Computer Physics Communications 247 (2020) 106949

Contents lists available at ScienceDirect

Computer Physics Communications

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

# DScribe: Library of descriptors for machine learning in materials science ☆

Lauri Himanen [a,*], Marc O.J. Jäger [a], Eiaki V. Morooka [a], Filippo Federici Canova [a,b], Yashasvi S. Ranawat [a], David Z. Gao [b,c], Patrick Rinke [a,f], Adam S. Foster [a,d,e]

[a] Department of Applied Physics, Aalto University, P.O. Box 11100, 00076 Aalto, Espoo, Finland
[b] Nanolayers Research Computing Ltd., 1 Granville Court, Granville Road, London, N12 0HL, United Kingdom
[c] Department of Physics, Norwegian University of Science and Technology, NO-7491, Trondheim, Norway
[d] Graduate School Materials Science in Mainz, Staudinger Weg 9, 55128, Germany
[e] WPI Nano Life Science Institute (WPI-NanoLSI), Kanazawa University, Kakuma-machi, Kanazawa 920-1192, Japan
[f] Theoretical Chemistry and Catalysis Research centre, Technische Universität München, Lichtenbergstr. 4, D-85747 Garching, Germany

ARTICLE INFO

*Article history:*
Received 17 April 2019
Received in revised form 14 August 2019
Accepted 24 August 2019
Available online 26 September 2019

*Keywords:*
Machine learning
Materials science
Descriptor
Python
Open source

ABSTRACT

DScribe is a software package for machine learning that provides popular feature transformations (*"descriptors"*) for atomistic materials simulations. DScribe accelerates the application of machine learning for atomistic property prediction by providing user-friendly, off-the-shelf descriptor implementations. The package currently contains implementations for Coulomb matrix, Ewald sum matrix, sine matrix, Many-body Tensor Representation (MBTR), Atom-centered Symmetry Function (ACSF) and Smooth Overlap of Atomic Positions (SOAP). Usage of the package is illustrated for two different applications: formation energy prediction for solids and ionic charge prediction for atoms in organic molecules. The package is freely available under the open-source Apache License 2.0.

**Program summary**

*Program Title:* DScribe
*Program Files doi:* http://dx.doi.org/10.17632/vzrs8n8pk6.1
*Licensing provisions:* Apache-2.0
*Programming language:* Python/C/C++
*Supplementary material:* Supplementary Information as PDF
*Nature of problem:* The application of machine learning for materials science is hindered by the lack of consistent software implementations for feature transformations. These feature transformations, also called descriptors, are a key step in building machine learning models for property prediction in materials science.
*Solution method:* We have developed a library for creating common descriptors used in machine learning applied to materials science. We provide an implementation the following descriptors: Coulomb matrix, Ewald sum matrix, sine matrix, Many-body Tensor Representation (MBTR), Atom-centered Symmetry Functions (ACSF) and Smooth Overlap of Atomic Positions (SOAP). The library has a python interface with computationally intensive routines written in C or C++. The source code, tutorials and documentation are provided online. A continuous integration mechanism is set up to automatically run a series of regression tests and check code coverage when the codebase is updated.



## 1. Introduction

Machine learning of atomistic systems is a highly active, interdisciplinary area of research. The power of machine learning lies in the ability of interpolating existing calculations with surrogate models to accelerate predictions for new systems [1–4]. The set of possible applications is very rich, including high-throughput search of stable compounds with machine learning

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
*E-mail address:* lauri.himanen@aalto.fi (L. Himanen).

https://doi.org/10.1016/j.cpc.2019.106949

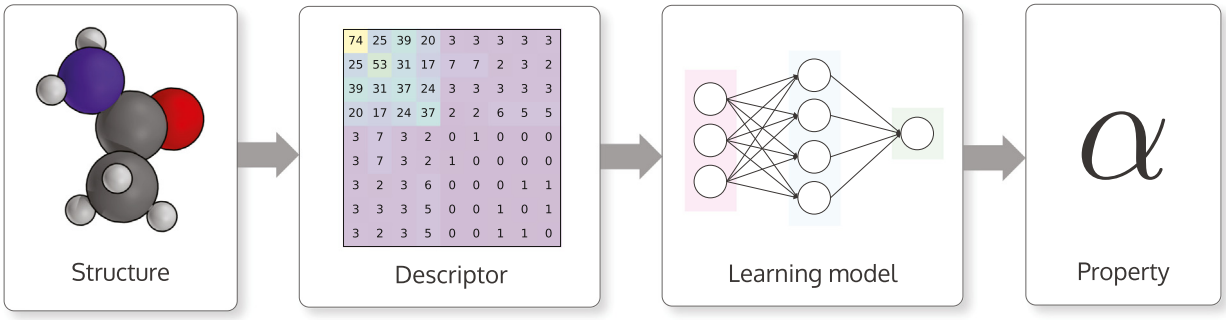


**Fig. 1.** Typical workflow for making machine learning based materials property predictions for atomistic structures. An atomic structure is transformed into a numerical representation called a descriptor. This descriptor is then used as an input for a machine learning model that is trained to output a property for the structure. There is also a possibility of combining the descriptor and learning model together into one inseparable step.

based energy predictions for solids [5–8], accelerated molecular property prediction [9–11], creation of new force-fields based on quantum mechanical training data [12–19], search for catalytically active sites in nanoclusters [20–24] and efficient optimization of complex structures [25–27].

Atomistic machine learning establishes a relationship between the atomic structure of a system and its properties. This so called structure–property relation is illustrated in Fig. 1. It is analogous to the structure–property relation in quantum mechanics. For a set of nuclear charges $\{Z_i\}$ and atomic positions $\{R_i\}$ of a system, the solution of the Schrödinger equation $\hat{H}\Phi = E\Phi$ yields the properties of the system since both the Hamiltonian $\hat{H}$ and the wave function $\Phi$ depend only on $\{Z_i\}$ and $\{R_i\}$. Atomistic machine learning bypasses the computationally costly step of solving the Schrödinger equation[1] by training a surrogate model. Once trained, the surrogate model is typically very fast to evaluate facilitating almost instant structure–property predictions.

Unlike for the Schrödinger equation, the nuclear charges and atomic positions are not a suitable input representation of atomistic systems for machine learning. They are, for example, not rotationally or translationally invariant. If presented with atomic positions, the machine learning method would have to learn rotational and translational invariance for every dataset, which would significantly increase the amount of required training data. For this reason, the input data has to be transformed into a representation that is suitable for machine learning. This transformation step is often referred to as *feature engineering* and the selected features are called a *descriptor* [28].[2] Various feature engineering approaches have been proposed [5–9,14,16,33–40], and often multiple approaches have to be tested to find a suitable representation for a specific task [41]. Features are often based on the atomic structure, but it is also common to extend the input to other system properties [5,28,36,42].

There are several requirements for good descriptors in atomistic machine learning [6,7]. We identify the following properties to be most important for an ideal descriptor:

(i) Invariant with respect to spatial translation of the coordinate system: isometry of space.
(ii) Invariant with respect to rotation of the coordinate system: isotropy of space.
(iii) Invariant with respect to permutation of atomic indices: changing the enumeration of atoms does not affect the physical properties of the system.
(iv) Unique: there is a single way to construct a descriptor from an atomic structure and the descriptor itself corresponds to a single property.
(v) Continuous: small changes in the atomic structure should translate to small changes in the descriptor.
(vi) Compact: the descriptor should contain sufficient information of the system for performing the prediction while keeping the feature count to minimum.
(vii) Computationally cheap: the computation of the descriptor should be significantly faster than any existing computational model for directly calculating the physical property of interest.

In this article we present the DScribe package that can be used to transform atomic structures into machine-learnable input features. The aim of this software is to provide a coherent and easily extendable implementation for atomistic machine learning and fast prototyping of descriptors. There already exist libraries like QML [43], Amp [44], Magpie [45], quippy [46], ChemML [47] and matminer [48] which include a subset of descriptors as a part of a bigger framework for materials data analytics. DScribe follows this spirit but specializes on providing efficient and scalable descriptor transformations and is agnostic to the framework used for doing the actual data analytics.

Currently in the DScribe package we include descriptors that can be represented in a vectorial form and are not dependent on any specific learning model. By decoupling the descriptor creation from the machine learning model, the user can experiment in parallel with various descriptor/model combinations and has the possibility of directly applying emerging learning models on existing data. This freedom to switch between machine learning models becomes important because currently no universally best machine model exists for every problem, as stated by the "No Free Lunch Theorem" [49]. In practice this means that multiple models have to be tested to find optimal performance. Furthermore, vectorial features provide easier insight into the importance of certain features and facilitate the application of unsupervised learning methods, such as clustering and subsequent visualization with informative "materials maps" [50–52].

Descriptors that encode an atomic structure are typically designed to either depict a local atomic environment, or the structure as a whole. *Global* descriptors encode information about the whole atomic structure. These global descriptors can be used to predict properties related to the structure as a whole, such as molecular energies [9], formation energies [5] or band gaps [36]. In this work we cover four such global descriptors: the Coulomb matrix [9], the Ewald sum matrix [7], the sine matrix [7] and the

[1] Of course in practice the Schrödinger equation is not solved directly but by approximate methods. Also approximate solutions of the Schrödinger equation are computationally expensive, compared to surrogate machine learning models.

[2] Sometimes the separation between a descriptor and a learning model is however blurred. For example, in various models based on neural networks [29–32] the description of the atomistic structure is embedded inside the weights of the network, essentially mixing the descriptor and the learning model together.

