **2019**, *141*, 19879–19887.

(8) Reske, R.; Mistry, H.; Behafarid, F.; Roldan Cuenya, B.; Strasser, P. *Journal of the American Chemical Society* **2014**, *136*, 6978–6986.

(9) Verdaguer-Casadevall, A.; Li, C. W.; Johansson, T. P.; Scott, S. B.; McKeown, J. T.; Kumar, M.; Stephens, I. E.; Kanan, M. W.; Chorkendorff, I. *Journal of the American Chemical Society* **2015**, *137*, 9808–9811.

(10) Eilert, A.; Cavalca, F.; Roberts, F. S.; Osterwalder, J.; Liu, C.; Favaro, M.; Crumlin, E. J.; Ogasawara, H.; Friebel, D.; Pettersson, L. G., et al. *The Journal of Physical Chemistry Letters* **2017**, *8*, 285–290.

(11) Garza, A. J.; Bell, A. T.; Head-Gordon, M. *The Journal of Physical Chemistry Letters* **2018**, *9*, 601–606.

(12) Kim, S.-Y.; Kumar, N.; Persson, P.; Sofo, J.; Van Duin, A. C.; Kubicki, J. D. *Langmuir* **2013**, *29*, 7838–7846.

(13) Bartók, A. P.; Payne, M. C.; Kondor, R.; Csányi, G. *Physical Review Letters* **2010**, *104*, 136403.

(14) Bartók, A. P.; Kondor, R.; Csányi, G. *Physical Review B* **2013**, *87*, 184115.

(15) Cramer, C. J., *Essentials of computational chemistry: theories and models*; John Wiley & Sons: 2013.

(16) Born, M. *Annals of Physics* **1927**, *84*, 457.

(17) Marek, A.; Blum, V.; Johanni, R.; Havu, V.; Lang, B.; Auckenthaler, T.; Heinecke, A.; Bungartz, H.-J.; Lederer, H. *Journal of Physics: Condensed Matter* **2014**, *26*, 213201.

(18) Köppl, C.; Werner, H.-J. *Journal of Chemical Theory and Computation* **2016**, *12*, 3122–3134.

(19) Jensen, F., *Introduction to Computational Chemistry*; John wiley & sons: 2017.

(20) Hohenberg, P.; Kohn, W. *Physical Review* **1964**, *136*, B864.

(21) Kohn, W.; Sham, L. J. *Physical Review* **1965**, *140*, A1133.

(22) Becke, A. D. *The Journal of Chemical Physics* **2014**, *140*, 18A301.

(23) Parr, R. G. In *Horizons of Quantum Chemistry*; Springer: 1980, pp 5–15.

(24) Perdew, J. P.; Ernzerhof, M.; Burke, K. *The Journal of Chemical Physics* **1996**, *105*, 9982–9985.

(25) Haas, P.; Tran, F.; Blaha, P.; Schwarz, K. *Physical Review B* **2011**, *83*, 205117.

(26) Woods, N.; Payne, M.; Hasnip, P. *Journal of Physics: Condensed Matter* **2019**, *31*, 453001.

(27) Bloch, F. *Zeitschrift für Physik* **1928**, *52*, 555–600.

(28) Giannozzi, P.; Baroni, S.; Bonini, N.; Calandra, M.; Car, R.; Cavazzoni, C.; Ceresoli, D.; Chiarotti, G. L.; Cococcioni, M.; Dabo, I., et al. *Journal of Physics: Condensed Matter* **2009**, *21*, 395502.

(29) Schwerdtfeger, P. *ChemPhysChem* **2011**, *12*, 3143–3155.

(30) Garrity, K. F.; Bennett, J. W.; Rabe, K. M.; Vanderbilt, D. *Computational Materials Science* **2014**, *81*, 446–452.

(31) Xue, G. *Theoretical Computer Science* **1998**, *197*, 157–169.

(32) Harrison, J. A.; Schall, J. D.; Maskey, S.; Mikulski, P. T.; Knippenberg, M. T.; Morrow, B. H. *Applied Physics Reviews* **2018**, *5*, 031104.

(33) Lopes, P. E.; Guvench, O.; MacKerell, A. D. In *Molecular Modeling of Proteins*; Springer: 2015, pp 47–71.

(34) Maple, J. R.; Hwang, M.-J.; Stockfisch, T. P.; Dinur, U.; Waldman, M.; Ewig, C. S.; Hagler, A. T. *Journal of Computational Chemistry* **1994**, *15*, 162–182.

