An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for T...

Nongnuch Artrith

Computational Materials Science 114 (2016) 135–150

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Editor's Choice

An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for TiO_2

Nongnuch Artrith [a,*], Alexander Urban [b,*]

[a] Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[b] Department of Materials Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Received 21 September 2015
Received in revised form 8 November 2015
Accepted 28 November 2015
Available online 4 January 2016

Keywords:
Machine learning
Artificial neural networks
Atomistic simulations
Titanium dioxide (TiO_2)
Behler–Parrinello

ABSTRACT

Machine learning interpolation of atomic potential energy surfaces enables the nearly automatic construction of highly accurate atomic interaction potentials. Here we discuss the Behler–Parrinello approach that is based on artificial neural networks (ANNs) and detail the implementation of the method in the free and open-source **a**tomic **e**nergy **net**work (`æne t`) package. The construction and application of ANN potentials using `æne t` is demonstrated at the example of titanium dioxide (TiO_2), an industrially relevant and well-studied material. We show that the accuracy of lattice parameters, energies, and bulk moduli predicted by the resulting TiO_2 ANN potential is excellent for the reference phases that were used in its construction (rutile, anatase, and brookite) and examine the potential's capabilities for the prediction of the high-pressure phases columbite (α-PbO_2 structure) and baddeleyite (ZrO_2 structure).

1. Introduction

During the past decade, atomistic simulations have become an integral part of materials design. This trend owes much to accurate *first principles* methods that are free from empirical parameters and applicable to all chemical species. Especially density-functional theory (DFT) has demonstrated predictive power for diverse materials properties, such as crystal phase stability, elastic constants, and transport phenomena to name just a few examples (see Ref. [1] for a recent review). However, the accuracy of DFT comes at the expense of great computational cost which presently limits the applicability of the method to structures containing a few hundred atoms when extensive sampling is required and to simulations of time scales on the order of several hundred picoseconds. In addition, the algorithmic scaling of common DFT implementations is of order $\mathcal{O}(N^3)$ or $\mathcal{O}(N\ln(N))$, where N is the number of electrons, and hence the system size accessible by DFT calculations does not grow as fast as the computational power.

These system-size and time-scale limitations can be overcome by turning to empirically parametrized atomistic potentials that approximate atomic interactions by simple analytic functions, often inspired by physical interaction laws [2–7]. The computational cost of most empirical potentials scales linearly with the number of atoms and is even for small structures several orders of magnitude lower than for first principles methods. However, empirical potentials rely on parameters that are typically adjusted to reproduce either experimental or first principles references, for instance elastic constants or total energies. Here, the number of parameters, i.e., the model complexity of empirical potentials, correlates with the range of structural situations (e.g., bulk structures, surfaces, and molecules) and chemistries (metallic, covalent and ionic systems) for which the potential is suitable. A Lennard–Jones potential that accurately describes the interaction of elemental argon atoms, for example, only requires two model parameters, while modern empirical potentials that are applicable to a wide range of different materials, for example ReaxFF or COMB [8–10], rely on about 30 different parameters. The careful parametrization of such sophisticated potentials can become challenging and has to be validated for each system.

A number of recently proposed methods circumvent the parametrization problem by employing machine learning techniques to interpolate the first principles potential energy surface defined by a set of reference calculations [11–15,90,91]. The oldest members of this new class of interaction potentials are the method by Behler and Parrinello that is based on artificial neural networks [11,79] and the Gaussian Approximation Potential (GAP) by Csányi and coworkers that employs Gaussian process regression [12,92]. Such machine-learning potentials abandon the requirement of a physical model in favor of purely mathematical representations with large but adjustable numbers of hundreds to thousands of

* Present address: Department of Materials Science and Engineering, University of California, Berkeley, CA 94720, USA.
E-mail addresses: nartrith@mit.edu (N. Artrith), alexurba@mit.edu (A. Urban).

http://dx.doi.org/10.1016/j.commatsci.2015.11.047
0927-0256/

parameters, depending on the system and potential type. On one hand, this approach has the advantage that machine learning regression can be well automated, and new potentials can thus be constructed with minimal human intervention. Additionally, the flexibility of their model function enables machine learning potentials to reach an accuracy that is comparable to the first-principles reference method used in their construction, and permits the application of the potentials to arbitrary structures and chemistries. On the other hand, the non-physical form of machine learning potentials hampers the direct interpretation of the model parameters and prevents the extrapolation to regions of the configurational space that are not well described by the reference data set [16]. However, the impact of these limitations has, in our opinion, not been comprehensively explored, which is mostly due to the absence of public software for fitting and for using machine learning potentials. While the original implementations of the method by Behler and coworkers [17] have been used in a growing number of applications, the software is, at the time of writing, not yet publicly available.

In view of this situation, we present here an implementation of Behler–Parrinello (BP) machine learning potentials [11,18,16]. The BP method uses feedforward artificial neural networks (ANNs) [19] for the representation of atomic energies as a function of the local structural environment. Our open source software package, the *atomic energy network* (`ænet`) code [20], comprises tools for the construction and the use of ANN potentials, and it is designed in a modular way, so that it can be extended by other machine learning potential methods in future. The `ænet` source code is written in modern Fortran 95/2003 and provides a C-compatible library for interoperability with other programming languages (e.g., C, C++ and Python), so that it can be easily interfaced with existing software, for example for Monte-Carlo or molecular dynamics simulations. We envision establishing a free and open platform for machine learning potentials. Note that `ænet` has already been used for relevant applications, namely for modeling electrocatalytic CuAu nanoparticles in water [21,22] and Cu-doped ceria nanoparticles [23].

In the following, we describe the BP method (Section 2) and its particular implementation in the `ænet` package (Section 3), and then demonstrate the construction of ANN potentials at the example of titanium dioxide (TiO_2) (Section 4). TiO_2 is an extremely versatile material with applications in diverse areas and has recently gained renewed attention as photocatalyst for the water splitting reaction [24–26]. For catalytic activity, defected crystals and nano structures are of particular importance, and the required length scales render the construction of realistic structure models purely with first principles methods challenging. Therefore, the TiO_2 potential described in this article is not just a showcase but the first step towards a general potential for non-ideal TiO_2 structures. The potential and the reference data set used in its construction are available along with the source code.

2. Machine learning potentials and the implementation in aenet

The conventional approach towards the construction of atomic interaction potentials begins with the identification of an adjustable functional form, an appropriate *model*, either by systematic approximation of first principles expressions or by physical intuition. Not every model is suitable for every application: For example, embedded atom model (EAM) potentials [6] are, *per their design*, well suited for the description of metallic solids, but less adequate for the description of the atomic interactions in, for example, molecules. Once a suitable model has been determined, its parameters can be obtained by adjustment to either experimentally measured quantities (e.g., elastic constants or formation energies) or energies and forces from first principles calculations. On one hand, the physics of a simple but sufficiently accurate model may be transparently analyzed. On the other hand, every potential constructed in this way is only valid within the bounds defined by its underlying model.

The general idea behind *machine learning* in this context is to automate the model discovery step: Instead of fitting the parameters of a predefined model to reference data, an appropriate model that is able to describe the feature-space of the input data shall be automatically determined and parametrized by the machine learning method. Potential fitting, i.e., the non-linear regression of the high-dimensional potential energy surface of atomic systems, is an example of a *supervised learning* problem, as the model *learns* the relationship between atomic structure and energy from a set of reference structures and energies. This approach is universally applicable to any kind of material, be it bulk metals or proteins. Once the model has successfully learned the structure-energy relationship, it can essentially be used as a *black box* to predict the energies of previously unknown structures, provided that the new structures are sufficiently similar to the ones in the trained reference set (Fig. 1). Since machine learning models are nevertheless differentiable mathematical functions, analytic expressions of the atomic forces can be obtained as negative gradient of the energy. Special care has to be taken to assess the transferability of the machine learning model to structures not included in the reference set.

Although machine learning techniques for regression and pattern recognition have been developed since the 1950s [27,28], applications to atomistic simulations are only just emerging. The particular method utilized by `ænet` belongs to the oldest and arguably best studied class of machine learning algorithms, the family of *artificial neural network* (ANN) algorithms [27,29,19]. In the following section we will therefore briefly lay out the general formalism of ANNs as implemented in `ænet` and atomistic potentials based on these. Further details regarding the actual implementation are given in the next Section 3. Topical reviews about applications of ANNs for atomistic simulations can be found in Refs. [30,31].

2.1. Artificial neural networks

The designation *artificial neural network* (ANN) for a certain class of mathematical functions is motivated by the similarity of their graph representations to the network of neurons in our brains. This analogy has led to biologically inspired terminology.

Biological neurons are interconnected in a network structure, where each neuron receives electrical input signals and transmits a response when the total input exceeds a certain activation threshold. Each individual neuron can thus be pictured as a signal processor with a Heaviside step function as *activation function*. ANNs essentially attempt to mimic biological neural networks by representing neurons by step-like signal processing functions. For numerical reasons, the discontinuity of the step function is avoided in practice, and sigmoidal functions, such as the hyperbolic tangent

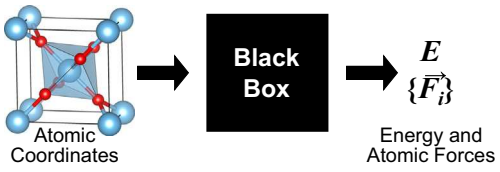

Fig. 1. Schematic of the machine-learning potential approach. In contrast to conventional atomic interaction potentials that are approximate descriptions of physical laws, machine learning potentials are *black boxes* whose internals have no direct physical interpretation.

or the logistic function, are employed as activation functions in ANNs. The activation function types supported by `ænet` are:

$$\begin{aligned}
&\text{linear function} && f_a^1(x) = x,\\
&\text{hyperbolic tangent} && f_a^2(x) = \tanh(x) = \tfrac{1-e^{-2x}}{1+e^{-2x}},\\
&\text{logistic function} && f_a^3(x) = \tfrac{1}{1+e^{-x}}, \quad \text{and}\\
&\text{tanh with linear twisting} && f_a^4(x) = 1.7159\,\tanh\left(\tfrac{2}{3}x\right) + ax,
\end{aligned} \tag{1}$$

where f_a^4 is recommended for general purposes in Ref. [19]. The activation functions and their derivatives (required, e.g., for the force evaluation) are shown in Fig. 2.

ANNs that connect input nodes with output nodes in a linear directed fashion, i.e., without cyclic connections between neurons, are called *feedforward* neural networks. In *multilayer perceptrons* (MLPs), the class of ANNs that is used by `ænet`, the artificial neurons are additionally organized in layers, and only neurons in adjacent layers are connected. The algebraic expression of the jth artificial neuron in the ith layer of an MLP with N layers is

$$x_{i,j}(\{x_{i-1,k}\}) = f_a^{i,j}\left(\sum_k w_{k,j}^i x_{i-1,k}\right) \quad \text{with} \quad i = 1,\ldots,(N-1), \tag{2}$$

where $\{x_{i-1,k}\}$ are the input signals from the previous layer $(i-1)$, $w_{k,j}^i$ is the weight of the signal from the kth neuron in layer $(i-1)$, and $f_a^{i,j}$ is the activation function. Equivalently, the vector of all neurons in layer i is given by

$$\mathbf{x}_i(\mathbf{x}_{i-1}) = f_a^i(\mathbf{W}_i\mathbf{x}_{i-1}) \quad \text{with} \quad i = 1,\ldots,(N-1), \tag{3}$$

and $\mathbf{W}_i$ is the *weight matrix* for signal transfer from layer $(i-1)$ to layer i with $(\mathbf{W}_i)_{k,j} = w_{k,j}^i$. In Eq. (3), the activation function is understood to act on each component of the argument vector.