Many-Body Tensor Representation (MBTR) [6]. *Local* descriptors are instead used to represent a localized region in an atomic structure, and are thus suitable for predicting localized properties, like atomic forces [13], adsorption energies [23], or properties that can be summed from local contributions. In this article we discuss two local descriptors, Atom-centered Symmetry functions (ACSFs) [16] and the Smooth Overlap of Atomic Positions (SOAP) [14].

We first introduce the descriptors that have been implemented in the DScribe package and then we discuss the structure and usage of the package. After this we illustrate the applicability of the package by showing results for formation energy prediction of periodic crystals and partial charge prediction for molecules. We conclude, by addressing the impact and future extensions of this package.

## 2. Descriptors

Here we briefly introduce the different descriptors that are currently implemented in DScribe. In some cases, we have deviated from the original literature due to computational or other reasons, and if so this is explicitly mentioned. For more in-depth presentations of the descriptors we refer the reader to the original research papers. At the end of this section we also discuss methods for organizing the descriptor output so that it can be effectively used in typical machine learning applications.

### 2.1. Coulomb matrix

The Coulomb matrix [9] encodes the atomic species and interatomic distances of a finite system in a pair-wise, two-body matrix inspired by the form of the Coulomb potential. The elements of this matrix are given by:

$$M_{ij}^{\text{Coulomb}} = \begin{cases} 0.5Z_i^{2.4} & \forall\ i = j \\ \frac{Z_i Z_j}{|\boldsymbol{R}_i - \boldsymbol{R}_j|} & \forall\ i \neq j \end{cases}$$

where $Z$ is the atomic number, and $|\boldsymbol{R}_i - \boldsymbol{R}_j|$ is the Euclidean distance between atoms $i$ and $j$. The form of the diagonal terms was determined by fitting the potential energy of neutral atoms [53].

### 2.2. Ewald sum matrix

The Ewald sum matrix [7] can be viewed as a logical extension of the Coulomb matrix for periodic systems. In periodic systems each atom is infinitely repeated in the three crystal lattice vector directions, **a**, **b** and **c** and the electrostatic interaction between two atoms becomes

$$\phi_{ij} = \sum_{\mathbf{n}} \frac{Z_i Z_j}{|\boldsymbol{R}_i - \boldsymbol{R}_j| + \mathbf{n}} \tag{1}$$

where $\sum_{\mathbf{n}}$ is the sum over all lattice vectors $\mathbf{n} = h\mathbf{a} + k\mathbf{b} + l\mathbf{c}$.

For $h, k, l \rightarrow \infty$, this sum converges only conditionally and will become infinite if the system is not charge neutral. In the Ewald sum matrix, the Ewald summation technique [54,55] and a neutralizing background charge [56] is used to force this sum to converge. One can separate the total Ewald energy into pairwise components, which will result in the following matrix:

$$M_{ij}^{\text{Ewald}} = \begin{cases} \phi_{ij}^{\text{real}} + \phi_{ij}^{\text{recip}} + \phi_{ij}^{\text{self}} + \phi_{ij}^{\text{bg}} & \forall\ i = j \\ 2\left(\phi_{ij}^{\text{real}} + \phi_{ij}^{\text{recip}} + \phi_{ij}^{\text{bg}}\right) & \forall\ i \neq j \end{cases}$$

where the terms are given by

$$\phi_{ij}^{\text{real}} = \frac{1}{2} Z_i Z_j \sum_{\boldsymbol{n}'} \frac{\operatorname{erfc}\left(\alpha |\boldsymbol{R}_i - \boldsymbol{R}_j + \mathbf{n}|\right)}{|\boldsymbol{R}_i - \boldsymbol{R}_j + \mathbf{n}|} \tag{2}$$

$$\phi_{ij}^{\text{recip}} = \frac{2\pi}{V} Z_i Z_j \sum_{\mathbf{G}} \frac{e^{-|\mathbf{G}|^2/(2\alpha)^2}}{|\mathbf{G}|^2} \cos\left(\mathbf{G} \cdot \left(\boldsymbol{R}_i - \boldsymbol{R}_j\right)\right) \tag{3}$$

$$\phi_{ij}^{\text{self}} = \begin{cases} -\frac{\alpha}{\sqrt{\pi}} Z_i^2 & \forall\ i = j \\ 0 & \forall\ i \neq j \end{cases} \tag{4}$$

$$\phi_{ij}^{\text{bg}} = \begin{cases} -\frac{\pi}{2V\alpha^2} Z_i^2 & \forall\ i = j \\ -\frac{\pi}{2V\alpha^2} Z_i Z_j & \forall\ i \neq j \end{cases} \tag{5}$$

Here the primed notation means that when $\boldsymbol{n} = \mathbf{0}$ the pairs $i = j$ are not taken into account. $\alpha$ is the screening parameter controlling the size of the gaussian charge distributions used in the Ewald method, **G** is a reciprocal space lattice vector with an implicit $2\pi$ prefactor and $V$ is the volume of the cell. A more detailed derivation is given in the supplementary information. By default we use the value $\alpha = \sqrt{\pi}\left(\frac{N}{V^2}\right)^{1/6}$ [57], where $N$ is the number of atoms in the unit cell.

It is important to notice that the off-diagonal contribution $\phi_{ij}^{\text{self}} + \phi_{ij}^{\text{bg}} = -\frac{\pi}{2V\alpha^2} Z_i Z_j\ \forall\ i \neq j$ given here differs from the original work. In the original formulation this sum was defined as [7] $\phi_{ij}^{\text{self}} + \phi_{ij}^{\text{bg}} = -\frac{\alpha}{\sqrt{\pi}}(Z_i^2 + Z_j^2) - \frac{\pi}{2V\alpha^2}(Z_i + Z_j)^2\ \forall\ i \neq j$. Our correction makes the total matrix elements independent of the screening parameter $\alpha$, which is not the case in the original formulation.

For numerical purposes, the sums in Eqs. (2) and (3) are cut off by $n \leq n_{\text{cut}}$ and $G \leq G_{\text{cut}}$. By default we use the values $G_{\text{cut}} = 2\alpha\sqrt{-\ln A}$ and $n_{\text{cut}} = \sqrt{-\ln A}/\alpha$ [57], where the small positive parameter $A$ controls the accuracy of the sum and can be determined by the user.

### 2.3. Sine matrix

The Ewald sum matrix encodes the correct Coulomb interaction between atoms, but can become computationally heavy especially for large systems. The sine matrix [7] captures some important features of interacting atoms in a periodic system with a much reduced computational cost. The matrix elements are defined by

$$M_{ij}^{\text{sine}} = \begin{cases} 0.5Z_i^{2.4} & \forall\ i = j \\ \phi_{ij} & \forall\ i \neq j \end{cases}$$

where

$$\phi_{ij} = Z_i Z_j |\mathbf{B} \cdot \sum_{k=\{x,y,z\}} \hat{\mathbf{e}}_k \sin^2\left(\pi \mathbf{B}^{-1} \cdot \left(\boldsymbol{R}_i - \boldsymbol{R}_j\right)\right)|^{-1} \tag{6}$$

Here **B** is a matrix formed by the lattice vectors and $\hat{\mathbf{e}}_k$ are the cartesian unit vectors. This functional form has no physical interpretation, but it captures some of the properties of the Coulomb interaction, such as the periodicity of the crystal lattice and an infinite energy when two atoms overlap.

The Coulomb, Ewald sum and sine matrices for diamond are depicted in Fig. 2. Notice that the matrices given here are not unique, as different cell sizes can be used for a periodic crystal, and the indexing of the rows and columns depends on the ordering of atomic indices in the structure. Section 2.7 discusses some ways to overcome the issues related to this non-uniqueness.

By construction the Coulomb matrix is not periodic as manifested by the unequivalent row elements in the matrix (one carbon in the system has four bonded neighbors, three carbons have two neighbors and four carbons have a single bonded neighbor). Conversely, both the Ewald sum and the sine matrix are periodic and correctly encode the identical environment of the carbon atoms in the lattice. As a result, each row and each column has the same matrix elements, but neighboring rows and columns

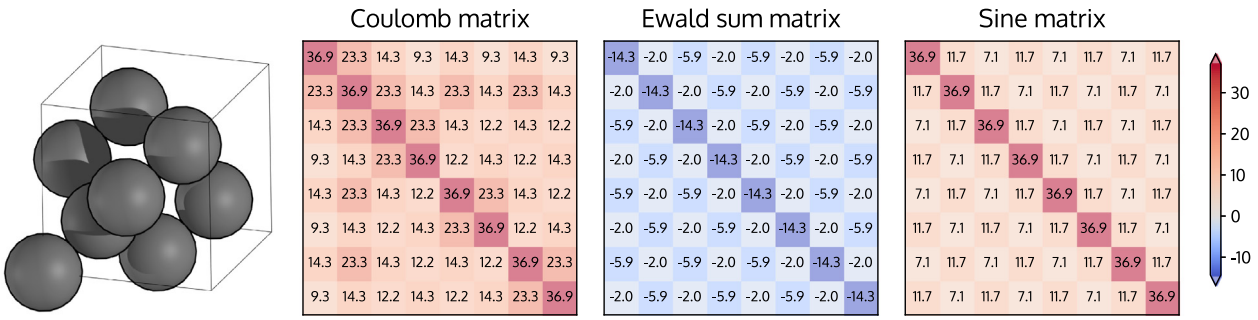


**Fig. 2.** Illustration of the Coulomb matrix, Ewald sum matrix and sine matrix for a periodic diamond structure. The used atomic structure for the conventional diamond cell is shown on the left. The color scale (legend on the right) is used to illustrate the magnitude of the matrix elements.

are shifted by one element relative to each other. Unlike the other matrices, Ewald sum matrix often contains negative elements due to the interaction of the positive atomic nuclei with the added uniform negative background charge. This energetically favorable interaction shifts the off-diagonal elements down in energy compared to the other two matrices. Moreover, the diagonal elements of the Ewald sum matrix encode the physical self-interaction of atoms with their periodic copies, instead of the potential energy of the neutral atoms.