(35) De Vivo, M.; Masetti, M.; Bottegoni, G.; Cavalli, A. *Journal of Medicinal Chemistry* **2016**, *59*, 4035–4061.

(36) Daw, M. S.; Baskes, M. I. *Physical Review B* **1984**, *29*, 6443.

(37) Abell, G. *Physical Review B* **1985**, *31*, 6184.

(38) Brenner, D. W. *Physical Review B* **1990**, *42*, 9458.

(39) Senftle, T. P.; Hong, S.; Islam, M. M.; Kylasa, S. B.; Zheng, Y.; Shin, Y. K.; Junkermeier, C.; Engel-Herbert, R.; Janik, M. J.; Aktulga, H. M., et al. *npj Computational Materials* **2016**, *2*, 1–14.

(40) Mitchell, T. Machine Learning 1. Edition., 1997.

(41) Behler, J. *Angewandte Chemie International Edition* **2017**, *56*, 12828–12840.

(42) Behler, J.; Parrinello, M. *Physical Review Letters* **2007**, *98*, 146401.

(43) Behler, J. *Physical Chemistry Chemical Physics* **2011**, *13*, 17930–17955.

(44) Singraber, A.; Morawietz, T.; Behler, J.; Dellago, C. *Journal of Chemical Theory and Computation* **2019**, *15*, 3075–3092.

(45) Behler, J. *The Journal of Chemical Physics* **2011**, *134*, 074106.

(46) Szlachta, W. J.; Bartók, A. P.; Csányi, G. *Physical Review B* **2014**, *90*, 104108.

(47) Williams, C. K.; Rasmussen, C. E., *Gaussian Processes for Machine Learning*; 3; MIT press Cambridge, MA: 2006; Vol. 2.

(48) Von Mises, R., *Mathematical Theory of Probability and Statistics*; Academic Press: 2014.

(49) Pizzi, G.; Cepellotti, A.; Sabatini, R.; Marzari, N.; Kozinsky, B. *Computational Materials Science* **2016**, *111*, 218–230.

(50) Perdew, J. P.; Burke, K.; Ernzerhof, M. *Physical Review Letters* **1996**, *77*, 3865.

(51) Marzari, N.; Vanderbilt, D.; De Vita, A.; Payne, M. *Physical Review Letters* **1999**, *82*, 3296.

(52) Chadi, D. J.; Cohen, M. L. *Physical Review B* **1973**, *8*, 5747.

(53) Persson, K.; Project, M. **2020**, DOI: 10.17188/1204433.

(54) Jain, A.; Castelli, I. E.; Hautier, G.; Bailey, D. H.; Jacobsen, K. W. *Journal of Materials Science* **2013**, *48*, 6519–6534.

(55) Plimpton, S. *Journal of Computational Physics* **1995**, *117*, 1–19.

(56) Harvey, S. C.; Tan, R. K.-Z.; Cheatham III, T. E. *Journal of Computational Chemistry* **1998**, *19*, 726–740.

(57) Mahoney, M. W.; Drineas, P. *Proceedings of the National Academy of Sciences* **2009**, *106*, 697–702.

(58) Jain, A.; Ong, S. P.; Hautier, G.; Chen, W.; Richards, W. D.; Dacek, S.; Cholia, S.; Gunter, D.; Skinner, D.; Ceder, G.; Persson, K. a. *APL Materials* **2013**, *1*, 011002.

(59) Larsen, A. H. et al. *Journal of Physics: Condensed Matter* **2017**, *29*, 273002.

(60) Birch, F. *Physical Review* **1947**, *71*, 809.

(61) Bernstein, N.; Csányi, G.; Deringer, V. L. *npj Computational Materials* **2019**, *5*, 1–9.

(62) Mills, I. et al., *Quantities, units and symbols in physical chemistry/prepared for publication by Ian Mills...[et al.]* Oxford; Boston: Blackwell Science; Boca Raton, Fla.: CRC Press [distributor], 1993.

(63) Andreussi, O.; Hormann, N. G.; Nattino, F.; Fisicaro, G.; Goedecker, S.; Marzari, N. *Journal of Chemical Theory and Computation* **2019**, *15*, 1996–2009.

(64) Schölkopf, B.; Smola, A.; Müller, K.-R. In *Advances in Kernel Methods: Support Vector Learning*, MIT Press: 1999, pp 327–352.

