Laakso, Jarno; Himanen, Lauri; Homm, Henrietta; Morooka, Eiaki V.; Jäger, Marc O.J.; Todorović, Milica; Rinke, Patrick

Updates to the DScribe library : New descriptors and derivatives

Published in:
Journal of Chemical Physics

DOI:
10.1063/5.0151031

Published: 21/06/2023

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Laakso, J., Himanen, L., Homm, H., Morooka, E. V., Jäger, M. O. J., Todorović, M., & Rinke, P. (2023). Updates to the DScribe library : New descriptors and derivatives. *Journal of Chemical Physics*, *158*(23), 1-8. Article 234802. https://doi.org/10.1063/5.0151031

The Journal of Chemical Physics

RESEARCH ARTICLE | JUNE 20 2023

Updates to the DScribe library: New descriptors and derivatives

Special Collection: Software for Atomistic Machine Learning

Jarno Laakso ; Lauri Himanen ; Henrietta Homm ; Eiaki V. Morooka; Marc O. J. Jäger; Milica Todorović ; Patrick Rinke

J. Chem. Phys. 158, 234802 (2023)
https://doi.org/10.1063/5.0151031

The Journal of Chemical Physics ARTICLE scitation.org/journal/jcp

Updates to the DScribe library: New descriptors and derivatives

Cite as: J. Chem. Phys. 158, 234802 (2023); doi: 10.1063/5.0151031
Submitted: 17 March 2023 • Accepted: 5 June 2023 •
Published Online: 20 June 2023

Jarno Laakso,[1] Lauri Himanen,[1] Henrietta Homm,[1] Eiaki V. Morooka,[1] Marc O. J. Jäger,[1] Milica Todorović,[2] and Patrick Rinke[1,a)]

AFFILIATIONS

[1] Department of Applied Physics, Aalto University, P.O. Box 11100, 00076 Aalto, Finland
[2] Department of Mechanical and Materials Engineering, University of Turku, FI-20014 Turku, Finland

Note: This paper is part of the JCP Special Topic on Software for Atomistic Machine Learning.
[a)] **Author to whom correspondence should be addressed:** patrick.rinke@aalto.fi

ABSTRACT

We present an update of the DScribe package, a Python library for atomistic descriptors. The update extends DScribe's descriptor selection with the Valle–Oganov materials fingerprint and provides descriptor derivatives to enable more advanced machine learning tasks, such as force prediction and structure optimization. For all descriptors, numeric derivatives are now available in DScribe. For the many-body tensor representation (MBTR) and the Smooth Overlap of Atomic Positions (SOAP), we have also implemented analytic derivatives. We demonstrate the effectiveness of the descriptor derivatives for machine learning models of Cu clusters and perovskite alloys.

I. INTRODUCTION

DScribe is a software library that provides atomistic descriptors to researchers in the natural sciences and engineering.[1] Descriptors represent the atomic structure of molecules, nanostructures, and materials in a machine-readable format. To facilitate machine learning (ML), descriptors should be invariant under transformations that conserve physical quantities, such as translations, mirroring, rotations, and atomic permutations.[2] Descriptors are also a powerful tool for defining distance metrics between atomic structures, which is helpful in many ML tasks, such as clustering or kernel-based regression. While ML is one of the main applications of descriptors, their usefulness is not limited to that. They can also be utilized, for example, in similarity analysis of atomic structures or visualization of atomistic data. In this article, we present new features that we have added to DScribe, including a new descriptor and the capability to calculate descriptor derivatives with respect to atomic positions.

At the time of its publication in 2019, DScribe included six descriptors: the Coulomb matrix,[3] the sine matrix,[4] the Ewald sum matrix,[4] the Many-Body Tensor Representation (MBTR),[5] the Atom-Centered Symmetry Functions (ACSF),[6] and the Smooth Overlap of Atomic Positions (SOAP).[7] The first four descriptors in this list are global descriptors, and the remaining two are local descriptors. DScribe made these descriptors available to a wide community and facilitated a variety of machine learning applications, including property prediction,[8–13] global structure search,[14] and data analysis and visualization.[15–17] All DScribe descriptors can output the representations in vector form, which makes them compatible with a multitude of existing ML methods and algorithms. Thus far, they have been employed, for example, with linear regression models,[9,12,13] neural networks,[8,9,11,12] random forests,[9,12] and Gaussian processes.[14]

In this work, we present a new descriptor we recently added to DScribe. It is based on a structural fingerprint proposed by Valle and Oganov for similarity analysis of crystal structures.[18] Since its formulation, it has been adopted for other applications. Bisbo and Hammer used the Valle–Oganov fingerprint to represent atomic structures in their global structure optimization algorithm.[19] Arrigoni and Madsen combined it with principal component analysis (PCA) for dimensionality reduction to facilitate data analysis.[20] According to its original definition, the Valle–Oganov fingerprint is constructed from the interatomic distances in an atomic structure. It greatly resembles the $k = 2$-term of the MBTR descriptor, but unlike MBTR, it was specifically tailored for periodic systems. The more specific use case reduces the number of user-defined parameters, which makes the Valle–Oganov descriptor easier to use than MBTR. Other studies have extended on the Valle–Oganov fingerprint by adding an angular term.[21] We also included such a higher-order

term in our implementation of the Valle–Oganov descriptor, but allow the user to decide whether to use it or not.

