The Journal of Chemical Physics

FCHL revisited: Faster and more accurate quantum machine learning

Cite as: J. Chem. Phys. **152**, 044107 (2020); https://doi.org/10.1063/1.5126701
Submitted: 04 September 2019 . Accepted: 07 January 2020 . Published Online: 27 January 2020

Anders S. Christensen, Lars A. Bratholm, Felix A. Faber, and O. Anatole von Lilienfeld

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

FCHL revisited: Faster and more accurate quantum machine learning

Cite as: J. Chem. Phys. 152, 044107 (2020); doi: 10.1063/1.5126701
Submitted: 4 September 2019 • Accepted: 7 January 2020 •
Published Online: 27 January 2020

Anders S. Christensen,[1] Lars A. Bratholm,[2,3] Felix A. Faber,[1] and O. Anatole von Lilienfeld[1,a)]

AFFILIATIONS

[1] Department of Chemistry, National Center for Computational Design and Discovery of Novel Materials (MARVEL), Institute of Physical Chemistry, University of Basel, Klingelbergstrasse 80, CH-4056 Basel, Switzerland
[2] School of Mathematics, University of Bristol, Bristol BS8 1TW, United Kingdom
[3] School of Chemistry, University of Bristol, Bristol BS8 1TS, United Kingdom

Note: This paper is part of the JCP Special Topic on Machine Learning Meets Chemical Physics.
[a)] **Author to whom correspondence should be addressed:** anatole.vonlilienfeld@unibas.ch

ABSTRACT

We introduce the FCHL19 representation for atomic environments in molecules or condensed-phase systems. Machine learning models based on FCHL19 are able to yield predictions of atomic forces and energies of query compounds with chemical accuracy on the scale of milliseconds. FCHL19 is a revision of our previous work [F. A. Faber *et al.*, J. Chem. Phys. 148, 241717 (2018)] where the representation is discretized and the individual features are rigorously optimized using Monte Carlo optimization. Combined with a Gaussian kernel function that incorporates elemental screening, chemical accuracy is reached for energy learning on the QM7b and QM9 datasets after training for minutes and hours, respectively. The model also shows good performance for non-bonded interactions in the condensed phase for a set of water clusters with a mean absolute error (MAE) binding energy error of less than 0.1 kcal/mol/molecule after training on 3200 samples. For force learning on the MD17 dataset, our optimized model similarly displays state-of-the-art accuracy with a regressor based on Gaussian process regression. When the revised FCHL19 representation is combined with the operator quantum machine learning regressor, forces and energies can be predicted in only a few milliseconds per atom. The model presented herein is fast and lightweight enough for use in general chemistry problems as well as molecular dynamics simulations.

I. INTRODUCTION

Approximate models have been used to make predictions in chemistry since the beginning of theoretical chemistry. In recent years, however, data-driven machine learning (ML) models which can make predictions across chemical space with chemical accuracy are becoming increasingly common in the literature.

Tasks such as molecular dynamics (MD) simulations and geometry optimizations have been a standard tool in the toolbox of the computational chemist for many years, and several machine learning models now provide the gradients necessary to carry out such tasks.[2–21] We have previously published a machine learning model based on the Faber–Christensen–Huang–Lilienfeld (FCHL18) representation which performs very well on chemical compounds across chemical space,[1] as well as a proof-of-concept implementation of learning and prediction of response properties based on this model,[22] such as atomic force, normal modes, dipole moments, and even IR spectra.

While our FCHL18-based models yielded state-of-the-art accuracy on several benchmark sets,[1,22] the applicability was, in some cases, hindered by poor computational performance, and proper hyperparameter optimization of the model has been computationally unfeasible. Whereas FCHL18 solves an analytical integral to compare atomic environments in order to learn properties of chemical compounds, other ML models use discretized representations that can be handled with far greater computational efficiency.[6,23–32]

In this work, we present a discretized representation for chemical compounds based on our earlier work in Ref. 1, which allows for a very fast evaluation of the L2-distance between two representations. A rigorous Monte Carlo optimization of the model parameters

is performed in order to find a set of universally transferable hyperparameters that yield ML models of high accuracy without any need for re-optimization. We include a detailed review of different kernel-based models with which the representation can be used and highlight their strengths, differences, and shortcomings. Finally, we thoroughly benchmark the predictive accuracy of our models on several established datasets of chemical compounds from the literature. In addition to benchmarking the accuracy of energy and force prediction, we also present timings of our model in order to demonstrate the applicability.

II. THEORY

This section first introduces the representation used to describe atomic environments throughout this work. Second, a number of kernel-based machine learning methods (MOB) which can be used with the representation are discussed. While the representation could in principle also be used favorably with feed-forward neural networks, this paper focuses solely on kernel-based methods.

A. Representation

We have previously compared ML models based on a number of different representations for the QM9 dataset.[1,33] Based on these studies, it is apparent that the currently best performing representations contain certain similarities, although the exact implementations differ. Some of the best performing representations for kernel-based machine learning for chemical compounds are the smooth overlap of atomic densities (SOAP),[34,35] spectrum of London and Axilrod–Teller–Muto (SLATM),[36] the many-body descriptor of Pronobis *et al.*,[37] and FCHL18 representations,[1] while variants of the atom-centered symmetry functions (ACSFs) of Behler[6,30,31] have been shown to perform well for feed-forward neural networks. In brief, these methods contain some terms that are similar: (1) a two-body term that relates to the radial distribution between a central atom and other nearby atoms in its local environment and (2) a three-body term that similarly relates to, for example, distribution of angles and/or distances between atoms in the local environment of the atom.

In this paper, we construct a new atom-centered representation termed FCHL19 that contains such two- and three-body terms and demonstrate that this leads to similar performance. The FCHL19 representation is based on the FCHL18 representation[1] but is discretized in a manner very similar to the well-known ACSF of Behler.[30]

In order to enable faster and more memory efficient machine learning models, it is key that the input representation is as small as possible compared to the information it holds, as evaluation and training times scale linearly/quadratically with representation size.

We show that when the parameters of our new representation are optimized properly, the result is a representation that is compact in size—ensuring faster machine learning algorithms—without loss in predictive accuracy.

Briefly described, the representation is a vector that encodes the atomic environment of an atom in a chemical compound. It consists of a two-body term which encodes radial distributions between the central atoms and neighboring atoms of a given element type. Additionally, the representation contains a three-body term that encodes the mean distances and angles between the atom and neighboring pairs of atoms of given element types.

The representation does not contain an explicit one-body term and, for performance reasons, we do not consider terms of higher order than three-body, but it is possible that the inclusion of such terms could lead to even higher predictive accuracy.[33]

The two- and three-body components of the representation are described in detail in the following text. The procedure to obtain the hyperparameters of the representation is detailed in Sec. IV B, while the optimized parameters are presented in Table III in Appendix A.

