The Journal of Chemical Physics

Efficient implementation of atom-density representations

Cite as: J. Chem. Phys. **154**, 114109 (2021); https://doi.org/10.1063/5.0044689
Submitted: 18 January 2021 • Accepted: 22 February 2021 • Published Online: 16 March 2021

Félix Musil, Max Veit, Alexander Goscinski, et al.

ARTICLES YOU MAY BE INTERESTED IN

Atom-density representations for machine learning
The Journal of Chemical Physics **150**, 154110 (2019); https://doi.org/10.1063/1.5090481

When do short-range atomistic machine-learning models fall short?
The Journal of Chemical Physics **154**, 034111 (2021); https://doi.org/10.1063/5.0031215

SchNet – A deep learning architecture for molecules and materials
The Journal of Chemical Physics **148**, 241722 (2018); https://doi.org/10.1063/1.5019779

AIP Publishing

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

Efficient implementation of atom-density representations

Cite as: J. Chem. Phys. 154, 114109 (2021); doi: 10.1063/5.0044689
Submitted: 18 January 2021 • Accepted: 22 February 2021 •
Published Online: 16 March 2021

View Online Export Citation CrossMark

Félix Musil,[1,2,a)] Max Veit,[1,2,b)] Alexander Goscinski,[1] Guillaume Fraux,[1] Michael J. Willatt,[1] Markus Stricker,[3,4] Till Junge,[3] and Michele Ceriotti[1,c)]

AFFILIATIONS

[1] Laboratory of Computational Science and Modeling, Institute of Materials, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

[2] National Center for Computational Design and Discovery of Novel Materials (MARVEL), Lausanne, Switzerland

[3] Laboratory for Multiscale Mechanics Modeling, Institute of Mechanical Engineering, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

[4] Interdisciplinary Centre for Advanced Materials Simulation, Ruhr-University Bochum, Universitätsstraße 150, 44801 Bochum, Germany

[a)] **Electronic mail:** felix.musil@epfl.ch
[b)] **Electronic mail:** max.veit@epfl.ch
[c)] **Author to whom correspondence should be addressed:** michele.ceriotti@epfl.ch

ABSTRACT

Physically motivated and mathematically robust atom-centered representations of molecular structures are key to the success of modern atomistic machine learning. They lie at the foundation of a wide range of methods to predict the properties of both materials and molecules and to explore and visualize their chemical structures and compositions. Recently, it has become clear that many of the most effective representations share a fundamental formal connection. They can all be expressed as a discretization of n-body correlation functions of the local atom density, suggesting the opportunity of standardizing and, more importantly, optimizing their evaluation. We present an implementation, named `librascal`, whose modular design lends itself both to developing refinements to the density-based formalism and to rapid prototyping for new developments of rotationally equivariant atomistic representations. As an example, we discuss smooth overlap of atomic position (SOAP) features, perhaps the most widely used member of this family of representations, to show how the expansion of the local density can be optimized for any choice of radial basis sets. We discuss the representation in the context of a kernel ridge regression model, commonly used with SOAP features, and analyze how the computational effort scales for each of the individual steps of the calculation. By applying data reduction techniques in feature space, we show how to reduce the total computational cost by a factor of up to 4 without affecting the model's symmetry properties and without significantly impacting its accuracy.

I. INTRODUCTION

Supervised machine learning (ML) methods are gaining increasing importance in the field of atomistic materials modeling, where they are often used to replace accurate, but computationally prohibitive, electronic structure calculations. Furthermore, unsupervised ML methods are gaining prominence as a way to interpret simulations of ever-increasing complexity.[1–7] All these methods fundamentally rely on a transformation of the system's atomic coordinates into a form amenable to the construction of efficient and transferable machine-learning models. Usually, this implies that the features that represent an atomic configuration reflect the transformations (invariance or covariance) of the target properties with respect to fundamental symmetry operations and that the prediction of the extensive properties of a structure is decomposed into that of local contributions, written as a function of a description of the neighborhood of individual atoms.[8,9]

We will focus, for the majority of this paper, on the problem of *regression* of a property expressed as a function of these transformed coordinates (hereafter called just "representation"). By far, the most common application of structure-property regression in the context of atomistic simulations is in the fitting of potential energy surfaces, which are used in molecular simulations or to compute thermodynamic averages. The majority of the considerations we make here applies to the prediction of any scalar property of the system, although the calculation of gradients might be less important than that in the case of potentials, and we use the terms "machine-learned interatomic potential" (MLIP) and "model" interchangeably in what follows. Figure 1 schematically illustrates the typical procedure of a single time step in an atomistic machine learning molecular dynamics (ML-MD) simulation. After the atomic coordinates are read in and the neighbor list is computed to determine the local environments around each atom, the coordinates are transformed into an intermediate representation. It is this representation that is then passed to the machine learning model—be it one based on neural networks (NNs),[10] Gaussian process regression (GPR),[11] or one of several other closely related methods. The accuracy and the transferability of the regression model are usually greatly improved by the use of representations that fulfill the requirements of symmetry and locality,[10,12–14] while at the same time being sensitive to all relevant structural changes,[12,15,16] being smooth functions of the atomic coordinates, and—ideally—being free of degeneracies, which map completely different structures to the same descriptor.[17]

Here, we focus on a class of representations that fulfill these requirements and that can be constructed starting from a description of a structure in terms of an atom density—which is naturally invariant to permutations of the atom labels—which is made translationally and rotationally invariant by first summing over $\mathbb{R}^3$ and then averaging ν-point correlations of the resultant atom-centered density over the $O(3)$ improper rotation group.[14] The smooth overlap of atomic position (SOAP) power spectrum is perhaps the best-known member of this class of representations,[12] but a wealth of other descriptors such as those underlying the spectral neighbor analysis potentials (SNAPs),[18] the atomic cluster expansion (ACE),[19,20] moment tensor potentials (MTPs),[21] and the equivariant extension λ-SOAP[22] and the N-point contractions of equivariant (NICE)[23] representation can be recovered as appropriate limits or extensions. Atom-centered symmetry functions[8,10] can also be seen as a projection of these atom-density representations on a bespoke set of basis functions.

Even though these representations are related through a common mathematical formalism,[14,19] the cost of evaluating them, and the accuracy of the resulting models, can vary greatly. In some cases, different frameworks have been shown to yield comparable errors,[24] while other studies have suggested a trade-off between accuracy and computational cost, with the combination of SOAP features and Gaussian process regression (hereafter termed just SOAP-GAP) emerging as the most accurate, but also the most computationally demanding method.[25,26]

In fact, the evaluation of SOAP features and their gradients can take anywhere from 10% to 90% (depending on the system and the parameters chosen) of the total computational cost of the energy and force evaluation in a typical molecular dynamics (MD) simulation with the SOAP-GAP method; almost all the remaining cost is taken up by the evaluation of the kernels (and their gradients) required to compute the GAP energy and forces. We therefore discuss optimization strategies aimed at reducing the computational cost of these two critical components. While we focus, at present, on a serial implementation, the modular structure that we introduce to optimize single-core performance is also very well-suited to parallelization, which becomes indispensable when aiming at extending simulation size and timescale.

We begin in Sec. II with an overview of density-based representations and present our benchmarking methodology in Sec. III. We continue in Sec. IV showing how the mathematical formulation of density-based representations reveals several opportunities for optimization, which we implement and systematically benchmark. We then expand these benchmarks to a variety of realistic simulation scenarios, shown in Sec. V, comparing against an existing simulation code and investigating the effect of convergence parameters. We present further experimental extensions of the code's capabilities in Sec. VI. Finally, in Sec. VII, we summarize the improvements and

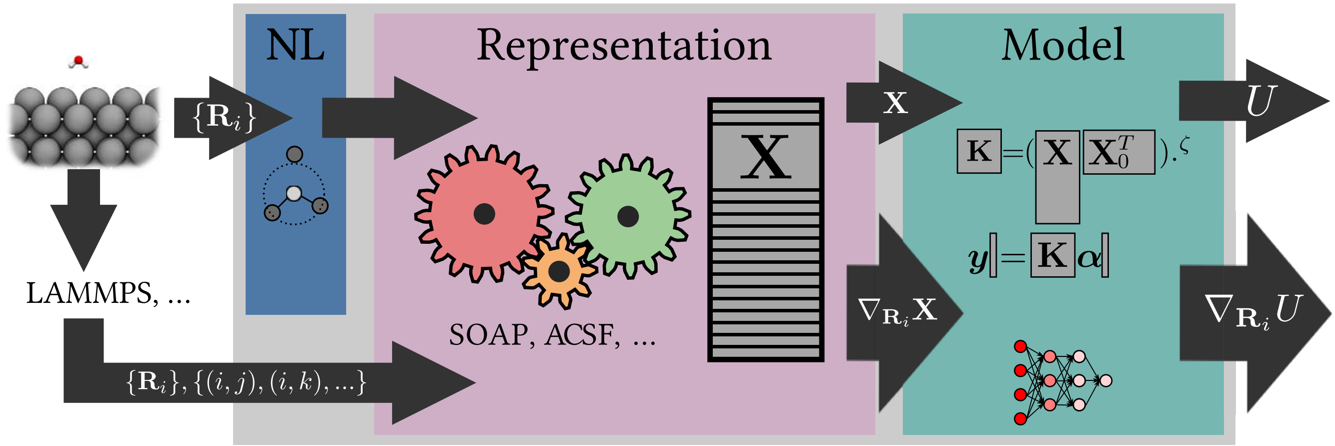
NL
Representation
Model
$\{\mathbf{R}_i\}$
LAMMPS, ...
$\{\mathbf{R}_i\}, \{(i,j),(i,k),...\}$
SOAP, ACSF, ...
X
X
$\mathbf{K} = (\mathbf{X}\,\mathbf{X}_0^T).^{\zeta}$
$\boldsymbol{y} = \mathbf{K}\boldsymbol{\alpha}$
$\nabla_{\mathbf{R}_i}\mathbf{X}$
U
$\nabla_{\mathbf{R}_i}U$

FIG. 1. A scheme showing the different components involved in the evaluation of energies and forces for an atomic-scale machine-learning model: Neighbor list (NL), representation calculation, and model evaluation. These steps need to be performed for each new structure in a screening procedure or each step in a molecular-dynamics simulation.

describe the role of our new, modular, efficient code `librascal` in the modern atomistic ML ecosystem.