Structural descriptors have facilitated quick and accurate property predictions of molecules and materials using ML.[4,22,23] Many useful properties are derivatives of other quantities, which means that the same ML model can be used to predict multiple quantities. For atomic structures, gradients of energy give access to atomic forces. Differentiating an ML model for energy with respect to atomic positions would, therefore, also provide force predictions. This is the working principle of ML potentials, which are increasingly employed in simulating the dynamics of atomic systems.[24–26] Having access to the derivatives of an ML model also helps with optimization of the predicted property. Energy minimization, for example, is one of the most common tasks in computational chemistry and physics. Using an ML model to relax atomic positions by optimizing the surrogate potential energy surface instead of using a more expensive method such as density functional theory (DFT) saves computational resources. Derivative implementations that enable these ML tasks are already publicly available for some descriptors. The interatomic potential package QUIP,[27] for example, offers gradients for the SOAP descriptor, while Huo and Rupp published code for evaluating MBTR derivatives along with the article in which they proposed the descriptor.[5] The original aim of DScribe was to provide a large selection of atomistic descriptors in one package to make them more accessible to the research community. Here, we build on this principle and present our work on implementing descriptor derivatives in DScribe.

Descriptor derivatives can be calculated either analytically or numerically. Numerical derivatives are easy to implement and can be transferred to all descriptors in the library. Their disadvantage is the increased numerical cost and potential discretization errors. The number of descriptor evaluations required to calculate the numerical derivatives with respect to all atomic coordinates, for example, scales linearly with the system size. Analytical derivatives do not suffer from limited accuracy, and the computational time for calculating all the derivatives of a descriptor analytically is usually comparable to a single descriptor calculation. The downside of analytical derivatives is that they need to be implemented separately for every descriptor, which can be very tedious. Here, we have implemented analytical derivatives for SOAP and MBTR and numerical derivatives for all descriptors. We demonstrate the effectiveness of the SOAP derivatives by employing them in a neural network model to fit a simple ML potential. We also showcase the application of MBTR derivatives in a structure optimization task.

This article is organized as follows: Sec. II presents a description of the numerical and analytical descriptor derivatives and the Valle–Oganov descriptor. In Sec. III, we elaborate on the practical implementation of the new features in DScribe and provide a guide on their usage. Section IV presents the results of tests that we conducted to make sure that the descriptor derivatives function properly and showcase their usefulness with two demonstrations. Finally, in Sec. V, we conclude our work.

II. METHODS

In this section, we detail the computational methods behind the features that we added to DScribe. We describe our approach for computing numerical descriptor derivatives with the central difference method and detail our implementations of the analytic SOAP and MBTR derivatives. We then briefly present two ML tasks that demonstrate the effectiveness of our implementation. We close the section by introducing the additional descriptor that is based on the structural fingerprint by Valle and Oganov and showing how we implemented it with small changes to the existing MBTR implementation.

A. Descriptor derivatives

1. Numerical derivatives

We collect the atomic positions in the coordinate matrix $\boldsymbol{R}$ with the shape $(N, 3)$, where N is the number of atoms in the system and 3 corresponds to the Cartesian coordinates x, y, and z. Any atomic representation $F(\boldsymbol{R})$ can be differentiated numerically using the central finite difference method, according to which the derivative of $F(\boldsymbol{R})$ with respect to the coordinate j of atom i is approximately

$$\frac{\partial F(\boldsymbol{R})}{\partial \boldsymbol{R}_{ij}} \approx \frac{F\left(\boldsymbol{R} + \frac{h}{2}\boldsymbol{E}^{ij}\right) - F\left(\boldsymbol{R} - \frac{h}{2}\boldsymbol{E}^{ij}\right)}{h}, \tag{1}$$

where $\boldsymbol{E}^{ij}$ is a single-entry matrix of the same shape as $\boldsymbol{R}$ with one non-zero element

$$(\boldsymbol{E}^{ij})_{ab} = \begin{cases} 1 & \text{when } a = i \text{ and } b = j, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

and h quantifies the magnitude of the atomic displacement. The error made by the central difference approximation is proportional to h^2, which means that the accuracy of the numerical derivatives improves quickly with smaller values of h. In practice, however, the limited floating-point accuracy makes the derivatives unstable when h is too small. In order to determine the optimal value of h for our implementation, we conducted a test comparing the numerical derivatives computed with different h values to the analytical derivatives that we derived for MBTR and SOAP descriptors. We calculated the relative error between the numerical and analytical derivatives using the mean relative percentage difference,

$$\text{MRPD} = \frac{2}{3NM}\sum_{i,j,k}\left|\frac{\partial^{\text{a}}_{ij}F_k - \partial^{\text{n}}_{ij}F_k}{|\partial^{\text{a}}_{ij}F_k| + |\partial^{\text{n}}_{ij}F_k|}\right|, \tag{3}$$

where i iterates over the atoms in an atomic structure, j runs over the three Cartesian coordinates, and k runs over the components of the representation vector. N is the number of atoms in the structure, and M is the number of components in the representation vector. ∂^{a}_{ij} and ∂^{n}_{ij} are the analytical and numerical derivatives with respect to coordinate $\boldsymbol{R}_{ij}$, respectively. The descriptor computations for the test were carried out using 64-bit floating-point numbers, which is the highest accuracy supported by DScribe.