1. Two-body function

For a given central atom, a set of radial basis functions is constructed for each unique type of element in the dataset. Each of the $n_{R_{s_2}}$ basis functions in this set is placed on an equidistant grid from $\frac{r_{cut}}{n_{R_{s_2}}}$ to r_{cut}, with r_{cut} being the cut-off radius. We found it advantageous to use log-normal distribution functions for the radial functions, compared to Gaussian functions as used in our previous work.[1] We note that this is an empirical choice and it is possible that a better distribution function could be found, for example, from using an optimization procedure. The log-normal radial basis functions take the form

$$G^{\text{2-body}} = \xi_2(r_{IJ}) f_{\text{cut}}(r_{IJ}) \frac{1}{R_s \sigma(r_{ij})\sqrt{2\pi}} \exp\left(-\frac{(\ln R_s - \mu(r_{ij}))^2}{2\sigma(r_{ij})^2}\right), \quad (1)$$

where R_s is the distance location of the grid point and $\mu(r_{ij})$ and $\sigma(r_{ij})$ are parameters of the log-normal distribution, which in turn depend on the interatomic distance, r_{IJ}, and a hyperparameter, w, given as follows:

$$\mu(r_{ij}) = \ln\left(\frac{r_{IJ}}{\sqrt{1+\frac{w}{r_{IJ}^2}}}\right) \quad \text{and} \quad \sigma(r_{ij})^2 = \ln\left(1+\frac{w}{r_{IJ}^2}\right). \quad (2)$$

The two-body scaling function, $\xi_2(r_{IJ})$, serves the purpose of applying a higher regression weight to terms that are more likely to contribute substantially to the total energy, thus increasing the accuracy of the machine learning procedure for properties that relate to the total energy. Similar to previous studies,[1,38] we found the following form to be suitable:

$$\xi_2(r_{IJ}) = \frac{1}{r_{IJ}^{N_2}}, \quad (3)$$

where the exponent N_2 is a hyperparameter of the representation. Finally, the soft cut-off function used here is

$$f_{\text{cut}}(r_{IJ}) = \begin{cases} \frac{1}{2}\left(\cos\left(\frac{\pi\, r_{IJ}}{r_{\text{cut}}}\right)+1\right) & \text{if} \quad r_{IJ} \le r_{\text{cut}} \\ 0 & \text{if} \quad r_{IJ} > r_{\text{cut}}. \end{cases} \quad (4)$$

Thus, the hyperparameters of the two-body term are the width parameter of the log-normal distributions, w; the exponent of the scaling function, N_2; the cut-off distance, r_{cut}; and the number of radial basis functions, $n_{R_{s_2}}$. Optimized values of these parameters are presented in Table III in Appendix A.

A graphical representation of the two-body function for an H and an O atom in a water molecule is displayed in Fig. 1. For each

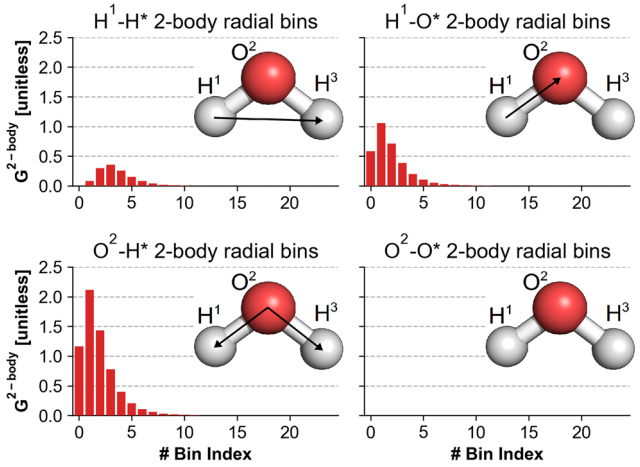

FIG. 1. The values of four unique types of two-body radial basis functions in a water molecule are displayed. The radial spectrum is divided into 24 bins, with r_{cut} = 8.0 Å, w = 0.32 Å, and N_2 = 1.8. The top row contains the radial basis functions for the first H atom and the bottom row for the oxygen atom. The distances that are used to produce the basis functions in each spectrum are marked with black arrows.

atomic environment in the water molecule, the minimal representation will contain two radial distributions, x-H and x-O. Thus, the size of the two-body term scales linearly with the number of possible elements in the atomic environment.

2. Three-body function

The three-body function encodes the distances from an atom to neighboring pairs of atoms in the environment of the atom, as well as the angle between the triplet, and the element types of the neighbors. The resulting function is a product of the following terms:

$$G^{\text{3-body}} = \xi_3 G^{\text{3-body}}_{\text{Radial}} G^{\text{3-body}}_{\text{Angular}} f_{\text{cut}}(r_{IJ}) f_{\text{cut}}(r_{JK}) f_{\text{cut}}(r_{KI}). \quad (5)$$

The radial part is similar to the radial part in the ACSFs used in the ANI-1 neural network,[7,30]

$$G^{\text{3-body}}_{\text{Radial}} = \sqrt{\frac{\eta_3}{\pi}} \exp\left(-\eta_3\left(\tfrac{1}{2}(r_{IJ} + r_{IK}) - R_s\right)^2\right), \quad (6)$$

where η_3 is a parameter that controls the width of the radial distribution functions and again R_s is the location of the radial gridpoints. Finally, the three-body scaling function, ξ_3, is the Axilrod–Teller–Muto term[39,40] with modified exponents,[1,38]

$$\xi_3 = c_3 \frac{1 + 3\cos(\theta_{KIJ})\cos(\theta_{IJK})\cos(\theta_{JKI})}{(r_{IK} r_{JK} r_{KI})^{N_3}}. \quad (7)$$

Here, θ_{KIJ} is the angle between the three atoms K, I, and J and c_3 is a weight term that balances the weight of the three-body part relative to the two-body part.

The angular term is similar to the Fourier series expansion previously introduced in Ref. 1,

$$G^{\cos}_n = \exp\left(-\frac{(\zeta n)^2}{2}\right)(\cos(n\theta_{KIJ}) - \cos(n(\theta_{KIJ} + \pi))), \quad (8)$$

$$G^{\sin}_n = \exp\left(-\frac{(\zeta n)^2}{2}\right)(\sin(n\theta_{KIJ}) - \sin(n(\theta_{KIJ} + \pi))), \quad (9)$$

where ζ is a hyperparameter describing the width of the angular Gaussian function and $n > 0$ is the expansion order. With a sufficiently large value of η_3, the angular spectrum can in many cases be almost completely recovered with only the first Fourier terms.[1] This is in part due to the fact that there is only room for a limited number of atoms in the local environment at a certain distance, and the angular spectra are, therefore, rarely very crowded for short distances. In the rest of this work, only the two $n = 1$ cosine and sine terms are used, i.e., $G^{\text{3-body}}_{\text{Angular}} \in \{G^{\cos}_1, G^{\sin}_1\}$.

Since the number of three-body functions scales as $\mathcal{O}(N^2)$ with the number of possible different elements in the chemical compounds, they comprise a much larger part of the representation than the two-body part. A graphical representation of the three-body terms for the atomic environments in a water molecule is displayed in Fig. 2.

B. Machine learning

In Subsections II B 1–II B 5, we discuss four kernel-based regressors that are also used in this study. First, the kernel ridge regression (KRR) method to learn the energy of chemical compounds is discussed. Next, three different regressors to learn forces and energies of chemical compounds are reviewed, namely, "operator quantum machine learning" (OQML),[22] Gaussian process regression (GPR),[41,42] and finally "gradient-domain machine learning" (GDML).[28,29]

In this section, lower-case indices denote the index of a chemical compound, while upper-case indices denotes the index of the atomic centers in the chemical compound, and finally, asterisks are used to denote relation to a query compound or query atomic center.