II. THEORY

We begin by giving a brief overview of the construction of a symmetrized atom-density representation,[14] introducing the notation we use in the rest of this paper to indicate the various components that are needed to evaluate the features associated with a given structure. The construction operates by a sequence of integrals over symmetry operations, applied to a smooth (or Dirac-δ-like) atom density that is taken to describe the structure. After summing over translations, one obtains a description of the atomic environment A_i around the *central atom* i, which depends on the neighbor positions $\mathbf{r}_{ji} = \mathbf{r}_j - \mathbf{r}_i$. Each atomic species a is associated with a separate density built as a superimposition of Gaussian functions centered on the interatomic distance vectors, $\langle \mathbf{x}|\mathbf{r}_{ji}; g\rangle \equiv g(\mathbf{x} - \mathbf{r}_{ji})$ with variance σ^2, restricted to a local spherical cutoff r_{cut} by a smooth function f_{cut},

$$\langle a\mathbf{x}|A;\rho_i\rangle = \sum_{j\in A_i} \delta_{aa_j} \langle \mathbf{x}|\mathbf{r}_{ji}; g\rangle f_{\text{cut}}(r_{ij}). \tag{1}$$

We use a notation that mimics the Dirac bra-ket formalism,[27] in which the ket indicates the entity being represented (the density field ρ centered on atom i of structure A or the Gaussian density that describes the position of neighbors) and the bra indicates the indices that label different features (in this case, the chemical species a and the position at which the field is evaluated, $\mathbf{x}$, that serves as a continuous index). To simplify the notation, when discussing the construction of the features for an arbitrary configuration, we omit the reference to the atomic structure such that $\langle a\mathbf{x}|A;\rho_i\rangle \to \langle a\mathbf{x}|\rho_i\rangle$. Following Ref. 14, we introduce the symmetrized $(\nu + 1)$-body correlation representation,

$$\langle a_1\mathbf{x}_1; \dots a_\nu\mathbf{x}_\nu|\overline{\rho_i^{\otimes\nu}}\rangle = \sum_{k=0,1} \int_{SO^3} \mathrm{d}\hat{R}\, \langle a_1\mathbf{x}_1|\hat{R}\hat{i}^k|\rho_i\rangle \dots \langle a_\nu\mathbf{x}_\nu|\hat{R}\hat{i}^k|\rho_i\rangle, \tag{2}$$

where $\overline{\rho_i^{\otimes\nu}}$ is a tensor product of ν atom centered fields averaged over all possible improper rotations. This object can be understood as a fixed ν-point stencil centered on atom i, which is applied continuously to the density field, hence accumulating correlations of body order $\nu + 1$. To perform the rotational average, it is convenient to expand the atom density on a basis whose expansion coefficients are given by

$$\begin{aligned}\langle anlm|\rho_i\rangle &= \sum_{j\in A_i} \delta_{aa_j} \int \mathrm{d}\mathbf{x}\, \langle nl|x\rangle\langle lm|\hat{\mathbf{x}}\rangle\langle \mathbf{x}|\mathbf{r}_{ji}; g\rangle \\ &= \sum_{j\in A_i} \delta_{aa_j} \langle nlm|\mathbf{r}_{ji}; g\rangle,\end{aligned} \tag{3}$$

where $x = \|\mathbf{x}\|$, $\hat{\mathbf{x}} = \mathbf{x}/x$. $\langle x|nl\rangle \equiv R_{nl}(x)$ are orthogonal radial basis functions, which may or may not depend explicitly on l (see, e.g., Ref. 28), and $\langle \hat{\mathbf{x}}|lm\rangle \equiv Y_l^m(\hat{\mathbf{x}})$ are spherical harmonics. As we discuss in Sec. IV A, the choice of $\langle x|nl\rangle$ is flexible and can thus be guided by considerations of computational and information efficiency. The angular dependence, on the other hand, is most naturally expanded using spherical harmonics, which results in compact expressions for the density correlation features of Eq. (2) in terms of products and contractions of the expansion coefficients of Eq. (3) (see Ref. 14 for more details). For the case of $\nu = 2$,

$$\begin{aligned}\langle a_1n_1; a_2n_2; l|\overline{\rho_i^{\otimes 2}}\rangle = \frac{1}{\sqrt{2l+1}} \sum_m (-1)^m \langle a_1n_1lm|\rho_i\rangle \\ \times \langle a_2n_2l(-m)|\rho_i\rangle,\end{aligned} \tag{4}$$

which corresponds to the SOAP power spectrum of Ref. 12 up to some inconsequential factors. In the following text, we discuss and benchmark the efficient implementation of the density expansion and the spherical invariant of order 2, i.e., the power spectrum, in `librascal`.

III. METHODS

In order to provide a concrete assessment of the impact of the optimizations we describe in this paper and to demonstrate the performance of the optimized code on a variety of realistic systems, we apply a comprehensive benchmarking strategy that compares different classes of systems and breaks down the overall computational cost into contributions associated with different steps of the evaluation of the ML model. We focus on a typical use case of SOAP features, namely, as the input to a kernel ridge regression (KRR) model for a property and its derivatives (e.g., energy and forces).[29] Even though `librascal` is mainly dedicated to the calculation of features, including the model evaluation step is crucial to assess the computational effort in the context of the overall cost of the model. In the same spirit of focusing on the most common, and relevant, use case scenario, we consider the cost of evaluating a previously trained model, rather than the cost to train the model itself. The training step must be performed only once per potential, and—at least in the case of KRR/GPR models—is usually limited by the need to store large kernel or feature matrices in memory, rather than by computation time. When running a simulation, such as an MD trajectory, the representation and model evaluation constitute the real limiting factors in what can be achieved with a given potential in terms of statistics, system size, and complexity of the target properties.

A. Benchmarks and key parameters

We report and examine the benchmarks separating the logical components of the overall calculation, as summarized in Figs. 1 and 2—namely, the construction of the neighbor list, the calculation of the local density expansion (that can be further broken down into the evaluation of radial and angular terms), the combination of the density coefficients to obtain an invariant representation, and the evaluation of the model itself. Most of these steps can also be broken down into the time required to compute just the representation (energy) vs the overhead for computing gradients (forces) in addition.

For the representation stage, it is possible to track the computational cost as a function of a few key parameters, namely, the radial and angular expansion limits (n_{max} and l_{max}, respectively). The benchmarks for this stage are reported not per atom but per *pair*, consistent with the overall scaling of this component of the calculation. The timings reported in this way are therefore also mostly independent of the system in question, i.e., the variation between systems is usually comparable to the variation between individual timing runs. The model stage has a less straightforward dependence on the spherical expansion parameters, and the system dependence

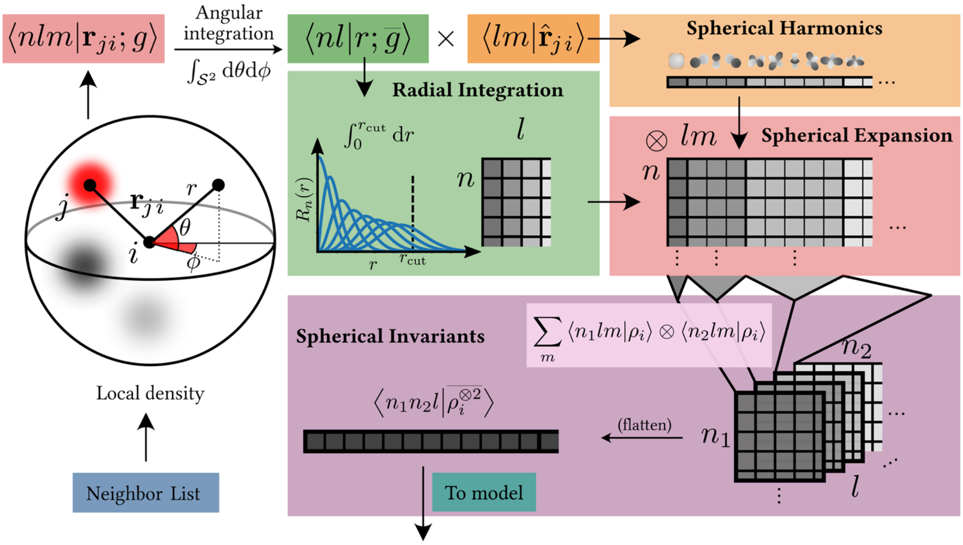
$\langle nlm|\mathbf{r}_{ji};g\rangle$
Angular integration
$\int_{S^2} \mathrm{d}\theta\mathrm{d}\phi$
$\langle nl|r;g\rangle$
×
$\langle lm|\hat{\mathbf{r}}_{ji}\rangle$
Spherical Harmonics
Radial Integration
$\int_0^{r_{\mathrm{cut}}} \mathrm{d}r$
l
n
$R_n(r)$
r
r_{cut}
$\otimes\ lm$
Spherical Expansion
n
j
$\mathbf{r}_{ji}$
r
θ
i
ϕ
Local density
Neighbor List
Spherical Invariants
$\sum_m \langle n_1lm|\rho_i\rangle \otimes \langle n_2lm|\rho_i\rangle$
$\langle n_1n_2l|\rho_i^{\otimes 2}\rangle$
(flatten)
n_2
n_1
l
To model

FIG. 2. Schematic showing the process of expanding the density in a radial and angular basis set and recombining those to form spherical invariants (or covariants).

is more subtle. The main influence on the computational cost is the feature space dimension n_{feat} and the number of environments n_{active} used to parameterize the model. As will be discussed in Sec. IV H, both of these parameters can be reduced significantly by the use of dimensionality reduction algorithms, with lower computational cost generally trading off with the accuracy of predictions (a pattern seen in many other machine learning frameworks[25]). In order to run and organize the large number of individual benchmarks required for this study, we have made an extensive use of the signac data management framework,[30,31] which can be accessed from an open repository.[32] The version of `librascal` used to run these benchmarks is archived to Zenodo.[33]

B. Datasets

The system dependence of the overall computation is influenced by two major factors: The first is the number density, which—together with the cutoff radius r_{cut}—determines the total number of pairs that must be iterated over to compute the representations, as well as the number of the degrees of freedom needed to fully characterize the local environment, which, in turn, affects the radial and angular expansion parameters necessary to represent it. The second is the number of chemical elements that is present, which directly affects the dimensionality of the representation. However, several optimizations are possible depending on the model and species composition, as well as the distribution of these species throughout the system, making this a subtle and nontrivial influence on the total model cost.

Therefore, we have decided to benchmark the overall cost (neighbor list, representation, and model together) on a selection of five realistic datasets that represent both typical and challenging applications of machine learning potentials. For a single-species system, we have chosen the bulk silicon dataset[34] from the work of Bartók *et al.*;[35] despite its simple species composition, it still represents a large array of structural diversity. The fluid methane dataset[36] from the work of Veit *et al.*[37] has two chemical species but is distributed homogeneously throughout the cell; the dataset additionally contains a range of different cell densities. In order to include more challenging multi-species systems, we have selected three additional datasets from different application areas. The solvation dataset from the work of Rossi *et al.*[38] consists of structures each containing a single molecule of methanesulfonic acid within a large cell of liquid phenol, where the presence of multiple species and the inhomogeneity of their distribution presents a challenge for both representation and fitting algorithms. The molecular crystal dataset "CSD1000r" used by Musil *et al.*[39] contains up to four species, where not all species are present in each separate structure. Finally, the widely used QM9 dataset[40] contains isolated molecules of up to nine heavy (non-hydrogen) atoms each and is composed of up to five chemical species—where, again, not every species is represented in every structure.

IV. IMPLEMENTATION OF INVARIANT REPRESENTATIONS

We begin by discussing the `librascal` implementation of the power spectrum SOAP features and by showing how a deeper understanding of the structure of the atom-density correlation features can be exploited to improve substantially the cost of evaluation. Benchmarks on all the datasets discussed above are included in the supplementary material—here, we choose a subset of the different test cases since, usually, the computational cost can be normalized in a way that minimizes the dependence on the specifics of the system at hand.

A. Density coefficients

The exact expression for the density coefficients depends on the specifics of the atom density field and on the basis used to expand it. To see this, it is advantageous to separate the integral in Eq. (3) into radial and angular coordinates. Then, regardless of the choice of the functional form of the atom density or the radial basis, the density coefficients can be written as a sum over functions of neighbor distances and orientations,

$$\langle anlm|\rho_i\rangle = \sum_{j\in A_i} \delta_{aa_j} f_{\text{cut}}(r_{ji})\langle nlm|\mathbf{r}_{ji};g\rangle, \tag{5}$$

where

$$\langle nlm|\mathbf{r}_{ji};g\rangle = \int \mathrm{d}\mathbf{x}\,\langle nl|x\rangle\langle lm|\hat{\mathbf{x}}\rangle\langle \mathbf{x}|\mathbf{r}_{ji};g\rangle. \tag{6}$$

For a Gaussian atom density, the integral can be factorized into

$$\langle nlm|\mathbf{r};g\rangle = \langle lm|\hat{\mathbf{r}}\rangle\langle nl|r;\overline{g}\rangle, \tag{7}$$

containing a radial integral

$$\langle nl|r;\overline{g}\rangle = 4\pi e^{-cr^2}\int_0^\infty \mathrm{d}x\,x^2\langle nl|x\rangle e^{-cx^2}\,\mathrm{i}_l(2cxr), \tag{8}$$

where $c = 1/2\sigma^2$, and the radial and angular degrees of freedom are explicitly coupled by the l dependence of the modified Bessel function. Thus, the density coefficients can be computed by evaluating spherical harmonics and radial integral functions for each pair of neighbors, and then, by summing over their products,

$$\langle anlm|\rho_i\rangle = \sum_{j\in A_i} \delta_{aa_j} f_{\text{cut}}(r_{ji})\langle lm|\hat{\mathbf{r}}_{ji}\rangle\langle nl|r_{ij};\overline{g}\rangle. \tag{9}$$

Alternative atom-centered density formulations such as in ACE[19] or TurboSOAP[28] lead to similar expressions for the radial function. For instance, TurboSOAP chooses a Gaussian atomic density that is symmetric about $\mathbf{r}_i$ instead of $\mathbf{r}_{ji}$, making it possible to factorize the radial term such that $\langle nl|r_{ij};\hat{g}\rangle = \langle n|r_{ij};\hat{g}\rangle\langle l|r_{ij};\hat{g}\rangle$. Both terms can be efficiently computed using recurrence relations in l and n. In `librascal`, the density expansion is implemented only for Gaussian atomic densities symmetric about $\mathbf{r}_{ji}$, using two types of radial basis sets: The Gaussian-type orbital (GTO) basis and the discrete variable representation (DVR) basis.