(65) Himanen, L.; Jäger, M. O. J.; Morooka, E. V.; Federici Canova, F.; Ranawat, Y. S.; Gao, D. Z.; Rinke, P.; Foster, A. S. *Computer Physics Communications* **2020**, *247*, 106949.

(66) Tran, R.; Xu, Z.; Radhakrishnan, B.; Winston, D.; Sun, W.; Persson, K. A.; Ong, S. P. *Scientific Data* **2016**, *3*, 1–13.

(67) Sun, W.; Ceder, G. *Surface Science* **2013**, *617*, 53–59.

(68) Persson, K. A.; Waldwick, B.; Lazic, P.; Ceder, G. *Physical Review B* **2012**, *85*, 235438.

(69) Rogal, J.; Reuter, K. *Ab initio atomistic thermodynamics for surfaces: A primer*; tech. rep.; Max-planck-gesellschaft zur foerderung der wissenschaften ev berlin (germany . . ., 2006.

(70) Reuter, K.; Scheffler, M. *Physical Review B* **2003**, *68*, 045407.

(71) Tran, F.; Stelzl, J.; Blaha, P. *The Journal of Chemical Physics* **2016**, *144*, 204120.

(72) Hörmann, N. G.; Andreussi, O.; Marzari, N. *The Journal of Chemical Physics* **2019**, *150*, 041730.

(73) Lozovoi, A. Y.; Alavi, A. *Physical Review B* **2003**, *68*, 245416.

# Appendix

## Feed-Forward Neural Network Potentials (FFNNPs)

FFNNPs have three types of layers. The input layer $G$ contains the geometric information of the input structure. $G$ is then processed by the hidden layer $y$, or more commonly by multiple hidden layers $y_i$. The hidden layer then maps the physical information in $G$ to the output layer $E$.[41]

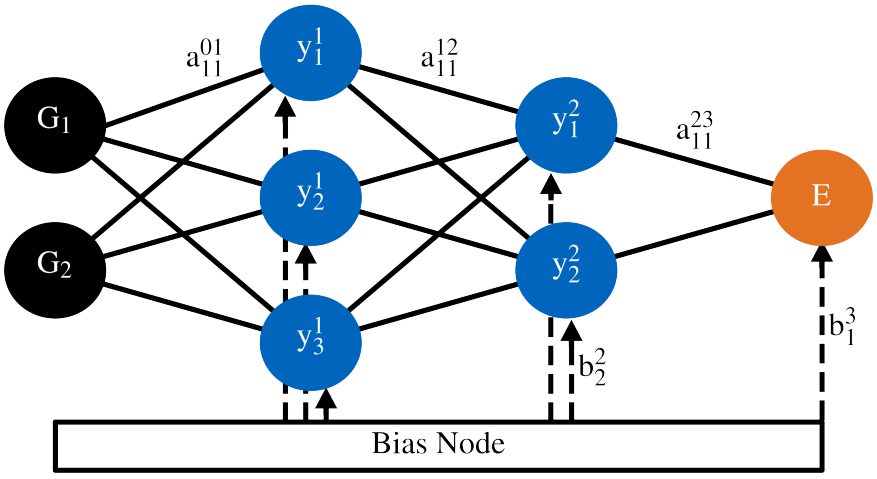



**Figure 6.1** Theoretical structure of FFNNP with two hidden layers. The figure is a simplified version from [41].

The regression of the physical information occurs in the hidden layers. These have no real physical meaning and are often referred to as a "black box".[43] Each hidden layer $j$ consists of nodes; the number of nodes per layer $N^j$ does not have to be the same for each layer.

The FFNNP assigns a value between 0 and 1 for each node in every layer. For the node $i$ in layer $j$, these values are determined by adding the weighted sum of all nodes in the layer $j-1$ to a bias weight $b_i^j$. The weighting occurs with the help of a weight parameter $a^{j-1,j}$ which is determined during the training of the neural network. In order to ensure that the outputs of each node are continuous values, an activation function $f_i^j$ is applied to

the entire sum.[41]

$$y_i^j = f_i^j \left( b_i^j + \sum_{k=1}^{N^{j-1}} a_k^{j-1,j} \cdot y_k^{j-1} \right) \tag{6.1}$$

In the case of multiple hidden layers, the same process occurs for each hidden layer. $G$ serves as the input layer for the first hidden layer.