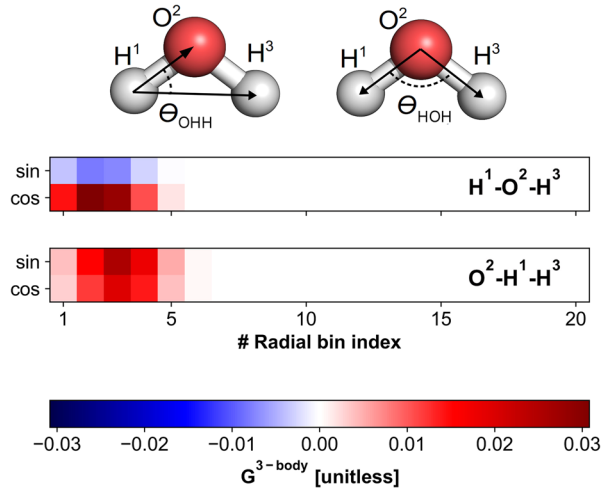

FIG. 2. The three-body basis functions are plotted for the two unique three-body terms in the water molecule, corresponding to the O^2-H^1-H^3 and H^1-O^2-H^3 angles displayed at the top. The atoms are numbered for clarity.

1. Kernel ridge regression (KRR)

It is well-established that KRR—despite its simplicity—is one of the most powerful methods to learn energies of chemical compounds.[1,24,25,33–36,43] In KRR, the energy, U^*, of a query compound, c, can be decomposed into the sum of atomic energies. These are calculated in a basis of kernel functions placed on the atoms of the chemical compounds in the training set. That is,

$$U_c^* = \sum_{I\epsilon c} U_{\text{local}}^*\left(\mathbf{q}_I^*\right) = \sum_{I\epsilon c}\sum_j\sum_{J\epsilon j} \mathcal{K}\left(\mathbf{q}_J, \mathbf{q}_I^*\right)\alpha_j, \tag{10}$$

where I and J are atoms in the query and training compounds c and j, respectively. $\mathbf{q}_J$, $\mathbf{q}_I^*$ are their representation and α_j is the jth regression coefficient. This can be written in matrix notation,

$$\mathbf{U} = \mathbf{K}^{\text{KRR}}\boldsymbol{\alpha}^{\text{KRR}}, \tag{11}$$

where the elements of the KRR kernel matrix are given by the sums over the pair-wise kernels between the atoms in two compounds,

$$\mathbf{K}_{ij}^{\text{KRR}} = \sum_{I\epsilon i}\sum_{J\epsilon j} \mathcal{K}\left(\mathbf{q}_J, \mathbf{q}_I^*\right) \tag{12}$$

and $\boldsymbol{\alpha}^{\text{KRR}}$ is the regression coefficient vector.

These regression coefficients can be obtained by fitting Eq. (11) to the energies of a training set in the basis of the same set of compounds. In KRR, this is done by minimizing the cost function

$$J\left(\boldsymbol{\alpha}^{\text{KRR}}\right) = \tfrac{1}{2}\left\|\mathbf{K}^{\text{KRR}}\boldsymbol{\alpha}^{\text{KRR}} - \mathbf{U}\right\|_2^2 + \tfrac{\lambda}{2}\left(\boldsymbol{\alpha}^{\text{KRR}}\right)^T\mathbf{K}^{\text{KRR}}\boldsymbol{\alpha}^{\text{KRR}}, \tag{13}$$

which has the following closed-form solution:

$$\boldsymbol{\alpha}^{\text{KRR}} = \left(\mathbf{K}^{\text{KRR}} + \mathbf{I}\lambda\right)^{-1}\mathbf{U}. \tag{14}$$

λ is a typically small number which is added to the diagonal of the kernel matrix in order to regularize and ensure numerical stability when the kernel is inverted.[44]

We have previously shown how KRR with FCHL18 yields systematically improving property predictions that reach state-of-the art accuracy for many system classes including molecules and materials.[1]

2. Operator quantum machine learning (OQML)

It is advantageous to also include forces in the training step if available as these both improve energy and force prediction. In the operator quantum machine learning (OQML) approach introduced in Ref. 22, the model is trained on the energy and forces simultaneously.

The kernel is expanded in a basis of kernel functions placed on the atomic environments of each atom in the training set. Effectively, this extends the number of regression coefficients to the total number of atoms in the training set rather than the number of chemical compounds as for KRR. In the following, we refer to this non-square kernel as $\mathbf{K}^{\text{OQML}}$.

In addition to the energies, $\mathbf{U}$, it is possible to include the forces, $\mathbf{F}$, in the training step by applying the force operator to the kernel and solving the regression coefficients for both the energy and forces simultaneously. The equation that is solved during the training step is

$$\begin{bmatrix}\mathbf{U}\\ \mathbf{F}\end{bmatrix} = \begin{bmatrix}\mathbf{K}^{\text{OQML}}\\ -\frac{\partial}{\partial\vec{r}^*}\mathbf{K}^{\text{OQML}}\end{bmatrix}\boldsymbol{\alpha}^{\text{OQML}}, \tag{15}$$

where $\mathbf{K}^{\text{OQML}}$ is the matrix of kernel elements between the atoms in the training set and the training or query molecules and $-\frac{\partial}{\partial\vec{r}^*}$ is the force operator. The presence of an asterisk in the operator denotes that the differentiation is with respect to a coordinate in the training/query compound, while the absence of an asterisk denotes that the differentiation is carried out with respect to the coordinate of an atom/molecule used to form the basis set. Thus, the dimension of the OQML kernel is $(3MN + N) \times MN$, where N is the number of molecules in the training set, and M is the average number of atoms in each molecule.

A solution to Eq. (15) can be obtained by minimizing the following cost function:

$$J\left(\boldsymbol{\alpha}^{\text{OQML}}\right) = \left\|\begin{bmatrix}\mathbf{U}\\ \mathbf{F}\end{bmatrix} - \begin{bmatrix}\mathbf{K}^{\text{OQML}}\\ -\frac{\partial}{\partial\vec{r}^*}\mathbf{K}^{\text{OQML}}\end{bmatrix}\boldsymbol{\alpha}^{\text{OQML}}\right\|_2^2 \tag{16}$$

with respect to $\boldsymbol{\alpha}^{\text{OQML}}$. This least-squares approach leads to a solution that looks similar to the normal equation. However, we found that this approach involves the product of kernel matrices that are ill-conditioned and, therefore, suffers from numerical instability, leading to large training and test errors.

A more numerically stable approach involves solving Eq. (15) directly, using a singular-value decomposition (SVD), which is used for OQML in this work. In similar spirit to the regularization used in KRR, the smallest singular values (below a certain threshold) can be ignored in the solution. This threshold, ϵ_{min}, can be treated as a hyperparameter in the model.