B. GTO radial basis

The Gaussian-type orbital radial basis is defined as

$$\langle x|n;\text{GTO}\rangle = \mathcal{N}_n x^n \exp[-d_n x^2], \tag{10}$$

where $d_n = 1/2\sigma_n^2$, $\sigma_n = r_{\text{cut}}\max(\sqrt{n},1)/n_{\max}$, $\mathcal{N}_n^2 = 2/(\sigma_n^{2n+3}\Gamma(n+3/2))$ is a normalization factor, and $0 \le n < n_{\max}$. In contrast to the displaced Gaussian basis in the original formulation of SOAP,[12] this choice of radial basis leads to a radial integral that can be evaluated analytically,

$$\langle nl;\text{GTO}|r;\overline{g}\rangle = \pi^{3/2}\exp[-cr^2]\mathcal{N}_n\frac{\Gamma\left(\frac{n+l+3}{2}\right)}{\Gamma\left(l+\frac{3}{2}\right)}c^l r^l(c+d_n)^{-\frac{n+l+3}{2}}$$
$$\times\,{}_1F_1\left(\frac{n+l+3}{2}, l+\frac{3}{2}, \frac{c^2r^2}{c+d_n}\right), \tag{11}$$

where ${}_1F_1$ is the confluent hypergeometric function of the first kind. Given that the overlap matrix $\mathbf{S}$ between GTOs of the form (10) can be computed analytically, it is then easy to obtain an orthogonal basis set,

$$\langle x|n;\text{o-GTO}\rangle = \sum_{n'}[\mathbf{S}^{-1/2}]_{nn'}\langle x|n';\text{GTO}\rangle. \tag{12}$$

Due to the linear nature of all the operations involved in the evaluation of the density expansion coefficients, the orthogonalization can be applied at any point of the procedure. In the case of the analytical evaluation of Eq. (11), it is convenient to first combine the contributions from all the neighbors to the density coefficients Eq. (8) and then orthogonalize just once. In Sec. IV D, when computing the coefficients numerically, it is instead more convenient to orthogonalize the radial integral Eq. (11) directly.

The total time required to compute the radial integral, as well as its derivative with respect to r_{ij} (needed for gradients of the model), is plotted in the top left panel of Fig. 3 as a function of the expansion parameters $n_{\max}$ and $l_{\max}$ and scales roughly linearly with respect to the expansion thresholds (see also the supplementary material for a more detailed figure). Despite the use of an efficient and robust algorithm, which is discussed in Appendix A, most of the computational cost in the evaluation of Eq. (11) is associated with the confluent hypergeometric function ${}_1F_1$.

C. DVR radial basis

Another possible choice of the basis is inspired by the idea of using a numerical, rather than analytical, evaluation of the radial integral. In fact, the numerical integral can be done exactly and with no discretization overhead if we choose the orthonormal DVR radial basis with a Gauss–Legendre quadrature rule.[41] This basis has the advantage of vanishing at every quadrature point except for one, i.e.,

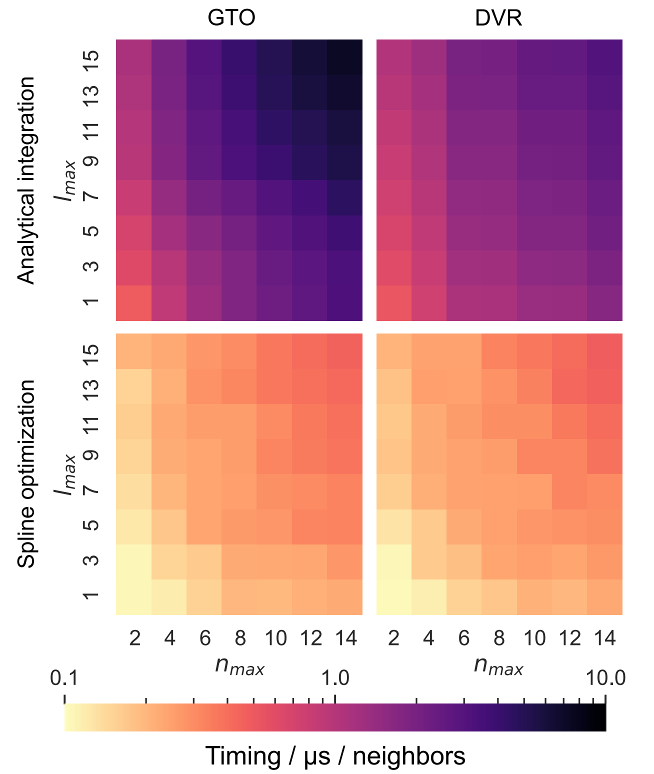

FIG. 3. Computational cost for the evaluation of the radial integral and its derivatives with different methods for structures taken from the QM9 dataset. Note that the dataset has very little influence on this benchmark since the radial integral and its derivative are always evaluated once per neighbor. For the splining, an accuracy of 10^{-8} was chosen.

$\langle x|n;\mathrm{DVR}\rangle = \sqrt{\omega_n}\delta(x-x_n)$, which simplifies the numerical radial integral into

$$\langle nl;\mathrm{DVR}|r;\bar{g}\rangle = x_n\sqrt{\omega_n}e^{-cx_n^2}\mathrm{i}_l(2cx_nr), \tag{13}$$

where x_n are the quadrature points, distributed across the range [0, $r_{\mathrm{cut}} + 3\sigma$] over which the integrand differs substantially from 0, and the ω_n are the corresponding quadrature weights.[42] The DVR basis is orthogonal by construction and only requires evaluating the modified spherical Bessel function rather than the more demanding ${}_1F_1$, leading to a reduction of about a factor of 2 of the cost of evaluating radial integrals (top right panel in Fig. 3). The computational cost of evaluating the radial integral in the DVR basis is again shown in the upper right-hand panel of Fig. 3. The computational cost is reduced by more than half compared to the integral in the GTO basis, although the scaling with the l_{max} and especially n_{max} parameters remains approximately linear (see plots in the supplementary material).

D. Spline optimization

Rather than devising basis functions that allow for a less demanding analytical evaluation of the radial integrals, one can evaluate inexpensively the full radial integral $\langle nl|r;\bar{g}\rangle$ by pre-computing its value on a grid and then using a cubic spline interpolator. For each combination of radial $0 \le n < n_{\mathrm{max}}$ and angular $0 \le l \le l_{\mathrm{max}}$ indices, the integral is tabulated and the spline is computed for the range [0, r_c]. A grid $\{r_k\}_{k=1}^{M}$ with constant step size Δ is used to achieve a constant time complexity for the search of the closest interval $[r_k, r_{k+1}]$ for a distance $r_{ij} \in [r_k, r_{k+1}]$. Following the implementation of Ref. 43, the computation of radial terms simplifies to the evaluation of a polynomial of degree of 3 in r_{ij} with precomputed coefficients c_k and d_k,

$$\begin{aligned}\langle nl|r;\bar{g}\rangle = \frac{1}{\Delta}\Big(&(r-r_{k+1})c_k + (r-r_{k+1})^3 d_k - (r-r_{k+1})d_k \\ &+ (r-r_{k+1})c_{k+1} + (r-r_{k+1})^3 d_{k+1} - (r-r_{k+1})d_{k+1}\Big).\end{aligned} \tag{14}$$

This expression requires only a small number of multiplications and additions, thus reducing the computational time of the radial integral by avoiding the evaluation of the complex hypergeometric, exponential, or Gamma functions present in the analytical GTO and DVR basis sets. Given that the expression is linear in the coefficients, it is straightforward to evaluate the coefficients for |o-GTO⟩ by simply applying the orthogonalization matrix to the coefficients of |GTO⟩. Smooth derivatives $\partial\langle nl|r;\bar{g}\rangle/\partial r$ of this piecewise polynomial function can also be computed by taking the derivative of the polynomial with minimal additional effort. As seen in Fig. 3, splining reduces the computational cost of the radial integrals by almost an order of magnitude and effectively eliminates the difference between the GTO and DVR bases.

Thus, the choice of $\langle x|nl\rangle$ should not be guided by the cost of evaluation but by a different metric—for instance, the information efficiency. As discussed in Ref. 44, even though in the $n_{\mathrm{max}} \to \infty$ limit all bases converge to the same feature space, they do so with different rates. Figure 4 shows that the GTO basis converges to the complete basis limit faster than either the DVR basis or the shifted-Gaussian basis used in the QUIP code (see Sec. V A).[12] This can be seen both in terms of the error in approximating the $n_{\mathrm{max}} \to \infty$ scalar-product SOAP kernel and in terms of the global feature space reconstruction error (GFRE),[44] a measure of the ability of a set of features to linearly predict a second set of features—in this case, the $n_{\mathrm{max}} = 20$ GTO power spectrum. For the same value of n_{max}, the GTO basis better approximates the complete basis set limit, which offsets the higher computational cost. When using splines, the computational overhead associated with GTOs is eliminated and this basis has a clear advantage overall. It is important to stress that this reduction in computational cost comes with essentially no negative side-effects: the splines approximate their target very closely (spline-based curves in Fig. 4 are indistinguishable from their analytical counterparts except for large n_{max}), and the approximation does not affect the exact translational, permutational, or rotational symmetries of the features. Finally, there is evidence that the size of the radial basis set n_{max} has a larger influence than the angular expansion threshold l_{max} on the accuracy of a SOAP-based potential.[45] Furthermore, a reduction in n_{max} does not only lower the cost of the spherical expansion coefficients but also of evaluating invariants. Together, these insights all point toward numerical optimization of the radial basis as a promising future line of investigation.

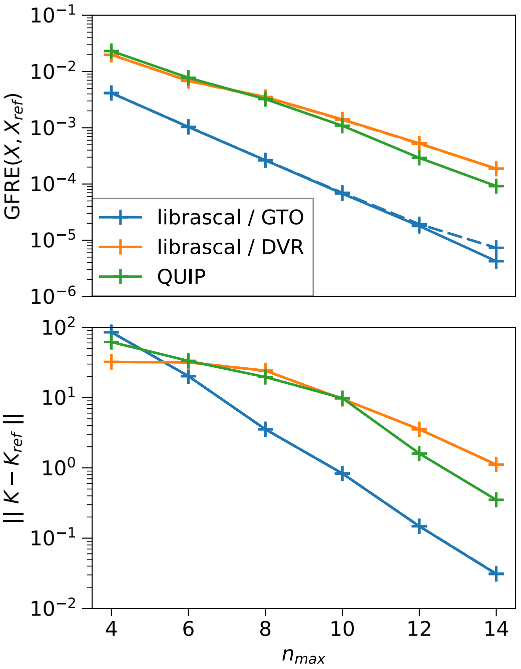

FIG. 4. Analysis of the convergence of the SOAP features built using the GTO and DVR basis set of `librascal` and the shifted-Gaussian basis of QUIP, with respect to reference features computed using the non-splined `librascal` GTO radial basis and $n_{\mathrm{max}} = 20$. The top panel shows global feature reconstruction error (GFRE), and the bottom panel shows the distance between kernel matrices, both computed on the bulk silicon dataset. All features are computed using $l_{\mathrm{max}} = 16$. Dashed curves correspond to the spline approximation of the GTO and DVR bases using an accuracy of 10^{-6} and are identical to the analytical version except for the highest n_{max}, where a small residual GFRE remains, indicating an approximation error that can be reduced by using a finer grid in the spline.