The setup of the hidden layers is crucial to the success of the FFNNP. The most obvious adjustment the user can make is changing the number of hidden layers, or the number of neurons within the hidden layers. An increase of these numbers leads to more complex hidden layers, which results in a more flexible FFNNP. This allows the NNP to detect more subtleties within the Potential Energy Surface of the phase.

However, if the number of hidden layers or nodes is too high, the force field will overfit the PES.[41] A method to avoid overfitting is schematically shown in Figure 6.2. By splitting the dataset into a validation and training set, and only fitting the potential on the training data, we are able to observe artificial features in the potential by comparing the RMSE of the validation data to the RMSE of the training data.

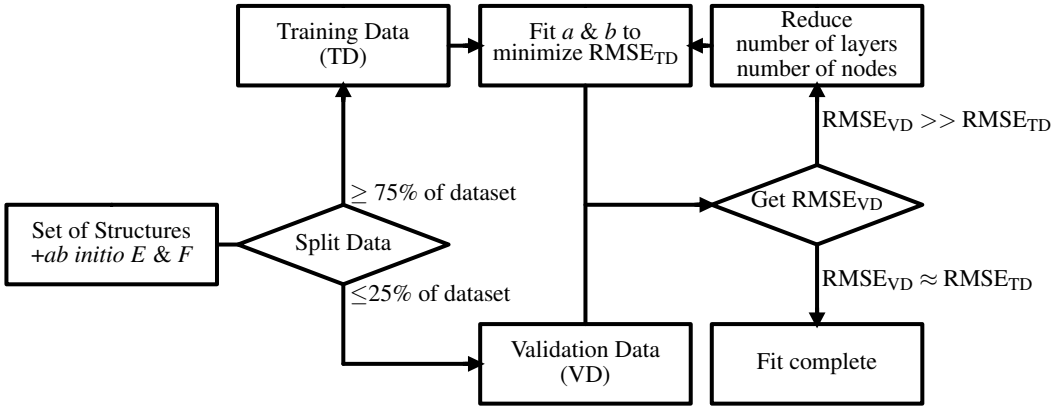



**Figure 6.2** Overview of the neural network potential fitting workflow.

Thus, we have an algorithm capable of determining the PES, the inputs being the set of atomic positions $\{G_i\}$, the set of weight parameters $\{a_i\}$, and the set of bias weights $\{b_i\}$. The weight parameters and bias weights are determined in the training phase. Using optimization algorithms such as the "back propagation" algorithm, or the Kalman Filter, $\{a\}$ and $\{b\}$ are changed in such a way that the RMSE of the output layer is minimized. As training can take a long time, there is currently a strong research focus on optimizing the involved processes.[44]

## High Dimensional NNPs (HDNNPS)

High Dimensional NNPs (HDNNPs), first presented by Behler and Parrinello in 2007[42], address a major issue of FFNNPs: The FFNNP can only be fit for input vectors with the

exact same number of nodes, meaning that the system size is prohibited from changing. This limits the applicability of FFNNP to low dimensional problems.
HDNNPs acquire the total energy of the system by taking the sum of the local, short-range energy of every atom in the system. This local energy is calculated by creating a feed-forward neural network around the central atom. The local environment is cutoff at a certain radius $r_c$. As this approach needs an exact prediction of the energies of each atom, the fitting process of the HDNNP focuses on interpreting the local environments of each atom.
The total energy of any system is unaffected by rotation, translation, or permutation of the atom ordering. The descriptor of the atomic coordinates must then be invariant to these changes as well. It can be achieved by mapping the Cartesian coordinates onto a vector consisting of atom-centered symmetry functions. By using these functions as descriptors we are able to make sure that the NNP maps equivalent geometries to identical energies. The Behler-Parrinello symmetry functions[45] and the 'Smooth Overlap of Atomic Positions' (SOAP) descriptor serves as examples of such functions (see section 2.3.2).

Once the atom has been transformed into a vector of symmetry functions, these are used as input for the atomic NN. A nice feature of the symmetry functions is that they can be constructed in such a way that their number remains constant despite changes in the number of atoms in the environment.[41] This means that the fixed nature of the input vector in FFNNP is unproblematic in this case.