The elements of $\mathbf{K}^{\text{OQML}}$ are given by

$$\mathbf{K}_{ij}^{\text{OQML}} = \sum_{I\epsilon i} \mathcal{K}\left(\mathbf{q}_J, \mathbf{q}_I^*\right), \tag{17}$$

where J is an atom in the training set and I is an atom in molecule i. In contrast to the kernel matrix in KRR, $\mathbf{K}^{\text{OQML}}$ is non-square and has a column for each of the atoms in the training set and a row corresponding to each of the molecules in the training or query set.

The kernel matrix elements that correspond to the atomic forces are calculated by taking the negative derivative of the matrix elements in Eq. (17) with respect to the coordinates of the query molecules, that is,

$$-\frac{\partial}{\partial\vec{r}_K^*}\mathbf{K}_{ij}^{\text{OQML}} = -\sum_{I\epsilon i}\frac{\partial\mathcal{K}\left(\mathbf{q}_J, \mathbf{q}_I^*\right)}{\partial\vec{r}_K^*}, \tag{18}$$

where $\vec{r}_K^*$ denotes the Kth coordinate of the query molecule. The resulting derivative kernel, thus, has a column for each of the atoms in the training set and a row corresponding to each of the gradient components in the training or query set.

Energies are predicted from the set of $\boldsymbol{\alpha}$-coefficients,

$$\mathbf{U} = \mathbf{K}^{\text{OQML}}\boldsymbol{\alpha}^{\text{OQML}}. \tag{19}$$

The force prediction is simply the derivative of the above equation with the same set of $\boldsymbol{\alpha}$-coefficients,

$$\mathbf{F} = -\frac{\partial}{\partial\vec{r}^*}\mathbf{K}^{\text{OQML}}\boldsymbol{\alpha}^{\text{OQML}}. \tag{20}$$

See Appendix B for the derivation of all kernel derivatives mentioned in this section.

In contrast to methods that learn forces directly as a vectorial quantity,[15,45] the use of the force operator guarantees that the machine learned potential will describe a conservative force field. This property is crucial for applications in molecular dynamics where energy conservation is necessary to obtain correct sampling without heavy use of thermostats.

3. Gaussian process regression including derivatives

It is also possible to define models that incorporate derivatives in the training set within the framework of Gaussian process regression.[41] The relevant equations for training a model on energies and forces for chemical compounds are presented below. For their derivation, we refer the reader to the work of Mathias[42] and the work of Bartók and Csányi.[2] The Gaussian process regression kernel matrix which simultaneously incorporates the energy, **U**, and the forces, **F**, is written as

$$\begin{bmatrix} \mathbf{U} \\ \mathbf{F} \end{bmatrix} = \begin{bmatrix} \mathbf{K}^{\mathrm{KRR}} & -\frac{\partial}{\partial \vec{r}}\mathbf{K}^{\mathrm{KRR}} \\ -\frac{\partial}{\partial \vec{r}^{*}}\mathbf{K}^{\mathrm{KRR}} & \frac{\partial^2}{\partial \vec{r}\partial \vec{r}^{*}}\mathbf{K}^{\mathrm{KRR}} \end{bmatrix} \boldsymbol{\alpha}^{\mathrm{GPR}}, \tag{21}$$

where $\mathbf{K}^{\mathrm{KRR}}$ is the same kernel matrix as used in KRR, as described previously. In the following, we abbreviate the above methodology of Gaussian process regression derivatives as "GPR."

The first of the two off-diagonal blocks contain only one derivative given by

$$-\frac{\partial}{\partial \vec{r}_K^{*}}\mathbf{K}_{ij}^{\mathrm{KRR}} = -\sum_{I\in i}\sum_{J\in j}\frac{\partial \mathcal{K}\left(\mathbf{q}_J, \mathbf{q}_I^{*}\right)}{\partial \vec{r}_K^{*}}, \tag{22}$$

where $\vec{r}_K^{*}$ denotes the Kth coordinate of the query compound. The other block is given analogously. The last block which comprises the largest part of the GPR kernel matrix is the double derivative given by

$$\frac{\partial^2}{\partial \vec{r}_L \partial \vec{r}_K^{*}}\mathbf{K}_{ij}^{\mathrm{KRR}} = \sum_{I\in i}\sum_{J\in j}\frac{\partial \mathcal{K}\left(\mathbf{q}_J, \mathbf{q}_I^{*}\right)}{\partial \vec{r}_L \partial \vec{r}_K^{*}} \tag{23}$$

where $\vec{r}_L$ and $\vec{r}_K^{*}$ denote the Lth and Kth coordinate of the basis and query compounds, respectively.

Thus, the dimension of the GPR kernel is $(3MN+N)\times(3MN+N)$, where N is the number of molecules in the training set and M is the average number of atoms in each molecule. The rows of the full GPR kernel matrix thus run over the same indices as the OQML kernel matrix. However, where the indices of the columns of the OQML kernel matrix run over the atoms in the training set, the indices of the columns GPR kernel run first over the molecules in the training set and second over the gradient components in each molecule. Thus, the main difference is the choice of basis in which the regression problem is expanded.

The regression coefficients $\boldsymbol{\alpha}^{\mathrm{GPR}}$ can be obtained by minimizing a cost function similar to that in Eq. (13), only with the difference that the matrix $\mathbf{K}^{\mathrm{KRR}}$ is replaced by the GPR kernel matrix in Eq. (21), and **U** is replaced by the vector that contains both the energies and atomic forces.

Compared to OQML, the GPR kernel matrix contains derivative terms of up to second order, whereas OQML only contains terms up to first order.

The second-order part of the GPR kernel matrix is the computationally heaviest term, and the time to compute it scales as $\mathcal{O}\left(36N^2M^4\right)$, where N is the number of molecules in the training set and M is the average number of atoms in each molecule.[22] In comparison, the heaviest term of the OQML kernel is only of first order and scales as $\mathcal{O}\left(6N^2M^3\right)$.[22] Both methods scale as $\mathcal{O}\left(kN^2\right)$ with the number of training molecules, but GPR has a higher prefactor and scales much less favorably with the number of atoms in the individual molecules (quartic rather than cubic).

We also note the existence of sparsification procedures which can be applied to the GPR model, such as those used within the Gaussian approximation potential methods.[2] The use of sparsification makes it possible to treat the problem without any second derivatives in the kernel matrix. The result here is then a kernel matrix that is very similar to the OQML kernel and requires far less time to compute compared to the full GPR kernel.

As the molecules used to benchmark force prediction methods in this study contain between 9 and 21 atoms, it is expected that the time to calculate the kernel for GPR is on the order of 50–200 times slower than OQML. This is demonstrated numerically in Sec. III C.

Additionally, we note that evaluation of the second-derivative matrix elements in GPR scales as $\mathcal{O}\left(N^2\right)$ with regard to the size of the representation, while evaluation of first derivative matrix elements (such as in OQML and the off-diagonal blocks in GPR) only scales as $\mathcal{O}(N)$ [see, for example, Eqs. (B5) and (B7) in Appendix B].

In terms of memory usage, the GPR kernel is roughly three times larger than the OQML kernel since it contains a column for each molecule and gradient component in the training set, whereas the OQML kernel only contains a column for each atom in the training set. As a result, GPR scales computationally less favorably compared to OQML, although the accuracy of the regression may be slightly increased due to more regression coefficients being fitted.