In the graph representation of the MLP, the neurons are the nodes, and $w_{k,j}^i$ are the weights of their connections, i.e., the graph edges. An example MLP graph is shown in Fig. 3a, and the graph representation of a single neuron is shown in Fig. 3b. The algebraic expression (*network function*) of the MLP in Fig. 3a is

$$\mathcal{N}(\mathbf{x}_0;\{\mathbf{W}_i\}) = f_a^3\left\{\mathbf{W}_3 f_a^2\left[\mathbf{W}_2 f_a^1(\mathbf{W}_1\mathbf{x}_0)\right]\right\} = \mathbf{x}_3. \tag{4}$$

It is evident from Eq. (4) that vector $\mathbf{x}_0$ is the argument (*input layer*, blue nodes in Fig. 3a) of the network function $\mathcal{N}$, and $\mathbf{x}_3$ is the function value (*output layer*, green nodes in Fig. 3a). The layers between input and output layers (gray nodes in Fig. 3a) are called *hidden layers*, as they have no intuitive interpretation. The components of the weight matrices $\mathbf{W}_i$, the weight parameters, are the model parameters that have to be determined during the learning process. The number of layers in an MLP and the number of neurons per layer define the network *architecture*. The sample MLP in Fig. 3a has a 3-2-3-1 architecture. In general, the number of output nodes, i.e., the dimension of the network function, is arbitrary, however, for the representation of potential energy surfaces (the subject of

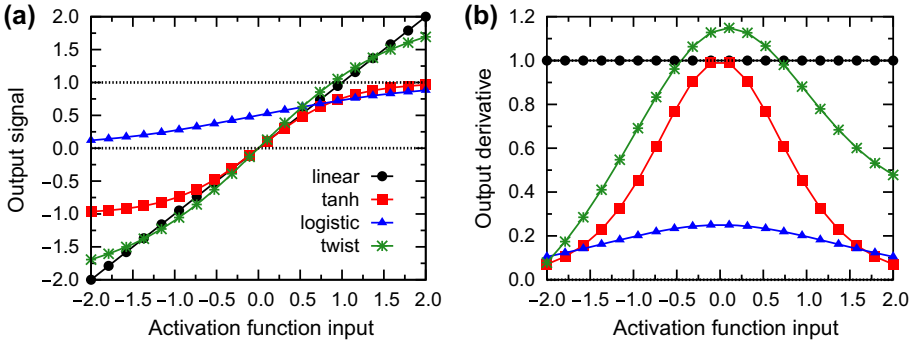

Fig. 2. Plot of the four activation functions of Eq. (1) that are currently available in `ænet`. (a) Function values (output signals) for input values between −2 and 2. (b) Function derivatives for the same input value range.

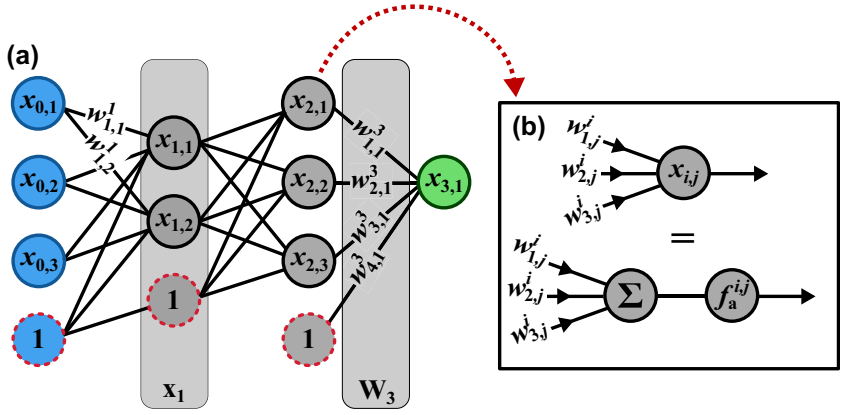

Fig. 3. (a) Graph representation of a multilayer perceptron artificial neural network. The nodes in the input layer (blue) represent the argument vector $\mathbf{x} = \mathbf{x}_0$ of the network function, and the nodes in the output layer (green) correspond to the function value $\mathbf{y} = \mathbf{x}_3$. Bias neurons are labeled with 1 and have a dotted red border. The nodes of the first network layer $\mathbf{x}_1$ and the edges corresponding to the third transfer matrix $\mathbf{W}_3$ are highlighted. (b) A single artificial neuron. Each node $x_{i,j}$ of the artificial neural network corresponds to a single artificial neuron that sums up input signals and transmits the output of an activation function $f_a^{i,j}$. Bias neurons transmit constant signals. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

this article) we only consider a scalar output corresponding to the potential energy.

Note that each layer of the MLP before the output layer contains one node that transmits a constant signal equal to 1. These *bias neurons* (depicted with dashed red borders in Fig. 3a) accommodate for a constant shift with the magnitude of their weight parameters. In addition, the activation function of the output layer is for our application always chosen to be the linear function, $f_a(x) = x$, so that the codomain of the network function is unrestricted, i.e., the potential energy can assume any arbitrary value.

Finally, we note that the signal propagation from one layer to another layer in MLPs, as implemented in `ænet`, simply is the matrix–vector multiplication of Eq. (3). Therefore, the evaluation of the network function of an MLP can be well vectorized on current SIMD (single instruction, multiple data) computer architectures and is highly efficient.

2.1.1. Batch training and online training

The weight parameters $\{\mathbf{W}_\ell\}$ for all network layers ℓ are determined during the *training* process. The regression of the potential energy surface is a *supervised learning* problem, i.e., the weight parameters ought to be adjusted such that a set of reference input–output samples $\{(\mathbf{x}_{0,n}, \mathbf{y}_n)\}$ is reproduced by the network function $\mathcal{N}$ as accurately as possible. Mathematically, this means the absolute value of the error e_n of each reference sample n

$$e_n(\mathbf{x}_{0,n}, \mathbf{y}_n; \{\mathbf{W}_\ell\}) = \mathcal{N}(\mathbf{x}_{0,n}; \{\mathbf{W}_\ell\}) - \mathbf{y}_n \tag{5}$$

has to be minimized. Hence, ANN training can be expressed as the high-dimensional optimization problem

$$\{\mathbf{W}_\ell^{\text{opt}}\} = \underset{\{\mathbf{W}_\ell\}}{\arg\min}\, \mathcal{E}(\{\mathbf{W}_\ell\}) \quad \text{with} \quad \mathcal{E}(\{\mathbf{W}_\ell\}) = \frac{1}{2}\sum_n^{\text{samples}} e_n^2, \tag{6}$$

where $\mathcal{E}$ is the quadratic ANN error function and $\{\mathbf{W}_\ell^{\text{opt}}\}$ is the set of optimal weight parameters.

In order to evaluate the complete error function $\mathcal{E}$ of Eq. (6), the error e_n of each sample n in the reference set has to be computed. Optimization methods that require the entire error function are called *batch training* methods. Alternatively, each reference sample can be presented separately to the ANN, and a weight correction can be estimated based on the error e_n of each individual reference sample (*online training*).

The training methods supported by `ænet` (see also Section 2.1.2) rely on the gradient of the error function $\mathcal{E}$ with respect to the weight parameters

$$\nabla\mathcal{E} = \mathbf{J}^T\mathbf{e} \quad \text{with} \quad (\mathbf{J})_{\alpha n} = \frac{\partial\mathcal{N}(\mathbf{x}_{0,n})}{\partial w_\alpha} \quad \text{and} \quad (\mathbf{e})_n = e_n \tag{7}$$

to estimate iterative weight updates. In Eq. (7) the Jacobian matrix, **J**, and the error vector, **e**, are introduced. Note that we enumerate the weight parameters of all layers with the unified index $\alpha = (i,j,k)$ to simplify the expression. The required derivatives of the network function are obtained by applying the chain rule of differentiation: For instance, the derivative of a neural output, Eq. (2), with respect to a weight in the previous layer is

$$\frac{\partial x_{i,j}}{\partial w^i_{m,j}} = f_a^{i,j\,(1)}\left(\sum_k w^i_{k,j} x_{i-1,k}\right)\cdot x_{i-1,m}, \tag{8}$$

where $f_a^{i,j\,(1)}$ is the first derivative of the activation function. Since the computation of the derivative of the network function requires descending from the output layer back to the input layer, this method is often called *error back-propagation* in the ANN literature [19].

2.1.2. Training methods

Currently, `ænet` provides three different training methods that can be used for the ANN weights optimization.

(i) *Gradient Descent (GD).* The simplest optimization algorithm for ANN training available in `ænet` is the *gradient descent* algorithm [29,19] that attempts the direct iterative minimization of the gradient of the error function, Eq. (7). At each I-th iteration of the GD method, the weight parameters are updated according to

$$\mathbf{w}^{(I+1)} = \mathbf{w}^{(I)} + \Delta\mathbf{w}^{\text{GD},(I+1)} \quad \text{with} \quad \Delta\mathbf{w}^{\text{GD},(I+1)} = -\gamma\nabla\mathcal{E}, \tag{9}$$

where $\mathbf{w}^{(I)}$ is the vector of all weight parameters, γ is an empirical *learning rate*, and the gradient of the error function, $\nabla\mathcal{E}$, is given in Eq. (7). However, more commonly, the GD method is used for online training, i.e., the weight parameters are updated according to the error of each separate sample n in the reference set, which is for each weight w_α

$$\Delta w_{\alpha,n}^{\text{OGD},(I+1)} = -\gamma\,\frac{\partial}{\partial w_\alpha}\left(\frac{1}{2}e_n^2\right) = -\gamma\, e_n \frac{\partial\mathcal{N}(\mathbf{x}_{0,n})}{\partial w_\alpha}. \tag{10}$$

Convergence of the online GD algorithm can be improved by introducing a *momentum* term, scaled by a parameter β, to introduce at each iteration a contribution from the previous weight update (I) [19]

$$\Delta w_{\alpha,n}^{\text{OGD},(I+1)} = -\gamma e_n \frac{\partial\mathcal{N}(\mathbf{x}_{0,n})}{\partial w_\alpha} + \beta\Delta w_{\alpha,n}^{\text{OGD},(I)} \tag{11}$$

to reduce fluctuations in the weight update.

(ii) *Limited-memory BFGS (L-BFGS).* *Quasi-Newton* methods are based on a second order expansion of the objective function and the iterative weight updates take the form

$$\Delta\mathbf{w}^{\text{QN},(I+1)} = -\left(\mathbf{H}^{(I)}\right)^{-1}\nabla\mathcal{E}^{(I)}, \tag{12}$$

where **H** is an approximation to the Hessian matrix, i.e., the second derivatives of the network function with respect to the weights. A popular quasi-Newton method that is also implemented in `ænet` is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method in which **H** is updated at each iteration [32–35]. For the training of ANNs with a large number of weight parameters, the *limited-memory BFGS* method is particularly useful, as it does not require to store the entire Hessian matrix [36].

(iii) *Levenberg–Marquardt (LM).* The *Levenberg–Marquardt* method [37–39,29] is another standard method for least squares fitting commonly used for ANN training that is supported by `ænet`. The LM weights update vector is given by

$$\Delta\mathbf{w}^{\text{LM},(I+1)} = -\left(\mathbf{J}^{T,(I)}\mathbf{J}^{(I)} + \lambda\mathbf{I}\right)^{-1}\mathbf{J}^{T,(I)}\mathbf{e}^{(I)}, \tag{13}$$

where **J** is the Jacobian matrix of Eq. (7), **I** is the identity matrix, and λ is a reciprocal learning rate. Note the similarity of the LM weight update to the quasi-Newton expression of Eq. (12), as $\mathbf{J}^T\mathbf{J}$ is the Gauss–Newton approximation for the Hessian matrix. For $\lambda = 0$ the method is identical to the Gauss–Newton algorithm, but for large λ it approaches the GD algorithm with a small learning rate. The idea behind the LM method is to start off with the more robust gradient descent method, but to switch to the faster convergent Gauss–Newton method as soon as the weights are sufficiently close to the optimum. To achieve this, the value of λ is *decreased* after each iteration that resulted in an error reduction, and λ is *increased* when the weight update would otherwise increase the error (the multiples used for this adjustment are method parameters).

The LM algorithm is implemented in `ænet` as batch training method including all reference samples, and as a variant using *mini batches* containing a subset of the samples.

2.2. Atomistic potentials based on high-dimensional ANNs

The idea to employ ANNs for the description of atomic interactions has been investigated since the early 1990s [40–43,30,31]. Early approaches have in common that ANNs were trained to represent the structural energy $E(\sigma)$ of an atomic configuration $\sigma = \{\mathbf{R}_i\}$, using directly the Cartesian atomic coordinates $\{\mathbf{R}_i\}$ as inputs for the ANN

$$E(\sigma) \approx E^{\text{ANN}}(\sigma) = \mathcal{N}(\{\mathbf{R}_i\}). \tag{14}$$

As a consequence of this technique, the number of ANN input nodes depended on the number of atoms in the structure, resulting in highly specialized potentials that were not transferable to systems with different numbers of atoms.

Recent machine-learning potential methods [11,12,14] have overcome this limitation by partitioning the total energy $E(\sigma)$ into atomic contributions

$$E(\sigma) = \sum_i^{\text{atoms}} E_i(\sigma) \approx \sum_i^{\text{atoms}} E_i(\sigma_i), \tag{15}$$

where $\sigma_i \subset \sigma$ only depends on the coordinates of atoms within a cutoff radius R_c around atom i. In other words, σ_i describes the *local structural environment* around atom i. Machine-learning approaches are then used to derive a model for the atomic energy E_i instead of the total structural energy.