### 2.4. Many-body tensor representation

The many-body tensor representation (MBTR) [6] encodes finite or periodic structures by breaking them down into distributions of differently sized structural motifs and grouping these distributions by the involved chemical elements. In MBTR, a geometry function $g_k$ is used to transform a configuration of $k$ atoms into a single scalar value representing that particular configuration. Our implementation provides terms up to $k = 3$, and provides the following geometry functions $g_1(Z_l)$: $Z_l$ (atomic number), $g_2(\boldsymbol{R}_l, \boldsymbol{R}_m)$: $|\boldsymbol{R}_l - \boldsymbol{R}_m|$ (distance) or $\frac{1}{|\boldsymbol{R}_l - \boldsymbol{R}_m|}$ (inverse distance) and $g_3(\boldsymbol{R}_l, \boldsymbol{R}_m, \boldsymbol{R}_n)$: $\angle(\boldsymbol{R}_l - \boldsymbol{R}_m, \boldsymbol{R}_n - \boldsymbol{R}_m)$ (angle) or $\cos(\angle(\boldsymbol{R}_l - \boldsymbol{R}_m, \boldsymbol{R}_n - \boldsymbol{R}_m))$ (cosine of angle). These scalar values are then broadened by using kernel density estimation with a gaussian kernel, leading to the following distributions $\mathcal{D}_k$

$$\mathcal{D}_1^l(x) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x-g_1(Z_l))^2}{2\sigma_1^2}} \tag{7}$$

$$\mathcal{D}_2^{l,m}(x) = \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x-g_2(\boldsymbol{R}_l,\boldsymbol{R}_m))^2}{2\sigma_2^2}} \tag{8}$$

$$\mathcal{D}_3^{l,m,n}(x) = \frac{1}{\sigma_3\sqrt{2\pi}} e^{-\frac{(x-g_3(\boldsymbol{R}_l,\boldsymbol{R}_m,\boldsymbol{R}_n))^2}{2\sigma_3^2}} \tag{9}$$

Here $\sigma_k$ is the standard deviation of the gaussian kernel and $x$ runs over a predefined range of values covering the possible values for $g_k$. A weighted sum of the distributions $\mathcal{D}_k$ is then made separately for each possible combination of $k$ chemical species present in the dataset. For $k = 1, 2, 3$ these distributions are given by

$$\text{MBTR}_1^{Z_1}(x) = \sum_l^{|Z_1|} w_1^l \mathcal{D}_1^l(x) \tag{10}$$

$$\text{MBTR}_2^{Z_1,Z_2}(x) = \sum_l^{|Z_1|} \sum_m^{|Z_2|} w_2^{l,m} \mathcal{D}_2^{l,m}(x) \tag{11}$$

$$\text{MBTR}_3^{Z_1,Z_2,Z_3}(x) = \sum_l^{|Z_1|} \sum_m^{|Z_2|} \sum_n^{|Z_3|} w_3^{l,m,n} \mathcal{D}_3^{l,m,n}(x) \tag{12}$$

where the sums for $l$, $m$ and $n$ run over all atoms with the atomic number $Z_1$, $Z_2$ or $Z_3$ respectively, and $w_k$ is a weighting function that is used to control the importance of different terms. When calculating MBTR for periodic systems, the periodic copies of atoms in neighboring cells are taken into account by extending the original cell with periodic copies. When a periodic system is extended in this way, certain sets of atoms may get counted multiple times due to translational symmetry. Like in the original formulation [6] we require that one of the atoms, $l$, $m$ or $n$, must be in the original cell. In addition, our implementation ensures that each translationally unique combination of the atoms is counted only once. This makes the MBTR output for different cells representing the same material identical up to a size extensive scalar multiplication factor. Unlike in the original formulation, we do not include the possible correlation between chemical elements directly in Eqs. (10)–(12). We do not however lose any generality, as the correlation between chemical elements can be introduced as a postprocessing step that combines information from the different species.

For $k = 1$, typically no weighting is used: $w_1^l = 1$. In the case of $k = 2$ and $k = 3$, the weighting function can, however be used to give more importance to values that correspond to configuration where the atoms are closer together. For fully periodic systems, a weighting function must be used, as otherwise the sums in Eqs. (10)–(12) do not converge. For $k = 2, 3$ we provide exponential weighting functions of the form

$$w_2^{l,m} = e^{-s_k|\boldsymbol{R}_l - \boldsymbol{R}_m|} \tag{13}$$

$$w_3^{l,m,n} = e^{-s_k(|\boldsymbol{R}_l - \boldsymbol{R}_m| + |\boldsymbol{R}_m - \boldsymbol{R}_n| + |\boldsymbol{R}_l - \boldsymbol{R}_n|)} \tag{14}$$

where the parameter $s_k$ can be used to effectively tune the cutoff distance. For computational purposes a cutoff of $w_k^{\min}$ can be defined to ignore any contributions for which $w_k < w_k^{\min}$.

Some of the distributions, for example $\text{MBTR}_2^{1,2}$ and $\text{MBTR}_2^{2,1}$, contain identical information. In our implementation this symmetry is taken into consideration by only calculating the distributions for which the last atomic number is bigger or equal to the first: $Z_2 \geq Z_1$ in the case of $\text{MBTR}_2^{Z_1,Z_2}$ or $Z_3 \geq Z_1$ in the case of $\text{MBTR}_3^{Z_1,Z_2,Z_3}$. This reduces the computational time and the number of features in the final descriptor without losing information. The final MBTR output for a water molecule is illustrated in Fig. 3.

There are multiple system-dependent parameters that have to be decided for the MBTR descriptor. At each level $k$, the broadness of the distribution $\sigma_k$ has to be chosen. A too small value for $\sigma_k$ will lead to a delta-like distribution that is very sensitive to differences in system configurations. Conversely, a too large value

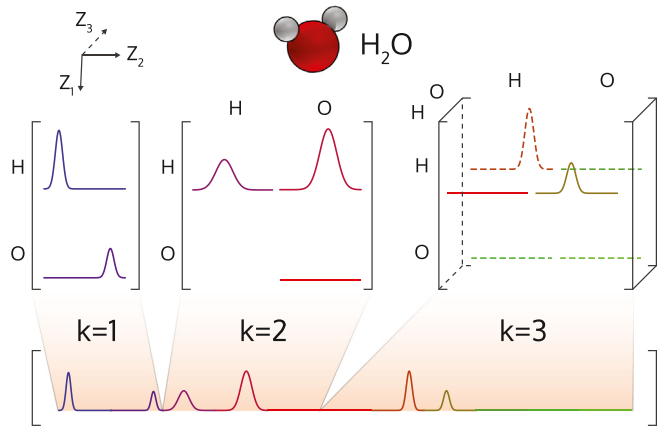


**Fig. 3.** MBTR output for a water molecule showing the distributions $\mathrm{MBTR_k}$ for $k = 1, 2, 3$ with different combinations of chemical elements. For each $k$-term, the distributions can be arranged into a $k$-dimensional grid, resulting in a $k+1$ dimensional tensor. If a flattened one-dimensional vector is needed by the learning model, the distributions may be concatenated together, possibly with some weighting, as shown in the lower panel.

will make it hard to resolve individual peaks as the distribution becomes broad and featureless. The choice of the weighting function also has a direct effect on the final distribution, as it controls how much importance is given to atom combinations that are physically far apart. When combining information from multiple $k$-terms, it can be beneficial to control the contribution from different terms. As the number of features related to higher $k$-values is bigger, the machine learning model may by default give more importance to these higher terms. For example if similarity between two MBTR outputs is measured by an Euclidean distance in the machine learning model, individually normalizing the total output for each term $k$ to unit length helps in equalizing the information content of the different terms.

### 2.5. Atom-centered Symmetry Functions

Atom-centered Symmetry Functions (ACSFs) [16] can be used to represent the local environment near an atom by using a fingerprint composed of the output of multiple two- and three-body functions that can be customized to detect specific structural features. ACSF encodes the configuration of atoms around a single central atom with index $i$ by using so called symmetry functions. The presence of atoms neighboring the central atom are detected by using three different two-body symmetry functions $G_i^{1,Z_1}$, $G_i^{2,Z_1}$ and $G_i^{3,Z_1}$, which are defined as

$$G_i^{1,Z_1} = \sum_j^{|Z_1|} f_c\left(R_{ij}\right)$$

$$G_i^{2,Z_1} = \sum_j^{|Z_1|} e^{-\eta\left(R_{ij}-R_s\right)^2} f_c\left(r_{ij}\right)$$

$$G_i^{3,Z_1} = \sum_j^{|Z_1|} \cos\left(\kappa R_{ij}\right) f_c\left(r_{ij}\right)$$

where the summation for $j$ runs over all atoms with atomic number $Z_1$, $\eta$, $R_s$ and $\kappa$ are user-defined parameters, $R_{ij} = |\boldsymbol{R}_i - \boldsymbol{R}_j|$ and $f_c$ is a smooth cutoff function defined as

$$f_c(r) = \frac{1}{2}\left[\cos\left(\pi \frac{r}{r_{\mathrm{cut}}}\right) + 1\right]$$

where $r_{\mathrm{cut}}$ is a cutoff radius.