4. Gradient-domain machine learning (GDML)

Since we will be comparing numerical results from the GDML[28] and the closely related sGDML[29] methods, we also briefly review these approaches for the sake of completeness. GDML can be seen as equivalent to the GPR approach detailed above, with the difference that the energy is left out of the training data, such that only forces are used in the training. In turn, the corresponding 0th and 1st derivative kernel blocks from the GPR kernel are not present in the GDML kernel. Thus, the kernel in the GDML approach is identical to the block in the GPR kernel which contains the second-order derivative. Effectively, the equation solved in the GDML approach is

$$-\mathbf{F} = \frac{\partial^2}{\partial \vec{r}\partial \vec{r}^{*}}\mathbf{K}^{\mathrm{KRR}}\boldsymbol{\alpha}^{\mathrm{GDML}}. \tag{24}$$

The GDML regression coefficients can be obtained, similar to those in GPR and KRR, by minimizing a cost function similar to that in Eq. (13) but only including the forces and second-derivative kernel matrix in the above equations.

Force predictions are then calculated using Eq. (24), while energy predictions in the GDML approach are done using a 1st derivative kernel,

$$\mathbf{U} = \frac{\partial}{\partial \vec{r}} \mathbf{K}^{\mathrm{KRR}} \boldsymbol{\alpha}^{\mathrm{GDML}}. \tag{25}$$

Note that this derivative is taken with respect to the basis and not the query molecule. Since the energy is not used in the training step, all predicted energies are only defined up an integration constant which can be inferred from the mean deviation between predicted energies for a training or validation set.

Compared to GPR, the computational cost of GDML is ever so slightly reduced as the three smaller blocks are being ignored in the kernel. However, the corresponding gain in computational cost is negligible. Leaving out the energy in the training set makes it difficult to regress any energy offset if a model is trained across chemical composition and molecular size. For GDML models that are only trained on one molecule, however, this seems to have very little effect.[28]

In the formulation of GDML and sGDML by Chmiela *et al.*,[28] one further performance enhancement is made, compared to the equations mentioned for KRR and GPR herein. Instead of using a representation for each atomic environment, GDML and sGDML both use one "global" representation for the entire molecule. In GDML, the inverse interatomic distance matrix is used, while sGDML is a variant of GDML which takes atoms with symmetry into account. Other notable global representations are the Coulomb matrix,[24] BoB,[25] SLATM,[36] the Fourier series of atomic radial distribution functions,[27] and the constant size descriptor of Collins *et al.*[26]

The use of such global representations reduces the double sum over atomic contributions in Eq. (23) to the evaluation of just the single "global" kernel derivative. The result is a massive reduction in the evaluation speed on the order of M^2, where M is the number of atoms in each molecule.

While this speedup is desirable, we note that based on our observation in Sec. III and our results for force and energy predictions, such global representations generally display lower predictive accuracy, especially for condensed-phase systems. There is some evidence that global representations are not ideal for learning size-extensive properties, such as energies, since the resulting kernel elements do not scale with the size.[38] This is naturally taken into account by kernels that sum over atomic contributions. For example, with a local representation, a system with two of the same molecule infinitely apart will naturally have twice the energy of a system containing the same molecule only once, whereas the same will not necessarily be true for a global kernel.

Future development of global representations could potentially be fruitful due to their computational efficiency.

We note that the speedup obtained from using global representations is similar to the difference in scaling cost between OQML and GPR, and the difference between our combined FCHL19/OQML model and a GDML-type model based on a global representation will likely be less than a factor M.

5. Kernel function

We introduce a variant of the Gaussian kernel function, augmented with an elemental screening function that only compares representations for atomic environments of atoms of the same element type,

$$\mathcal{K}(\mathbf{q}_I, \mathbf{q}_J^*) = \delta_{Z_I Z_J^*} \exp\left(-\frac{\|\mathbf{q}_I - \mathbf{q}_J^*\|_2^2}{2\sigma^2}\right), \tag{26}$$

where δ is the Kronecker delta function and the subscripts Z_I and Z_J^* are the nuclear charges of atoms I and J^*, respectively. The $\delta_{Z_I Z_J^*}$ term ensures that only relevant pairs of atoms are compared. For example, it is likely to be of little relevance to compare the atomic environment of a carbon atom to that of a hydrogen atom. Furthermore, the Kronecker delta function reduces the cost of a kernel evaluation since the calculation of many expensive combinations of kernels and their derivatives can be skipped. If needed, the Kronecker delta function could still be changed to a function that incorporates learning across alchemical space to increase the learning rate, as shown in our previous work.[1]

In principle, any suitable kernel function could be used besides a Gaussian function, and the choice could be treated as a hyperparameter of the model. For simplicity, however, only the Gaussian kernel is used in this work.

III. RESULTS

A. Energy learning

In this section, the FCHL19 representation is used with the "universal" set of hyper-parameters fitted to energies of non-equilibrium structures (see Sec. IV B). As the geometries in the QM9 and QM7b datasets used in this section are minimized with respect to energy, we expect a slight decrease in the predictive accuracy of FCHL19 compared to if the hyper-parameters had been optimized on similarly minimized structures. We compare KRR models with FCHL19 to similar KRR models with FCHL18 and a number of other models from the literature. We note that the model and data selection methodology used to obtain the various results found in the literature might differ from the 5-fold cross-validation methodology we used in this study. However, we assume that such differences only give rise to negligible differences in the predictive accuracy of the models.

1. Results for QM9

In Fig. 3, we compare the predictive accuracy of a number of kernel-based models for the atomization energy of molecules in the QM9 dataset.[46] We compare FCHL19 to five other well-performing representations: the SOAP multi-kernel model;[34,35] SchNet[17] and PhysNet,[21] which are the two best performing neural networks for this dataset; SLATM and aSLATM,[36] where the former uses one global representation for the entire molecule and the latter uses an atomic decomposition of the kernel; and finally the previous FCHL18[1] representation.

For the QM9 dataset, we find models based on FCHL19 to be among the models with the lowest out-of-sample mean absolute error (MAE) atomization energy predictions. Compared to the best performing model, FCHL18, the MAE at 20 000 training samples is 0.30 kcal/mol and 0.47 kcal/mol for FCHL18 and FCHL19, respectively. For the largest training split (75 000 training samples), the MAE for the FCHL19 model is 0.25 kcal/mol. Overall, we find that our previous FCHL18 model has the lowest prediction MAE, while SOAP, FCHL19, and aSLATM have virtually indistinguishable