To be useful as a universal atomistic potential, σ_i has to be represented in a set of coordinates that is invariant with respect to translation, rotation, and exchange of equivalent atoms. We indicate the transformation from Cartesian coordinates $\{\mathbf{R}_i\}$ to invariant coordinates by introducing the *fingerprint* function $\mathcal{F}$ with $\widetilde{\sigma}_i \equiv \mathcal{F}(\sigma_i)$. With this definition, the atomic energy in ANN potentials becomes

$$E_i^{\text{ANN}}(\sigma_i) = \mathcal{N}[\mathcal{F}(\sigma_i)] = \mathcal{N}(\widetilde{\sigma}_i), \tag{16}$$

i.e., the invariant coordinates in $\widetilde{\sigma}_i$ are the ANN input nodes (see Section 2.1). The total structural energy is approximated as sum of the atomic contributions

$$E(\sigma) \approx E^{\text{ANN}}(\sigma) = \sum_i^{\text{atoms}} E_i^{\text{ANN}}(\sigma_i). \tag{17}$$

2.2.1. Representations of the local structural environment

Several schemes for the invariant representation of the local structural environment have been suggested in the literature [45,93–96], and a recent review can be found in Ref. [44]. In their original work on ANN potentials, Behler and Parrinello (BP) introduced an invariant basis set of radial and angular *symmetry functions* [45] that has since been used for the construction of various ANN potentials [11,46–52,21,22]. Fundamentally, the BP symmetry functions are a set of atom-centered radial and angular coordination functions. The radial functions

$$G_i^{\text{r}}(\sigma_i) = \sum_{j\neq i}^{\text{neighbors}} g^{\text{r}}(R_{ij}) \quad \text{with} \quad R_{ij} = |\mathbf{R}_j - \mathbf{R}_i| \tag{18}$$

map the distribution of distances $R_{ij} = |\mathbf{R}_j - \mathbf{R}_i|$ between a central atom i and its neighbors j within a certain cutoff radius. In the same manner, the angular functions depend on three atomic coordinates $\mathbf{R}_i, \mathbf{R}_j$, and $\mathbf{R}_k$

$$G_i^{\text{a}}(\sigma_i) = \sum_{k\neq j\neq i}^{\text{neighbors}} g^{\text{a}}(\theta_{ijk}) \quad \text{with} \quad \theta_{ijk} = \angle(\mathbf{R}_j - \mathbf{R}_i, \mathbf{R}_k - \mathbf{R}_i) \tag{19}$$

and essentially represent the distribution of bond angles θ_{ijk}. A set of radial and angular functions with different parameters is used as basis for the representation of the local structural environment. Behler suggested several different choices for the radial and angular kernels g^{r} and g^{a} of Eqs. (18) and (19) that can be found in Ref. [45], and all of them are available in `ænet`.

The BP approach requires to evaluate the symmetry function values for each combination of atomic species separately (e.g., for a binary system with species A and B the interactions A–A, B–B, and A–B are treated individually), and hence the number of basis functions scales with the number of species. Since the number of basis functions dictates the number of input nodes of the ANN which in turn determines the computational cost of the evaluation of the network function, such growth of dimensionality imposes an upper limit on the number of species that can be described with the ANN potential. So far, the greatest number of different atom types reported for an ANN potential with BP basis is four, which was required for the representation of Cu–Au nanoalloys in liquid water (species: Au, Cu, O, and H) [21].

2.2.2. ANN potential training

The partitioning of the total energy into atomic contributions, Eq. (17), introduces an additional layer of complexity as compared to standard ANNs. Unlike in the case of regular ANN training, it is not possible to fit the weight parameters directly to input/output reference samples, as the atomic energies are not known *a priori*. Therefore, the training of high-dimensional ANN potentials has to occur simultaneously for all atoms in a structure, and the quadratic error function that is to be minimized becomes

$$\begin{aligned}\mathcal{E}(\{\mathbf{W}_\ell\}) &= \frac{1}{2}\sum_\sigma^{\text{structures}} \left[E^{\text{ANN}}(\sigma;\{\mathbf{W}_\ell\}) - E_{\text{ref}}^\sigma\right]^2 \\ &= \frac{1}{2}\sum_\sigma^{\text{structures}} \left[\sum_i^{\text{atoms}} E_i^{\text{ANN}}(\sigma;\{\mathbf{W}_\ell\}) - E_{\text{ref}}^\sigma\right]^2,\end{aligned} \tag{20}$$

where E_{ref}^σ is the reference energy of structure σ. However, the evaluation of the gradient of the error function

$$\frac{\partial \mathcal{E}}{\partial w_\alpha} = \sum_\sigma^{\text{structures}} \left[\sum_i^{\text{atoms}} E_i^{\text{ANN}}(\sigma;\{\mathbf{W}_\ell\}) - E_{\text{ref}}^\sigma\right] \sum_i^{\text{atoms}} \frac{\partial E_i^{\text{ANN}}(\sigma;\{\mathbf{W}_\ell\})}{\partial w_\alpha} \tag{21}$$

only requires the derivatives of the individual atomic ANNs which can be obtained through standard back-propagation (Section 2.1.1). Once again, $\alpha = (i,j,k)$ is a unified weight index.

The training progress with optimization iterations is typically monitored through the evolution of the *root mean squared error* (RMSE) and the *mean absolute error* (MAE) of the reference structures

$$\begin{aligned}\text{RMSE} &= \sqrt{\frac{1}{N}\sum_\sigma^{\text{structures}} \left[E^{\text{ANN}}(\sigma) - E_{\text{ref}}^\sigma\right]^2}, \\ \text{MAE} &= \frac{1}{N}\sum_\sigma^{\text{structures}} \left|E^{\text{ANN}}(\sigma) - E_{\text{ref}}^\sigma\right|,\end{aligned} \tag{22}$$

where N is the total number of reference structures.

To estimate the convergence of the training process, it is useful to split the set of reference structures into a *training set* that is actually used for ANN potential training and an independent *testing set* that is exclusively used as a measure for the accuracy of the potential. The testing set can also be used as a proxy to detect *overfitting*, i.e., the situation when the model reproduces the reference points including their statistical errors and random noise. Overfitting is indicated by the divergence of the RMSE of the training and testing sets, as schematically shown in Fig. 4, and training should be aborted when this point has been reached (*early stopping*).

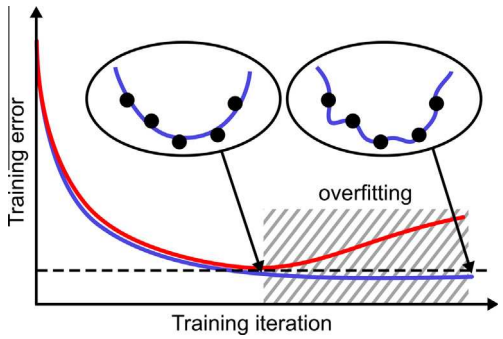

Fig. 4. Schematic of the evolution of the ANN error function of the training and testing sets during the ANN training. The blue and red curves represent the mean errors of the training and testing sets, respectively. Overfitting occurs when training and testing set errors diverge (hatched region). The insets show schematically how overfitting affects the potential energy surface (points: reference data; lines: ANN function). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3. Implementation details: the aenet code

We have implemented the high-dimensional ANN potential approach outlined in Section 2.2 in the **a**tomic **e**nergy **net**work (`ænet`) package. `ænet` provides tools for both, the construction and application of ANN potentials. The `ænet` source code is open and can be freely obtained from http://ann.atomistic.net. The core of the package has been implemented in modern Fortran 95/2003, and ANN potential training and evaluation are parallelized for the efficient use on compute clusters and supercomputers. The individual `ænet` modules can be used independently. A flow chart of the module hierarchy of the `ænet` package is shown in Fig. 5.

Potential construction with `ænet` is broken down into two separate tasks (yellow[1] and green regions in Fig. 5): (i) the compilation of reference structures and energies into a single collection, the training set file, using the tool `generate.x`, and (ii) the actual training of the atomic energy ANNs using a second tool, `train.x`.

Simulations based on existing ANN potentials are facilitated by the `ænetLib` library (blue region in Fig. 5), which provides bindings for Fortran and C. Note that C libraries are interoperable with many other programming languages, such as C++, Python, or Java. `ænetLib` provides routines for parsing ANN potential files and for the evaluation of the energies and internal forces of atomic configurations. Part of the `ænet` package are two sample tools that interface with `ænetLib`, `predict.x` and `predict.py`, showcasing implementations in Fortran and Python, respectively.

3.1. Construction of ANN potentials with `ænet`

The first step for the construction of an ANN potential is the selection of suitable reference structures and the computation of their energy, e.g., with first principles methods. This process is independent of `ænet` and is discussed for the example of TiO_2 in Section 4.1, for which we used DFT reference calculations based on the Quantum ESPRESSO package [53].

For the input and output of atomic structure information, `ænet` employs a subset of the XCrySDen Structure Format (XSF) [54]. The open and well-documented XSF format has the advantage that it can be generated and visualized by various common programs, such as VMD [55], Vesta [56,57], ASE [58], and of course XCrysDen [59,60]. The structural energy required for the ANN potential fit is included in the XSF file as a comment, so that the file remains valid XSF format.

[1] For interpretation of color in Fig. 5, the reader is referred to the web version of this article.

Provided a set of reference structures and energies, the remaining steps for the ANN potential construction with `ænet` are the transformation of the Cartesian atomic coordinates to an invariant basis representation (see Section 2.2.1) and the actual training of the ANNs.

3.1.1. Training set generation with `generate.x`

Before the actual ANN training, the Cartesian atomic coordinates of the structures in the reference set need to be transformed to an invariant, atom-centered basis, as described in Section 2.2.1. This is the purpose of the tool `generate.x`, which iterates over a list of reference structures and transforms each structure's coordinates using the method specified in the input file.

For each atomic species in the reference set, `generate.x` requires the definition of a *structural fingerprint* setup that defines the representation methods and all required parameters for the coordinate transform. Currently, `ænet` supports the BP *symmetry functions* of Section 2.2.1 as invariant basis, but the code is designed such that implementing additional methods is straightforward.

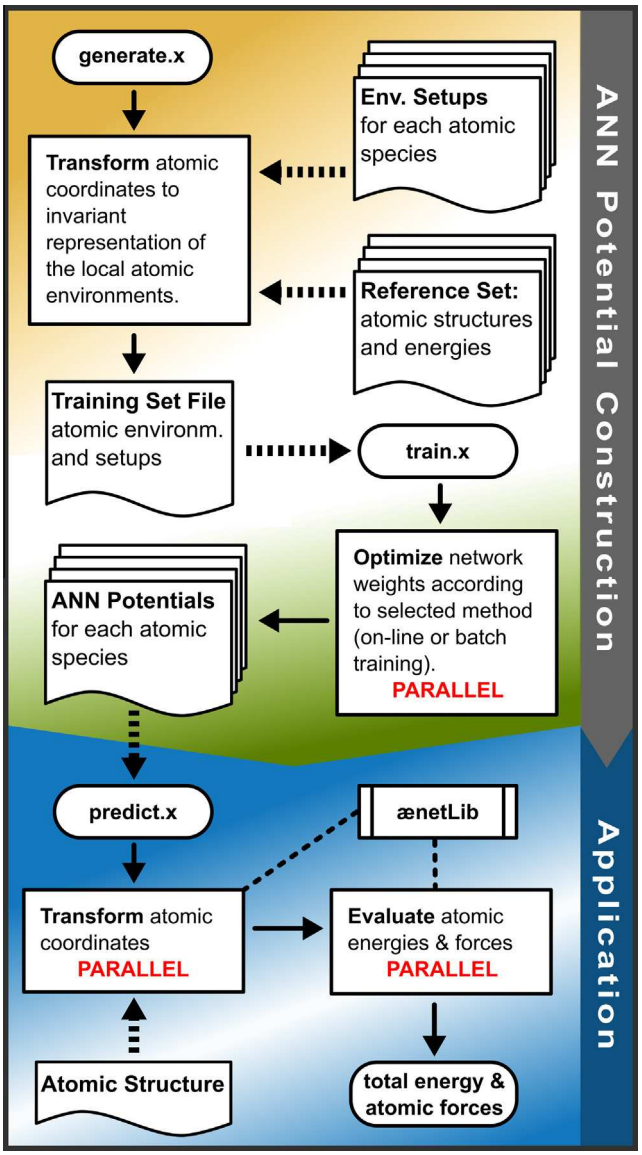

Fig. 5. Flow chart of the module hierarchy of the `ænet` package. The tool `generate.x` is used to bundle reference structures in training set files that are subsequently used as input for ANN training with `train.x`. The final ANN potentials can be used either directly with the `predict.x` tool, or in third-party programs by interfacing with the `ænetLib` library.

Provided a principle input file containing the paths to all reference structures and all required structural fingerprint setups, `generate.x` will generate a training set file that can subsequently be used for the training of ANN potentials with `train.x`.