E. Spherical harmonics

In contrast to the relatively obscure special functions needed for the radial integrals, the spherical harmonics needed for the angular part of the density coefficients [see Eq. (7)] are much more widely used due to their importance in any problem with spherical symmetry. Correspondingly, there has been much research into finding efficient algorithms to evaluate spherical harmonics, leading to many good algorithms becoming publicly available. In `librascal`, we have chosen to implement the algorithm described by Limpanuparb and Milthorpe,[46] which makes use of efficient recurrence relations optimized for speed and numerical stability and is similar to the algorithm implemented in the GNU Scientific Library.[47] Gradients are computed from analytical expressions given in the supplementary material.

As shown in Fig. 5, the cost scales linearly with the angular expansion parameter l_{max} and including gradients consistently increases the cost by a factor of about 4, consistent with the need to compute three additional values per spherical harmonic. The cost to compute the spherical harmonics and gradients is typically comparable to, or larger than, the cost to compute the splined radial integral; this cost is discussed in more detail and in the context of the whole invariant computation in Sec. IV F.

F. Spherical expansion and invariants

Having discussed how to implement an efficient procedure to evaluate the radial and the angular terms contributing to the density expansion, let us now consider the cost of the remaining steps to obtain the full SOAP feature vector $\langle a_1 n_1; a_2 n_2; l|\overline{\rho_i^{\otimes 2}}\rangle$. Figure 6 presents an overview of the timings for all evaluation steps for different $(n_{\text{max}}, l_{\text{max}})$, comparing a dataset of bulk Si configurations and a database of molecular crystals. For a few selected parameter sets, it also shows the breakdown of the evaluation time into the part associated with the evaluation of radial integrals and spherical harmonics for each neighbor, the combination of the two into the full density expansion coefficients, and the calculation of the SOAP invariants. The spline interpolation makes the cost of radial integrals negligible, and even the evaluation of spherical harmonics usually requires less than 25% of the total timing. For the silicon dataset, which has only one atomic species, the cost is typically dominated by the combination of radial and angular terms. Indeed, the computational cost of this step scales roughly as $n_{\text{neigh}} n_{\text{max}} (l_{\text{max}} + 1)^2$, which can easily dominate the total cost for realistic parameter sets.

For the molecular materials, on the other hand, the evaluation of the invariants becomes more expensive, comparable to the computation of the density coefficients. The difference can be explained as follows. Given that the coefficients $\langle anlm|\rho_i\rangle$ are combined to obtain spherically equivariant representations of the atomic environment by averaging over the group symmetries their tensor products, as outlined in Eq. (2), their evaluation exhibits a very different scaling. The cost is independent of the number of neighbors and instead depends strongly on the size of the basis used to expand the atom density and on the number of chemical species n_{species}. For the special case of spherical *invariants* of body order $(\nu + 1) = 3$, corresponding to classic SOAP features,[12] evaluating Eq. (4) essentially amounts to computing an outer product over the (a, n) dimension of expansion coefficients that is then summed over m—which requires a number of multiplications of the order of ${n_{\text{species}}}^2 {n_{\text{max}}}^2 (l_{\text{max}} + 1)^2$. In summary, the cost of the different steps varies substantially depending on the system, the cutoff, and the expansion parameters, and there is no contribution that dominates consistently in all use cases.

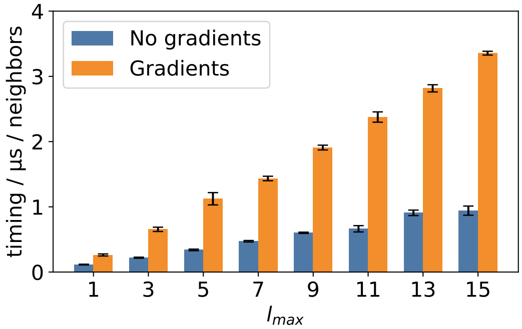

FIG. 5. Timings for the computation of the spherical harmonics as a function of the angular expansion threshold for the QM9 dataset.

G. Cost of gradients

Evaluating the gradients of the invariant features with respect to the atomic coordinates is a necessary step to compute model derivatives, e.g., forces and stresses for MD simulations—but it also entails a substantial overhead, as shown in the right-hand panels of Fig. 6. This overhead is ultimately a consequence of the direct evaluation of the gradients of the features, which requires a separate contraction for each of the $\langle nn'l|$ components in the SOAP vector,

$$\frac{\partial\langle nn'l|\overline{\rho_i^{\otimes 2}}\rangle}{\partial \mathbf{r}_j} \propto \sum_m \langle n'lm|\rho_i\rangle^{\star} \frac{\partial\langle nlm|\mathbf{r}_{ji};g\rangle}{\partial \mathbf{r}_j} + \cdots. \qquad (15)$$

While some speedup could be attained by reordering the summation, the core issue is the need to compute a separate term for each feature and each neighbor of the central atom, which means that the computational effort, for the typical values of $(n_{\text{max}}, l_{\text{max}})$, is overwhelmingly dominated by the construction of the invariants. These issues indicate that the evaluation of gradients would benefit from further optimizations—in particular, trading-off modularity for speed by optimizing the expansion coefficients together with the model evaluation. This way, it will be possible to avoid the (re)computation of certain intermediate quantities, analogous to the optimization of the order of matrix multiplications involved in the evaluation of the chain rule, linking the model target and the input atomic coordinates.

H. Feature dimensionality reduction

A more straightforward, and potentially more impactful, optimization involves performing a data-driven selection to reduce the number of invariant features to be computed and used as inputs of the model. Even though representations based on systematic orthonormal basis expansions, such as the SOAP power spectrum, provide a complete linear basis to describe three-body correlations[20,48] and even though they do *not* provide an injective representation of an atomic environment,[17] one often finds that for realistic structural datasets, different entries in the SOAP feature vector exhibit a high degree of correlation. This means that they span a much larger space than what is effectively needed for the prediction of typical atomistic properties.

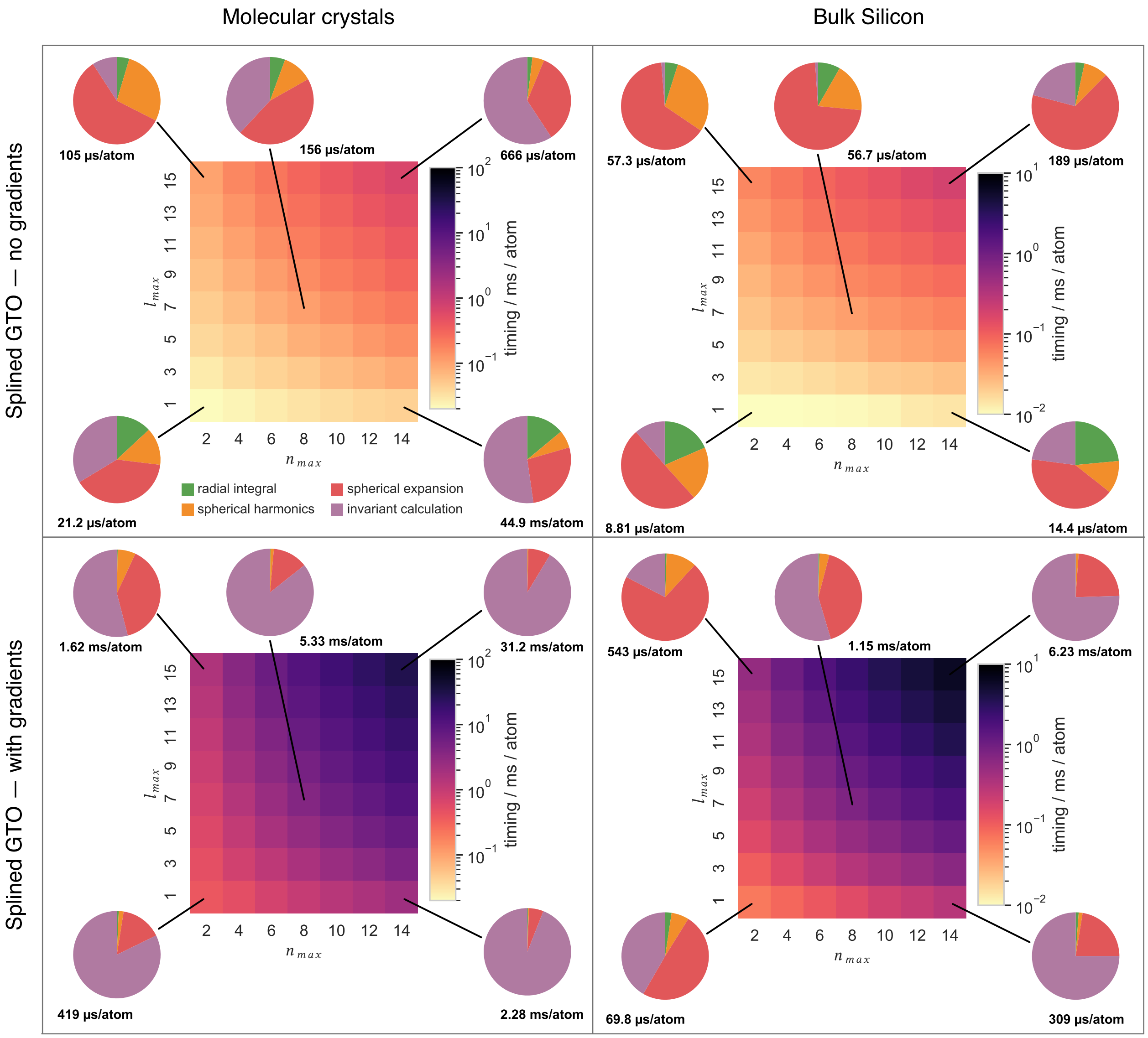

FIG. 6. Effect of radial and angular cutoffs on the overall timing of calculating spherical invariants. (left) Molecular crystal dataset (with, on average, 27 neighbors per center and four elements). (right) Bulk silicon dataset (16 neighbors per center and a single element). (top) SOAP power spectrum only. (bottom) SOAP power spectrum and gradients. All calculations use the GTO radial basis with a spline optimization. For selected points, we also show, as pie charts, the relative time spent in the different phases.

Therefore, a subset of features—usually a small fraction of the full set—can be selected with little impact on the model error.[15,49] Both CUR[50] and farthest point sampling (FPS)[49,51,52] selection strategies are available in `librascal` and can be performed as a preliminary step in the optimization of a model, using Python utility functions. Feature selection can reduce the time spent both on computing the features and the model parameters and on making predictions (see Sec. V B). For kernel models and reasonably simple forms of the kernel function, the evaluation of both the features and the kernels scale linearly with n_{feat}.