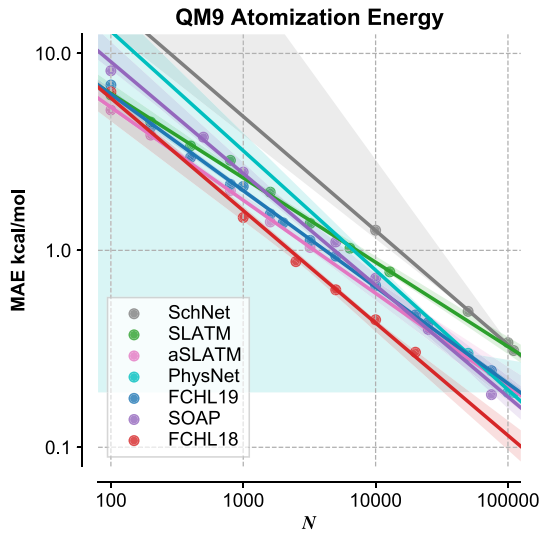

FIG. 3. Learning curves for the QM9 dataset: The mean absolute error (MAE) of atomization energy prediction is plotted for 5 KRR models based on different representations and one neural network vs the training set size (see text). Linear fits are displayed for clarity, and shaded areas denote the 95% confidence intervals for the fits as obtained via boot-strapping.[47]

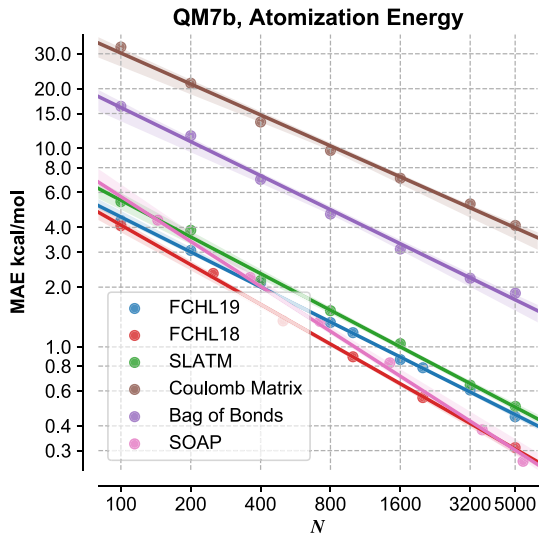

FIG. 4. Learning curves for QM7b: The mean absolute error (MAE) is plotted for KRR models with 6 different representations (see text) vs the training set size. Linear fits are displayed for clarity, and shaded areas denote the 95% confidence intervals for the fits as obtained via boot-strapping.[47]

MAE. Finally, the Global SLATM model and SchNet perform a bit worse for QM9.

We reiterate again at this point that the hyperparameters of FCHL19, in contrast to some other models, have not been optimized on the QM9 dataset.

2. Results for QM7b

Similarly, Fig. 4 compares the predictive accuracy of a number of kernel-based models for the atomization energy of the QM7b dataset.[48] We compare our model to the following representations: FCHL18,[1] SLATM,[36] the Coulomb matrix,[24] Bags-of-Bonds (BoB),[25] and finally SOAP.[34,35] Data for these models are obtained from Ref. 1.

As expected, the dataset is too small for the Coulomb matrix and BoB to reach chemical accuracy. In contrast, all other models (FCHL19, FCHL18, SLATM, and SOAP) reach chemical accuracy when trained on between 800 and 1600 samples. Additionally, the fitted learning curves (Fig. 4) display similar predictive accuracies. For example, all these models are within an MAE of ±0.3 kcal/mol of FCHL19 at 1000 training samples.

3. Results for QM7b-T and GDB13-T

While the QM7b and QM9 datasets contain energies for the equilibrium geometry of small molecules, the QM7b-T[49] and GDB13-T[49] datasets contain non-equilibrium geometries of molecules from QM7b[48] and GDB-13.[50] In addition to gauging the accuracy of a model on unseen samples from the same dataset by training and predicting on subsets of QM7b-T, we also benchmark how well a model can extrapolate to prediction on samples from a dataset containing larger molecules by predicting on GDB13-T with models trained on QM7b-T. Learning curves for these tests can be seen in Fig. 5.

First, we compare FCHL19 to FCHL18 and the Molecular-Orbital-Based (MOB) machine learning method[49,51] by training on the QM7b-T dataset and predicting on unseen samples from the same dataset. FCHL19 and FCHL18 both reach 1 kcal/mol accuracy for this dataset at between 400 and 800 training samples, and above 1000, the difference is less that 0.1 kcal/mol, with FCHL18 being consistently slightly more accurate. The MOB method, which requires a Hartree–Fock calculation for every query to calculate the localized molecular orbitals used to generate the representation, reaches 1 kcal/mol at about 200 training samples and is consistently more accurate with about a 2–3 times improvement in accuracy.

Second, we test the extrapolative power of the three models by training models on the QM7b-T dataset and predicting on the GDB13-T dataset. In this test, the differences observed previously seem to be magnified. Neither the FCHL19 nor the FCHL18 models reach chemical accuracy for the GDB13-T dataset with the amount of training data available in the QM7b-T dataset. At 1000 training samples, the MAE for the two models is 2.7 kcal/mol and 2.2 kcal/mol, respectively. In comparison, MOB reaches this error at around 100–200 samples; however, a larger MOB training set is unavailable due to the difficulty of training large models for MOB.[49,51]

4. Results for Water40

The Water40 dataset consists of 10 000 MD snapshots of a water cluster with 40 water molecules for which a density functional theory (DFT) single-point energy has been calculated.[22] As such, this dataset probes the performance of ML models on chemical systems that approach the condensed-phase behavior. Here, we compare

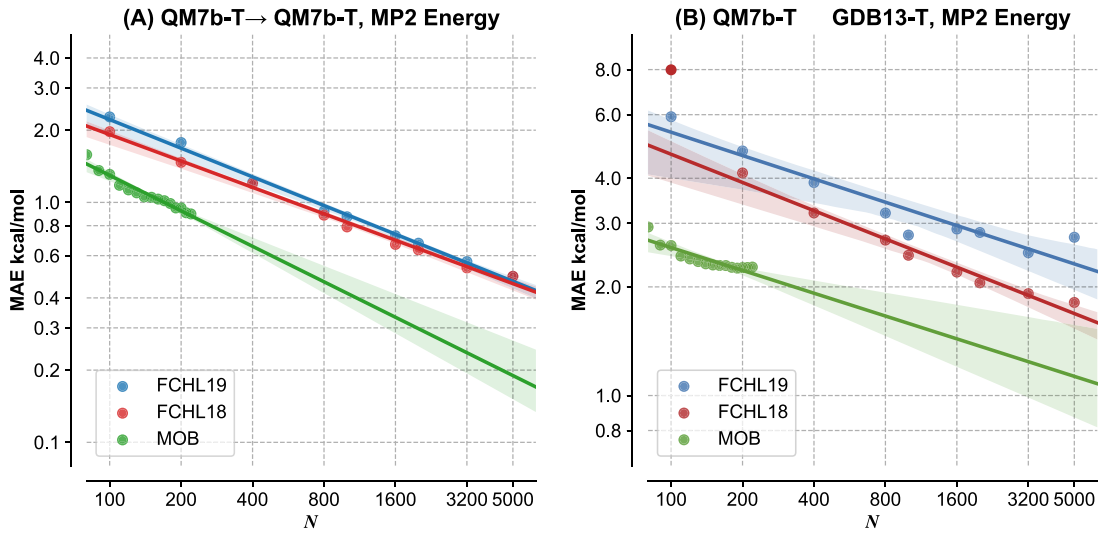

FIG. 5. The two figures display the out-of-training error for models trained on subsets of the QM7b-T dataset. In (a), the models predict the MP2 correlation energy of unseen samples from the same QM7b-T dataset, while in (b), the same models predict the energy on unseen samples from a subset of GDB13-T dataset. Linear fits are displayed for clarity, and shaded areas denote the 95% confidence intervals for the fits as obtained via boot-strapping.[47] For FCHL18, the fit and boot strapping are performed without including the first data point as the fit would otherwise appear unreasonably steep.

our model to the following representations: FCHL18,[1] SLATM and aSLATM,[36] the Coulomb matrix,[24] and BoB.[25] The learning curves for these models on the Water40 dataset are displayed in Fig. 6.