# Excess surface free energy $g_{exc}$

## *Ab-initio* thermodynamics for bulk materials

A bulk system's Gibbs formation energy $G^{\alpha}_{\text{form}}$ is defined as the cost of moving the elements $i$ out of their respective reference state and into their configuration, assuming that the system $\alpha$ consists of a set of atoms $\{N_i^{\alpha}\}$.[68]

$$G^{\alpha}_{\text{form}}\left(\{N_i^{\alpha}\}\right) = G^{\alpha}_{\text{bulk}}\left(\{N_i^{\alpha}\}\right) - \sum_i N_i^{\alpha}\mu_i \tag{6.2}$$

The reference state chemical potential of the element $i$ is $\mu_i$. Choosing a reference state for an element $i$ is arbitrary, however for solid elements it is typical to select the most stable configuration at standard conditions ($RT$) ($T = 298\,\text{K}, p = 1\,\text{bar}$).[68]
For solid bulk materials, the Gibbs free energy splits up into following energetic contributions:[69]

$$G = E^{\text{total}} + F^{\text{vib}} + F^{\text{conf}} + pV \tag{6.3}$$

With $E^{\text{total}}$ being the total internal energy, $F^{\text{vib}}$ the vibrational free energy, and $F^{\text{conf}}$ the configurational free energy.
The $pV$ term can typically be ignored, as even at high pressures the energetic values in a bulk are negligible.[70] The vibrational and configurational contributions can usually be neglected as well, as their magnitude is usually very small and the computational cost can be high.[69]
This leaves the total internal energy, which for metals can be approximated very well by Density Functional Theory (DFT) calculations, which we assume is close to our machine learned Gaussian Approximation Potential (see chapter 4).[71]

$$G^{\alpha}_{\text{bulk}} \approx E^{\alpha,\,\text{DFT}}_{\text{bulk}} \approx E^{\alpha,\,\text{GAP}}_{\text{bulk}} \tag{6.4}$$

To better compare the stability of systems among one another, we normalize the energies of bulk systems by the total number of atoms in the system $N^{\alpha}_{\text{tot}}$.[69] We denote these normalized energies by lower case letters.

$$g^{\alpha}_{\text{form}} = G^{\alpha}_{\text{form}} / N^{\alpha}_{\text{tot}} \tag{6.5}$$

This allows us to approximate the chemical potential of a solid element as:

$$\mu_i^{(RT)} = g_i^{(RT)} \approx e_i^{\text{GAP}}. \tag{6.6}$$

This means that the formation energy of the reference state (where $\alpha = \text{ref}$) is: $G^{\text{Ref.}}_{\text{form}} = 0$.
For materials where the set of constituent atoms contains more than one type of atom, formation energies are typically normed per formula unit $g^{\alpha,\,\text{f.u.}}_{\text{form}}$.[69]

## *Ab-initio* thermodynamics of interfaces

Similarly as above, we view the formation energy of a surface as an energy cost. Thus, we can define the formation energy of a surface (or interface) as the formation energy of a system $\alpha$ with said surface and the composition $(\{N_i^{\alpha}\})$, subtracted by the chemical potentials from its respective constituents $\mu_i$.[72]
In this context, the formation energy is frequently referred to as surface excess energy $G^{\alpha}_{\text{exc}}$.[73]

$$G^{\alpha}_{\text{exc}}(\{N_i^{\alpha}\}) = G^{\alpha}_{\text{surf}}(\{N_i^{\alpha}\}) - \sum_i N_i^{\alpha}\mu_i \tag{6.7}$$

The relevant normalization of the energetics is the surface area $A$:

$$g^{\alpha}_{\text{exc}} = G^{\alpha}_{\text{exc}} / A \tag{6.8}$$

### Symmetric Slab Setup

In pratice, we calculate the interfacial system using calculations of slabs, periodic in the $x, y$-vectors of the unit cell. In the $z$-vector we add a vacuum layer of at least 5 atomic layers. From this it follows that we are actually calculating two surfaces, a top layer and a bottom layer of the slab.
Should that setup exhibit an identical top layer $T$ and bottom layer $B$, the appropriate normalization in (6.8) is $2A$. We call such structures symmetric, we also sometimes refer to this as a "clean" slab.
In the case of a monoatomic metal, we proceed as follows. First, we obtain the GAP energy of our slab calculation $E_{\text{slab}}^{\text{clean, GAP}}$. Next, we calculate the bulk energy $E_{\text{bulk}}^{\text{clean, GAP}}$. We define the chemical potential per atom of this system as:

$$\mu_i = \frac{E_{\text{bulk}}^{\text{clean, GAP}}}{N^{\text{clean}}} \tag{6.9}$$