Once a list of selected features has been obtained, their indices $\{q\} \equiv \{(a_q n_q;\, a'_q n'_q;\, l_q)\}$ can be passed to the C++ code. The sparse feature computation is simply implemented as a selective evaluation of the pre-selected invariants $\langle q|\overline{\rho_i^{\otimes 2}}\rangle$. The effect of this optimization on the overall cost of computing spherical invariants is shown in Fig. 7, with realistic n_{max} and l_{max} parameters, which are comparable to those used in applications (i.e., (n_{max}, l_{max}) equal to (10, 12) for Si,[35] (8, 6) for methane,[37] and (9, 9) for molecular crystals[39]). The overall trend is that of a constant contribution (from the local density expansion) plus a linear term (from the construction of spherical invariants). Although most datasets do not reach linear scaling even for the largest number of features, selecting a small n_{feat} can reduce the computational cost by up to an order of magnitude. The impact of feature dimensionality on both

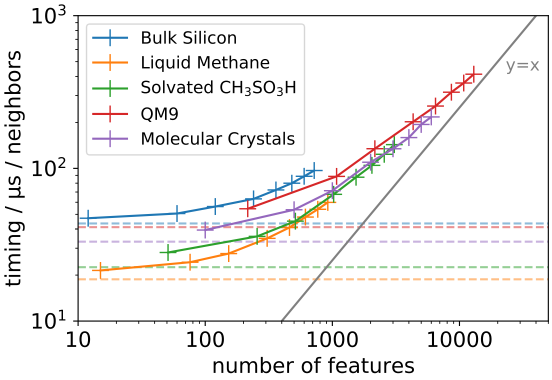

FIG. 7. Timing for the calculation of the SOAP power spectrum with gradients as a function of the requested number of features. Horizontal lines represent the time taken by the local density expansion step for each dataset. The gray line is a guide for the eye representing a linear relation between time and the number of features. The ($n_{\max}$, $l_{\max}$) parameters used are the following: bulk silicon (10,11), liquid methane (8,7), solvated CH_3SO_3H (8,7), QM9 (12,9), and molecular crystals (10,9).

the computational cost and accuracy of models trained on realistic data is discussed in Sec. V C; oftentimes, the features can be sparsified fairly aggressively (up to a factor of about 5–10, depending on the dataset) without any significant impact on the prediction error.

V. COMPARATIVE BENCHMARKS

Now that we have analyzed separately the different components of the calculation of the SOAP features, we turn our attention to the end-to-end benchmarking of a full energy and force evaluation, similar to what one would encounter when running a MD simulation. As in Sec. IV, we run comprehensive tests on each of the five datasets described in Sec. III B and we report here those that are most telling of the scaling of the different terms with the system parameters, most notably the neighbor density and the number of chemical elements. We include a simple but complete implementation of kernel ridge regression, a framework that is often used together with SOAP features and that allows us to comment on the interplay between the calculation of the representation and the model. Thus, we can compare the computational effort associated with the use of `librascal` with that of QUIP, an existing, well-established code to train and evaluate Gaussian approximation potentials (GAPs) based on SOAP features,[53] and investigate the effect of the various optimizations described above on the overall model efficiency.

A. Existing implementations

Over the past couple of years, several codes have been released that can be used to fit and run ML potentials supporting different representations, especially for neural-network-type potentials such as n2p2[54] (which uses Behler–Parrinello ACSF[10]), ANI-1,[55] PANNA,[56] or DeepMD.[57] Here, we focus on kernel methods, for which there is a smaller number of actively used codes. The first, and still widely adopted, is the QUIP library, part of the libAtoms framework,[58] which has been used for almost all published Gaussian approximation potentials (GAPs)[9,28,37,59–62] and continues to be actively maintained. SOAP features are also implemented in several other packages, including DScribe,[63] TENSOAP,[64] and TurboSOAP[28] (now integrated into QUIP). Other notable kernel-learning potential packages are GDML, which implements the "gradient-domain machine learning" method of Chmiela *et al.*[65] (the full-kernel equivalent of the sparse kernel model we implement here), and QML,[66] which implements the FHCL-type representations[67] and the OQML framework.[68] We finally note for completeness several codes used for linear high-body-order models, such as the SNAP method[18] implemented in LAMMPS,[69] aPIPs[70] and ACE[19,20] implemented in JuLIP,[71] and the NICE descriptors[23] implemented in a separate code[72] interfaced with `librascal` (see Sec. VI). Here, we focus only on the QUIP code, which is the most mature implementation available and matches most closely the application domain of `librascal`.

B. Kernel models

To benchmark the performances of `librascal` in the context of the GAP framework typically used to build potentials with SOAP, we have implemented the same regression scheme used in QUIP to build a MLIP based on the SOAP power spectrum representation. We summarize the key ideas, emphasizing the aspects that are important to achieve optimal performance. In a GAP, as in the vast majority of regression models based on atom-centered features, the energy is defined as a sum of atomic contributions,

$$E(A) \equiv \langle E|A\rangle = \sum_{i\in A} E(A_i) \equiv \sum_{i\in A}\langle E|A_i\rangle, \tag{16}$$

where A_i indicates a local environment centered on atom i. An accurate, yet simple and efficient GAP can be built using a "projected process approximation"[73] form of kernel ridge regression, which mitigates the unfavorable scaling with train set size n_{train} of the cost of fitting (cubic) and predicting (linear) energies using a "full" ridge regression model. A small, representative subset M of the atomic environments usually found in the training set—the so-called "active," "pseudo-," or "sparse" points—is used, together with a positive-definite kernel function k, as a basis to expand the atomic energy,

$$\langle E|A_i\rangle = \sum_{I\in M} \delta_{a_i a_I}\langle E; a_I|M_I\rangle k(M_I, A_i), \tag{17}$$

where M_I indicates the Ith sparse point, $\langle E; a_I|M_I\rangle$ indicates the regression weights, and a separate energy model is determined for each atomic specie, which also means that the active set is partitioned with respect to the central atom-type. The sparse model (17) exhibits a much more favorable scaling with training set size, both during fitting $[\mathcal{O}(n_{\text{train}}{n_{\text{active}}}^2 + {n_{\text{active}}}^3)$, for the implementation in `librascal`] and when predicting a new structure $[\mathcal{O}(n_{\text{active}})]$. Obviously, the accuracy of the approximation relies on a degree of redundancy being present in the training set, and in practice, a suitable size of the active set M scales with the "diversity" of the structures contained within. Usually, however, an accuracy close to that of a full model can be reached even with $n_{\text{active}} \ll n_{\text{train}}$. The gradient of the energy with respect to the coordinates of an atom j can be obtained as a special case of the general form (B1),

$$\nabla_j\langle E|A\rangle = \sum_{I\in M}\langle E; a_I|M_I\rangle \sum_{i\in A}\delta_{a_i a_I}\nabla_j k(M_I, A_i), \tag{18}$$

and the virial (the derivative with respect to deformations $\boldsymbol{\eta}$ of the periodic cell) can be obtained as a special case of (B2),

$$\frac{\partial}{\partial \boldsymbol{\eta}}\langle E|A\rangle = \sum_{I\in M}\langle E; a_I|M_I\rangle \sum_{i\in A}\delta_{a_i a_I}\sum_{j\in A_i}\mathbf{r}_{ji}\otimes\nabla_j k(M_I,A_i). \tag{19}$$

In both Eqs. (18) and (19), the sum over the neighbors of atom i extends also over periodic replicas of the system. Both equations require the evaluation of kernel gradients, which can, in turn, be expressed using the chain rule in terms of the derivatives of the kernel function with respect to atomic features, and the atomic gradient of such features,

$$\nabla_j k(M_I,A_i) = \sum_q \nabla_j\langle q|A_i\rangle \frac{\partial k(M_I,A_i)}{\partial\langle q|A_i\rangle}. \tag{20}$$

When computing the model derivatives, it is important to contract the sums in the optimal order, by first summing the derivatives of the kernel over the active set. For instance, for the force,

$$\nabla_j\langle E|A\rangle = \sum_{i\in A}\delta_{a_i a_I}\sum_q \nabla_j\langle q|A_i\rangle\left[\sum_{I\in M}\langle E; a_I|M_I\rangle\frac{\partial k(M_I,A_i)}{\partial\langle q|A_i\rangle}\right]. \tag{21}$$

This form shows that the cost of evaluating forces scales with $n_{\text{feat}}n_{\text{neigh}}n_{\text{active}}$, indicating how the reduction in the number of sparse points *and* features combines to accelerate the evaluation of energy and forces using a sparse GAP model.

The fitting procedure that is implemented in `librascal` has been discussed in Ref. 74, and we do not repeat it here. It only requires the evaluation of kernels and kernel derivatives between the active set environments and the environments in the structures that are part of the training set and is usually limited by memory more than by computational expense. In the following benchmarks we adopt the polynomial kernel, which has been widely used to introduce non-linearity into SOAP-based GAP models,[9,37,59–61]

$$k_\zeta(M_I,A_i) = \left[\sum_q\langle M_I|q\rangle\langle q|A_i\rangle\right]^\zeta, \tag{22}$$

whose derivative can be simply computed as

$$\frac{\partial k_\zeta(M_I,A_i)}{\partial\langle q|A_i\rangle} = \zeta\langle M_I|q\rangle k_{\zeta-1}(M_I,A_i). \tag{23}$$

C. Benchmarks of sparse models

Having summarized the practical implementation of a sparse GPR model based on SOAP features, we can systematically investigate the effect of the sparsification parameters—the number of sparse environments n_{active} and the number of sparse features n_{feat}—on the different components of an energy and force calculation. Figures 8 and 9 show the full cost of evaluating a MLIP for different classes of materials, both with and without the evaluation of forces, for different levels of sparsification in terms of both n_{active} and n_{feat}. The cost is broken down in the contributions from the evaluation of the neighbor list, the representation, and model evaluation (prediction) steps.

Figure 8 shows that, when using the full feature vector in the model,[75] the evaluation of the kernels contributes substantially to the cost of predicting energies. In QUIP, this cost, which scales linearly with the number of active points, matches the cost of evaluating the representations, independent of n_{active}, since the same number of representations must always be computed for the target structure, at $n_{\text{active}} \approx 2000$ for Si and $n_{\text{active}} \approx 200$ for the molecular crystals. Due to the optimization of the feature evaluation step in `librascal`, the

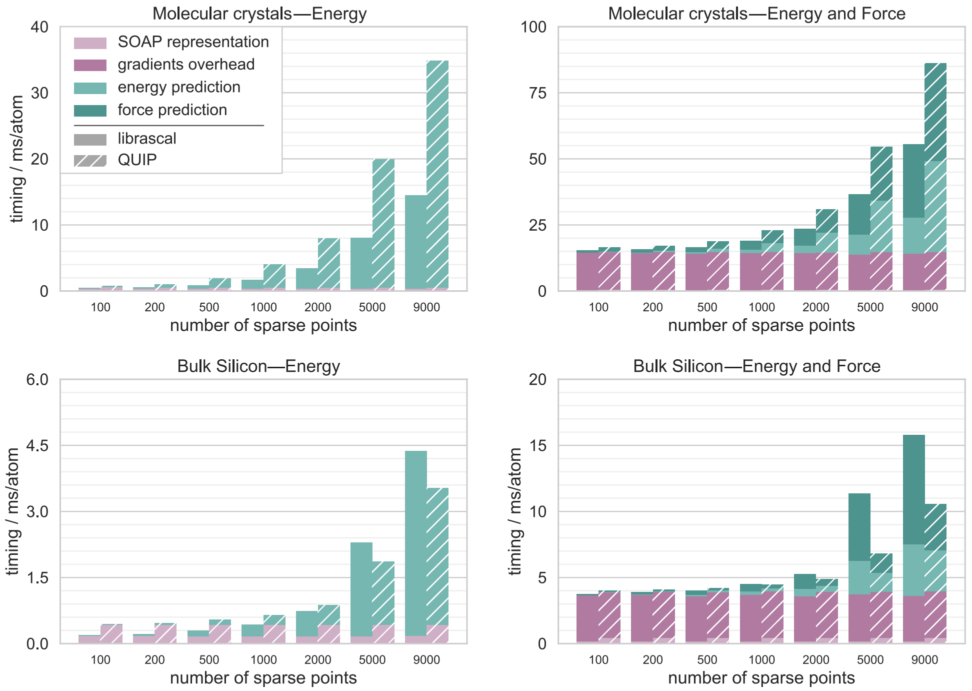

FIG. 8. Prediction timings for GAP models as a function of the number of sparse points, with (right) and without (left) the evaluation of forces, with minimal feature sparsification, i.e., just enough to eliminate redundant symmetric terms (these are retained in `librascal` for simpler bookkeeping). We used all unique SOAP features for each system here, meaning 6660 features for the molecular crystals and 715 features for bulk silicon.