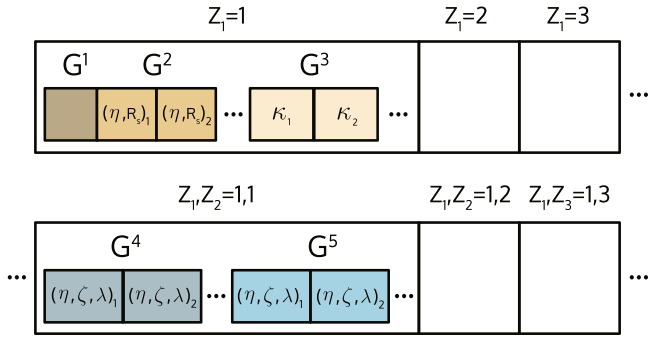


**Fig. 4.** Structure of the ACSF output vector. The values of the two-body symmetry functions $G^1$, $G^2$ and $G^3$ are given first for each chemical species present in the dataset. Next the values of the three-body symmetry functions $G^4$ and $G^5$ are listed for each unique combination of two chemical species. All symmetry functions except $G^1$ may also have multiple parameterizations as indicated by the subindices.

Additionally, three-body functions may be used to detect specific motifs defined by three atoms, one being the central atom. These three-body functions include a dependence on the angle between triplets of atoms within cutoff, as well as their mutual distance. The package implements the following functions

$$G_i^{4,Z_1,Z_2} = 2^{1-\zeta} \sum_{j\neq i}^{|Z_1|} \sum_{k\neq i}^{|Z_2|} (1+\lambda\cos\theta)^\zeta \cdot e^{-\eta\left(R_{ij}^2+R_{ik}^2+R_{jk}^2\right)} f_c\left(R_{ij}\right) f_c\left(R_{ik}\right) f_c\left(R_{jk}\right) \quad (15)$$

$$G_i^{5,Z_1,Z_2} = 2^{1-\zeta} \sum_{j\neq i}^{|Z_1|} \sum_{k\neq i}^{|Z_2|} (1+\lambda\cos\theta)^\zeta \cdot e^{-\eta\left(R_{ij}^2+R_{ik}^2\right)} f_c\left(R_{ij}\right) f_c\left(R_{ik}\right) \quad (16)$$

where the summation of $j$ and $k$ runs over all atoms with atomic numbers $Z_1$ or $Z_2$ respectively, $\zeta$, $\lambda$ and $\eta$ are user-defined parameters and $\theta$ is the angle between the three atoms ($i$th atom in the center).

In practice, multiple symmetry functions of each type are used in the descriptor, each with a different parametrization ($\zeta$, $\lambda$, $\eta$, $R_s$ and $\kappa$), encoding different portions of the chemical environment. As there is no single set of symmetry functions that optimally fits all applications, the selection is guided by the generally desired properties of a descriptor previously listed (unique, continuous and compact) and the specifics of the application. Different sets of symmetry functions used for various applications can be found in the literature [58–60].

The final fingerprint for a single atom can be constructed by concatenating the output from differently parametrized symmetry functions with consistent ordering, as illustrated in Fig. 4. The list starts with the two-body ACSFs, ordered by atom type $Z_1$. For each type, $G^1$ appears first, bringing only one value since it has no parameter dependence. Next we find all values of $G^2$ calculated with different $(\eta, R_s)$ parameter pairs given by the user. The values of $G^3$ for all $\kappa$ are found last. This sequence is repeated for each atomic type, sorted from lighter to heavier. Three-body ACSFs appear afterward: for each unique combination of chemical elements, we find the values of $G^4$ and $G^5$ given by all specified triplets of $(\zeta, \lambda, \eta)$.

### 2.6. Smooth Overlap of Atomic Orbitals

The Smooth Overlap of Atomic Positions (SOAP) [14] can be used to encode a local environment within an atomic structure by using an expansion of a gaussian smeared atomic density based

on spherical harmonics and radial basis functions. In SOAP, the atomic structure is first transformed into atomic density fields $\rho^Z$ for each species by using un-normalized gaussians centered on each atom

$$\rho^Z(\boldsymbol{r}) = \sum_i^{|Z|} e^{-\frac{1}{2\sigma^2}|\boldsymbol{r}-\boldsymbol{R}_i|^2}. \tag{17}$$

Here the summation for $i$ runs over atoms with the atomic number $Z$ to build a separate density for each atomic element and the width of the gaussian is controlled by $\sigma$.

When the origin $\boldsymbol{r} = \boldsymbol{0}$ is chosen to be centered at the local point of interest, the atomic density may then be expanded with a set of orthonormal radial basis functions $g_n$ and spherical harmonics $Y_{lm}$ as

$$\rho^Z(\boldsymbol{r}) = \sum_{nlm} c^Z_{nlm} g_n(r) Y_{lm}(\theta, \phi) \tag{18}$$

where the coefficients can be obtained through

$$c^Z_{nlm} = \iiint_{\mathcal{R}^3} \mathrm{d}V g_n(r) Y_{lm}(\theta, \phi) \rho^Z(\boldsymbol{r}). \tag{19}$$

Instead of using the complex spherical harmonics as in the original work [14], we use the real (tesseral) spherical harmonics as they are computationally preferable when expanding real-valued functions such as the atomic density defined by Eq. (17). The real spherical harmonics $Y_{lm}$ are defined as

$$Y_{lm}(\theta, \phi) = \begin{cases} \sqrt{2}(-1)^m \mathrm{Im}[Y_l^{|m|}(\theta, \phi)] & \text{if } m < 0 \\ Y_l^0 & \text{if } m = 0 \\ \sqrt{2}(-1)^m \mathrm{Re}[Y_l^m(\theta, \phi)] & \text{if } m > 0 \end{cases} \tag{20}$$

where $Y_l^m$ corresponds to the complex orthonormalized spherical harmonics defined as

$$Y_l^m(\theta, \phi) = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi} \tag{21}$$

and $P_l^m$ are the associated Legendre polynomials.

The final rotationally invariant output from our SOAP implementation is the partial power spectra [51] vector $\mathbf{p}$ where the individual vector elements are defined as:

$$p^{Z_1,Z_2}_{nn'l} = \pi \sqrt{\frac{8}{2l+1}} \sum_m \left(c^{Z_1}_{nlm}\right)^* c^{Z_2}_{n'lm} \tag{22}$$

The vector $\mathbf{p}$ is constructed by concatenating the elements $p^{Z_1,Z_2}_{nn'l}$ for all unique atomic number pairs $Z_1, Z_2$, all unique pairs of radial basis functions $n, n'$ up to $n_{\max}$ and the angular degree values $l$ up to $l_{\max}$.

Spherical harmonics are a natural orthogonal and complete set of functions for the angular degrees of freedom. For the radial degree of freedom the selection of the basis set is not as trivial and multiple approaches may be used. In our implementation we, by default, use a set of spherical primitive gaussian type orbitals $g_{nl}(r)$ as radial basis functions. These basis functions are defined as

$$g_{nl}(r) = \sum_{n'=1}^{n_{\max}} \beta_{nn'l} \phi_{n'l}(r) \tag{23}$$

$$\phi_{nl}(r) = r^l e^{-\alpha_{nl} r^2}. \tag{24}$$

This basis set allows analytical integration of the $c_{nlm}$ coefficients defined by Eq. (19). This provides a speedup over various other radial basis functions that require numerical integration. Our current implementation provides the analytical solutions up to $l \leq 9$, with the possibility of adding more in the future.

The decay parameters $\alpha_n$ are chosen so that each non-orthonormalized function $\phi_{nl}$ decays to a threshold value of $10^{-3}$ at a cutoff radius taken on an evenly spaced grid from 1Å to $r_{\text{cut}}$ with a step of $\frac{r_{\text{cut}}-1}{n_{\max}}$. Thus the parameter $r_{\text{cut}}$ controls the maximum reach of the basis and a better sampling can be obtained by increasing the number of basis functions $n_{\max}$.

The weights $\beta_{nn'l}$ are chosen so that the radial basis functions are orthonormal. For each value of angular degree $l$, the orthonormalizing weights $\beta_{nn'l}$ can be calculated with Löwdin orthogonalization [61]:

$$\boldsymbol{\beta} = \boldsymbol{S}^{-1/2} \tag{25}$$

$$S_{nn'} = \langle \phi_{nl} | \phi_{n'l} \rangle = \int_0^\infty \mathrm{d}r r^2 r^l e^{-\alpha_{nl} r^2} r^l e^{-\alpha_{n'l} r^2} \tag{26}$$

where the matrix $\boldsymbol{\beta}$ contains the weights $\beta_{nn'l}$ and $\boldsymbol{S}$ is the overlap matrix.

We also provide an option for using the radial basis consisting of cubic and higher order polynomials, as introduced in the original SOAP article [14]. This basis set is defined as:

$$g_n(r) = \sum_{n'=1}^{n_{\max}} \beta_{nn'} \phi_{n'}(r) \tag{27}$$

$$\phi_n(r) = (r - r_{\text{cut}})^{n+2} \tag{28}$$

The calculations with this basis are performed with efficient numerical integration and currently support $l_{\max} \leq 20$.

The two different basis sets are compared in Fig. 5. Most notably the form of the gaussian type orbitals depends on the angular degree $l$, whereas the polynomial basis is independent of this value. It is also good to notice that between these two radial basis functions the definition of $r_{\text{cut}}$ is somewhat different — whereas the polynomial basis is guaranteed to decay to zero at $r_{\text{cut}}$, the gaussian basis only approximately decays near this value and the decay is also affected by the orthonormalization.

### 2.7. Descriptor usage as machine learning input

In this section we discuss some of the methods for organizing the output from descriptors so that it can be efficiently used as input for machine learning.