We find that the accuracy of machine learning models based on the FCHL19 representation is far greater compared to any other representation. For example, for models trained on 1000 training instances, the FCHL18-based model yields an MAE of 0.22 kcal/mol/molecule, while the model trained using the FCHL19 representation yields an MAE test error of 0.12 kcal/mol/molecule. The FCHL19 representation reduces the data required to reach a given accuracy by roughly 5 times compared to FCHL18 and by roughly 10 times compared to aSLATM.

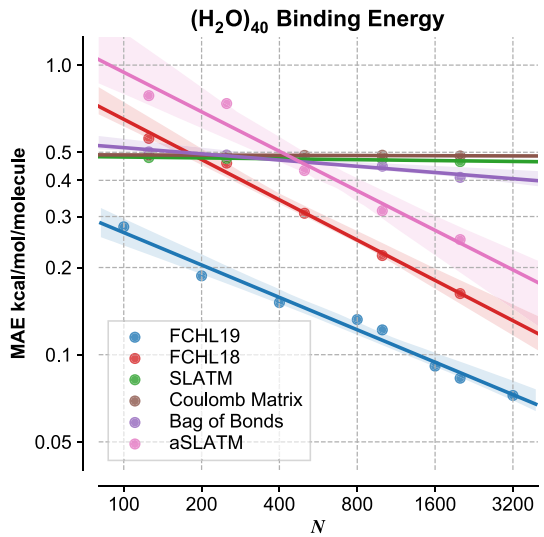

FIG. 6. Learning curves for the Water40 dataset: The mean absolute error (MAE) binding energy per molecule is plotted for 6 different representations vs the training set size. Linear fits are displayed for clarity, and shaded areas denote the 95% confidence intervals for the fits as obtained via boot-strapping.[47]

Note that for Water40, and in contrast to molecular datasets, the use of global representations (i.e., those that do not use a decomposition of the kernel in local, atomic contributions), such as the Coulomb matrix, BoB, and SLATM, results in models which hardly display any learning at all, with a constant error of about 0.5 kcal/mol/molecule, regardless of training set size.

Although FCHL19 is parameterized for the atomization energy of small molecules, it, nevertheless, yields superior accuracy for the binding energy of water clusters where accurate handling of non-covalent interactions is key to determining the energy. This suggests that the parameters in the representation have a high degree of transferability and do not necessarily need to be re-parameterized for every new dataset.

While the accuracy of models based on FCHL19 is better than that of models based on other representations, we expect that models based on, for example, FCHL18 and aSLATM are likely to reach a similar accuracy if the model parameters of those representations are obtained similar to those of FCHL19.

B. Force learning

In Sec. III B 1, the FCHL19 representation is used with parameters that are optimized for both force and energy prediction simultaneously (see Sec. IV B).

1. Results for MD17

Figure 7 reports the MAE force and energy prediction as a function of the number of training samples taken from 7 molecules from

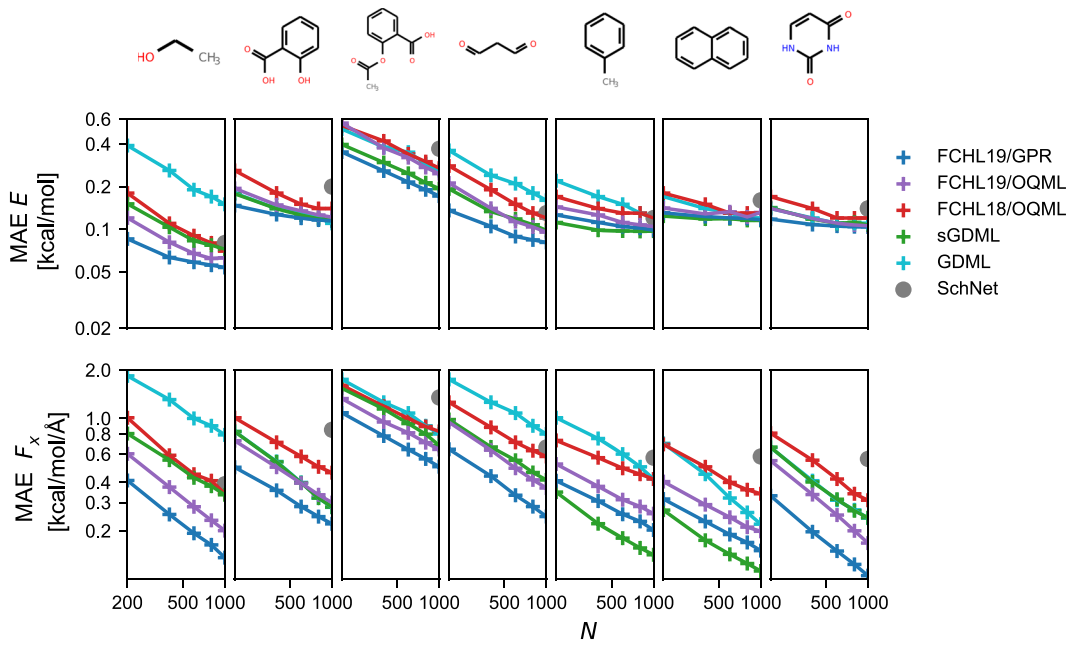

FIG. 7. Here, we present learning curves for force and energy learning for seven molecules from the MD17 dataset. Learning curves are presented for 6 different QML models (see text). The top row contains learning curves for the out-of-sample MAE energy prediction (MAE E), and the bottom row contains corresponding learning curves for out-of-sample MAE force component prediction (MAE F_x), for the molecules (from left to right) ethanol, salicylic acid, aspirin, malonaldehyde, toluene, naphthalene, and uracil.

the MD17 dataset.[28] We note that the original MD17 dataset also includes a dataset for benzene. However, due to low accuracy in the reported energies for this dataset, we have chosen to exclude this from our tests as the high noise level would be the dominating error and as such would not reflect differences in the machine learning procedures. We compare OQML and GPR models based on FCHL19 to OQML models based on FCHL18.[1,22] In addition, we compare to GDML[28] and sGDML[29] which are two state-of-the-art kernel-based methods closely related to GPR. Furthermore, we compare to one of the best performing neural networks for forces, SchNet,[17] which is based on a continuous-filter convolutional neural network.

In general, we note that the reparameterized FCHL19 representation leads to models that have improved accuracy compared to the FCHL18 prediction errors reported in our previous paper.[22] Learning curves for these models are presented in Fig. 7.