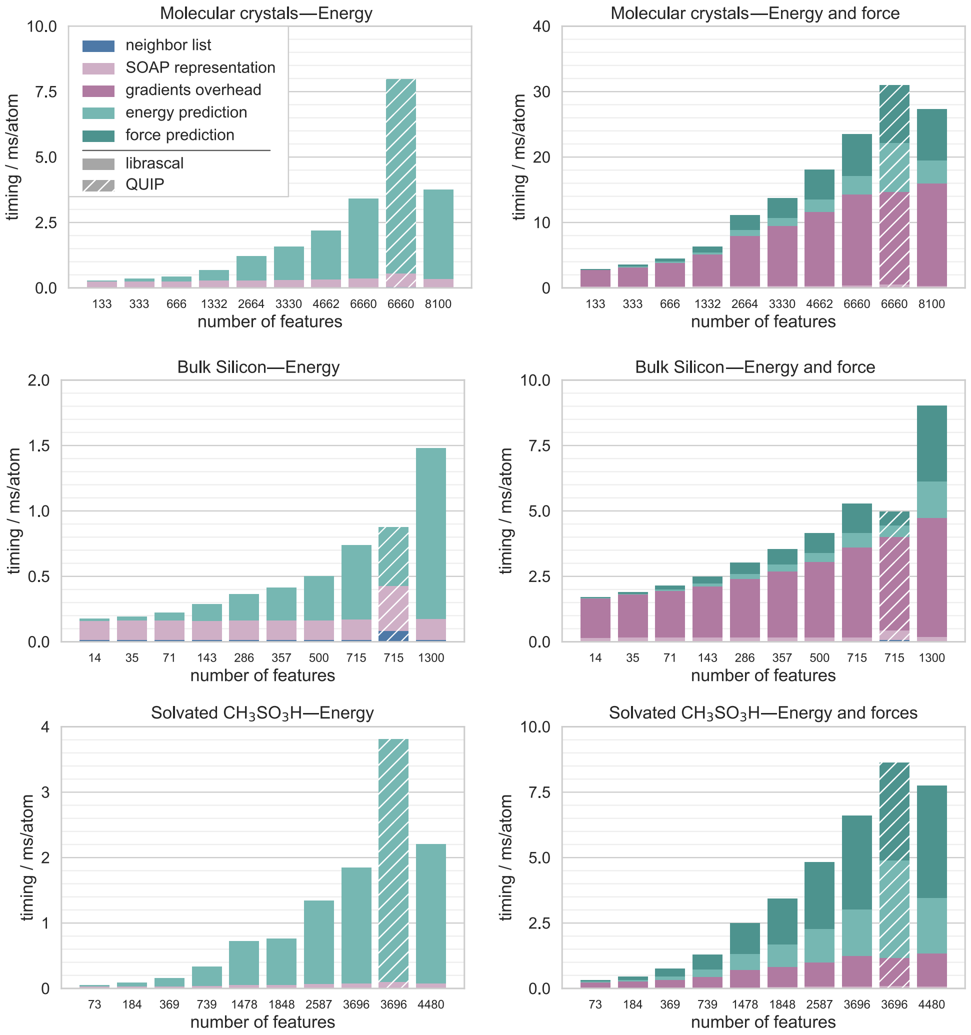

FIG. 9. Prediction timings for GAP models as a function of the number of features, with (right) and without (left) the evaluation of forces. All models use 2000 sparse points for the sparse kernel basis. The rightmost column in each plot shows the cost with some redundant features, which are computed by default in `librascal` for simpler bookkeeping. In practical applications, though, we recommend these be eliminated automatically through feature sparsification. The column directly to the left of the QUIP benchmark corresponds to a lossless `librascal` calculation and can be used as a comparison of the performance without including feature selection. Figure 10 demonstrates the trade-off between accuracy and cost for aggressive feature selection.

kernel evaluation dominates down to even smaller n_{active}. Note also the lower cost of the kernel evaluation for the molecular crystals in `librascal`, which can be explained by the fact that only the features associated with chemical species that are present in each structure are computed, while in QUIP, they yield blocks of zeros that are multiplied to compute scalar products. We observe, however, that QUIP achieves better performance for the evaluation of the kernel (and kernel derivatives) for a single-element dataset, possibly due to a better vectorization of the memory-bound operations, which is allowed by the contiguous storage of the features. Evaluating also forces (right-hand panels of Fig. 8) introduces a very large overhead to feature calculation (up to one order of magnitude, as discussed in Sec. IV G) and roughly doubles the cost of the other steps involved in obtaining model predictions. Since the cost of feature evaluation is independent of n_{active}, the active set can be expanded up to thousands of environments before the model evaluation becomes comparable to feature evaluation.

In order to accelerate calculations further, it is then necessary to reduce not only the time needed to compute the model but also the time needed to compute the representation itself. In Fig. 7, we show how restricting the evaluation of SOAP features to a smaller subset of the $\langle a_1 n_1; a_2 n_2; l|$ indices reduces by up to an order of magnitude the cost of evaluating the feature vector and its gradients. Figure 9 demonstrates how this speedup combines with the acceleration of the model evaluation step, whose nominal complexity also scales linearly with n_{feat}, for an intermediate size of the active set $n_{\text{active}} = 2000$. For simple, single-component systems such as bulk silicon, the cost saturates to that of evaluating the density expansion coefficients, and so, the overall speedup that can be achieved by feature sparsification is limited to about a factor of 2 with respect to the full SOAP power spectrum. For multi-component systems, such as the molecular crystals dataset or the solvated CH_3SO_3H dataset, a speedup of nearly an order of magnitude is possible.

D. Accuracy-cost tradeoff

While the performance optimization discussed in Sec. IV can dramatically increase the efficiency of a MLIP based on SOAP features and sparse GPR, one should obviously ensure that models with reduced n_{active} and n_{feat} still achieve the desired accuracy. The data-driven determination of the most representative and diverse set of features and samples is a very active area of research, using both unsupervised[15,29,49–51] and, very recently, semi-supervised[76] criteria to select an optimal subset. Here, we use the well-established FPS to sort features and environments in decreasing order of importance, starting from the full list of environments for the Si dataset and a pool of 715 features, corresponding to $n_{\max} = 10$, $l_{\max} = 12$. We train MLIPs to reproduce energy and forces and report the fourfold cross-validation error as well as the cost for evaluating the energy and its gradients in Fig. 10, using only the "best" n_{active} active points and n_{feat} features. We also report the "Δ" measure introduced in Ref. 77 as an indication of the ability of the ML model to reproduce properties that are indirectly related to the accuracy of the PES,[35,78]

$$\Delta = \sqrt{\frac{\int_{0.95\,V_0}^{1.06\,V_0}\left[E^{\mathrm{GAP}}(V) - E^{\mathrm{DFT}}(V)\right]^2 \mathrm{d}V}{0.12\,V_0}}, \tag{24}$$

where E^{GAP} and E^{DFT} are the GAP and DFT energies relative to the diamond energy minimum and V_0 is the volume of the minimum DFT energy structure for each phase.

The results clearly indicate that it is possible to considerably reduce the cost of the MLIP with little impact on the accuracy of the model. Severe degradation of model performance occurs in the regime in which the computational cost is dominated by the calculation of the density expansion coefficients, suggesting that further optimization of the evaluation of $\langle anlm|\rho_i\rangle$ might bring only small performance gains in most practical use cases.

VI. EXPERIMENTAL FEATURES

The spherical expansion coefficients can also be used to compute equivariant features and kernels,[22,79] as well as higher-body-order invariants.[14,19] This evaluation is easily and efficiently achieved with an external library, as it is done in the current implementation[72] of the N-body iterative contraction of equivariant (NICE) framework.[23] Furthermore, `librascal` contains experimental

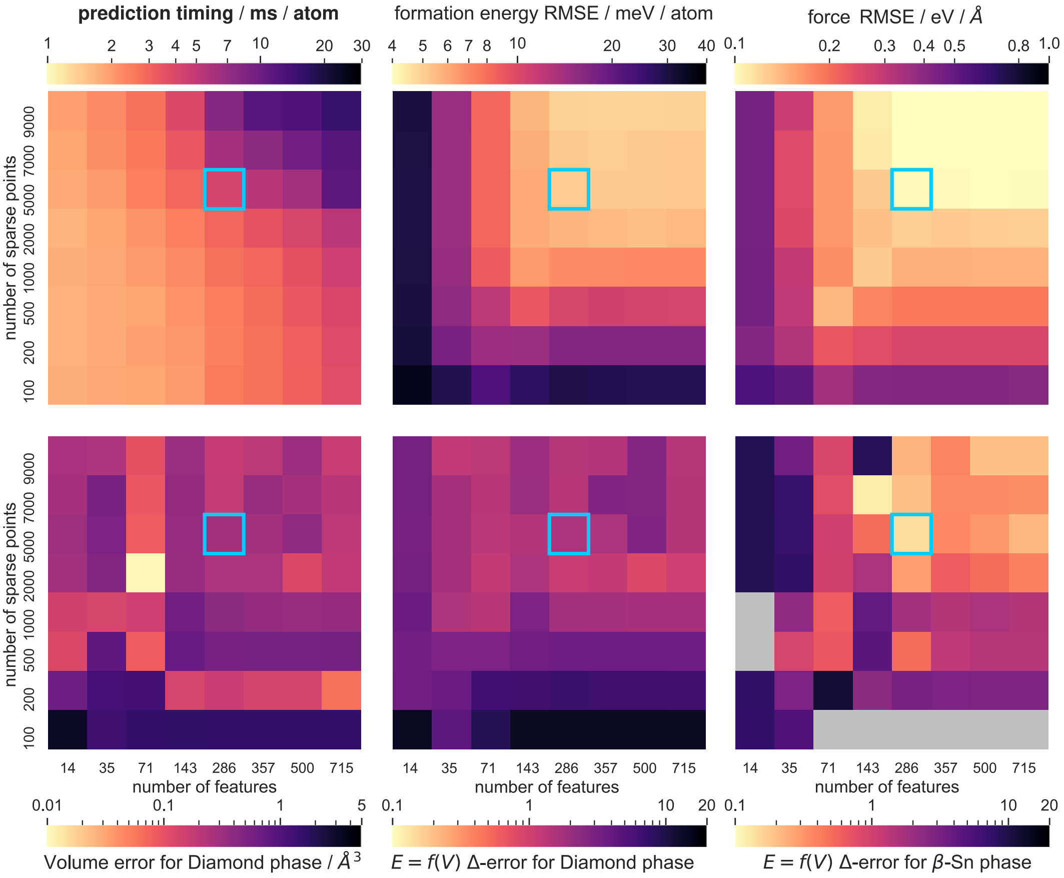

FIG. 10. Assessment of the performance of a GAP model for bulk silicon. The cost of evaluating energy and forces as a function of n_{feat} and n_{active} is shown in the top-left panel. The other panels benchmark the accuracy of the model: clockwise, they show the root mean square error for the predicted energies and forces, the Δ-error—see Eq. (24)—for the energy/volume curve for the β-Sn and diamond phases of Si, and the error on the equilibrium volume of the diamond phase. All errors are computed relative to the reference density functional theory (DFT) values.[35] One can assess the trade-off between model accuracy and computational cost by comparing corresponding cells in each plot. For instance, (n_{feat} = 286, n_{active} = 5000), highlighted with a blue border, has comparable accuracy to the full model, while requiring only a third of the evaluation time.

implementations of other representations based on the SOAP framework, for example, the bispectrum[12,18] and the $\nu = 2$ equivariants that underlie the λ-SOAP kernels[22] (which is also available as an independent implementation[64]). As development progresses, these libraries will be further integrated with `librascal`, harmonizing and streamlining the user-facing APIs and achieving the best balance between modularity and evaluation efficiency.