The descriptor invariance against permutations of atomic indices – property (iii) in the introduction – is directly achieved in MBTR, SOAP and ACSF by stratifying the output according to the involved chemical elements. The output is always ordered by a predefined order determined by the chemical elements that are included in the dataset, making the output independent of the indexing of individual atoms. The three matrix descriptors – the Coulomb matrix, Ewald sum matrix, and sine matrix – are, however, not invariant with respect to permutation of atomic indices, as the matrix columns and rows are ordered by atomic indices. However, there are different approaches for enforcing invariance for these matrices. One way is to encode the matrices by their eigenvalues, which are invariant to changes in the column and row ordering [9]. Another way is to order the rows and columns by a chosen norm, typically the Euclidean norm [33]. A third approach is to augment the dataset by creating multiple slightly varying matrices for each structure. In this approach multiple matrices are drawn from a statistical set of sorted matrices where Gaussian noise is added to the row norms before sorting [33]. When the learning algorithm is trained over this ensemble of matrices it becomes more robust against small sorting differences that can be considered noise. All of these three approaches are available in our implementation.

Machine learning algorithms also often require constant-sized input. Once again the stratification of the descriptor output by

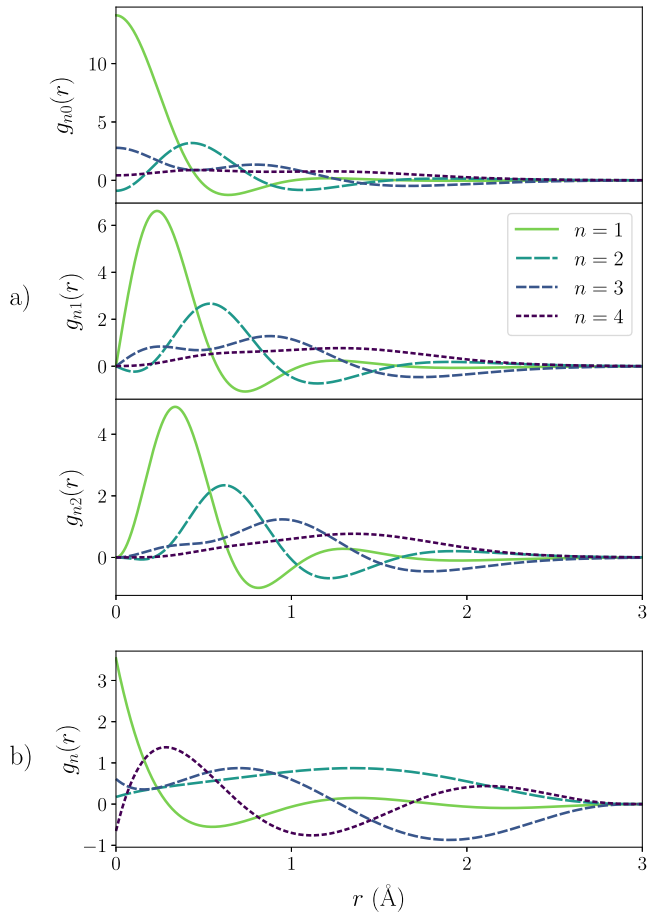


**Fig. 5.** Plot of the (a) gaussian type orbital and (b) polynomial radial basis functions, defined by Eqs. (24) and (28) respectively. The basis functions here correspond to the orthonormalized set with $r_{\text{cut}} = 3$ and up to $n_{\text{max}} = 4$. Notice that the polynomial basis is independent of the spherical harmonics degree $l$, whereas the form of the gaussian type orbital basis depends on $l$ and the examples here are given for $l = 0, 1, 2$.

chemical elements makes the output for MBTR, ACSF and SOAP constant size. For the matrix descriptors a common way to achieve a uniform size for geometries with different amount of atoms, is by introducing zero-padding. This means that we first have to determine the largest system in the dataset. If this system has $N_{\text{max}}$, we allocate matrices of size $N_{\text{max}} \times N_{\text{max}}$ or a vectors or size $N_{\text{max}}$ if using matrix eigenvectors. The descriptor for each system will fill the first $N^2$ or $N$ many entries, with the rest being set to zero. If the machine-learning algorithms expect a one-dimensional vector as input, the two-dimensional matrices can be flattened by concatenating the rows together into a single vector.

Local descriptors, such as ACSF and SOAP, encode only local spatial regions and cannot be directly used as input for performing predictions related to entire structures. There are, however, various ways for combining information from multiple local sites to form a prediction for an entire structure. The descriptor output for multiple local sites can simply be averaged, a custom kernel can be used to combine information from multiple sites [51,62] or the predicted property can in some cases be directly modeled as a sum of local contributions [13].

```python
from ase.build import molecule
from dscribe.descriptors import SOAP

# Define the atomic system
system = molecule("H2O")

# Setup the descriptor
soap = SOAP(
    species=["H", "O"],
    rbf="gto",
    rcut=3,
    nmax=10,
    lmax=8,
    sparse=True
)

# Query the number of features
n_features = soap.get_number_of_features()

# Get output as a dense or sparse array
output = soap.create(system)

# Get output for multiple systems
systems = [molecule("H2"), molecule("O2")]
outputs = soap.create(systems, n_jobs=2)
```

**Fig. 6.** Example of creating descriptors with DScribe. The structures are defined as ase.Atoms objects, in this case by using predefined molecule geometries. The usage of all descriptors follows the same pattern: (a) a descriptor object is initialized with the desired configuration (b) the number of features can be requested with `get_number_of_features` (c) the actual output is created with `create`-method that takes one or multiple atomic structures and possibly other arguments, such as the number of parallel jobs to use.

## 3. Software structure

We use python as the default interfacing language through which the user interacts with the library. This decision was motivated by the existence of various python libraries, including ase [63], pymatgen [64] and quippy [46], that supply tools for creating, reading, writing and manipulating atomic structures. Our python interface does not, however, restrict the implementation to be made entirely in python. Python can easily interact with software libraries written with high-performance, statically typed languages such as C, C++ and Fortran. We use this dual approach by performing some of the most computationally heavy calculations either in C or C++.

An example of creating a descriptor for an atomic structure with the library is demonstrated in Fig. 6. It demonstrates the workflow that is common to all descriptors in the package. For each descriptor we define a class, from which objects can be instantiated with different descriptor specific setups.

All the descriptors have the `sparse`-parameter that controls whether the created output is a dense or a sparse matrix. The possibility for creating a sparse output is given so that large and sparsely filled output spaces can be handled, as typically encountered when a dataset contains large amounts of different chemical elements. Various machine learning algorithms can make use of this sparse matrix output with linear algebra routines specifically designed for sparse data structures.

Once created, the descriptor object is ready to be used and provides different methods for interacting with it. All of the descriptors implement two methods: `get_number_of_features` and `create`. The `get_number_of_features`-method can be used for querying the final number of features for the descriptor, even before a structure has been provided. This dimension can be used for initializing and reserving storage space for the resulting output array. `create` accepts one or multiple atomistic structures as an argument, and possibly other descriptor-specific arguments.

It returns the final descriptor output that can be used in machine learning applications. To define atomic structures we use the ase.Atoms-object from the ase package [63]. The Atoms-objects are easy to create from structure files or build with the utilities provided by ase.

As the creation of a descriptor for an atomic system is completely independent from the other systems, it can be easily parallelized with data parallelism. For convenience we provide a possibility of parallelizing the descriptor creation for multiple samples over multiple processes. This can be done by simply providing the number of parallel jobs to instantiate with the `n_jobs`-parameter as demonstrated in Fig. 6.

The DScribe package is structured such that new descriptors can easily be added. We provide a python base-class that defines a standard interface for the descriptors through abstract classes. One of our design goals is to provide a codebase in which researchers can make their own descriptors available to the whole community. All descriptor implementations are accompanied by a test module that defines a set of standard tests. These tests include tests for rotational, translational and index permutation invariance, as well as other tests for checking the interface and functionality of the descriptor. We have adapted a continuous integration system that automatically runs a series of regression tests when changes in the code are introduced. The code coverage is simultaneously measured as a percentage of visited code lines in the python interface.

The source code is directly available in github at https://github.com/SINGROUP/dscribe and we have created a dedicated home page at https://singroup.github.io/dscribe/ that provides additional tutorials and a full code documentation. For easy installation the code is provided through the python package index (pip) under the name `dscribe`.

## 4. Results and discussion

The applicability of the software is demonstrated by using the different descriptors in building a prediction model for formation energies of inorganic crystal structures and ionic charges of atoms in organic molecules. The used datasets are publicly available at Figshare (https://doi.org/10.6084/m9.figshare.c.4607783). These examples demonstrate the usage of the package in supervised machine learning tasks, but the output vectors can be as easily used in other learning tasks. For example the descriptors can be used as input for unsupervised clustering algorithms such as T-distributed stochastic neighbor embedding (T-SNE) [65] or Sketchmap [50] to analyze structure–property relations in structural and chemical landscapes.

For simplicity we here restrict the machine learning model to be kernel ridge regression (KRR) as implemented in the scikit-learn package [66]. However, the vectorial nature of the output from all the introduced descriptors does not impose any specific learning scheme, and many other regressors can be used, including neural networks, decision trees and support vector regression.