2. MBTR derivatives

In DScribe, three MBTR terms are implemented, each corresponding to atomic motifs of a different size. The $k = 1$ term encodes single atoms, the $k = 2$ term encodes atomic pairs, and the $k = 3$ term encodes atom triples. Each term is a function of a continuous variable x,

$$F(x) = N\sum_i w^i d^i(x), \tag{4}$$

where the sum runs over the motifs in an atomic structure. N is a normalization term, w^i is the weight related to motif i and $d^i(x)$ is a distribution function. DScribe uses the Gaussian distribution,

$$d^i(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-g^i)^2}{2\sigma^2}\right), \tag{5}$$

where σ is the standard deviation of the distribution and g^i is a function that maps the atomic motif i to a scalar value. Assuming that N is independent of the atomic positions, the gradient of the representation function with respect to the coordinates of an atom is

$$\nabla F(x) = N\sum_i d^i(x)\left[\nabla w^i + w^i \frac{1}{\sigma^2}(x-g^i)\nabla g^i\right]. \tag{6}$$

∇w^i and ∇g^i depend on the choice of the weighting and geometry functions. The assumption of N being independent of atomic positions is not true for the normalization option `l2 each` that normalizes the L2-norm of the representation vector to one, and analytical derivatives are currently not available for it. For a detailed derivation of the MBTR gradients with the different options for w^i and g^i, we refer to Sec. S2.B of the supplementary material.

To demonstrate the analytical MBTR derivatives, we used them in the geometry optimization of perovskite materials. We fitted an energy ML model that combines the MBTR with kernel ridge regression (KRR) for a $CsPb(Cl/Br)_3$ dataset. We generated the dataset and used the same ML model for an earlier study that contains detailed information on model fitting and structure optimization.[28] For this demonstration, we optimized the atomic positions of 25 perovskite structures using the ML model and compared the obtained energies and geometries to DFT relaxation results.

3. SOAP derivatives

The SOAP descriptor represents local atomic environments in a rotationally invariant way by expanding Gaussian smeared atomic densities on the basis of spherical harmonics and radial basis functions. In DScribe, SOAP outputs a vector of partial power spectra[29] $\boldsymbol{p}$, where the individual components are defined as

$$p^{Z_1,Z_2}_{nn'l} = \pi\sqrt{\frac{8}{2l+1}}\sum_m \left(c^{Z_1}_{nlm}\right)^* c^{Z_2}_{n'lm} \tag{7}$$

$$= \pi\sqrt{\frac{8}{2l+1}}\sum_m \left(\sum_j^{|Z_1|} c^j_{nlm}\right)\left(\sum_k^{|Z_2|} c^k_{n'lm}\right). \tag{8}$$

The complex conjugation in (7) can be omitted because DScribe uses real spherical harmonics. The summations for j and k run over atoms in the environment with the atomic numbers Z_1 and Z_2, respectively. The coefficients c^i_{nlm} are defined as

$$c^i_{nlm} = \iiint_{\mathcal{R}^3} \mathrm{d}V g_{nl}(r)\rho(\boldsymbol{r},\boldsymbol{R}_i)Y_{lm}(\theta,\phi), \tag{9}$$

where $\rho(\boldsymbol{r},\boldsymbol{R}_i)$ is the Gaussian smeared atomic density, $g_{nl}(r)$ is a radial basis function, and $Y_{lm}(\theta,\phi)$ is a spherical harmonic. The vector $\boldsymbol{p}$ consists of elements $p^{Z_1,Z_2}_{nn'l}$ for all unique atomic number pairs (Z_1,Z_2), and unique combinations of radial and spherical basis functions (n,n',l).

The gradient of $p^{Z_1,Z_2}_{nn'l}$ with respect to the coordinates of an atom is

$$\nabla p^{Z_1,Z_2}_{nn'l} = \pi\sqrt{\frac{8}{2l+1}}\sum_m \left[\left(\sum_j^{|Z_1|}\nabla c^j_{nlm}\right)\sum_k^{|Z_2|} c^k_{n'lm} + \sum_j^{|Z_1|} c^j_{nlm}\left(\sum_k^{|Z_2|}\nabla c^k_{n'lm}\right)\right]. \tag{10}$$

The final derivative equation depends on the choice of the radial basis function. For now, we have implemented the analytical derivatives for spherical primitive Gaussian type orbitals in the non-periodic case. For polynomial radial basis functions and periodic systems, numerical differentiation is used instead. See Sec. S3.B of the supplementary material for the full derivation of the SOAP gradients.

We tested the SOAP derivatives in an ML potential model that we trained for Cu clusters. We generated data for training and testing the ML potential by running a classical molecular dynamics simulation of a 55-atom Cu cluster at 500 K for 5.0 ns and uniformly picking 10,000 snapshots from the simulation. The simulation used a 5.0 fs timestep, embedded atom method[30] (EAM) for the Cu interactions, and a Nose–Hoover thermostat[31] for the temperature control. It was performed using the LAMMPS simulation tool.[32]

We built an ML model that first represents all 55 atomic environments in the vector form using SOAP, and then uses a feedforward neural network to map the atomic environments to atomic energies, which are summed to the total energy. To make model training easier, we decreased the dimensionality of the SOAP vectors with principal component analysis (PCA) following an earlier study by Zhou *et al.*[9] The full ML model architecture is shown in Fig. 1.