3.1.2. ANN potential training with `train.x`

Once a training set file has been generated, ANN potentials can be fitted using `train.x`. The user has to specify ANN architectures for all chemical species and activation function types for the hidden network layers (see Section 2.1 for the functions supported by `ænet`).

For the actual weight optimization, `train.x` provides the training algorithms of Section 2.1.2. Note that the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm is made available through a third-party numerical library [61,62].

For batch training, the errors of all reference samples within the batch have to be computed before the neural weight update can be determined (Section 2.1.1). Since the sample errors are independent, this process can be well parallelized. Batch training with `train.x` uses the message passing interface (MPI) to enable parallel batch training on hundreds of compute cores. The parallel efficiency of batch training with `ænet` is discussed for the example of TiO_2 in Section 4.5.

To accelerate the convergence of the weight optimization, `train.x` shifts the mean input and output values of the reference set to 0 and scales all values such that their standard deviation is 1 using the normalization method described in Ref. [19]. The initial weights are drawn from a uniform random distribution in such a way that the output values of each network layer also follow a zero-centered distribution with a standard deviation of 1 [19].

A parameter to be defined by the user is the size of the testing set (see Section 2.2.2). By default, `train.x` randomly selects 10% of the reference structures for testing. After each training iteration, `train.x` reports the MAE and the RMSE, Eq. (22), of the samples in the training and testing sets.

3.2. Prediction of structural energies and atomic forces

Once an ANN potential has been constructed using `generate.x` and `train.x`, it can be used in the same way as conventional atomistic potentials to predict structural energies and, via their analytic gradients, the atomic forces. In practice, the potential itself is only one ingredient for atomistic simulations, and the implementation of sampling techniques, such as molecular dynamics (MD) or Monte–Carlo (MC) methods, can be sophisticated as well. To utilize existing simulation software, `ænet` can be compiled as a library with Fortran and C bindings, `ænetLib`, which provides Fortran subroutines and C functions for the evaluation of atomic energies and forces from atomic coordinates.

To showcase the `ænetLib` API, a Fortran tool (`predict.x`) and a Python tool (`predict.py`) are included in the `ænet` package. Both tools interface with `ænetLib` to predict the energy and atomic forces of input structures, however, their implementations are quite different: `predict.x` utilizes routines from the `ænet` code for the particle neighbor list and for MPI parallelization to demonstrate the concurrency of the ANN potential evaluation, for which each atom can essentially be treated separately (see Section 4.5 for a discussion of the parallel scaling). In contrast, `predict.py` has been implemented as a *Calculator* class for the Atomic Simulation Environment (ASE) Python toolkit [58]. This approach makes it possible to utilize ASE's vast functionality, such as MD, nudged elastic band (NEB) calculations, and geometry optimization, for simulations based on `ænet` ANN potentials.

4. An ANN potential for bulk TiO_2

The versatility of titanium dioxide (TiO_2) makes it an ideal benchmark material to demonstrate the features of the `ænet` package and the construction of ANN potentials. Industrial applications of TiO_2 range from the use as a pigment in paints [63] to the role of photoactive component in sunscreens [64]. The oxide is also one of the most widely used photocatalysts, especially for energy related applications, such as photocatalytic water splitting [24,26]. TiO_2 has been extensively studied both, experimentally and computationally (see Refs. [25,65–67] for recent reviews), and this vast experience allows us to draw confidence in computational predictions.

TiO_2 occurs naturally in three polymorphs, rutile (the most common phase), anatase, and brookite (Fig. 6a–c). In addition, a large number of synthetic TiO_2 polymorphs is known (see Ref. [67] and references therein). Apart from the three natural polymorphs, we consider in this work two high-pressure TiO_2 phases,

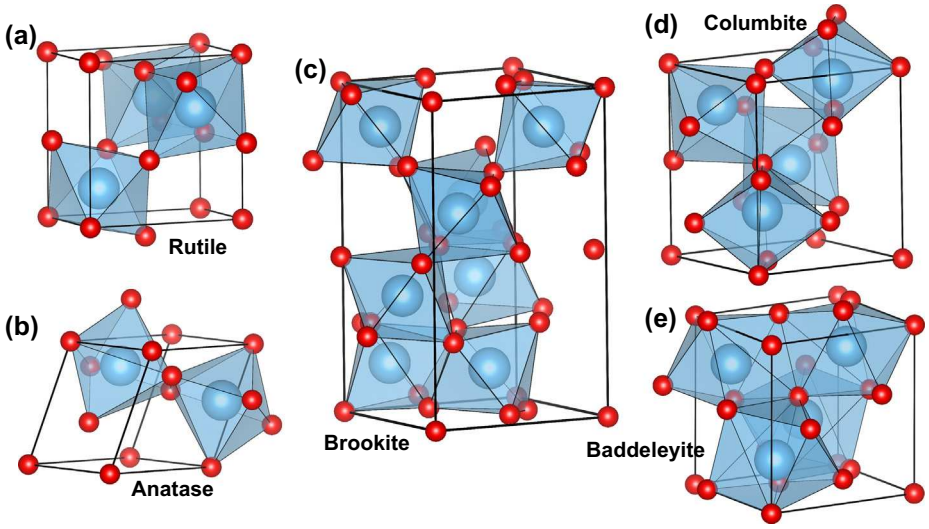

Fig. 6. Unit cells of the five TiO_2 polymorphs discussed in this work, space groups in parentheses: (a) rutile ($P4_2/mnm$), (b) anatase ($I4_1/amd$), (c) brookite (Pbca), (d) columbite (α-PbO_2 structure, Pbcn), and (e) baddeleyite (ZrO_2 structure, $P2_1/c$). Small red spheres indicate oxygen atoms and large blue spheres are titanium atoms. The local oxygen coordination environment of each titanium site is highlighted with a blue polyhedron. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

columbite and baddeleyite (Fig. 6d–e) to benchmark the transferability of the ANN potential [68].

Several empirical atomic interaction potentials for TiO_2 have been developed since the 1990s, including rigid-ion models [69,70] and polarizable models [71,72]. A discussion of the accuracy and shortcomings of those potentials can be found in Ref. [73]. Recently, Kim et al. reported a ReaxFF potential for the TiO_2/water system, achieving good accuracy for bulk TiO_2 phases. However, the largest body of computational work related to TiO_2 is based on *first principles* DFT, which allows the prediction of many of the material's properties, such as crystal shapes, surface reconstructions, and defect formation energies [67]. We note that DFT incorrectly predicts anatase to be more stable than rutile at standard conditions [74,75]. Conesa attributed this discrepancy to the lack of van-der-Waals interactions on this level of theory and proposed a semi-empirical correction [76]. Arroyo-de Dompablo et al. argue that the self-interaction error of DFT is the origin of the incorrect phase energetics and propose a DFT+U correction [77]. However, for properties that do not depend on the relative phase stability of anatase and rutile, DFT has been found to be generally reliable [67], and in this work we therefore consider DFT without any empirical correction terms as reference for the construction of an ANN potential.

Despite the great scientific efforts of the previous decade, important questions regarding the structure and composition of reduced TiO_2 [78] and nano-scale effects [80] have so far eluded any detailed computational investigation, as it is challenging to capture the properties of those systems with conventional empirical potentials, and the required length scales are out of reach for DFT. An accurate ANN TiO_2 potential will be able to bridge the gap between accurate first principles calculations and computationally efficient atomistic potentials. In the remaining part of this article, we demonstrate the construction of an ANN potential for TiO_2 based on DFT reference calculations. As an additional source of information, we note that the construction of ANN potentials has recently been reviewed by Behler [79].

On a technical note, transition metal oxides exhibit strong charge transfer from the metal to the oxygen atoms, which gives rise to localized charges on atoms of both species. In conventional empirical potentials, the long ranged electrostatic interaction between charged species is usually treated separately according to Coulomb's law. It is possible to do the same within the ANN potential method [48], however, it was found in previous benchmarks for zinc oxide supported copper clusters [51] and protonated water clusters [81] that electrostatic interactions can, with great accuracy, be captured in an effectively screened way by short-ranged ANN potentials. Since we are in the present work interested in the condensed phases of TiO_2, we will follow this insight and omit an explicit electrostatic interaction term as well.

4.1. TiO_2 reference data set

The quality of an ANN potential critically depends on the selection of reference structures and energies. For the present work, structural energies were obtained from DFT calculations using the PBE exchange–correlation functional [82] as implemented in PWSCF of the Quantum ESPRESSO package [53]. Wave functions and electron densities were represented in plane wave basis sets with energy cutoffs of 40 Ry and 200 Ry, respectively, using GBRV ultrasoft pseudopotentials for the atomic core regions [83]. For the Brillouin zone integration, all calculations employed gamma-centered k-point meshes with a density corresponding to $8 \times 8 \times 8$ k-points for the primitive rutile unit cell with 6 atoms. SCF energies were generally converged to 1.0×10^{-6} Ry and the convergence threshold for variable-cell optimizations was 0.5 kbar.

An initial set of reference structures for the potential construction was generated by distorting ideal rutile, anatase, and brookite structures in three systematic ways: (i) the lattice constants of the three crystal structures were scaled to a range of ±10% around the ground state, (ii) the crystal cells were gradually tilted by applying volume-conserving monoclinic strain (see inset of Fig. 14a), and (iii) the structures were stretched (compressed) by tensile (compressive) orthorhombic strain in one crystal dimension (see inset of 14b). In addition, supercell structures with oxygen vacancies were also included in the initial set of reference structures. Subsequently, a preliminary potential, fit against this initial data set, was used to generate additional reference structures by short molecular dynamics simulations at various temperatures. This process, i.e., fitting a preliminary potential and using it for the generation of additional reference structures, was repeated until the newly generated structures were already found to be accurately represented by the potential thereby confirming its predictive power. A schematic of this iterative refinement process is shown in Fig. 7.

Note that the DFT calculations for the initial generation and refinement of the reference data set can be automated to a large extend, for example using a high-throughput framework, such as AFLOW [84], Python Materials Genomics [85], or AiiDA [86].

For the present work, the reference set was restricted to periodic bulk structures. As a consequence of the interpolating nature of the method, the constructed TiO_2 ANN potential can therefore not be expected to describe surface models (slab structures) or isolated particles accurately. Note, however, that ANN potentials have previously been successfully used for the description of surfaces, interfaces, and nanostructures by including relevant reference structures in the training set [50,51,21].

The final reference data set comprises a total of 7694 structures containing between 6 and 95 atoms, from which 10% were randomly selected as independent testing set that was not used for the potential fit. This *testing set* allows to assess the transferability of the potential and makes it possible to detect overfitting, as described in Section 2.2.2. The distribution of formation energies in the training and testing sets is similar, as seen in Fig. 8, indicating that the testing set adequately represents the structures in the reference set. Also note that, although a small percentage of strongly distorted structures in the reference set give rise to a large energy range of ~3 eV/atom, the majority of reference structures are within a range of 0.5 eV/atom from the ground state.

4.2. Structural fingerprint setup for TiO_2

The structural fingerprint setup used for the description of the local atomic environment in this work comprises 8 radial and 18 angular Behler–Parrinello basis functions (Section 2.2.1) for each combination of atomic species. This means, the dimension of the input layer of each atomic ANN potential is 70, as the input nodes of the ANN potentials for the energy of oxygen and titanium atoms, respectively, are the values of each 8 radial functions for interactions with O and Ti atoms and 18 angular functions for the interactions with O–O, O–Ti, and Ti–Ti atom pairs ($2 \times 8 + 3 \times 18 = 70$).