### 4.1. Formation energy prediction for inorganic crystals

We demonstrate the use of multiple descriptors on the task of predicting the formation energy of inorganic crystals. The data comes from the Open Quantum Materials Database (OQMD) 1.1 [67]. We selected structures with a maximum of 10 atoms per unit cell and a maximum of 6 different atomic elements. Unconverged systems were filtered by removing samples which have a formation energy that is more than two standard deviations away from the mean, resulting in the removal of 96 samples. After these selections, 222 215 samples were left. The distribution

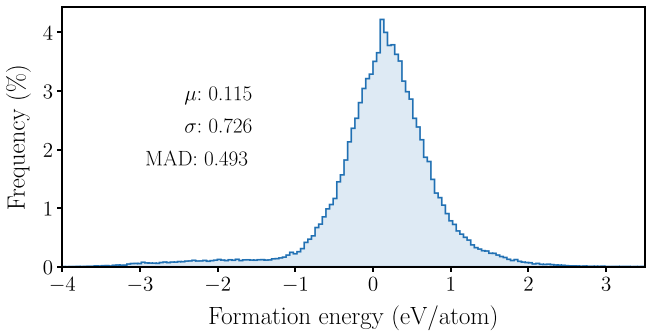


**Fig. 7.** Distribution of the formation energies together with the mean ($\mu$), standard deviation ($\sigma$) and mean absolute deviation (MAD).

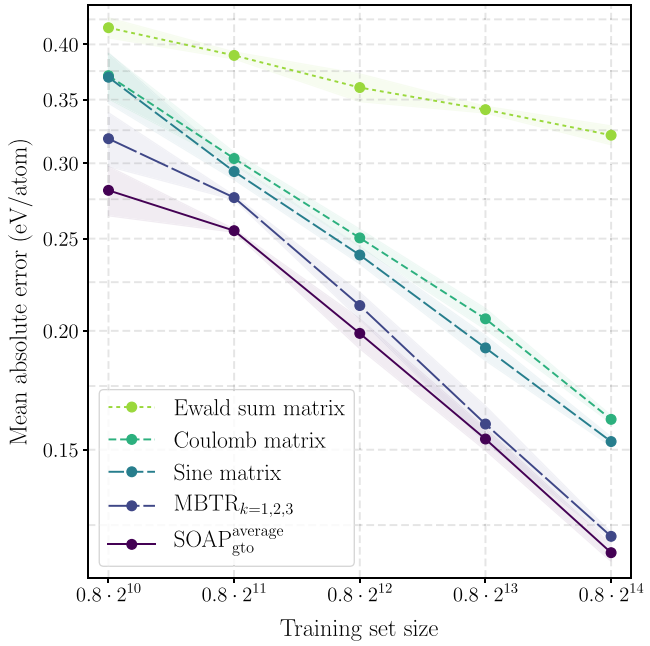


**Fig. 8.** Mean absolute error for formation energies in the test set as a function of training set size. The data consists of inorganic crystals from the OQMD database. The predictions are performed with kernel ridge regression and five different descriptors: Ewald sum matrix, Coulomb matrix, sine matrix, MBTR$_{k=1,2,3}$ and an averaged SOAP output for all atoms in the crystal. The figure shows an average over three randomly selected datasets, with the standard deviation shown by the shaded region.

of the formation energies is shown in Fig. 7. The models are trained and tested on total dataset sizes of 1024, 2048, 4096, 8192 and 16384, from which 80% is used as training data and 20% as test data. These sizes are selected as they are successive powers of two making them equidistant on a logarithmic grid. For each dataset size the results are averaged over three different random selections. The resulting mean absolute errors are given in Fig. 8. A full breakdown of the results for each descriptor and dataset size along with other performance metrics – including root mean square error, squared Pearson correlation coefficient and maximum error – is given in the Supplementary Information.

The Coulomb matrix, Ewald sum matrix and sine matrix are used for the prediction with matrix rows and columns sorted by their Euclidean norm, and using the unit cell that was used for performing the formation energy calculation. The Coulomb matrix does not take the periodicity of the structure into account,

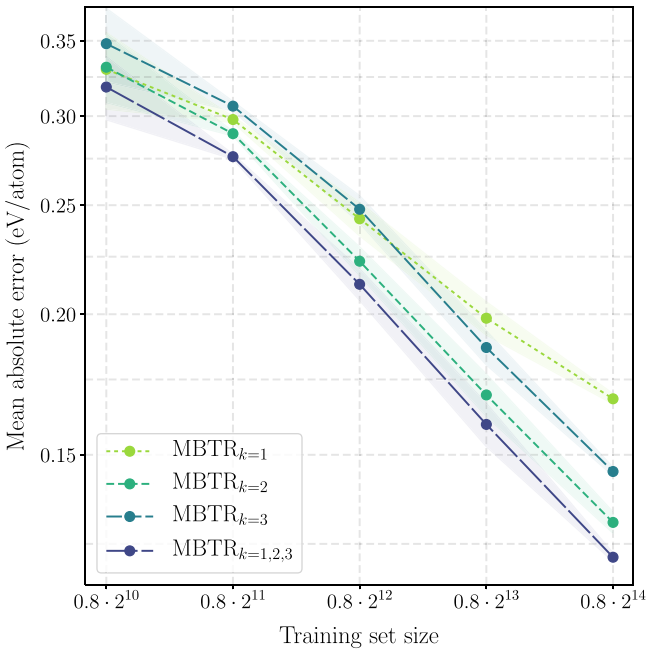


**Fig. 9.** Breakdown of the error for formation energies in the test set for different MBTR-terms. The predictions are performed with kernel ridge regression and four different MBTR configurations: MBTR$_{k=1}$, MBTR$_{k=2}$, MBTR$_{k=3}$ and MBTR$_{k=1,2,3}$ which includes all three terms, each term normalized to unit length. The figure shows an average over three randomly selected datasets, with the standard deviation shown by the shaded region.

but is included as a baseline for the other methods. We include MBTR with different values of $k$ and for each $k$ we individually optimize $\sigma$ and $s_k$ with grid search. Fig. 9 shows the error for each tested MBTR term, and the best performing one is included in Fig. 8. To test the energy prediction by combining information from multiple local descriptors, as discussed in 2.7, we also include results using a simple averaged SOAP output for all atoms in the simulation cell. For SOAP we use the gaussian type orbital basis and fix $n_{\text{max}} = 8$ and $l_{\text{max}} = 8$, but optimize the cutoff $r_{\text{cut}}$ and gaussian width $\sigma$ individually with grid search.

The possible descriptor hyperparameters are optimized at a subset of $2^{12} = 4096$ samples with 5-fold cross-validation and 80%/20%-training/test split. The KRR kernel width and the regularization parameter are also allowed to vary on a logarithmic grid during the descriptor hyperparameter search. The use of a smaller subset allows much quicker evaluation for the hyperparameters than optimizing the hyperparameters for each size individually, but the transferability of these optimized hyperparameters to different sizes may affect the results slightly. After finding the optimal descriptor setup, it is used in training a model for all the different dataset sizes. The same cross-validation setup as for the descriptor hyperparameter optimization is used, but now with a finer grid for the KRR kernel width. The hyperparameter grids and optimal values for both the descriptors and kernel ridge regression are found in the Supplementary Information together with additional details.

### 4.2. Ionic charge prediction for organic molecules

To demonstrate the prediction of local properties with the DScribe package, a prediction of ionic charges in small organic molecules is performed with the different local descriptors included in the package. The dataset consists of Mulliken charges calculated at the CCSD level for the GDB-9 dataset of 133 885 neutral molecules [68]. The structures contain up to nine atoms and five different chemical species: hydrogen, carbon, nitrogen, oxygen, and fluorine with 1 230 122, 846 557, 139 764, 187 996 and 3314 atoms present for each species respectively. The distribution of the ionic charges for each species is shown in Fig. 11. The geometries have been relaxed at the B3LYP/6-31G(2df,p) level and no significant forces were present in the static CCSD calculation. The models are trained and tested on a subset of 10 000 samples per chemical species (except fluorine, for which only 3314 atoms were available and all are used), from which 80% is used as training data and 20% as test data. The combined parity plots for all five chemical species together with error metrics are given in Fig. 10. A breakdown of the results for each species separately is given in the Supplementary Information.

The prediction is performed with the two local descriptors included in the package, SOAP and ACSF. For SOAP we perform the prediction with both radial basis functions: the polynomial basis (SOAP$_{\text{poly}}$) and the gaussian type orbital radial basis (SOAP$_{\text{gto}}$). For them we fix $n_{\text{max}} = 8$ and $l_{\text{max}} = 8$, but optimize the cutoff

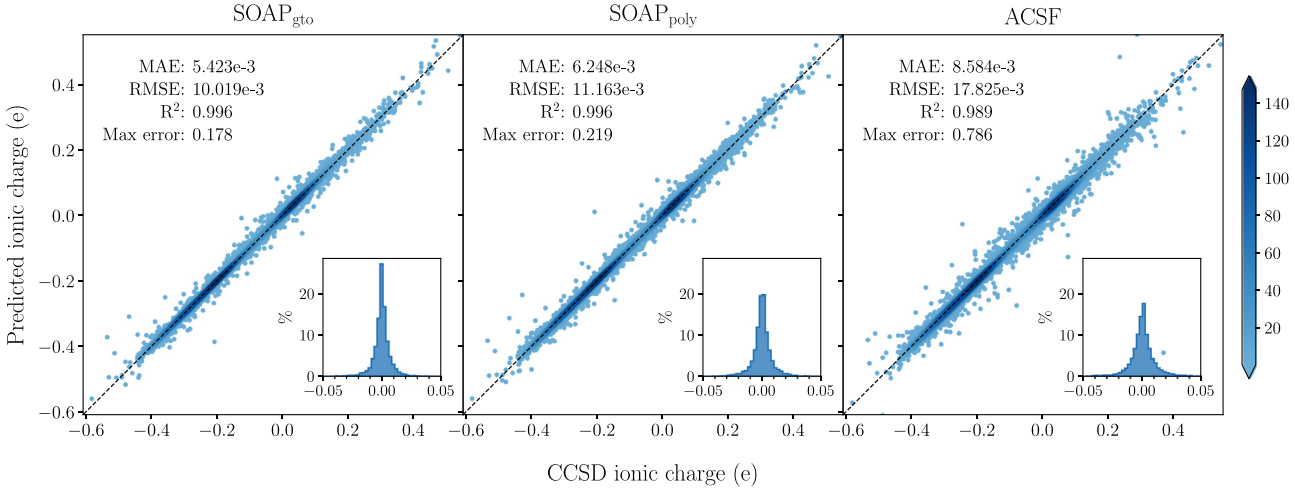


**Fig. 10.** Parity plot of ionic charge prediction results from the test set against true CCSD ionic charges. The predictions are performed with kernel ridge regression using SOAP$_{\text{gto}}$ (gaussian type orbital basis), SOAP$_{\text{poly}}$ (polynomial basis) and ACSF. The mean absolute error (MAE), root mean square error (RMSE), squared Pearson correlation coefficient (R$^2$) and maximum error are also shown together with the total error distribution in the inset.