To generate the SOAP vectors, we used a radial cutoff of 6.0 Å and a basis of eight radial basis functions and six spherical harmonics. These settings result in 252-dimensional SOAP vectors. We reduced the dimensionality of these vectors to 50 with PCA. The neural network that we used to map these reduced SOAP vectors to atomic energies had two hidden layers. Both hidden layers had 50 nodes that used the sigmoid activation function. The output layer used linear activation. We implemented the neural network using Keras[33] and Tensorflow[34] Python packages. The weights of the network were optimized using the Adam algorithm.[35]

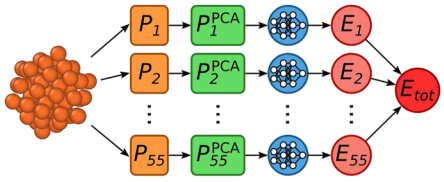

FIG. 1. The ML model for Cu cluster energy prediction. All 55 atomic environments of a Cu cluster are first represented in vector form using SOAP (P_i). Then, the dimensionality of these vectors is reduced with PCA. The PCA-reduced SOAP vectors P_i^{PCA} are mapped to atomic energies E_i with a feedforward neural network with two hidden layers. Finally, E_i are summed together to obtain the total potential energy E_{tot} of the Cu cluster.

We trained the ML potential on total energies of 8000 Cu clusters. Then, we assessed the quality of the fit by predicting the total energies of the remaining 2000 clusters and compared the results to the EAM energies. Atomic forces are derivatives of the total energy with respect to the atomic positions. By combining our implementation of the SOAP derivatives with the gradients of the neural network, we were able to use the ML model to predict atomic forces. We used the ML model to predict the forces of all 2000 test set clusters and compared the results to the EAM forces.

B. Valle–Oganov descriptor

In their article, Valle and Oganov defined an atomic structure representation for periodic systems.[18] The full representation has a distinct term for every unique pair of elements (A and B) present within the structure. Each of these terms is a function,

$$F_{AB}(x) = \sum_{A_i B_j} \frac{d^{i,j}(x)}{4\pi r_{ij}^2 (N_A N_B / V)} - 1. \tag{11}$$

The index i runs over all atoms of type A and j over all atoms of type B. r_{ij} is the distance between atoms i and j, V is the volume of the cell, and N_A and N_B are the number of atoms of each type. $d^{i,j}(x)$ is the Gaussian distribution of Eq. (5) with the geometry function set to be the distance between the atoms $g^{i,j} = r_{ij}$.

A closer inspection of Eq. (11) reveals a close similarity to the MBTR, which is already implemented in DScribe. The only fundamental difference between the two representations is the constant term −1 in the Valle–Oganov representation. The constant term, however, is not significant in most use cases as it does not affect the distances between the vectors. Furthermore, if the constant term is needed for some application, it can be added to the representation vectors afterward. We, therefore, decided to omit the constant term −1 from our implementation. Now, $F_{AB}(x)$ can be cast into the general MBTR formalism defined in Eq. (4) by setting the weighting function to

$$w^{i,j} = \frac{1}{r_{ij}^2} \tag{12}$$

and the normalization term to

$$N_{AB} = \frac{V}{4\pi N_A N_B}. \tag{13}$$

The Valle–Oganov descriptor implementation in DScribe, therefore, only requires the addition of new weighting and normalization options to the already existing MBTR implementation.

The original article by Valle and Oganov only considers interatomic distances, but the representation has been extended with an angular term by Bisbo and Hammer.[21] We define a similar third-order term $F_{ABC}(x)$, again utilizing the MBTR formalism. Now, the sum in Eq. (4) runs over atom triplets (i, j, k) and the geometry function in Eq. (5) is the angle between the atoms $g^{i,j,k} = \theta_{ijk}$. The normalization term is

$$N_{ABC} = \frac{V}{4\pi N_A N_B N_C}, \tag{14}$$

and the weight function is

$$w^{i,j,k} = f_c(r_{ij}) f_c(r_{ik}), \tag{15}$$

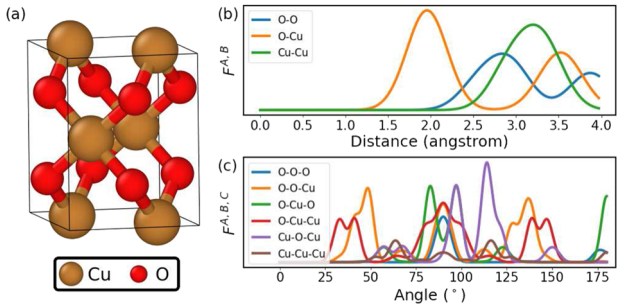

FIG. 2. Valle–Oganov descriptor construction. (a) CuO atomic structure. (b) Second-order and (c) third-order Valle–Oganov fingerprints of CuO. The full Valle–Oganov representation vector is obtained by concatenating the elemental contributions.

where

$$f_c(r) = 1 + \gamma \left(\frac{r}{r_{\text{cut}}}\right)^{\gamma+1} - (\gamma + 1)\left(\frac{r}{r_{\text{cut}}}\right)^{\gamma}. \tag{16}$$

The weight function and its derivative approach 0 when r_{ij} or r_{ik} approaches the cutoff distance r_{cut}. If either of the two distances is larger than r_{cut}, the atom triplet does not contribute to the representation. γ controls the sharpness of the cutoff, and in our implementation, it has a default value of 2.