VII. CONCLUSIONS

In this paper, we have made practical use of recent insights into the relationships between several families of representations that are typically applied to the construction of machine-learning models of the atomic-scale properties of molecules and materials. We have demonstrated how these insights can be translated into algorithms for more efficient computation of not only these representations, most notably SOAP, but also the atom-density bispectrum and the λ-SOAP equivariants. We have shown how the radial basis used to expand the density can be chosen at will and computed quickly using a spline approximation. Together with a fast gradient evaluation, this reduces the cost of computing the density expansion to the point where it is no longer the rate limiting step of the calculation in typical settings. Further optimizations can be obtained by a "lossy" strategy, which trades off some accuracy for efficiency by discarding redundant or highly correlated entries both in the active set of a projected-process regression model and in the invariant features. We have implemented all these optimizations in `librascal`, a modular, user-friendly, and efficient open-source library purpose-built for the computation of atom-density features (especially SOAP).

In order to test these optimizations in practice, we have run benchmarks over different kinds of datasets spanning elemental materials and organic molecules in isolation, in crystalline phases, and in bulk liquid phases. Using one of the most widespread codes for the training and evaluation of SOAP-based machine-learning interatomic potentials as a reference, we have found that our implementation of the SOAP representation is much faster, but that the advantage is less dramatic when considering also the calculation of a kernel model, which scales with the number of features, and that of the gradients, which is dominated by a term that scales with the number of neighbors in both codes. Feature selection addresses these additional overheads and allows for an acceleration of the end-to-end evaluation time of energy and forces by a factor anywhere between 4 and 10 with a minimal increase in the prediction errors. Our tests show that in the current implementation, when using realistic values of the parameters, the different steps of the calculation contribute similarly to the total cost, indicating that there is no single obvious bottleneck. Further improvements, although possible, should consider the model as a whole and especially improve the accuracy/cost balance of lossy model compression techniques.

SUPPLEMENTARY MATERIAL

See the supplementary material for figures describing the benchmark results for all the datasets mentioned in this work.

AUTHORS' CONTRIBUTIONS

F.M. and M.V. contributed equally to this work.

ACKNOWLEDGMENTS

F.M., M.V., M.S., and M.C. acknowledge support from the NCCR MARVEL, funded by the Swiss National Science Foundation (SNSF). A.G. and M.C. acknowledge support from the Swiss National Science Foundation (Project No. 200021-182057). G.F. acknowledges support from the European Center of Excellence MaX, Materials at the Exascale—Grant No. 676598.

APPENDIX A: EFFICIENT IMPLEMENTATION OF ${}_1F_1$

The confluent hypergeometric function of the first kind is defined as

$$ {}_1F_1(a,b,z) = \sum_{s=0}^{\infty} \frac{(a)_s}{(b)_s s!} z^s, \tag{A1}$$

where $(a)_s$ is the Pochhammer symbol [Ref. 80, Chap. 5.2(iii)]. To efficiently compute Eq. (11), we implement a restricted version of ${}_1F_1$,

$$ G(a,b,r_{ij}) = \frac{\Gamma(a)}{\Gamma(b)} \exp[-cr_{ij}^2]\,{}_1F_1\left(a, b, \frac{c^2 r_{ij}^2}{c+d_n}\right), \tag{A2}$$

where $a = \frac{n+l+3}{2}$ and $b = l + \frac{3}{2}$. We take into account that the arguments of ${}_1F_1$ are real and positive, and we avoid its artificial overflow by using the asymptotic expansion [Eqs. (13.2.4) and (13.7.1) in Ref. 80],

$$ \lim_{z\to\infty} {}_1F_1(a,b,z) = e^z z^{a-b} \frac{\Gamma(b)}{\Gamma(a)} \sum_{s=0}^{\infty} \frac{(b-a)_s (1-a)_s}{s!} z^{-s} \tag{A3}$$

since the exponential in Eq. (11) can be factorized as $\exp[\frac{c^2 r_{ij}^2}{c+d_n}]$ $\exp[-cr_{ij}^2] = \exp[cr_{ij}^2(\frac{c}{c+d_n} - 1)]$ and $\frac{c}{c+d_n} - 1 < 0$. Note that G is implemented as a class so that the switching point between the direct series and the asymptotic expansion evaluations is determined at construction for particular values of a and b using the bisection method.

For each value of n, the function G and its derivatives with respect to r_{ij} can be efficiently evaluated using the two step recurrence downward relation,

$$ \begin{aligned} G(a+1,b+1,r_{ij}) = {} & \frac{c^2 r_{ij}^2}{c+d_n} G(a+2,b+3,r_{ij}) \\ & + (b+1) G(a+1,b+2,r_{ij}), \end{aligned} \tag{A4}$$

$$ \begin{aligned} G(a,b,r_{ij}) = {} & \frac{c^2 r_{ij}^2}{c+d_n} \frac{a-b}{a} G(a+1,b+2,r_{ij}) \\ & + \frac{b}{a} G(a+1,b+1,r_{ij}), \end{aligned} \tag{A5}$$

with $\partial G(a,b,r_{ij})/\partial r_{ij} = \frac{2c^2 r_{ij}}{c+d_n} G(a+1,b+1,r_{ij}) - 2cr_{ij} G(a,b,r_{ij})$. We found empirically that only the downward recurrence relation was numerically stable for our range of parameters. Note that $a + 1$ corresponds effectively to steps of $l + 2$, so computing G and $\frac{\mathrm{d}G}{\mathrm{d}r_{ij}}$ for all $l \in [0, l_{\max}]$ and all $n \in [0, n_{\max}]$ requires $4n_{\max}$ evaluations when using this recurrence relation.

APPENDIX B: DERIVATIVES OF THE ENERGY FUNCTION

We have defined an atom centered energy model such that the energy associated with structure A can be written as in Eq. (16), $E(A) = \sum_{i\in A} E(A_i)$. The structure A is determined by the set of atomic coordinates and species $\{\mathbf{r}_i, a_i\}$ and (for periodic structures) unit cell vectors $\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\}$. The atom-centered environment A_i is entirely characterized by the atom centered vectors $\{\mathbf{r}_{ji} = \mathbf{r}_j - \mathbf{r}_i\}$, with $r_{ji} < r_{\text{cut}}$. The derivatives of E with respect to the position of atom k (the negative of the force acting on the atom) can be computed using the chain rule,

$$\frac{\partial E(A)}{\partial \mathbf{r}_k} = \sum_{i\in A} \frac{\partial E(A_i)}{\partial \mathbf{r}_k} = \sum_{i\in A}\sum_{j\in A_i} \frac{\partial E(A_i)}{\partial \mathbf{r}_{ji}} \cdot \frac{\partial \mathbf{r}_{ji}}{\partial \mathbf{r}_k}. \qquad \text{(B1)}$$

Here, index j runs over the neighbors of atom i, which includes periodic images, if the system is periodic. The term $\partial \mathbf{r}_{ji}/\partial \mathbf{r}_k$ is 0 unless $k = i$ (in which case, it evaluates to -1) or if $j = k$ (in which case, it evaluates to 1). In the periodic case, the derivative with respect to $\mathbf{r}_k$ has to be interpreted as one in which all periodic images of atom k are displaced simultaneously. Thus, when the neighbor j is a periodic image of k, $\partial \mathbf{r}_{ji}/\partial \mathbf{r}_k = 1$, and if $k = i$, the total contribution of the periodic images of i is 0.

For the virial, we need to compute the derivative of the energy with respect to infinitesimal strain deformations of the unit cell $\boldsymbol{\eta}$. Using the atom-centered decomposition and applying the chain rule as mentioned above, one gets

$$\begin{aligned}\frac{\partial E(A)}{\partial \boldsymbol{\eta}} &= \sum_{i\in A} \frac{\partial E(A_i)}{\partial \boldsymbol{\eta}} = \sum_{i\in A}\sum_{j\in A_i} \frac{\partial E(A_i)}{\partial \mathbf{r}_{ji}} \frac{\partial \mathbf{r}_{ji}}{\partial \boldsymbol{\eta}} \\ &= \sum_{i\in A}\sum_{j\in A_i} \frac{\partial E(A_i)}{\partial \mathbf{r}_{ji}} \otimes \mathbf{r}_{ji}, \end{aligned} \qquad \text{(B2)}$$

where again j runs over all the neighbors including periodic images. It is interesting to note that the periodic images of i have a non-zero contribution to the virial.

DATA AVAILABILITY

Data supporting the findings in this paper are available from public repositories as referenced or upon reasonable request to the authors. The source code of `librascal` is available from an open-source repository,[81] and workflows that can reproduce the benchmarks reported in this paper are available from a separate repository.[32]

REFERENCES

[1] M. A. Rohrdanz, W. Zheng, M. Maggioni, and C. Clementi, "Determination of reaction coordinates via locally scaled diffusion map," J. Chem. Phys. **134**, 124116 (2011).

[2] G. Pérez-Hernández, F. Paul, T. Giorgino, G. De Fabritiis, and F. Noé, "Identification of slow molecular order parameters for Markov model construction," J. Chem. Phys. **139**, 015102 (2013).

[3] T. D. Huan, A. Mannodi-Kanakkithodi, and R. Ramprasad, "Accelerated materials property predictions and design using motif-based fingerprints," Phys. Rev. B **92**, 014106 (2015).

[4] P. Gasparotto, R. H. Meißner, and M. Ceriotti, "Recognizing local and global structural motifs at the atomic scale," J. Chem. Theory Comput. **14**, 486–498 (2018).

[5] M. Ceriotti, "Unsupervised machine learning in atomistic simulations, between predictions and understanding," J. Chem. Phys. **150**, 150901 (2019).

[6] J. Rogal, E. Schneider, and M. E. Tuckerman, "Neural-network-based path collective variables for enhanced sampling of phase transformations," Phys. Rev. Lett. **123**, 245701 (2019).

[7] B. A. Helfrecht, R. K. Cersonsky, G. Fraux, and M. Ceriotti, "Structure-property maps with Kernel principal covariates regression," Mach. Learn.: Sci. Technol. **1**, 045021 (2020).

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

[9] 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).

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

[11] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, MA, 2006).

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

[13] S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko, "Towards exact molecular dynamics simulations with machine-learned force fields," Nat. Commun. **9**, 3887 (2018).

[14] M. J. Willatt, F. Musil, and M. Ceriotti, "Atom-density representations for machine learning," J. Chem. Phys. **150**, 154110 (2019).

[15] B. Onat, C. Ortner, and J. R. Kermode, "Sensitivity and dimensionality of atomic environment representations used for machine learning interatomic potentials," J. Chem. Phys. **153**, 144106 (2020).

[16] B. Parsaeifard, D. S. De, A. S. Christensen, F. A. Faber, E. Kocer, S. De, J. Behler, A. von Lilienfeld, and S. Goedecker, "An assessment of the structural resolution of various fingerprints commonly used in machine learning," Mach. Learn.: Sci. Technol. (published online, 2020).

[17] S. N. Pozdnyakov, M. J. Willatt, A. P. Bartók, C. Ortner, G. Csányi, and M. Ceriotti, "Incompleteness of atomic structure representations," Phys. Rev. Lett. **125**, 166001 (2020).

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

[19] R. Drautz, "Atomic cluster expansion for accurate and transferable interatomic potentials," Phys. Rev. B **99**, 014104 (2019).

[20] M. Bachmayr, G. Csanyi, R. Drautz, G. Dusson, S. Etter, C. van der Oord, and C. Ortner, "Atomic cluster expansion: Completeness, efficiency and stability," arXiv:1911.03550 [cs, math] (2020).

[21] A. Shapeev, "Accurate representation of formation energies of crystalline alloys with many components," Comput. Mater. Sci. **139**, 26–30 (2017).

[22] A. Grisafi, D. M. Wilkins, G. Csányi, and M. Ceriotti, "Symmetry-adapted machine learning for tensorial properties of atomistic systems," Phys. Rev. Lett. **120**, 036002 (2018).

[23] J. Nigam, S. Pozdnyakov, and M. Ceriotti, "Recursive evaluation and iterative contraction of N-body equivariant features," J. Chem. Phys. **153**, 121101 (2020).