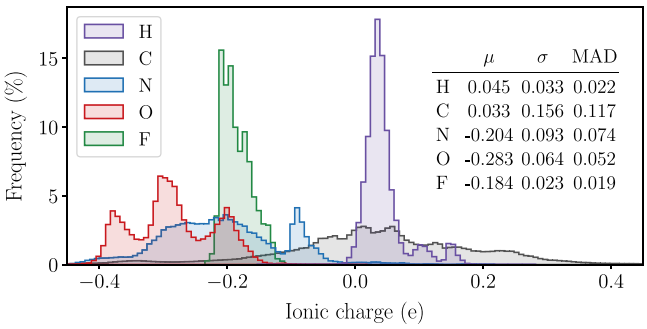


**Fig. 11.** Distribution of the ionic charges for each chemical species together with the mean ($\mu$), standard deviation ($\sigma$) and mean absolute deviation (MAD).

$r_{\text{cut}}$ and Gaussian width $\sigma$ with grid search. For ACSF we use 10 radial functions $G^2$ and 8 angular functions $G^3$. The cutoff value $r_{\text{cut}}$ is shared between the radial and angular functions and it is optimized with grid search. More details about the used ACSF symmetry functions are found in the Supplementary Information.

Descriptor hyperparameters are optimized with grid search separately for each species on a smaller set of 2500 sample atoms with 5-fold cross-validation and 80%/20%-training/test split. Both the KRR kernel width and the regularization parameter are allowed to vary on a logarithmic grid during the descriptor hyperparameter search. After finding the optimal descriptor setup, it is used in training a model for full dataset of 10 000 samples (except for fluorine with 3314 total samples). The training is done with the same cross-validation setup as for the descriptor hyperparameter optimization, but now with finer grid for the KRR kernel width. The hyperparameter grids and optimal values for both the descriptors and kernel ridge regression are found in the Supplementary Information together with additional details.

### 4.3. Discussion

The formation energy prediction demonstrates that our implementation performs consistently and offers insight into the performance of the different descriptors. Special care must be taken in interpreting the results, as there exist different variations of the different descriptors. For example, as discussed in Section 2.7, there are different ways to combine information from multiple local SOAP-outputs, and different geometry functions and cutoff types may be used for the MBTR. The learning rates also depend on the chosen machine learning model.

With $\text{SOAP}^{\text{average}}_{\text{gto}}$ and a training set of $0.8 \cdot 2^{14} = 13107$ samples the best mean absolute error of 0.117 eV/atom is achieved. It has been demonstrated that a similar mean absolute error (0.09 eV/atom [5] and 0.12 eV/atom [40]) can be used for virtual screening of materials by stability. The fact that the training data contains 89 chemical elements and various structural phases makes highly accurate predictions challenging and the error is still relatively large when compared against the mean absolute deviation of 0.493 eV/atom for the labels. As shown by earlier research [29,32,69], the prediction error can be reduced further by using a learning model with a more intelligent scheme for combining local structural information.

Our results for the Ewald sum matrix and the sine matrix reflect the results reported earlier, where a formation energy prediction was performed for a similar set of data from the Materials Project [70]. They report MAE for the Ewald sum matrix to be 0.49 eV and for the sine matrix to be 0.37 eV [7] with a training set of 3000 samples, whereas we find MAE for the Ewald sum matrix to be 0.36 eV and for the sine matrix to be 0.24 eV with a training set of 3276 samples. The performance improvement in our results can be explained by differences in the contents of the used dataset. We, however, recover the same trend of the sine matrix performing better, even when issues in the original formulation of the Ewald sum matrix (as discussed in Section 2.2) were addressed. The low performance of the more accurate charge interaction in the Ewald model and the relatively small difference between the performance of the Coulomb and sine matrix may indicate that for this task the information of the potential energy of the neutral atoms – contained on the diagonal of both the sine and Coulomb matrix – largely controls the performance.

With respect to the individual performance of the different MBTR parts, the $k = 2$ terms containing distance information performs best, whereas the angle information contained in $k = 3$ and the simple composition information contained by $k = 1$ lag behind. However, the best MBTR performance is achieved by combining the information from all of the terms. It is also surprising how well the simple averaging scheme for SOAP performs in the tested dataset range. When extrapolating the performance to larger datasets, it can however be seen that MBTR may provide better results.

The charge prediction test illustrates that the ionic charges of different species in organic molecules may be learned accurately on the CCSD level just by observing the local arrangement of atoms up to a certain radial cutoff. On average the mean absolute error is around 0.005–0.01 $e$ when using up to 10 000 samples for each species.

The best mean absolute error of 0.0054 $e$ and root mean square error of 0.0100 $e$ is achieved with $\text{SOAP}_{\text{gto}}$. A similar root mean square error of 0.016 $e$ was achieved in a recent machine learning based partial charge prediction for drug-like molecules using charges extracted from DFT electron density [71]. The machine learned partial charges offer a great balance between accuracy and computational cost, making them an attractive alternative to full quantum chemical calculations or empirical charge models. Potential applications include the parametrization of partial charges in classical molecular dynamics and quantitative structure–activity relationship (QSAR) models [71].

Fig. 11 shows that the deviation of the charges in the dataset depends on the species, which is also transferred to a species-specific variation of the prediction error included in the Supplementary Information. As to be expected, the charge of the multi-valent species – C, N, O – varies much more in the CCSD data and is much harder to predict than the charge of the low valence species H and F. Predicting the ionic charge of carbon is most difficult and so most of the outliers correspond to carbon atoms, with a few noticeable outliers corresponding also to oxygen and nitrogen atoms.

Our comparison shows that there is little difference between the predictive performance of the two radial bases used for SOAP. With our current implementation there is, however, a notable difference in the speed of creating these descriptors. For identical settings ($n_{\text{max}} = 8$, $l_{\text{max}} = 8$, $r_{\text{cut}} = 5$, and $\sigma = 0.1$), the gaussian type orbital basis is over four times faster to calculate than the polynomial basis. This difference originates largely from the numerical radial integration, which is required for the polynomial basis but not for the gaussian type orbital basis. The prediction performance of ACSF does not fall far behind SOAP and it might be possible to achieve the same accuracy by using a more advanced parameter calibration for the symmetry functions. The symmetry functions used in ACSF are easier to tune for capturing specific structural properties, such as certain pairwise distances or angles formed by three atoms. This tuning can, however, be done only if such intuition is available *a priori*, and in general consistently improving the performance by changing the used symmetry functions can be hard.

## 5. Conclusions

The recent boom in creating machine learnable fingerprints for atomistic systems, or descriptors, has led to a plethora of available options for materials science. The software implementation for these descriptors is, however, often scattered across different libraries or missing altogether, making it difficult to test and compare different alternatives.

We have collected several descriptors in the DScribe software library. DScribe has an easy-to-use python-interface, with C/C++ extensions for the computationally intensive tasks. We use a set of regression tests to ensure the validity of the implementation, and provide the source code together with tutorials and documentation. We have demonstrated the applicability of the package with the supervised learning tasks of formation energy prediction for crystals and the charge prediction for molecules. The DScribe descriptors are compatible with general-purpose machine learning algorithms, and can also be used for unsupervised learning tasks. In the future we plan to extend the package with new descriptors and also welcome external contributors.

## Acknowledgments

We acknowledge the computational resources provided by the Aalto Science-IT project. This project has received funding from the Jenny and Antti Wihuri Foundation and the European Union's Horizon 2020 research and innovation programme under grant agreements number no. 676580 NOMAD and no. 686053 CRITCAT.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2019.106949.