Figure 2 shows the Valle–Oganov fingerprint of CuO. The full representation vector is obtained by concatenating the different elemental contributions of the second- and third-order terms. Although we use the MBTR framework for the Valle–Oganov descriptor, analytical derivatives have not yet been fully implemented for its normalization and weighting options and numerical differentiation is used instead.

III. SOFTWARE STRUCTURE

Python has solidified its position as a go-to language for several domains, including data science. In order to seamlessly integrate with these Python-based data science workflows, the main entry point for our software is a Python API through which the descriptors can be configured and used. Figure 3 shows an example of the new DScribe interface for calculating derivatives.

DScribe works with atomic structures defined using the `ase.Atoms`-object from the `ase` package.[36] These objects are easy to create from existing structure files or to build with the utilities provided by `ase`. In addition to any descriptor-specific arguments, all descriptors accept the `sparse`-parameter that controls whether the created output is a dense or a sparse matrix. Especially in large systems where the interactions between atoms and centers of interest are very localized, sparsity provides memory and storage efficiency. Some ML algorithms can use sparse matrix formats directly, but it is also easy to restore the dense format for other algorithms.

All descriptors implement the new `derivatives` method. The first argument accepts one or multiple atomic structures. The argument positions can be used to define the `positions` of interest for local descriptors. It defaults to using all individual atoms as centers and cannot be specified for global descriptors. The arguments