[24] T. T. Nguyen, E. Székely, G. Imbalzano, J. Behler, G. Csányi, M. Ceriotti, A. W. Götz, and F. Paesani, "Comparison of permutationally invariant polynomials, neural networks, and Gaussian approximation potentials in representing water interactions through many-body expansions," J. Chem. Phys. **148**, 241725 (2018).

[25] Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csányi, A. V. Shapeev, A. P. Thompson, M. A. Wood, and S. P. Ong, "Performance and cost assessment of machine learning interatomic potentials," J. Phys. Chem. A **124**, 731–745 (2020).

[26] C. W. Rosenbrock, K. Gubaev, A. V. Shapeev *et al.* "Machine-learned interatomic potentials for alloys and alloy phase diagrams," NPJ Comput. Mater. **7**, 24 (2021).

[27] M. J. Willatt, F. Musil, and M. Ceriotti, "Feature optimization for atomistic machine learning yields a data-driven construction of the periodic table of the elements," Phys. Chem. Chem. Phys. **20**, 29661–29668 (2018).

[28] M. A. Caro, "Optimizing many-body atomic descriptors for enhanced computational performance of machine learning based interatomic potentials," Phys. Rev. B **100**, 024112 (2019); arXiv:1905.02142.

[29] A. P. Bartók and G. Csányi, "Gaussian approximation potentials: A brief tutorial introduction," Int. J. Quantum Chem. **115**, 1051–1057 (2015).

[30] C. S. Adorf, P. M. Dodd, V. Ramasubramani, and S. C. Glotzer, "Simple data and workflow management with the signac framework," Comput. Mater. Sci. **146**, 220–229 (2018).

[31] C. S. Adorf, V. Ramasubramani, B. D. Dice, M. M. Henry, P. M. Dodd, and S. C. Glotzer (2019). "Glotzerlab/signac," Zenodo. https://doi.org/10.5281/zenodo.2581327

[32] F. Musil, LIBRASCAL benchmark workflows, https://github.com/felixmusil/rascal_benchmarks.

[33] F. Musil, M. Stricker, A. Goscinski, F. Giberti, M. Veit, T. Junge, G. Fraux, M. Ceriotti, R. Cersonsky, M. Willatt, and A. Grisafi (2021). "cosmo-epfl/librascal," Zenodo. https://doi.org/10.5281/zenodo.4526063

[34] J. Kermode (2018). "Silicon testing framework," Zenodo. https://zenodo.org/record/1250555

[35] A. P. Bartók, J. Kermode, N. Bernstein, and G. Csányi, "Machine learning a general-purpose interatomic potential for silicon," Phys. Rev. X **8**, 041048 (2018).

[36] M. Veit (2018). "Bulk methane models and simulation parameters" Apollo, Dataset https://www.repository.cam.ac.uk/handle/1810/279000.

[37] M. Veit, S. K. Jain, S. Bonakala, I. Rudra, D. Hohl, and G. Csányi, "Equation of state of fluid methane from first principles with machine learning potentials," J. Chem. Theory Comput. **15**, 2574–2586 (2019).

[38] K. Rossi, V. Jurásková, R. Wischert, L. Garel, C. Corminbœuf, and M. Ceriotti, "Simulating solvation and acidity in complex mixtures with first-principles accuracy: The case of CH_3SO_3H and H_2O_2 in phenol," J. Chem. Theory Comput. **16**, 5139–5149 (2020); arXiv:2006.12597.

[39] F. Musil, M. J. Willatt, M. A. Langovoy, and M. Ceriotti, "Fast and accurate uncertainty estimation in chemical machine learning," J. Chem. Theory Comput. **15**, 906–915 (2019).

[40] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," Sci. Data **1**, 140022 (2014).

[41] J. C. Light and T. Carrington, Jr., *Discrete-Variable Representations and Their Utilization*, (John Wiley & Sons, Ltd., 2000), pp. 263–310.

[42] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables* (Dover Publications, Inc., 1972).

[43] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. (Cambridge University Press, 2007).

[44] A. Goscinski, G. Fraux, G. Imbalzano, and M. Ceriotti, "The role of feature space in atomistic learning," Mach. Learn.: Sci. Technol. (published online, 2021).

[45] P. Rowe, V. L. Deringer, P. Gasparotto, G. Csányi, and A. Michaelides, "An accurate and transferable machine learning potential for carbon," J. Chem. Phys. **153**, 034702 (2020).

[46] T. Limpanuparb and J. Milthorpe, "Associated Legendre polynomials and spherical harmonics computation for chemistry applications," arXiv:1410.1748 (2014).

[47] M. Galassi *et al.*, *GNU Scientific Library Reference Manual*, 3rd ed. (Network Theory, 2009), p. 573.

[48] A. Glielmo, C. Zeni, and A. De Vita, "Efficient nonparametric n-body force fields from machine learning," Phys. Rev. B **97**, 184307 (2018).

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

[50] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," Proc. Natl. Acad. Sci. U. S. A. **106**, 697–702 (2009).

[51] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," IEEE Trans. Image Process. Publ. IEEE Signal Process. Soc. **6**, 1305–1315 (1997).

[52] M. Ceriotti, G. A. Tribello, and M. Parrinello, "Demonstrating the transferability and the descriptive power of sketch-map," J. Chem. Theory Comput. **9**, 1521–1532 (2013).

[53] We use the GAP version "1611600208" implemented in the QUIP code.

[54] A. Singraber (2020). "N2P2," GitHub https://github.com/CompPhysVienna/n2p2.

[55] J. S. Smith, O. Isayev, and A. E. Roitberg, "ANI-1: An extensible neural network potential with DFT accuracy at force field computational cost," Chem. Sci. **8**, 3192–3203 (2017).

[56] R. Lot, F. Pellegrini, Y. Shaidu, and E. Küçükbenli, "PANNA: Properties from artificial neural network architectures," Comput. Phys. Commun. **256**, 107402 (2020).

[57] H. Wang, L. Zhang, J. Han, and W. E, "DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics," Comput. Phys. Commun. **228**, 178–184 (2018).

[58] A. Bartók-Pártay, L. Bartók-Pártay, F. Bianchini, A. Butenuth, M. Caccin, S. Cereda, G. Csányi, A. Comisso, T. Daff, S. John, C. Gattinoni, G. Moras, J. Kermode, L. Mones, A. Nichol, D. Packwood, L. Pastewka, G. Peralta, I. Solt, O. Strickson, W. Szlachta, C. Varnai, M. Veit, and S. Winfield (2020). "libAtoms+QUIP," GitHub https://github.com/libatoms/QUIP.

[59] V. L. Deringer and G. Csányi, "Machine learning based interatomic potential for amorphous carbon," Phys. Rev. B **95**, 094203 (2017).

[60] F. C. Mocanu, K. Konstantinou, T. H. Lee, N. Bernstein, V. L. Deringer, G. Csányi, and S. R. Elliott, "Modeling the phase-change memory material, $Ge_2Sb_2Te_5$, with a machine-learned interatomic potential," J. Phys. Chem. B **122**, 8998–9006 (2018).

[61] M. A. Caro, V. L. Deringer, J. Koskinen, T. Laurila, and G. Csányi, "Growth mechanism and origin of high sp^3 content in tetrahedral amorphous carbon," Phys. Rev. Lett. **120**, 166101 (2018).

[62] Z. Zhang, G. Csányi, and D. Alfè, "Partitioning of sulfur between solid and liquid iron under Earth's core conditions: Constraints from atomistic simulations with machine learning potentials," Geochim. Cosmochim. Acta **291**, 5–18 (2020).

[63] L. Himanen, M. O. J. Jäger, E. V. Morooka, F. F. 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).

[64] D. M. Wilkins and A. Grisafi (2020). "TENSOAP," GitHub, https://github.com/dilkins/TENSOAP.

[65] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, "Machine learning of accurate energy-conserving molecular force fields," Sci. Adv. **3**, e1603015 (2017); arXiv:1611.04678.

[66] A. S. Christensen, F. A. Faber, B. Huang, L. A. Bratholm, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld (2017). "QML," Zenodo. https://doi.org/10.5281/zenodo.817332.

[67] F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld, "Alchemical and structural distribution based representation for universal quantum machine learning," J. Chem. Phys. **148**, 241717 (2018).

[68] A. S. Christensen, F. A. Faber, and O. A. von Lilienfeld, "Operators in quantum machine learning: Response properties in chemical space," J. Chem. Phys. **150**, 064105 (2019).

[69] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," J. Comput. Phys. **117**, 1–19 (1995).

[70] C. van der Oord, G. Dusson, G. Csányi, and C. Ortner, "Regularised atomic body-ordered permutation-invariant polynomials for the construction of interatomic potentials," Mach. Learn.: Sci. Technol. **1**, 015004 (2020).

[71] C. Ortner (2019). "JuLIP: Julia Library for Interatomic Potentials" GitHub, https://github.com/JuliaMolSim/JuLIP.jl.

[72] S. Pozdnyakov (2020). "NICE libraries" GitHub, https://github.com/cosmo-epfl/nice.

[73] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems* (MIT Press, 2006), pp. 1257–1264.

[74] G. Csányi, M. J. Willatt, and M. Ceriotti, "Machine-learning of atomic-scale properties based on physical principles," in *Machine Learning Meets Quantum Physics*, edited by K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller (Springer International Publishing, Cham, 2020), Vol. 968, pp. 99–127.

[75] In practice, to match the number of features computed by QUIP, we use a mild feature sparsification in `librascal` that corresponds to the same use of the $\langle an_1; an_2; l| = \langle an_2; an_1; l|$ symmetry that is implemented in QUIP.

[76] R. K. Cersonsky, B. A. Helfrecht, E. A. Engel, and M. Ceriotti, "Improving sample and feature selection with principal covariates regression," arXiv:2012.12253 (2020).

[77] K. Lejaeghere, G. Bihlmayer, T. Bjorkman, P. Blaha, S. Blugel, V. Blum, D. Caliste, I. E. Castelli, S. J. Clark, A. Dal Corso, S. de Gironcoli, T. Deutsch, J. K. Dewhurst, I. Di Marco, C. Draxl, M. Du ak, O. Eriksson, J. A. Flores-Livas, K. F. Garrity, L. Genovese, P. Giannozzi, M. Giantomassi, S. Goedecker, X. Gonze, O. Granas, E. K. U. Gross, A. Gulans, F. Gygi, D. R. Hamann, P. J. Hasnip, N. A. W. Holzwarth, D. Iu an, D. B. Jochym, F. Jollet, D. Jones, G. Kresse, K. Koepernik, E. Kucukbenli, Y. O. Kvashnin, I. L. M. Locht, S. Lubeck, M. Marsman, N. Marzari, U. Nitzsche, L. Nordstrom, T. Ozaki, L. Paulatto, C. J. Pickard, W. Poelmans, M. I. J. Probert, K. Refson, M. Richter, G.-M. Rignanese, S. Saha, M. Scheffler, M. Schlipf, K. Schwarz, S. Sharma, F. Tavazza, P. Thunstrom, A. Tkatchenko, M. Torrent, D. Vanderbilt, M. J. van Setten, V. Van Speybroeck, J. M. Wills, J. R. Yates, G.-X. Zhang, and S. Cottenier, "Reproducibility in density functional theory calculations of solids," Science **351**, aad3000 (2016).

[78] W. J. Szlachta, A. P. Bartók, and G. Csányi, "Accuracy and transferability of Gaussian approximation potential models for tungsten," Phys. Rev. B **90**, 104108 (2014).

[79] A. Glielmo, P. Sollich, and A. De Vita, "Accurate interatomic force fields via machine learning with covariant kernels," Phys. Rev. B **95**, 214302 (2017).

[80] *NIST Digital Library of Mathematical Functions*, edited by F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain (NIST, 2020), http://dlmf.nist.gov/.

[81] F. Musil, M. Veit, T. Junge, M. Stricker, A. Goscinki, G. Fraux, and M. Ceriotti (2018). "LIBRASCAL," GitHub, https://github.com/cosmo-epfl/librascal.