The sum in (6.9) vanishes due to there only being one atom type. We now invoke (6.7). The normed surface excess energy is simply $G_{\text{exc}}^{\text{clean}}$ divided by twice the surface area, as the slab contains two surfaces.

$$g_{\text{exc}}^{\text{clean}} = \frac{G_{\text{exc}}^{\text{clean}}}{2 \cdot A} = \frac{E_{\text{slab}}^{\text{clean, GAP}} - \sum_i N_i^{\text{clean}} \mu_i}{2 \cdot A} \tag{6.10}$$

### Asymmetric slab setup

In case the top layer $T$ differs in geometry and/or composition to our bottom layer $B$, calculating $g_{\text{exc}}^{T}$ is less straightforward.
In surface simulations the atomic positions of $B$ will be fixed, usually also some layers above as well. This makes it simpler to recreate the surface energy of $B$. In order to do this, we calculate the symmetric case, with both surfaces of the slab being of configuration $B$. Thus, we can obtain both slab and bulk energies of clean$= B$.

$$g_{\text{exc}}^{T} = \frac{G_{\text{exc}}^{T} - g_{\text{exc}}^{\text{clean}} \cdot A}{A} = \frac{E_{\text{slab}}^{T, \text{GAP}} - \sum_i N_i^{T} \mu_i}{A} - g_{\text{exc}}^{\text{clean}} \tag{6.11}$$

## Obtaining chemical potentials

As our study deals with exclusively with copper and oxygen, we will now derive the reference chemical potentials for these species.

### Copper $\mu_{\text{Cu}}$

We define the copper chemical potential $\mu_{\text{Cu}}$, as the energy of one copper atom in the bulk, a common reference state for bulk solid materials.[68] Table 6.1 shows the reference

structure parameters as well as the total energy results of structure relaxations with the Cu – O GAP and with DFT.

The calculation parameters for the DFT relaxations are the same as mentioned in section 3.1. For GAP relaxations we used the BFGS algorithm implemented in the Atomic Simulation Environment.[59] The convergence criterion was set to $0.01\,\mathrm{eV\,\AA^{-1}}$.

**Table 6.1** Inputs and energetic outputs of bulk and symmetric slab calculations for copper.

| | a [Å] | b [Å] | c [Å] | $\alpha$ | $\beta$ | $\gamma$ | $E^{\mathrm{GAP}}$[eV] | $E^{\mathrm{DFT}}$ [eV] |
|---|---|---|---|---|---|---|---|---|
| $Cu_{16(\mathrm{bulk})}$ | 5.12 | 5.12 | 8.88 | 90.0 | 73.2 | 60.0 | −87 899.710 | −87 899.894 |
| $Cu_{16(\mathrm{sym.\ slab})}$ | 5.12 | 5.12 | 26.41 | 90.0 | 90.0 | 120.0 | −87 896.114 | −87 896.128 |
| $Cu_{16(\mathrm{clean})}$ | 5.12 | 5.12 | 26.41 | 90.0 | 90.0 | 120.0 | −87 896.115 | −87 896.130 |

$$\begin{aligned}\frac{E^{\mathrm{DFT}}_{\mathrm{bulk}}}{N_{\mathrm{Cu}}} &= \frac{-87\,899.894\,\mathrm{eV}}{16} = -5493.743\,\mathrm{eV} = \mu^{\mathrm{DFT}}_{\mathrm{Cu}}\\ \frac{E^{\mathrm{GAP}}_{\mathrm{bulk}}}{N_{\mathrm{Cu}}} &= \frac{-87\,899.710\,\mathrm{eV}}{16} = -5493.732\,\mathrm{eV} = \mu^{\mathrm{GAP}}_{\mathrm{Cu}}\end{aligned} \tag{6.12}$$