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

# Define the atomic systems
systems = [molecule("H2"), molecule("O2")]

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

# Calculate derivatives and descriptor features
derivatives, features = soap.derivatives(
    systems,
    positions=None,
    include=[0],
    exclude=None,
    method="auto",
    return_descriptor=True,
    n_jobs=2
)
```

FIG. 3. Example of creating derivatives with DScribe. All descriptors have the derivatives-function that can be used to retrieve both derivatives and descriptor features simultaneously. Only the first argument that specifies the used systems is required, and the additional arguments can be used to further control the methodology, parallelization, and with respect to which atoms the positional derivatives are calculated for.

`include` and `exclude` are used to control which atoms are considered in the derivative calculations. By using this method, the user can explicitly change between analytical and numerical differentiation. The default value auto will automatically choose the analytical implementation if it is available, and the numerical one otherwise. Descriptors and their derivatives can be created simultaneously by setting `return_descriptor=True`, as this is often computationally favorable. Finally, descriptor calculations can be parallelized over several CPU cores using the `n_jobs` parameter. This parallelization is based on evenly distributing the given atomic structures to different cores for data parallelism.

We have decided to retain as much structure in the derivative output as possible. This approach allows the user to better understand and access the different components, while it is still relatively easy to re-arrange the output into a lower-dimensional shape if needed. Generally, the output is a multidimensional array with the shape `[n_systems, n_centers, n_atoms, 3, n_features]`. Here, the dimension with `n_systems` runs over the different atomic structures, `n_centers` loops through the different centers of interest, `n_atoms` loops through the atoms for which the derivatives were calculated, the second-to-last dimension with three components loops through the x, y, and z components, and the last dimension with `n_features` loops through the different descriptor features. Global descriptors effectively have only one region of interest that covers the entire structure, meaning that `n_centers` = 1, and the corresponding dimension is not present. Similarly, when creating the derivatives only for one system, `n_systems` = 1, and that dimension is left out.

As many of the descriptor calculations require significant CPU resources, many of the heavier calculations are internally handled by an underlying C++ implementation that is accessed through the Python interface. This hybrid approach is similar to many other common numerical Python packages such as `numpy`[37] and `scipy`.[38] The communication between Python and C++ is implemented using the `pybind11`[39] library.

The source code is structured using an object-oriented programming approach. Each descriptor is represented by its own class, which inherits from a generic base class. When adhering to the base class interface, the subclasses can automatically take advantage of the functionality already defined in the base class—such as the numerical derivatives—in addition to ensuring that the user can expect each descriptor to have similar functionality. Each descriptor class is associated with a code test suite that is used to ensure the validity of the implementation. This test suite forms part of a continuous integration pipeline that is performed every time the source code is modified.

DScribe is distributed as a Python package and can be installed from the Python package index (PyPI)[40] under the package name `DScribe`. Alternatively, the package can be installed using the `conda-forge`[41] ecosystem, where it is distributed with the same name. Access to the full source code is also provided through GitHub at https://github.com/SINGROUP/dscribe. The full documentation and several tutorials are available on the DScribe homepage https://singroup.github.io/dscribe/.

IV. RESULTS AND DISCUSSION

This section showcases the outcomes of the tests conducted on the descriptor derivatives. First, we assessed the accuracy of our numerical derivative implementation by comparing it to analytical derivative values. We used the results of the test to determine the optimal h-value for DScribe's numerical derivative implementation. We then present MBTR derivatives for the perovskite structures and the SOAP derivative test for the Cu cluster.

A. Numerical descriptor derivatives

We tested the accuracy of our numerical descriptor derivative implementation by comparing it to analytical MBTR and SOAP derivatives. Figure 4 shows the relative error between the numerical and analytical derivatives for a water molecule. The errors are at

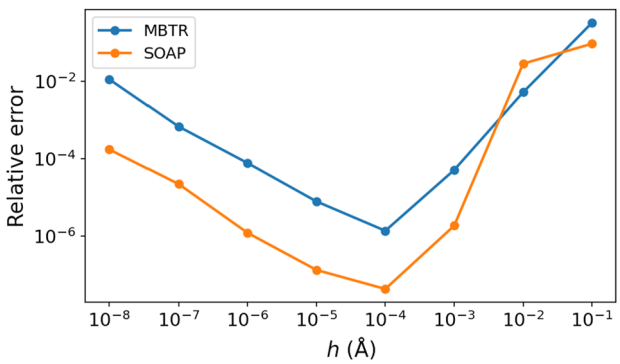

FIG. 4. Relative error of numerical derivatives compared to analytical MBTR and SOAP derivatives with different central difference step sizes h.

their highest at the higher end of the tested $h-$ range. With decreasing h, the relative error reduces due to the h^2 scaling of the central difference error. The MBTR and SOAP errors both reach their minima at $h = 1 \times 10^{-4}$ Å, after which they start to increase again due to the limited floating-point accuracy. For both descriptors, the relative error reaches 10^{-6} at its lowest. In practical applications, this is likely to be insignificant compared to errors related to model fitting or the underlying data. Based on the results of the test, we fixed $h = 1 \times 10^{-4}$ Å for the numerical derivatives of all descriptors in DScribe. The good agreement between the numerical and analytical derivatives shows that the numerical derivative implementation is highly accurate and that our analytical MBTR and SOAP derivatives are error-free.

B. Perovskite structure optimization with MBTR