## References

[1] K. Takahashi, Y. Tanaka, Dalton Trans. 45 (26) (2016) 10497–10499.
[2] L. Zdeborová, Nat. Phys. 13 (5) (2017) 420–421.
[3] J.E. Gubernatis, T. Lookman, Phys. Rev. Mater. 2 (12) (2018) 120301.
[4] K.T. Butler, D.W. Davies, H. Cartwright, O. Isayev, A. Walsh, Nature 559 (7715) (2018) 547–555.
[5] L. Ward, R. Liu, A. Krishna, V.I. Hegde, A. Agrawal, A. Choudhary, C. Wolverton, Phys. Rev. B 96 (2) (2017) 1–12.
[6] H. Huo, M. Rupp, Unified Representation of Molecules and Crystals for Machine Learning, arXiv e-prints arXiv:1704.06439.
[7] F. Faber, A. Lindmaa, O.A.v. Lilienfeld, R. Armiento, Int. J. Quantum Chem. 115 (16) (2015) 1094–1101.
[8] A. Seko, H. Hayashi, K. Nakayama, A. Takahashi, I. Tanaka, Phys. Rev. B 95 (14) (2017) 144110.
[9] M. Rupp, A. Tkatchenko, K.-R. Müller, O.A. von Lilienfeld, Phys. Rev. Lett. 108 (2012) 058301.
[10] A. Stuke, M. Todorović, M. Rupp, C. Kunkel, K. Ghosh, L. Himanen, P. Rinke, J. Chem. Phys. 150 (20) (2019) 204121.
[11] K. Ghosh, A. Stuke, M. Todorović, P.B. Jørgensen, M.N. Schmidt, A. Vehtari, P. Rinke, Adv. Sci. 6 (9) (2019) 1801367.
[12] J.S. Smith, O. Isayev, A.E. Roitberg, Chem. Sci. 8 (2017) 3192–3203.
[13] A.P. Bartók, M.C. Payne, R. Kondor, G. Csányi, Phys. Rev. Lett. 104 (13) (2010) 1–4.
[14] A.P. Bartók, R. Kondor, G. Csányi, Phys. Rev. B 87 (2013) 184115.
[15] S. Chmiela, A. Tkatchenko, H.E. Sauceda, I. Poltavsky, K.T. Schütt, K.-R. Müller, Sci. Adv. 3 (5) (2017) e1603015.
[16] J. Behler, J. Chem. Phys. 134 (7) (2011) 074106.
[17] J. Behler, J. Chem. Phys. 145 (17) (2016) 170901.
[18] Y. Li, H. Li, F.C. Pickard, B. Narayanan, F.G. Sen, M.K.Y. Chan, S.K.R.S. Sankaranarayanan, B.R. Brooks, B. Roux, J. Chem. Theory Comput. 13 (9) (2017) 4492–4503.
[19] T.L. Jacobsen, M.S. Jørgensen, B. Hammer, Phys. Rev. Lett. 120 (2) (2018) 026102.
[20] Z. Li, S. Wang, W.S. Chin, L.E. Achenie, H. Xin, J. Mater. Chem. A 99 (2017) 016105.
[21] B.R. Goldsmith, J. Esterhuizen, J.X. Liu, C.J. Bartel, C. Sutton, AIChE J. 64 (7) (2018) 2311–2323.
[22] A.J. Chowdhury, W. Yang, E. Walker, O. Mamun, A. Heyden, G.A. Terejanu, J. Phys. Chem. C 122 (49) (2018) 28142–28150.
[23] M.O.J. Jäger, E.V. Morooka, F. Federici Canova, L. Himanen, A.S. Foster, NPJ Comput. Mater. 4 (2018) 37.
[24] A.F. Zahrt, J.J. Henle, B.T. Rose, Y. Wang, W.T. Darrow, S.E. Denmark, Science 363 (6424) (2019) eaau5631.
[25] S. Kiyohara, H. Oda, T. Miyata, T. Mizoguchi, Sci. Adv. 2 (11) (2016) e1600746.
[26] P. Zalake, S. Ghosh, S. Narasimhan, K.G. Thomas, Chem. Mater. 29 (17) (2017) 7170–7182.
[27] M. Todorović, M.U. Gutmann, J. Corander, P. Rinke, NPJ Comput. Mater. 5 (2019) 35.
[28] L.M. Ghiringhelli, J. Vybiral, S.V. Levchenko, C. Draxl, M. Scheffler, Phys. Rev. Lett. 114 (2015) 105503.
[29] T. Xie, J.C. Grossman, Phys. Rev. Lett. 120 (14) (2018) 145301.
[30] F. Arbabzadah, S. Chmiela, K.R. Müller, A. Tkatchenko, Nature Commun. 8 (2017) 6–13.
[31] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural Message Passing for Quantum Chemistry, Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1263–1272.
[32] K.T. Schütt, H.E. Sauceda, P.J. Kindermans, A. Tkatchenko, K.R. Müller, J. Chem. Phys. 148 (24) (2018) 241722.
[33] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, A.V. Lilienfeld, K.-R. Müller, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012, pp. 440–448.
[34] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O.A. Von Lilienfeld, K.R. Müller, A. Tkatchenko, J. Phys. Chem. Lett. 6 (12) (2015) 2326–2331.
[35] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, P. Marquetand, J. Chem. Phys. 148 (24) (2018) 241709.
[36] O. Isayev, C. Oses, C. Toher, E. Gossett, S. Curtarolo, A. Tropsha, Nature Commun. 8 (2017) 15679.
[37] F.A. Faber, A. Lindmaa, O.A. Von Lilienfeld, R. Armiento, Phys. Rev. Lett. 117 (13) (2016) 2–7.
[38] F.A. Faber, A.S. Christensen, B. Huang, O.A. von Lilienfeld, J. Chem. Phys. 148 (24) (2018) 241717.
[39] W. Pronobis, A. Tkatchenko, K.-R. Müller, J. Chem. Theory Comput. 14 (6) (2018) 2991–3003.
[40] K. Choudhary, B. DeCost, F. Tavazza, Phys. Rev. Mater. 2 (2018) 083801.
[41] F.A. Faber, L. Hutchison, B. Huang, J. Gilmer, S.S. Schoenholz, G.E. Dahl, O. Vinyals, S. Kearnes, P.F. Riley, O.A. Von Lilienfeld, J. Chem. Theory Comput. 13 (11) (2017) 5255–5264.
[42] R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler, L.M. Ghiringhelli, Phys. Rev. Mater. 2 (2018) 083802.
[43] A.S. Christensen, F.A. Faber, B. Huang, L.A. Bratholm, A. Tkatchenko, K.-R. Muller, O.A. von Lilienfeld, QML: A Python Toolkit for Quantum Machine Learning, URL https://github.com/qmlcode/qml, 2019.
[44] A. Khorshidi, A.A. Peterson, Comput. Phys. Comm. (ISSN: 0010-4655) 207 (2016) 310–324, URL http://www.sciencedirect.com/science/article/pii/S0010465516301266.
[45] Magpie: A Materials-Agnostic Platform for Informatics and Exploration, URL https://bitbucket.org/wolverton/magpie, 2019.
[46] QUIP and quippy documentation, URL http://libatoms.github.io/QUIP, 2019.
[47] M. Haghighatlari, G. Vishwakarma, D. Altarawy, R. Subramanian, B.U. Kota, A. Sonpal, S. Setlur, J. Hachmann, ChemRxiv (2019) 8323271.
[48] L. Ward, A. Dunn, A. Faghaninia, N. Zimmermann, S. Bajaj, Q. Wang, J. Montoya, J. Chen, K. Bystrom, M. Dylla, K. Chard, M. Asta, K. Persson, G. Snyder, I. Foster, A. Jain, Comput. Mater. Sci. (ISSN: 0927-0256) 152 (2018) 60–69.
[49] D.H. Wolpert, W.G. Macready, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[50] M. Ceriotti, G.A. Tribello, M. Parrinello, Proc. Natl. Acad. Sci. USA 108 (32) (2011) 13023–13028.
[51] S. De, A.P. Bartók, G. Csányi, M. Ceriotti, Phys. Chem. Chem. Phys. 18 (20) (2016) 13754–13769.
[52] O. Isayev, D. Fourches, E.N. Muratov, C. Oses, K. Rasch, A. Tropsha, S. Curtarolo, Chem. Mater. 27 (3) (2015) 735–743.
[53] R. Ramakrishnan, M. Hartmann, E. Tapavicza, O.A. von Lilienfeld, J. Chem. Phys. 143 (8) (2015) 084111.
[54] P.P. Ewald, Ann. Phys. 369 (3) (1921) 253–287.
[55] A.Y. Toukmaji, J.A. Board, Comput. Phys. Comm. 95 (2) (1996) 73–92.
[56] J.S. Hub, B.L. de Groot, H. Grubmüller, G. Groenhof, J. Chem. Theory Comput. 10 (1) (2014) 381–390.
[57] R.A. Jackson, C.R. Catlow, Mol. Simul. 1 (4) (1988) 207–224.
[58] N. Gerrits, K. Shakouri, J. Behler, G.-J. Kroes, J. Phys. Chem. Lett. 10 (2019) 1763–1768.
[59] N. Artrith, A. Urban, Comput. Mater. Sci. 114 (2016) 135–150.

[60] T.T. Nguyen, E. Székely, G. Imbalzano, J. Behler, G. Csányi, M. Ceriotti, A.W. Götz, F. Paesani, J. Chem. Phys. 148 (24) (2018) 241725.
[61] P. Löwdin, J. Chem. Phys. 18 (3) (1950) 365–375.
[62] M.J. Willatt, F. Musil, M. Ceriotti, A Data-Driven Construction of the Periodic Table of the Elements, arXiv e-prints arXiv:1807.00236.
[63] A.H. Larsen, J.J. Mortensen, J. Blomqvist, I.E. Castelli, R. Christensen, M. Dułak, J. Friis, M.N. Groves, B. Hammer, C. Hargus, E.D. Hermes, P.C. Jennings, P.B. Jensen, J. Kermode, J.R. Kitchin, E.L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J.B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K.S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K.W. Jacobsen, J. Phys.: Condens. Matter 29 (27) (2017) 273002.
[64] 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.
[65] L. van der Maaten, G. Hinton, J. Mach. Learn. Res. 9 (2008) 2579–2605.
[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[67] J.E. Saal, S. Kirklin, M. Aykol, B. Meredig, C. Wolverton, JOM 65 (11) (2013) 1501–1509.
[68] R. Ramakrishnan, P.O. Dral, M. Rupp, O.A. von Lilienfeld, Sci. Data 1 (2014) 140022.
[69] C. Chen, W. Ye, Y. Zuo, C. Zheng, S.P. Ong, Chem. Mater. 31 (9) (2019) 3564–3572.
[70] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.a. Persson, APL Mater. 1 (1) (2013) 011002.
[71] P. Bleiziffer, K. Schaller, S. Riniker, J. Chem. Inf. Model. 58 (3) (2018) 579–590.