In this work, we chose radial basis functions of type G^2 and angular basis functions of type G^4, following Behler's original notation [45]. The radial function centered at atom i is defined as

$$G_i^2 = \sum_{j \neq i} e^{-\eta (R_{ij} - R_s)^2} \cdot f_c(R_{ij}), \tag{23}$$

where R_{ij} is the distance between atoms i and j, η and R_s are adjustable parameters. The sum in Eq. (23) runs over all atoms within the radius R_c, and a smooth truncation is achieved using a cosine cutoff function

$$f_c(R_{ij}) = \begin{cases} 0.5\left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right] & \text{for } R_{ij} \leqslant R_c, \\ 0 & \text{for } R_{ij} > R_c. \end{cases} \tag{24}$$

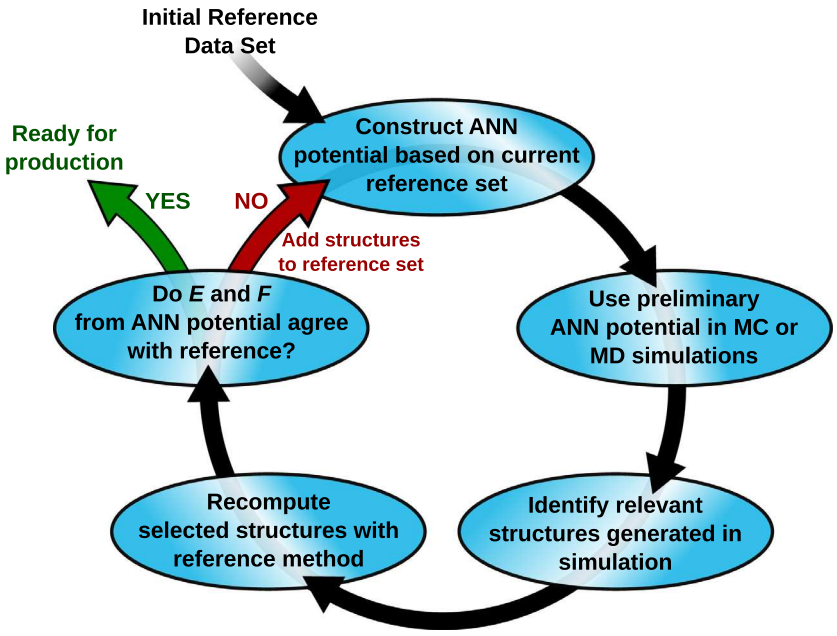

Fig. 7. Flow chart of the iterative refinement used to generate the reference data set. Preliminary ANN potentials are used to generate additional reference samples until the ANN potential energy of the new samples is in agreement with DFT.

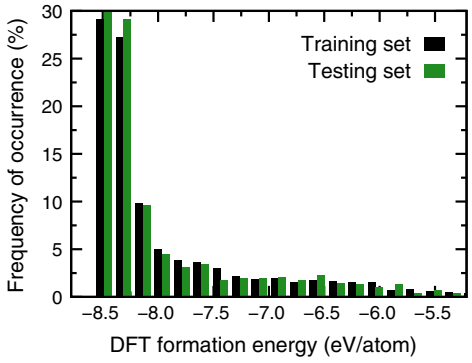

Fig. 8. Distribution of formation energies of all structures in the training (black) and testing (green) sets used for the fit of the TiO_2 ANN potential. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The angular three-body function centered at atom i is given by

$$G_i^4 = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} (1 + \lambda \cos\theta_{ijk})^{\zeta} \cdot e^{-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)} \cdot f_c(R_{ij}) \cdot f_c(R_{ik}) \cdot f_c(R_{jk}), \quad (25)$$

where R_{ij}, R_{ik}, R_{jk} are the interatomic distances between atoms i, j, and k, θ_{ijk} is the angle defined by the three atoms, and ζ, λ, and η are parameters. All basis function parameters of the present fingerprint setup are listed in Tables 1 and 2.

4.3. Selecting the best network architecture

The balance between model complexity and transferability (predictive power) of an ANN potential is controlled by its architecture, i.e., by the number of nodes per network layer (see also Section 2.1). If the network architecture offers too few model parameters, the ANN will not be flexible enough to represent the reference structures adequately. In our experience, too few nodes also tend to result in rougher ANN potentials, leading to numerical instabilities when the potential is used in simulations. On the other hand, architectures with more model parameters than required result in an unnecessary overhead in training time and may lead to overfitting and thus reduced transferability.

Fig. 9a shows the RMSE of the ANN potential energy with respect to the DFT energies of the structures in the testing set after 1000 batch L-BFGS training iterations (red squares) and after 100 LM iterations (black circles) for various ANN architectures. Note that the LM method usually converges more rapidly with the number of training iterations but is computationally more demanding,

Table 1
Parameters of the 2×8 G^2 type radial basis functions of Eq. (23) used for the description of the local structural environment within a cutoff radius $R_c = 6.5$ Å. The shift parameter R_s is 0 for all basis functions. The functions come in pairs, as each one is required for neighboring Ti and O atoms, respectively.

No.	η (Å^{-2})	No.	η (Å^{-2})
1–2	0.003214	9–10	0.214264
3–4	0.035711	11–12	0.357106
5–6	0.071421	13–14	0.714213
7–8	0.124987	15–16	1.428426

Table 2
Parameters of the 3×18 angular G^4 type basis functions of Eq. (25) used for the description of the local structural environment within a cutoff radius $R_c = 6.5$ Å. Each set of parameters listed in the table corresponds to 3 equivalent functions for the 3 possible species combinations Ti–Ti, O–O, and Ti–O.

No.	η (Å^{-2})	λ	ζ	No.	η (Å^{-2})	λ	ζ
17–19	0.000357	−1.0	1.0	44–46	0.000357	1.0	1.0
20–22	0.028569	−1.0	1.0	47–49	0.028569	1.0	1.0
23–25	0.089277	−1.0	1.0	50–52	0.089277	1.0	1.0
26–28	0.000357	−1.0	2.0	53–55	0.000357	1.0	2.0
29–31	0.028569	−1.0	2.0	56–58	0.028569	1.0	2.0
32–34	0.089277	−1.0	2.0	59–61	0.089277	1.0	2.0
35–37	0.000357	−1.0	4.0	62–64	0.000357	1.0	4.0
38–40	0.028569	−1.0	4.0	65–67	0.028569	1.0	4.0
41–43	0.089277	−1.0	4.0	68–70	0.089277	1.0	4.0

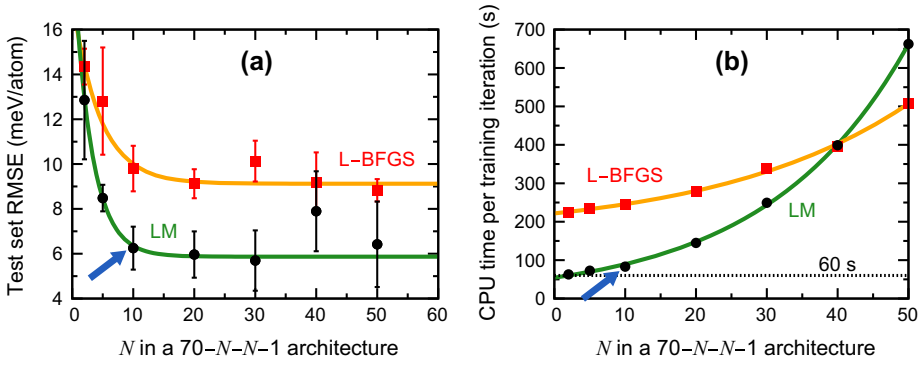

Fig. 9. (a) Root mean squared error of the ANN energy for the structures in the testing set after 1000 L-BFGS (red squares) and 100 LM (black circles) training iterations, respectively, as function of the ANN architecture. The data points are the mean of 7 independent training runs, and the standard deviation is shown as error bars. (b) Average CPU time per training iteration as function of the ANN architecture for 7694 reference structures using the L-BFGS method (red squares) and the LM method (black circles). The solid green and yellow curves in both graphs are exponential fits. The blue arrows indicate the architecture that was eventually selected. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

so that the L-BFGS method may in practice be faster. To ease the comparison, the graph only shows data for 70-*N*-*N*-1 ANN architectures, i.e., ANNs possessing two hidden layers with each N nodes and $N = 2, 5, 10, 20, 30, 40$, and 50. Note that the number of weight parameters, i.e., optimization parameters, for these architectures is given by $(70+1)N + (N+1)N + (N+1)\cdot 1 = N^2 + 73N + 1$, where one additional weight per layer is due to the bias nodes (see also Fig. 3). For small N (2–10 nodes per hidden layer), the RMSE decreases rapidly with increasing number of nodes, however, for $N > 10$ the improvement is minor. Additionally, for larger ANN sizes the standard deviation of the RMSE values increases (error bars in Fig. 9a), indicating either overfitting or insufficient training iterations.

The time required per training iteration is shown in Fig. 9b as function of the ANN size. Each training method has a constant computational base requirement that is independent of the network architecture. Note that this constant is particularly large for the L-BFGS method in Fig. 9b, as the training was run in parallel on 64 cores which introduces an additional overhead. Up to $N = 10$ the network architecture does not significantly contribute to the training time, i.e., the ANN evaluation is fast compared to the other computational operations. For larger ANN sizes the CPU time quickly rises.

Based on the above assessment, we settled on the 70-10-10-1 ANN architecture (831 weight parameters), since larger ANNs do not deliver significantly lower training errors while being computationally significantly more demanding.

4.4. Selecting a training method

The three training methods that are currently available in `æneto` (see Sections 2.1.2 and 3.1.2) are each suitable for different situations: The online gradient descent (GD) method is computationally least demanding and can be used with any ANN architecture, but the algorithm cannot be well parallelized. Due to its second order corrections, the Levenberg–Marquardt (LM) method is for small ANN architectures often more efficient than the online gradient descent method. However, the dimension of the Hessian matrix that has to be inverted at each LM step is determined by the number of weight parameters and reference structures, so that the LM method becomes computationally expensive for larger ANN architectures and training sets. Finally, the batch (limited memory) L-BFGS method can be well parallelized in reference structure space and scales linearly with the number of reference structures. It is therefore the method of choice for large reference sets and large ANN architectures.

Fig. 10a shows the RMSE of the testing set during ANN potential training using the training runs for the three different approaches that resulted in the smallest error after 500 iterations. For each of the training methods, we benchmarked the three different

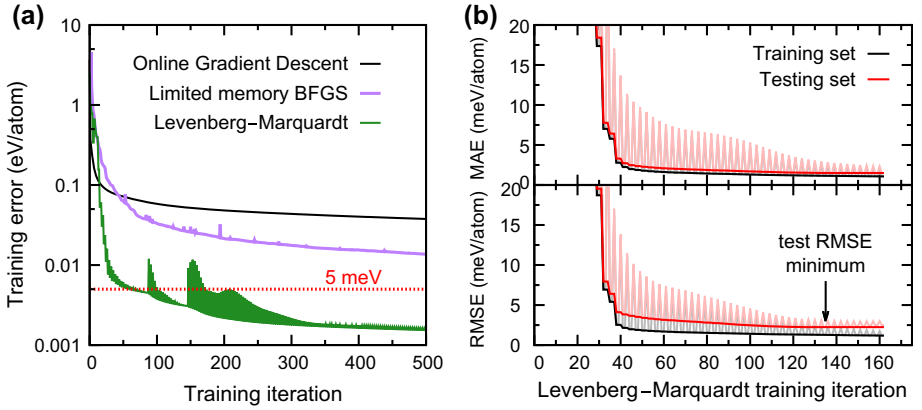

Fig. 10. (a) Evolution of the root mean squared error (RMSE) of the testing set during the training of a 70-10-10-1 ANN potential using the online gradient descent method (black line), the limited memory BFGS method (purple line), and the Levenberg–Marquardt method (green line). The target error of 5 meV is indicated by a red dotted line. (b) Convergence of the mean absolute error (MAE, top) and RMSE (bottom) of the training (black) and testing (red) sets during weight optimization for an ANN with a 70-40-40-1 architecture. The Levenberg–Marquardt algorithm results in an oscillating error function. The lower hull is highlighted for clarity. The arrow indicates the minimum of the testing set RMSE. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sigmoidal activation functions available in `ænet` (see Section 2.1). In general, we found that the hyperbolic tangent and the twisted hyperbolic tangent resulted in the best convergence behavior. The training runs depicted in Fig. 10a employed hyperbolic tangent activation functions (for online GD and L-BFGS) and the twisted function (for LM). The relatively small 70-10-10-1 architecture of the TiO_2 ANN potential and the small training set size of less than 8000 structures allows efficient training with the LM method, which converges most rapidly in the example. Our target accuracy of better than 5 meV/atom for the ANN potential is shown as dotted red line in Fig. 10a. The LM method reaches this threshold after less than 100 iterations while the other methods result in RMSEs above 10 meV/atom even after 1000 iterations. Note that the number of training iterations does not directly translate to compute time, since the algorithms have different computational requirements. However, for the particular dimensions of the ANN architecture (see Section 4.3) and the training set in this work, the LM algorithm is the most efficient training method.

Fig. 10b shows the training and testing RMSE and mean absolute error (MAE) during training of a 70-40-40-1 ANN potential with the LM method to illustrate the early stopping technique of Section 2.2.2. As seen in the figure, the testing set RMSE reaches its minimum value after 135 optimization iterations, and only the training set RMSE decreases afterward, indicating overfitting. Therefore, the optimal ANN weights are obtained after 135 iterations and the weights resulting from later iterations should not be used.

4.5. Scaling

The parallel scaling efficiency of batch training with `ænet` depends on the size of the reference data set, as all reference structures are distributed equally over the available processes. Hence, the larger the training set, the more cores may be efficiently used for training. For the case of the 7694-structure TiO_2 reference set, the scaling is shown in Fig. 11a. Even for this modestly sized training set, training on 100 cores achieves around 80% parallel efficiency on a supercomputer.