To showcase the analytical MBTR derivatives, we fitted an ML model that combines MBTR and KRR to predict the energies of $CsPb(Cl/Br)_3$ perovskite structures. To assess the accuracy of the fit, we used the model to predict the energies and forces of structure snapshots from DFT relaxation of 25 perovskite test structures. The mean absolute error (MAE) of energy predictions was only 0.14 meV/atom, while the force prediction MAE was 16.7 meV/Å.

We then used the derivatives of the trained ML model to relax the atomic positions of the same 25 $CsPb(Cl/Br)_3$ test geometries. Figure 5(a) shows an example of how the ML predicted energy of a perovskite structure decreases during ML relaxation and how that compares to the DFT relaxation of the same structure. The DFT relaxation reaches the minimum structure in fewer iterations than the ML model, but the final energies are very close, deviating only by 0.42 meV/atom. The final structures from the two relaxation methods are almost identical, with the root-mean-square deviation (RMSD) of atomic positions between them being only 0.016 Å. Furthermore, relaxing the structure with the ML model is over four orders of magnitude faster than with DFT.

In Fig. 5(b), we plot the relaxed energies of all 25 perovskite test structures. For most structures, the DFT and ML optimized energies are nearly identical, although, in some cases, the energies differ by up to 0.70 meV/atom. The mean absolute error between the two energies is 0.26 meV/atom, and the average RMSD of atomic positions is 0.031 Å. The good agreement in terms of both energy and structure

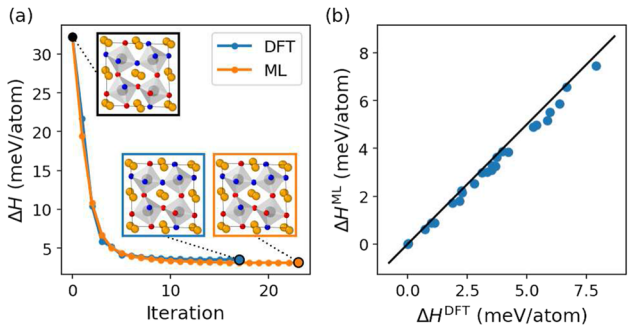

FIG. 5. Results from perovskite structure optimization tests utilizing MBTR gradients. (a) One perovskite structure optimized with both DFT and the ML model. (b) Comparison of ML and DFT optimized energies of 25 perovskite structures.

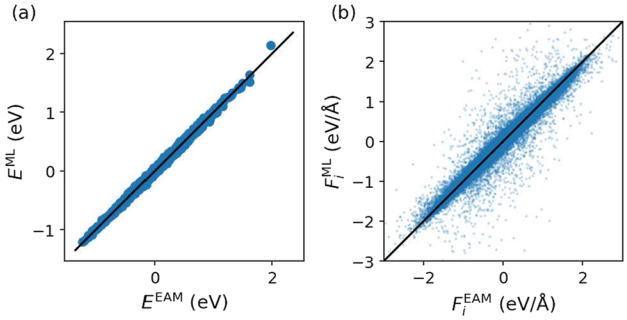

FIG. 6. ML model predictions on the 2000 Cu clusters. (a) ML predicted energies compared to EAM values. (b) Atomic force components compared to EAM forces.

demonstrates that an ML approach utilizing descriptor derivatives can effectively accelerate structure optimization computations.

C. Cu cluster ML potential using SOAP

We assessed the accuracy of the Cu cluster model fit by predicting the total potential energies of 2,000 Cu test clusters and comparing them to EAM energies. Figure 6(a) shows the results of this comparison. The absolute error of the ML model predictions is only 0.34 meV/atom on average. Next, we tested the force prediction accuracy of the ML model by comparing the ML-predicted atomic forces of the same 2,000 Cu clusters to the EAM forces. Figure 6(b) shows atomic force components computed with both methods. The mean absolute error of the predictions is 39 meV/Å.

By combining the SOAP descriptor with a simple neural network architecture, we were able to achieve energy predictions with quantum mechanical precision. The accuracy of the force predictions is lower in comparison, falling behind the state of the art ML potentials while being comparable to classical force fields. To assess whether the model accuracy could be improved by increasing the amount of training data, we computed learning curves for the model (see Fig. S1 of the supplementary material). The results of our test show that the model prediction errors are not yet converged with the full training set of 8000 structures, and training the ML model with more data would improve its accuracy. Here, we trained the ML model on classical MD data for demonstrative purposes, and no speedup was achieved. The same methodology, however, could be applied on data from a more accurate method, such as DFT, in which it would greatly accelerate simulations.

V. CONCLUSIONS

We have presented an update of the DScribe package. We have introduced a new structural representation, expanding DScribe's descriptor selection. In addition, we have extended the capabilities of DScribe by implementing descriptor derivatives. We have compared the accuracy of our numerical derivative implementation to analytical MBTR and SOAP derivatives and observe relative errors of less than 10^{-6}. We have also demonstrated the effectiveness of the analytical derivative implementations through two machine learning tasks involving force prediction and structure optimization. Our results show that our derivative implementations are accurate and effective, and we believe that the new version of DScribe will be a

valuable tool for researchers applying machine learning to materials science problems.

SUPPLEMENTARY MATERIAL

See the supplementary material for the detailed derivation of the analytical MBTR and SOAP derivatives.

ACKNOWLEDGMENTS

We acknowledge the funding from the European Union's Horizon program under Grant Agreement No. 951786, the Academy of Finland through Project No. 334532, and the Center of Excellence Virtual Laboratory for Molecular Level Atmospheric Transformations (VILMA; Project No. 346377). We further acknowledge the CSC-IT Center for Science, Finland, and the Aalto Science-IT project.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Jarno Laakso: Methodology (equal); Software (equal); Visualization (lead); Writing – original draft (lead). **Lauri Himanen**: Conceptualization (equal); Software (lead); Writing – original draft (equal). **Henrietta Homm**: Software (equal). **Eiaki V. Morooka**: Software (equal). **Marc O. J. Jäger**: Software (equal). **Milica Todorović**: Conceptualization (equal); Supervision (equal); Writing – review & editing (equal). **Patrick Rinke**: Conceptualization (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The newest version of the DScribe package is available through GitHub at https://github.com/SINGROUP/dscribe. All the codes related to the perovskite and Cu cluster ML model fitting and testing are available through a GitLab repository at https://gitlab.com/cest-group/dscribe-derivative-examples.

REFERENCES

[1]L. Himanen, M. O. J. Jäger, E. V. Morooka, F. Federici Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke, and A. S. Foster, "Dscribe: Library of descriptors for machine learning in materials science," Comput. Phys. Commun. **247**, 106949 (2020).

[2]M. F. Langer, A. Goeßmann, and M. Rupp, "Representations of molecules and materials for interpolation of quantum-mechanical simulations via machine learning," npj Comput. Mater. **8**, 41 (2022).