Next, we calculate the area-normed surface excess energy of the unrelaxed $Cu_{16}$-slab. For this we use (6.10) and the chemical potential for copper obtained in (6.12):

$$\begin{aligned}g^{\text{bottom, DFT}}_{\mathrm{exc}} &= \frac{E^{\mathrm{DFT}}_{\mathrm{slab}} - N_{\mathrm{Cu}}\mu_{\mathrm{Cu}}}{2\cdot A} = \frac{3.76\,\mathrm{eV}}{2\cdot 22.740\,\mathrm{\AA}^2} = 82.8\,\mathrm{meV\,\AA^{-2}}\\ g^{\text{bottom, GAP}}_{\mathrm{exc}} &= \frac{E^{\mathrm{GAP}}_{\mathrm{slab}} - N_{\mathrm{Cu}}\mu_{\mathrm{Cu}}}{2\cdot A} = \frac{3.60\,\mathrm{eV}}{2\cdot 22.740\,\mathrm{\AA}^2} = 79.1\,\mathrm{meV\,\AA^{-2}}\end{aligned} \tag{6.13}$$

Finally, the excess surface energy of the relaxed "clean" Cu(111) surface is obtained by using the asymmetric formula (6.11).

$$\begin{aligned}g^{\text{clean, DFT}}_{\mathrm{exc}} &= \frac{E^{\mathrm{DFT}}_{\mathrm{clean}} - N_{\mathrm{Cu}}\mu_{\mathrm{Cu}}}{A} - g^{\text{bottom, DFT}}_{\mathrm{exc}} = 82.7\,\mathrm{meV\,\AA^{-2}}\\ g^{\text{clean, GAP}}_{\mathrm{exc}} &= \frac{E^{\mathrm{GAP}}_{\mathrm{clean}} - N_{\mathrm{Cu}}\mu_{\mathrm{Cu}}}{A} - g^{\text{bottom, GAP}}_{\mathrm{exc}} = 79.0\,\mathrm{meV\,\AA^{-2}}\end{aligned} \tag{6.14}$$

### Oxygen $\mu_{\mathrm{O}}$

We derive $\mu_{\mathrm{O}}$ from the formation energy of bulk copper oxides:

$$\mu_{\mathrm{O}} = \frac{E^{\mathrm{GAP}}_{\text{opt CuO}} - n_{\mathrm{Cu}}\cdot\mu_{\mathrm{Cu}}}{n_{\mathrm{O}}} \tag{6.15}$$

Therefore we relax the bulk oxides shown in table 6.2 with our Cu – O GAP and calculate $\mu_{\mathrm{O}}$.

**Table 6.2** Evaluation of $\mu_O$ for bulk copper oxides included in the Cu – O GAP.

| Bulk $Cu_xO_y$ polymorph | $E_{opt}^{GAP}[eV]$ | $n_{Cu}$ | $n_O$ | $\mu_O^{GAP}$ [eV] |
|---|---|---|---|---|
| $Cu_2O_{3cubic}$ | −98414.975 | 16 | 24 | −438.136 |
| $Cu_2O_{3monoclinic}$ | −98411.350 | 16 | 24 | −437.985 |
| $Cu_2O_{3orthorhombic}$ | −98409.150 | 16 | 24 | −437.893 |
| $Cu_3O_4$ | −72935.991 | 12 | 16 | −438.201 |
| $CuO_{tetragonal}$ | −94917.731 | 16 | 16 | −438.626 |
| $CuO_{monoclinic}$ | −94917.572 | 16 | 16 | −438.616 |
| $CuO_{orthorhombic}$ | −94917.525 | 16 | 16 | −438.613 |
| $Cu_4O_3$ | −93162.940 | 16 | 12 | −438.602 |
| $Cu_2O$ | −182817.521 | 32 | 16 | −438.631 |

From these $\mu_O$ we select the value of $Cu_2O$. For comparison, from a previous study we computed $\mu_O$ using DFT and a $Cu_2O$ reference, resulting in $\mu_O^{DFT} = -438.628\,eV$.

# Acknowledgments

There are many people whom I want to thank for supporting me throughout my Master Thesis.

First of all I would like to thank my supervisors Dr. Nicolas Hörmann and Prof. Karsten Reuter for welcoming me into the Theory Department and allowing me the freedom to figure things out by myself and being patient with me throughout the entire thesis.

I am especially grateful to Simeon Beinlich, M.Sc. for always taking the time to help me with any ssh or AiiDA trouble. Without his constant guidance this thesis would not have been possible.

I also thank all members of the FHI's Theory Department for the warm welcome in Berlin, and for the companionship in these last couple of months.

# Eigenständigkeitserklärung

Hiermit erkläre ich, die von mir eingereichte Arbeit selbständig erstellt und keine anderen als die von mir angegeben Quellen und Hilfsmittel benutzt zu haben.

Berlin, der 23.04.2021

Nicolas Bergmann