The parallel implementation of the ANN potential evaluation works by distributing the structure's atoms over all processes. Hence, the scaling is optimal when the number of atoms is a multiple of the number of cores. For a TiO_2 bulk structure with one million atoms we find that the evaluation of energies and atomic forces achieves greater than 90% parallel efficiency on up to 64 cores on a current supercomputer (TACC Stampede). For larger numbers of cores the scaling performance is hampered by inter-process communication.

Arguably, the most important advantage of ANN potentials over first principles methods is the scaling behavior with system size. Due to the decomposition of the total energy into atomic contributions, Eq. (17), the computational effort formally scales linearly with the number of atoms. Fig. 11b shows that, using our current implementation, the compute time per atom on a standard desktop computer is indeed independent of the system size for structures with up to one million atoms.

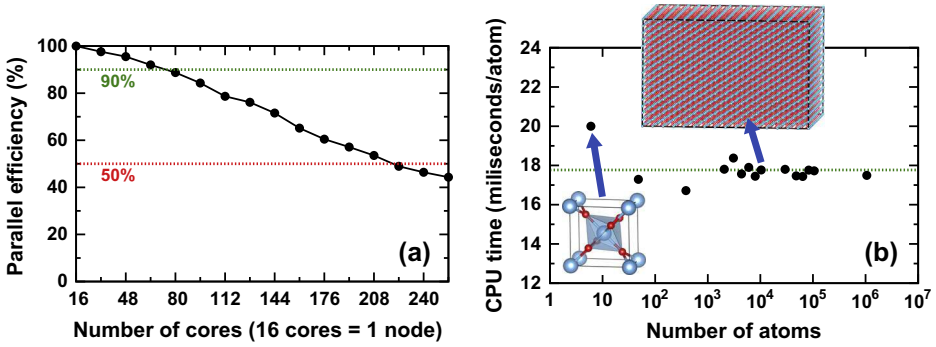

Fig. 11. (a) Parallel scaling of the weight optimization on up to 256 cores on TACC Stampede (batch BFGS, 7694 reference structures). On 64 cores a parallel efficiency greater than 90% is achieved (dotted green line). The parallel efficiency drops below 50% for more than 208 cores (dotted red line). (b) ANN potential evaluation time per atom (using `predict.x`) as function of the number of atoms. The green dotted line indicates the mean. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

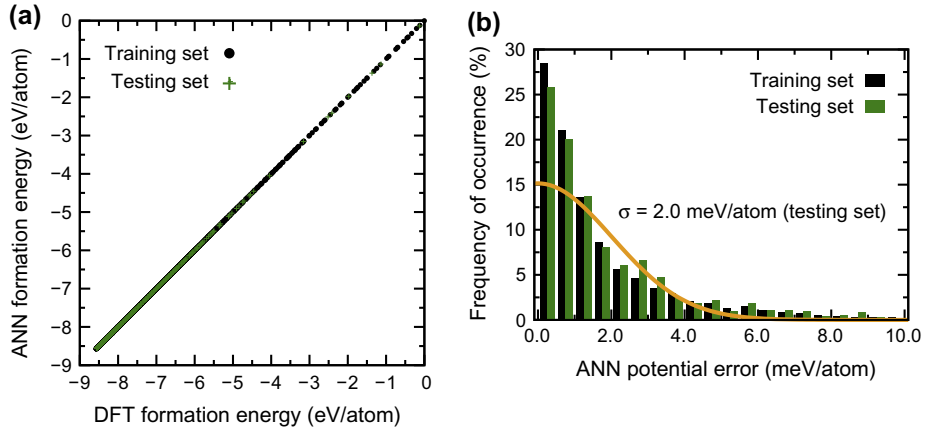

Fig. 12. (a) Correlation of the ANN potential formation energies with the corresponding DFT reference energies for all structures in the training (black dots) and testing (green crosses) sets. (b) Distribution of absolute ANN potential errors for all structures in the training (black) and testing (green) sets. A normal distribution fitted to the testing set errors is shown as yellow line (standard deviation σ). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.6. Accuracy of the TiO_2 ANN potential

Fig. 12a shows the correlation of the ANN potential energies of all structures in the training and testing sets with their corresponding DFT reference energies. Since the $y = x$ diagonal corresponds to a perfect fit, clearly, structures throughout the entire energy range are well captured by the ANN potential. The distribution of absolute errors (shown in Fig. 12b) indicates that more than 95% of the testing set structures are predicted with an error of less than ±4 meV/atom ($= 2\sigma$).

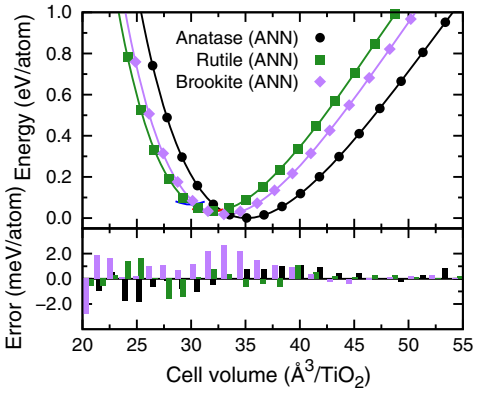

Fig. 13. Energy per atom as function of the cell volume for the three most stable TiO_2 crystal structures, anatase (black circles), rutile (green squares), and brookite (purple diamonds). Top: Symbols correspond to the ANN potential energies and lines interpolate the DFT reference energies. Bottom: Difference of ANN potential energy and DFT reference. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Although, these error statistics are encouraging, for practical purposes it is crucial that the relative energies of similar structures are reproduced correctly. If the structural errors on the order of 4 meV/atom were completely stochastic, the ANN potential energy surface would be rugged and unsuitable for simulations that require continuous energies, such as molecular dynamics simulations. However, as can be seen in Fig. 13, the ANN potential smoothly reproduces the energy as function of the lattice volume of rutile, anatase, and brookite. Note that the errors in the structural energies (bottom panel of Fig. 13) vary continuously, so that the relative errors of related structures are even smaller than expected based on the overall accuracy. A similar behavior is observed for the energy upon cell deformations. Fig. 14a and Fig. 14b depict the change of the structural energy with applied monoclinic (a) and orthorhombic (b) strain, respectively (the brookite structure is omitted for clarity, as its energy changes on a far greater scale).

Table 3 summarizes the relative phase energies, unit cell volumes, and bulk moduli as obtained by a fit of the Birch–Murnaghan equation of states [87] to the DFT and ANN potential energy curves. For the three reference TiO_2 phases rutile, anatase, and brookite used in the ANN potential construction the agreement between the values predicted by the ANN potential and their DFT references is excellent and close to the numerical precision of the fit. In addition to those three phases, Table 3 also includes the columbite (α-PbO_2) and baddeleyite (ZrO_2) high-pressure phases that were not included in the training set. These phases allow us to assess the transferability of the potential to unknown crystal phases. While the columbite crystal structure exhibits similar structural motifs to anatase and rutile, the baddeleyite structure is far denser and each titanium atom is sevenfold coordinated by oxygen (as opposed to the octahedral coordination in the other phases).

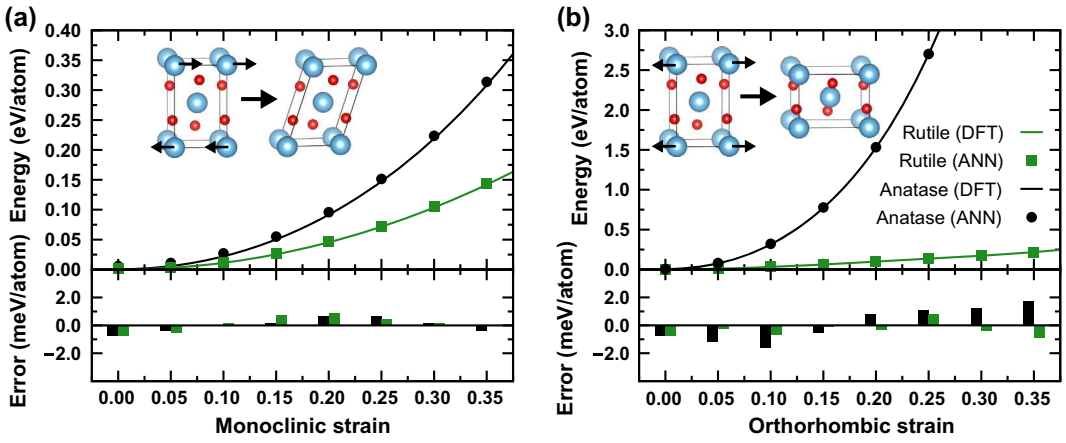

Fig. 14. (a) Energy change per atom upon volume conserving monoclinic strain acting on rutile (black) and anatase (green); comparison of the ANN potential prediction (symbols) with the DFT references (solid lines). The lower panel depicts the absolute difference of the ANN potential energy and the DFT reference. (b) The same for volume conserving orthorhombic strain. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Relative energies (E_0), unit cell volumes (V_0) per atom, and bulk moduli (B_0) of different TiO_2 phases as computed using density-functional theory (DFT) and the ANN potential of this work. The difference between the ANN potential and DFT is given in parentheses next to the ANN values.

Phase	DFT			ANN potential		
	E_0 (eV/atom)	V_0 (Å^3/atom)	B_0 (GPa)	E_0 (eV/atom)	V_0 (Å^3/atom)	B_0 (GPa)
Anatase ($I4_1/amd$)	0.000	11.71	211	0.000 (0.000)	11.70 (0.01)	211 (0)
Rutile ($P4_2/mnm$)	0.033	10.68	235	0.032 (−0.001)	10.68 (0.00)	234 (−1)
Brookite (Pbca)	0.016	11.01	225	0.017 (0.001)	11.01 (0.00)	223 (−2)
Columbite (Pbcn)	0.029	10.50	230	0.065 (0.036)	10.48 (−0.02)	218 (−12)
Baddeleyite ($P2_1/c$)	0.066	9.96	240	0.118 (0.052)	10.01 (0.05)	289 (49)

Nevertheless, the unit cell volumes and bulk moduli predicted by the ANN potential agree well with DFT, culminating in a peak error of around 0.5% in the volume and 20% in the bulk modulus of baddeleyite.

The relative phase energies of columbite and baddeleyite predicted by the ANN potential are greater than the DFT reference energies, however, as discussed above, DFT itself suffers from inaccuracies in the relative phase energies when the atomic density of the compared phases is significantly different. As shown in Fig. 15, the ANN potential energy of the two phases is essentially shifted, and the relative accuracy within a given phase is much higher than the absolute error seems to imply. This quick sanity check demonstrates that, despite the limited extrapolation capabilities of ANN potentials, the TiO_2 potential based on the rutile, anatase and brookite phases would likely be useful for the prediction of other TiO_2 crystal structures. We also stress that the ANN potential could be improved for additional crystal phases in a straightforward fashion by adding relevant structures to the reference set and continuing the ANN potential training.

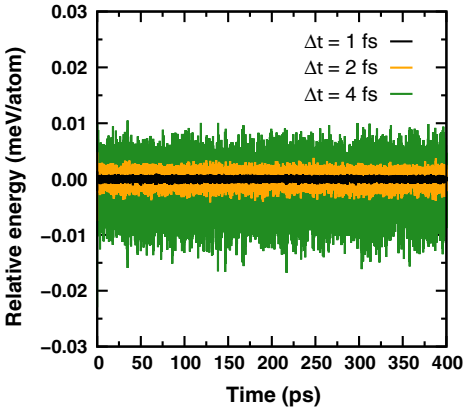

Fig. 16. Energy fluctuations during microcanonical MD simulations of rutile (16 TiO_2 formula units) using time steps of 1 fs (black line), 2 fs (yellow line), and 4 fs (green line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.7. Reliability of the ANN potential for MD simulations

As a final benchmark, we assess the reliability of the TiO_2 ANN potential during molecular dynamics (MD) simulations. For MD simulations it is especially important that the potential provides a smooth, continuous energy surface to facilitate the numerical integration of the equation of motion. Fig. 16 shows the total energy during MD simulations of a rutile supercell with 16 TiO_2 formula units, carried out with the ASE [58] interface to `ænetLib`, over 400 ps using time steps of 1 fs, 2 fs, and 4 fs. The atomic velocities were initialized with a Maxwell–Boltzmann distribution for a temperature of 1000 K. As seen in the figure, even with the largest time step the energy is well conserved and fluctuations do not exceed 0.01 meV/atom. Another indication for the numerical stability is the absence of an energy drift, i.e., the mean change of the total energy over the entire trajectory is zero.