[3]M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning," Phys. Rev. Lett. **108**, 058301 (2012).

[4]F. Faber, A. Lindmaa, O. A. von Lilienfeld, and R. Armiento, "Crystal structure representations for machine learning models of formation energies," Int. J. Quantum Chem. **115**, 1094–1101 (2015).

[5]H. Huo and M. Rupp, "Unified representation of molecules and crystals for machine learning," Mach. Learn.: Sci. Technol. **3**, 045017 (2022).

[6]J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," J. Chem. Phys. **134**, 074106 (2011).

[7]A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," Phys. Rev. B **87**, 184115 (2013).

[8]V. Fung, J. Zhang, E. Juarez, and B. G. Sumpter, "Benchmarking graph neural networks for materials chemistry," npj Comput. Mater. **7**, 84 (2021).

[9]X.-Y. Zhou, J.-H. Zhu, Y. Wu, X.-S. Yang, T. Lookman, and H.-H. Wu, "Machine learning assisted design of FeCoNiCrMn high-entropy alloys with ultra-low hydrogen diffusion coefficients," Acta Mater. **224**, 117535 (2022).

[10]A. Pihlajamäki, J. Hämäläinen, J. Linja, P. Nieminen, S. Malola, T. Kärkkäinen, and H. Häkkinen, "Monte Carlo simulations of $Au_{38}(SCH_3)_{24}$ nanocluster using distance-based machine learning methods," J. Phys. Chem. A **124**, 4827–4836 (2020).

[11]O. Rahaman and A. Gagliardi, "Deep learning total energies and orbital energies of large organic molecules using hybridization of molecular fingerprints," J. Chem. Inf. Model. **60**, 5971–5983 (2020).

[12]Q. Sun, Y. Xiang, Y. Liu, L. Xu, T. Leng, Y. Ye, A. Fortunelli, W. A. Goddard, and T. Cheng, "Machine learning predicts the x-ray photoelectron spectroscopy of the solid electrolyte interface of lithium metal battery," J. Phys. Chem. Lett. **13**, 8047–8054 (2022).

[13]H. Hirai, T. Iizawa, T. Tamura, M. Karasuyama, R. Kobayashi, and T. Hirose, "Machine-learning-based prediction of first-principles XANES spectra for amorphous materials," Phys. Rev. Mater. **6**, 115601 (2022).

[14]M. P. Lourenço, L. B. Herrera, J. Hostaš, P. Calaminici, A. M. Köster, A. Tchagang, and D. R. Salahub, "Taking the multiplicity inside the loop: Active learning for structural and spin multiplicity elucidation of atomic clusters," Theor. Chem. Acc. **140**, 116 (2021).

[15]L. Sun, Y.-X. Zhou, X.-D. Wang, Y.-H. Chen, V. L. Deringer, R. Mazzarello, and W. Zhang, "*Ab initio* molecular dynamics and materials design for embedded phase-change memory," npj Comput. Mater. **7**, 29 (2021).

[16]B. Cheng, R.-R. Griffiths, S. Wengert, C. Kunkel, T. Stenczel, B. Zhu, V. L. Deringer, N. Bernstein, J. T. Margraf, K. Reuter, and G. Csanyi, "Mapping materials and molecules," Acc. Chem. Res. **53**, 1981–1991 (2020).

[17]B. Monserrat, J. G. Brandenburg, E. A. Engel, and B. Cheng, "Liquid water contains the building blocks of diverse ice phases," Nat. Commun. **11**, 5757 (2020).

[18]M. Valle and A. R. Oganov, "Crystal fingerprint space—a novel paradigm for studying crystal-structure sets," Acta Crystallogr. A **66**, 507–517 (2010).

[19]M. K. Bisbo and B. Hammer, "Efficient global structure optimization with a machine-learned surrogate model," Phys. Rev. Lett. **124**, 086102 (2020).

[20]M. Arrigoni and G. K. H. Madsen, "Evolutionary computing and machine learning for discovering of low-energy defect configurations," npj Comput. Mater. **7**, 71 (2021).

[21]M. K. Bisbo and B. Hammer, "Global optimization of atomic structure enhanced by machine learning," Phys. Rev. B **105**, 245404 (2022).

[22]A. Stuke, M. Todorović, M. Rupp, C. Kunkel, K. Ghosh, L. Himanen, and P. Rinke, "Chemical diversity in molecular orbital energy predictions with kernel ridge regression," J. Chem. Phys. **150**, 204121 (2019).

[23]Y. Jiang, D. Chen, X. Chen, T. Li, G.-W. Wei, and F. Pan, "Topological representations of crystalline compounds for the machine-learning prediction of materials properties," npj Comput. Mater. **7**, 28 (2021).

[24]J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," Phys. Rev. Lett. **98**, 146401 (2007).

[25]A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons," Phys. Rev. Lett. **104**, 136403 (2010).

[26]K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "SchNet—a deep learning architecture for molecules and materials," J. Chem. Phys. **148**, 241722 (2018).

[27]G. Csányi, S. Winfield, J. R. Kermode, A. De Vita, A. Comisso, N. Bernstein, and M. C. Payne, "Expressive programming for computational physics in Fortran 95+," IoP Computational Physics Newsletter, Spring 2007 (2007).
[28]J. Laakso, M. Todorović, J. Li, G.-X. Zhang, and P. Rinke, "Compositional engineering of perovskites with machine learning," Phys. Rev. Mater. **6**, 113801 (2022).
[29]S. De, A. P. Bartók, G. Csányi, and M. Ceriotti, "Comparing molecules and solids across structural and alchemical space," Phys. Chem. Chem. Phys. **18**, 13754–13769 (2016).
[30]S. M. Foiles, M. I. Baskes, and M. S. Daw, "Embedded-atom-method functions for the fcc metals Cu, Ag, Au, Ni, Pd, Pt, and their alloys," Phys. Rev. B **33**, 7983–7991 (1986).
[31]W. G. Hoover, "Canonical dynamics: Equilibrium phase-space distributions," Phys. Rev. A **31**, 1695–1697 (1985).
[32]A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, "LAMMPS—A flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," Comput. Phys. Commun. **271**, 108171 (2022).
[33]F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[34]M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," arXiv:1603.04467 (2015).
[35]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv.1412.6980 (2014).
[36]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, and K. W. Jacobsen, "The atomic simulation environment—A Python library for working with atoms," J. Phys.: Condens. Matter **29**, 273002 (2017).
[37]C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," Nature **585**, 357–362 (2020).
[38]P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nat. Methods **17**, 261–272 (2020).
[39]W. Jakob, J. Rhinelander, and D. Moldovan, "pybind11 – seamless operability between C++11 and Python," https://github.com/pybind/pybind11, 2017.
[40]See https://pypi.org/ for Python Package Index—PyPI.
[41]Conda-Forge Community (2015). "The Conda-Forge Project: Community-based software distribution built on the Conda Package format and ecosystem," Zenodo. https://doi.org/10.5281/zenodo.4774217