To check how well the ANN potential performs in MD simulations at elevated temperatures, we carried out a short MD simulation of an anatase cell with 54 TiO_2 formula units (162 atoms) at 500 K over 100 ps with a time step of 1 fs. For this check the TINKER package [88] was interfaced with `ænetLib`, and the canonical statistical ensemble was imposed with a Bussi–Parrinello thermostat [89]. After every 1000th MD step of the resulting 100,000 steps trajectory, the atomic positions were extracted and recomputed with DFT, providing a total of 100 structures that were not included in the training set. Fig. 17 shows the ANN energies of these 100 structures along the MD trajectory in comparison to their DFT reference. Clearly, the ANN potential captures the course of the energy well, and no energy deviates by more than 4 meV/atom from the DFT value. This is especially remarkable as the sampled structures are on average around 50–60 meV/atom higher in energy than the anatase ground state.

To achieve physical dynamics, MD simulations require reasonably accurate atomic forces. Since the present ANN potential was not fitted against the DFT atomic forces, the statistical error in the atomic forces has to be expected to be larger than the error in the structural energy. Fig. 18a shows the correlation of the magnitude of the atomic forces predicted by the ANN potential and DFT for the same 100 structures as shown in Fig. 17. While some scattering is evident, the ANN forces strongly correlate with their DFT

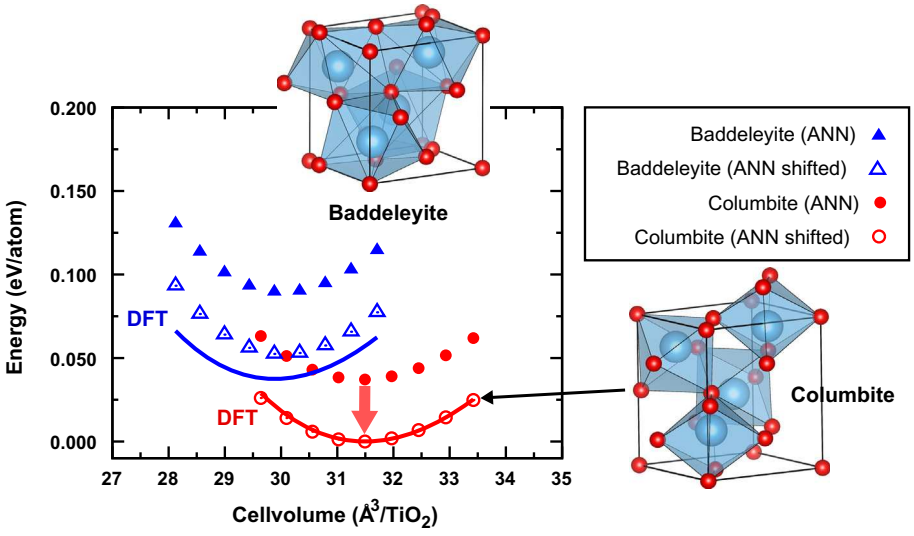

Fig. 15. (a) Energy per atom as function of the cell volume for the two high-pressure phases that were not included in the training set: Columbite (red squares and red line) and baddeleyite (blue triangles and blue line). Filled symbols correspond to the ANN potential energies and lines interpolate the DFT reference energies. Empty symbols are the ANN energies corrected by the columbite error at the minimum volume. All energies are relative. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

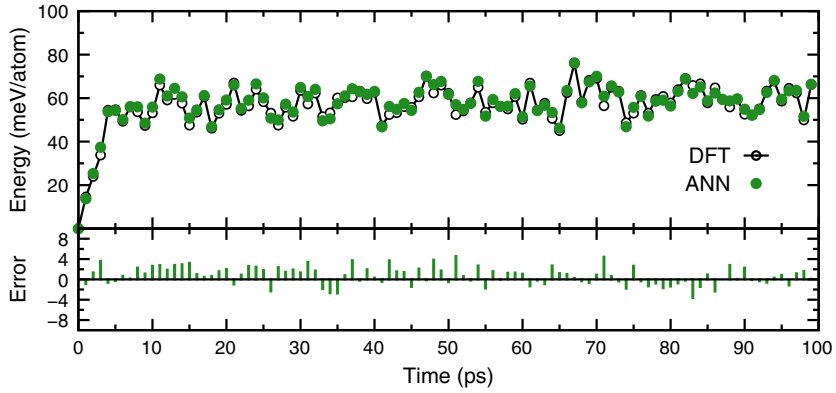

Fig. 17. Comparison of the total energies of structures along a molecular dynamics (MD) trajectory. The upper panel shows the DFT (empty circles) and ANN (green filled circles) energies of 100 structures that were selected from a 100 ps MD trajectory every 1000 steps using a time step of 1 fs. The difference between the ANN prediction and the DFT reference energy is shown in the lower panel (also in units of meV/atom). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

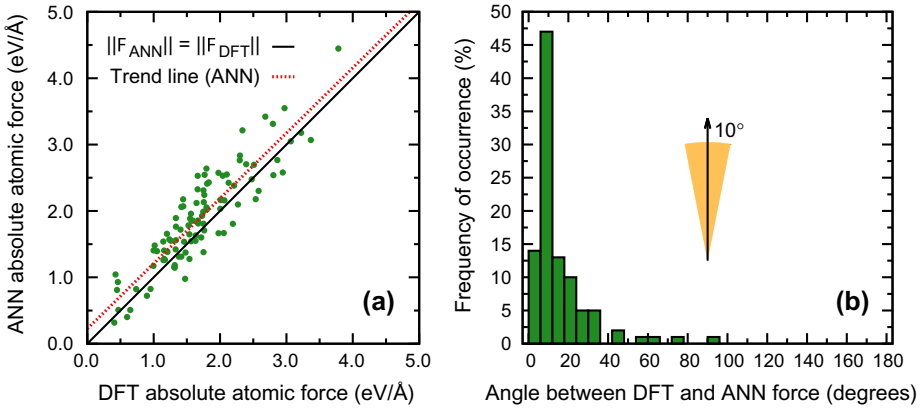

Fig. 18. (a) Correlation of the absolute value of the force acting on the first atom of the structures of Fig. 17 as predicted by the ANN potential with the DFT reference (green filled circles). The solid black line indicates perfect correlation, and the red dotted line is a linear fit to the actual data points. (b) Histogram of the absolute directional errors in the ANN prediction of the atomic force. The majority of ANN force vectors are within 10° of their DFT references; this *trust region* is depicted as yellow area in the inset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

references and are generally within 0.5 eV/Å of the DFT values. The histogram in Fig. 18b further confirms that the direction of the majority of the ANN force vectors is within 10° of the DFT reference.

5. Summary and perspective

Machine learning techniques offer an exciting new avenue towards the construction of accurate atomic interaction potentials. In this article we introduced the free and open-source atomic energy network (`ænet`) package for the construction and application of artificial neural network (ANN) potentials. With `ænet`, we envisage to encourage the exploration of ANN potentials as alternative to conventional atomistic force fields. We are releasing the source code under an open license in the hope to stimulate contributions from other researchers, not only to improve the `ænet` software itself, but also to advance the ANN potential method in general. The development of `ænet` is ongoing, and a number of features, such as using the atomic forces for ANN potential training and providing alternative structural fingerprint methods, are planned for future releases. However, even in its current state `ænet` is fully functional, as demonstrated herein at the example of an ANN potential for TiO_2 which shows promising performance for TiO_2 crystal phases. The interpolating nature of the method prevents the application of ANN potentials to systems that are very different from the reference structures used in the potential training. As a consequence, the present TiO_2 potential cannot be used to study structures containing atoms with lower coordination, such as surfaces and particles. In future work, we therefore plan to extend the TiO_2 potential to further classes of structures to apply it to actual research questions.

Acknowledgments

NA is indebted to Jörg Behler at RUB for introducing the ANN potential method and for providing the opportunity to work on the development of the method and its applications to large realistic simulations of complex materials. NA further thanks Alexie M. Kolpak for the discussions during the time at MIT. AU thanks Gerbrand Ceder for helpful discussions. The authors also appreciate the stimulating exchange with the members of the THEOS group at EPFL. Computational resources from the Extreme Science and Engineering Discovery Environment (XSEDE) are gratefully acknowledged.

References

[1] K. Burke, J. Chem. Phys. 136 (2012) 150901, http://dx.doi.org/10.1063/1.4704546.
[2] M.W. Finnis, J.E. Sinclair, Philos. Mag. A 50 (1984) 45–55, http://dx.doi.org/10.1080/01418618408244210.

[3] F.H. Stillinger, T.A. Weber, Phys. Rev. B 31 (1985) 5262–5271, http://dx.doi.org/10.1103/PhysRevB.31.5262.
[4] J. Tersoff, Phys. Rev. B 37 (1988) 6991–7000, http://dx.doi.org/10.1103/PhysRevB.37.6991.
[5] D.W. Brenner, Phys. Rev. B 42 (1990) 9458–9471, http://dx.doi.org/10.1103/PhysRevB.42.9458.
[6] M.S. Daw, S.M. Foiles, M.I. Baskes, Mater. Sci. Rep. 9 (1993) 251–310, http://dx.doi.org/10.1016/0920-2307(93)90001-U.
[7] S.J. Stuart, A.B. Tutein, J.A. Harrison, J. Chem. Phys. 112 (2000) 6472, http://dx.doi.org/10.1063/1.481208.
[8] A.C.T. van Duin, S. Dasgupta, F. Lorant, W.A. Goddard, J. Phys. Chem. A 105 (2001) 9396–9409, http://dx.doi.org/10.1021/jp004368u.
[9] T.-R. Shan, B.D. Devine, T.W. Kemper, S.B. Sinnott, S.R. Phillpot, Phys. Rev. B 81 (2010) 125328, http://dx.doi.org/10.1103/physrevb.81.125328.
[10] T. Liang, Y.K. Shin, Y.-T. Cheng, D.E. Yilmaz, K.G. Vishnu, O. Verners, C. Zou, S.R. Phillpot, S.B. Sinnott, A.C. van Duin, Annu. Rev. Mater. Res. 43 (2013) 109–129, http://dx.doi.org/10.1146/annurev-matsci-071312-121610.
[11] J. Behler, M. Parrinello, Phys. Rev. Lett. 98 (2007) 146401, http://dx.doi.org/10.1103/PhysRevLett.98.146401.
[12] A.P. Bartók, M.C. Payne, R. Kondor, G. Csányi, Phys. Rev. Lett. 104 (2010) 136403, http://dx.doi.org/10.1103/PhysRevLett.104.136403.
[13] M. Rupp, A. Tkatchenko, K.-R. Müller, O.A. von Lilienfeld, Phys. Rev. Lett. 108 (2012) 058301, http://dx.doi.org/10.1103/PhysRevLett.108.058301.
[14] A. Thompson, L. Swiler, C. Trott, S. Foiles, G. Tucker, J. Comput. Phys. 285 (2015) 316–330, http://dx.doi.org/10.1016/j.jcp.2014.12.018.
[15] C.M. Handley, J. Behler, Eur. Phys. J. B 87 (2014) 152, http://dx.doi.org/10.1140/epjb/e2014-50070-0.
[16] J. Behler, Phys. Chem. Chem. Phys. 13 (2011) 17930–17955, http://dx.doi.org/10.1039/C1CP21668F.
[17] J. Behler, The Runner Code, 2007–2015. <http://www.theochem.rub.de/~joerg.behler/runner.htm>.
[18] J. Behler, R. Martonak, D. Donadio, M. Parrinello, Phys. Status Solidi B 245 (2008) 2618–2629, http://dx.doi.org/10.1002/pssb.200844219.
[19] G. Montavon, G.B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, second ed., vol. 7700, Springer, Berlin Heidelberg, 2012.
[20] N. Artrith, A. Urban, The Atomic Energy Network Package, 2015. <http://ann.atomistic.net>.
[21] N. Artrith, A.M. Kolpak, Nano Lett. 14 (2014) 2670–2676, http://dx.doi.org/10.1021/nl5005674.
[22] N. Artrith, A.M. Kolpak, Comput. Mater. Sci. 110 (2015) 20–28, http://dx.doi.org/10.1016/j.commatsci.2015.07.046.
[23] J.S. Elias, N. Artrith, M. Bugnet, L. Giordano, G.A. Botton, A.M. Kolpak, Y. Shao-Horn, Elucidating the Nature of the Active Phase in Copper/ceria Catalysts for Co-oxidation, 2015 (submitted for publication).
[24] L. Kavan, M. Grätzel, S.E. Gilbert, C. Klemenz, H.J. Scheel, J. Am. Chem. Soc. 118 (1996) 6716–6723, http://dx.doi.org/10.1021/ja954172l.
[25] U. Diebold, Surf. Sci. Rep. 48 (2003) 53–229, http://dx.doi.org/10.1016/S0167-5729(02)00100-0.
[26] S.U.M. Khan, M. Al-Shahry, W.B. Ingler, Science 297 (2002) 2243–2245, http://dx.doi.org/10.1126/science.1075035.
[27] F. Rosenblatt, Psychol. Rev. 65 (1958) 386–408, http://dx.doi.org/10.1037/h0042519.
[28] E. Alpaydin, Introduction to Machine Learning, second ed., Adaptive Computation and Machine Learning, MIT Press, 2010.
[29] K.L. Priddy, P.E. Keller, SPIE-Int. Soc. Opt. Eng. (2005), http://dx.doi.org/10.1117/3.633187.
[30] C.M. Handley, P.L.A. Popelier, J. Phys. Chem. A 114 (2010) 3371–3383, http://dx.doi.org/10.1021/jp9105585.
[31] L. Raff, R. Komanduri, M. Hagan, S. Bukkapatnam, Neural Networks in Chemical Reaction Dynamics, Oxford University Press, Inc., 2012.
[32] C. Broyden, J. Inst. Math. Appl. 6 (1970) 222–231, http://dx.doi.org/10.1093/imamat/6.1.76.
[33] R. Fletcher, Comput. J. 13 (1970) 317–322, http://dx.doi.org/10.1093/comjnl/13.3.317.
[34] D. Goldfarb, Math. Comput. 24 (1970) 23–26, http://dx.doi.org/10.2307/2004873.
[35] D. Shanno, Math. Comput. 24 (1970) 647–656, http://dx.doi.org/10.1090/S0025-5718-1970-0274029-X.
[36] R. Byrd, P. Lu, J. Nocedal, C. Zhu, SIAM J. Sci. Comput. 16 (1995) 1190–1208, http://dx.doi.org/10.1137/0916069. <http://epubs.siam.org/doi/pdf/10.1137/0916069>.
[37] K. Levenberg, Quart. Appl. Math. 2 (1944) 164–168.
[38] D.W. Marquardt, J. Soc. Ind. Appl. Math. 11 (1963) 431–441, http://dx.doi.org/10.1137/0111030.
[39] M. Hagan, M. Menhaj, IEEE Trans. Neural Netw. 5 (1994) 989–993, http://dx.doi.org/10.1109/72.329697.
[40] B.G. Sumpter, D.W. Noid, Chem. Phys. Lett. 192 (1992) 455–462, http://dx.doi.org/10.1016/0009-2614(92)85498-Y.
[41] K. Tai No, B. Ha Chang, S. Yeon Kim, M. Shik Jhon, H.A. Scheraga, Chem. Phys. Lett. 271 (1997) 152–156, http://dx.doi.org/10.1016/S0009-2614(97)00448-X.
[42] S. Lorenz, Reactions on Surfaces with Neural Networks, PhD Thesis, Technischen Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften, 2001.
[43] S. Lorenz, A. Groß, M. Scheffler, Chem. Phys. Lett. 395 (2004) 210–215, http://dx.doi.org/10.1016/j.cplett.2004.07.076.
[44] A.P. Bartók, R. Kondor, G. Csányi, Phys. Rev. B 87 (2013) 184115, http://dx.doi.org/10.1103/PhysRevB.87.184115.
[45] J. Behler, J. Chem. Phys. 134 (2011) 074106, http://dx.doi.org/10.1063/1.3553717.
[46] H. Eshet, R.Z. Khaliullin, T.D. Kühne, J. Behler, M. Parrinello, Phys. Rev. B 81 (2010) 184107, http://dx.doi.org/10.1103/physrevb.81.184107.
[47] R.Z. Khaliullin, H. Eshet, T.D. Kühne, J. Behler, M. Parrinello, Phys. Rev. B 81 (2010) 100103, http://dx.doi.org/10.1103/physrevb.81.100103.
[48] N. Artrith, T. Morawietz, J. Behler, Phys. Rev. B 83 (2011) 153101, http://dx.doi.org/10.1103/PhysRevB.83.153101.
[49] G.C. Sosso, G. Miceli, S. Caravati, J. Behler, M. Bernasconi, Phys. Rev. B 85 (2012) 174103, http://dx.doi.org/10.1103/physrevb.85.174103.
[50] N. Artrith, J. Behler, Phys. Rev. B 85 (2012) 045439, http://dx.doi.org/10.1103/PhysRevB.85.045439.
[51] N. Artrith, B. Hiller, J. Behler, Phys. Status Solidi B 250 (2013) 1191–1203, http://dx.doi.org/10.1002/pssb.201248370.
[52] T. Morawietz, J. Behler, J. Phys. Chem. A 117 (2013) 7356–7366, http://dx.doi.org/10.1021/jp401225b.
[53] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, J. Phys.: Condens. Matter 21 (2009) 395502, http://dx.doi.org/10.1088/0953-8984/21/39/395502.
[54] A. Kokalj, The XSF Format Specification, 2009. <http://www.xcrysden.org/doc/XSF.html>.
[55] W. Humphrey, A. Dalke, K. Schulten, J. Mol. Graph. 14 (1996) 33–38, http://dx.doi.org/10.1016/0263-7855(96)00018-5.
[56] K. Momma, F. Izumi, J. Appl. Cryst. 41 (2008) 653–658, http://dx.doi.org/10.1107/s0021889808012016.
[57] K. Momma, F. Izumi, J. Appl. Cryst. 44 (2011) 1272–1276, http://dx.doi.org/10.1107/s0021889811038970.
[58] S. Bahn, K. Jacobsen, Comput. Sci. Eng. 4 (2002) 56–66, http://dx.doi.org/10.1109/5992.998641.
[59] A. Kokalj, J. Mol. Graph. 17 (1999) 176–179, http://dx.doi.org/10.1016/s1093-3263(99)00028-5.
[60] A. Kokalj, Comput. Mater. Sci. 28 (2003) 155–168, http://dx.doi.org/10.1016/s0927-0256(03)00104-6.
[61] C. Zhu, R.H. Byrd, P. Lu, J. Nocedal, ACM Trans. Math. Softw. 23 (1997) 550–560, http://dx.doi.org/10.1145/279232.279236.
[62] J.L. Morales, J. Nocedal, ACM Trans. Math. Softw. 38 (2011) 1–4, http://dx.doi.org/10.1145/2049662.2049669.
[63] G. Buxbaum, Industrial Inorganic Pigments, John Wiley & Sons, 2008.
[64] A.P. Popov, A.V. Priezzhev, J. Lademann, R. Myllylä, J. Phys. D: Appl. Phys. 38 (2005) 2564–2570, http://dx.doi.org/10.1088/0022-3727/38/15/006.
[65] U. Diebold, N. Ruzycki, G. Herman, A. Selloni, Catal. Today 85 (2003) 93–100, http://dx.doi.org/10.1016/s0920-5861(03)00378-x.
[66] W.-K. Li, X.-Q. Gong, G. Lu, A. Selloni, J. Phys. Chem. C (2008) 6594–6596, http://dx.doi.org/10.1021/jp802335h.
[67] F. De Angelis, C. Di Valentin, S. Fantacci, A. Vittadini, A. Selloni, Chem. Rev. 114 (2014) 9708–9753, http://dx.doi.org/10.1021/cr500055q.
[68] T. Arlt, M. Bermejo, M.A. Blanco, L. Gerward, J.Z. Jiang, J. Staun Olsen, J.M. Recio, Phys. Rev. B 61 (2000) 14414–14419, http://dx.doi.org/10.1103/physrevb.61.14414.
[69] M. Matsui, M. Akaogi, Mol. Simul. 6 (1991) 239–244, http://dx.doi.org/10.1080/08927029108022432.
[70] P.M. Oliver, G.W. Watson, E.T. Kelsey, S.C. Parker, J. Mater. Chem. 7 (1997) 563–568, http://dx.doi.org/10.1039/a606353e.
[71] V. Swamy, J. Gale, Phys. Rev. B 62 (2000) 5406–5412, http://dx.doi.org/10.1103/physrevb.62.5406.
[72] V. Swamy, J. Gale, L. Dubrovinsky, J. Phys. Chem. Solids 62 (2001) 887–895, http://dx.doi.org/10.1016/s0022-3697(00)00246-8.
[73] P.D. Mitev, K. Hermansson, Surf. Sci. 601 (2007) 5359–5367, http://dx.doi.org/10.1016/j.susc.2007.08.031.
[74] J. Muscat, V. Swamy, N.M. Harrison, Phys. Rev. B 65 (2002) 224112, http://dx.doi.org/10.1103/PhysRevB.65.224112.
[75] F. Labat, P. Baranek, C. Domain, C. Minot, C. Adamo, J. Chem. Phys. 126 (2007) 154703, http://dx.doi.org/10.1063/1.2717168.
[76] J.C. Conesa, J. Phys. Chem. C 114 (2010) 22718–22726, http://dx.doi.org/10.1021/jp109105g.
[77] M.E. Arroyo-de Dompablo, A. Morales-García, M. Taravillo, J. Chem. Phys. 135 (2011) 054503, http://dx.doi.org/10.1063/1.3617244.
[78] M.V. Ganduglia-Pirovano, A. Hofmann, J. Sauer, Surf. Sci. Rep. 62 (2007) 219–270, http://dx.doi.org/10.1016/j.surfrep.2007.03.002.
[79] J. Behler, Int. J. Quant. Chem. 115 (2015) 1032–1050, http://dx.doi.org/10.1002/qua.24890.
[80] T.A. Kandiel, A. Feldhoff, L. Robben, R. Dillert, D.W. Bahnemann, Chem. Mater. 22 (2010) 2050–2060, http://dx.doi.org/10.1021/cm903472p.
[81] S. Kondati Natarajan, T. Morawietz, J. Behler, Phys. Chem. Chem. Phys. 17 (2015) 8356–8371, http://dx.doi.org/10.1039/c4cp04751f.
[82] J. Perdew, K. Burke, M. Ernzerhof, Phys. Rev. Lett. 77 (1996) 3865–3868, http://dx.doi.org/10.1103/PhysRevLett.77.3865.
[83] K.F. Garrity, J.W. Bennett, K.M. Rabe, D. Vanderbilt, Comput. Mater. Sci. 81 (2013) 446–452, http://dx.doi.org/10.1016/j.commatsci.2013.08.053.
[84] S. Curtarolo, W. Setyawan, G.L.W. Hart, M. Jahnatek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M.J. Mehl, H.T. Stokes, D.O. Demchenko, D. Morgan, Comput. Mater. Sci. 58 (2012) 218–226, http://dx.doi.org/10.1016/j.commatsci.2012.02.005.

[85] S.P. Ong, W.D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V.L. Chevrier, K.A. Persson, G. Ceder, Comput. Mater. Sci. 68 (2013) 314–319, http://dx.doi.org/10.1016/j.commatsci.2012.10.028.
[86] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, B. Kozinsky, Comput. Mater. Sci. 111 (2016) 218–230, http://dx.doi.org/10.1016/j.commatsci.2015.09.013.
[87] F. Birch, Phys. Rev. 71 (1947) 809–824, http://dx.doi.org/10.1103/physrev.71.809.
[88] J.W. Ponder, F.M. Richards, J. Comput. Chem. 8 (1987) 1016–1024, http://dx.doi.org/10.1002/jcc.540080710.
[89] G. Bussi, D. Donadio, M. Parrinello, Canonical sampling through velocity rescaling, J. Chem. Phys. 126 (2007) 014101, http://dx.doi.org/10.1063/1.2408420.
[90] M. Rupp, Int. J. Quantum Chem. 115 (2015) 1058–1073, http://dx.doi.org/10.1002/qua.24954.
[91] S. Manzhos, R. Dawes, T. Carrington, Int. J. Quantum Chem. 115 (2014) 1012–1020, http://dx.doi.org/10.1002/qua.24795.
[92] A.P. Bartók, G. Csányi, Int. J. Quantum Chem. 115 (2015) 1051–1057, http://dx.doi.org/10.1002/qua.24927.
[93] A. Sadeghi, S.A. Ghasemi, B. Schaefer, S. Mohr, M.A. Lill, S. Goedecker, J. Chem. Phys. 139 (2013) 184118, http://dx.doi.org/10.1063/1.4828704.
[94] K.T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K.R. Müller, E.K.U. Gross, Phys. Rev. B 89 (2014) 205118, http://dx.doi.org/10.1103/PhysRevB.89.205118.
[95] F. Faber, A. Lindmaa, O.A. von Lilienfeld, R. Armiento, Int. J. Quantum Chem. 115 (2015) 1094–1101, http://dx.doi.org/10.1002/qua.24917.
[96] O.A. von Lilienfeld, R. Ramakrishnan, M. Rupp, A. Knoll, Int. J. Quantum Chem. 115 (2015) 1084–1093, http://dx.doi.org/10.1002/qua.24912.