For all molecules in the MD17 dataset, the FCHL19 representation with both the GPR and OQML regressors display faster learning compared to FCHL18 with the OQML regressor for both energy and force learning. As a general trend, FCHL19/OQML requires about half the samples to reach a given accuracy compared to FCHL18/OQML. Changing to the GPR regressor, FCHL19/GPR in turn requires about half the samples to reach the same accuracy as FCHL19/OQML. For example, for ethanol, an MAE force error of 0.4 kcal/mol/Å error is obtained at roughly 200, 400, and 800 samples for FCHL19/GPR, FCHL19/OQML, and FCHL18/OQML, respectively.

Similar trends are observed for both force and energy learning for salicylic acid, aspirin, malonaldehyde, and uracil. For these molecules, we find that FCHL19/GPR has the highest accuracy in all cases, with the sGDML method and FCHL19 with the OQML regressor also performing very well and at much reduced computational costs.

We note that GDML and FCHL18/OQML overall have the lowest accuracy of the kernel methods, and SchNet is slightly worse on average, although the time-to-train for SchNet reportedly is much more favorable for larger training sizes.[18] For toluene, naphthalene, and uracil, we note very slow energy learning for all the presented methods, with almost flat learning curves at around 0.1 kcal/mol error. However, this seems to be an inherent property of the dataset and likely to be related to noise in the calculated DFT energies, for example, from use of unconverged integration grids.[52,53]

Furthermore, for the molecules toluene and naphthalene, we observe that sGDML performs very well for force learning, compared to the FCHL19 variants. We speculate that the comparably poor performance of variants of FCHL is due to the high degree of symmetry in the 6-membered rings of the molecules; when the molecule has many atoms of the same element type at very close radial distances, the Fourier transform of the angular histogram in the three-body term becomes very crowded, and this might lead to slower learning for molecules containing such moieties of high symmetry. This might be improved upon by reoptimizing the hyperparameters specifically for this system. Nevertheless, FCHL19 with both the OQML and GPR regressors is within 0.1 kcal/mol energy

error and 0.1 kcal/mol/Å force component error of the best performing method (sGDML) at 1000 training samples in both of these cases.

C. Timings

In this section, we report timings for generating the training kernel which is the most costly step for these kernel models. For force predictions, we additionally report the prediction time per atom of FCHL19-based models trained with the OQML regressor. All timings in this section were carried out on a 24-core compute node equipped with two Intel Xeon E5-2680v3 @ 2.50 GHz central processing units (CPUs) and 128 GB RAM.

1. Timings for energy learning

Using the implementations in the QML software package,[54] we compare timings for calculating kernels for three representations that all use a decomposition of the kernel into atomic contributions, namely, FCHL19, FCHL18, and aSLATM. For FCHL18 and aSLATM, all parameters are set to the default values in QML, and for FCHL19, the values in Appendix A are used. These timings are given in Table I. In all cases, the training times scale as $\mathcal{O}(N^2)$ with the training set size, while the prediction time scales as $\mathcal{O}(N)$.

To illustrate the effects of elemental complexity, we compare timings for both QM7 and QM7b. The two datasets contain molecules with up to 7 non-hydrogen atoms, with the largest molecule being 23 atoms total in both sets, and both datasets contain about 7 K molecules. They differ, however, in the elements that are present in the two datasets: QM7 contains HCNOS, while QM7b additionally contains Cl. As the size of the three-body terms in aSLATM and FCHL19 representations scale cubically and quadratically, respectively, with the number of elements in the dataset, the result will be a substantial increase in kernel evaluation time for models based on these representations.

For aSLATM, the two datasets take 4 s, 955 s and 7 s, 727 s to compute, respectively, whereas for FCHL19, the same numbers are 216 s and 310 s. In contrast, FCHL18 is largely unaffected by chemical complexity, with kernel evaluation times of 3 s and 164 s, and 3 s and 286 s for the two sets, respectively.

Additionally, we present timings for the QM9 dataset. These timings are also presented in Table I. This dataset contains 133 855 molecules with the elements HCNOF and molecules with up to 9 non-hydrogen atoms, where the largest molecule contains 29 atoms. Using the previous implementations of aSLATM and FCHL18, calculating the kernel matrix for this dataset can only be done on a reasonable time scale on a cluster with several nodes. For aSLATM and FCHL18, the time to calculate the QM9 kernel is 728 h and 548 h on our 24-core node, respectively. In contrast, for FCHL19, the time to calculate the kernel is 27 h on the same node. The speedup compared to aSLATM comes from the reduced size of the representation and the element-wise kernel function which is not normally used with aSLATM.[36]

Note that these timings only cover calculating the training kernel, and not the representation generation or regression solver. Generating the representations scales as $\mathcal{O}(N)$ with the number of training or prediction samples and is insignificant in comparison. While solvers to obtain the regression coefficients typically scale as $\mathcal{O}(N^3)$ with the number of training samples, this step is in practice insignificant compared to generating the kernel, even for the largest kernels due to a lower prefactor. For example, the `QML` software package uses a Cholesky decomposition as implemented in libraries such as Intel Math Kernel Library (MKL), and using this implementation for the largest kernel studied in this section (QM9), this step takes less than 1 h, whereas the time to generate the kernel takes between 27 h and 728 h.

2. Timings for force learning

Next, we report timings for kernel evaluations for calculating the training kernel for force and energies for a set of 1 K molecules taken from the MD17 dataset. These timings are given in Table II. Again, in all cases, the training times scale as $\mathcal{O}(N^2)$ with the training set size, while the prediction time scales as $\mathcal{O}(N)$. Compared to the FCHL18 representation, the speedup using the same regressor (OQML) is as low as 5 times for the smallest molecules, ethanol and malonaldehyde, and up to almost 20 times for the largest molecule, aspirin. For models based on FCHL19 with the OQML regressor, the training times vary between around 51 s for malonaldehyde and 527 s for aspirin. These numbers correspond to force prediction times (also given in Table II, with a graphical overview in Fig. 8) in the range of 5.7–25.3 ms/atom for models trained on 1000 training samples, excluding generation of the representation. The time to generate the representations and Cartesian derivatives of the representation was found to be very negligible in comparison: for one of the two smallest molecules in MD17, namely, ethanol with 9 atoms, the time was found to be 0.27 ms/atom, while for the largest molecule (aspirin with 21 atoms), the time to compute the representation was found to be 1.0 ms/atom.

TABLE I. Timings for kernel evaluation for the QM7b and QM9 datasets, with the three different atomic representations: aSLATM, FCHL18, and FCHL19. To illustrate the effects of molecular size and elemental complexity on the kernel evaluation time, data for the three datasets QM7, QM7b, and QM9 are presented. Timings are presented in seconds (s) or hours (h). Additionally, the size of each dataset and the elements present in the datasets are listed. All calculations are done on a 24-core node equipped with two Intel Xeon E5-2680v3 @ 2.50 GHz CPUs.

Dataset	Molecules	Elements	aSLATM	FCHL18	FCHL19
QM7	7 165	H C N O S	4966 s	3164 s	216 s
QM7b	7 211	H C N O S Cl	7727 s	3286 s	310 s
QM9	133 885	H C N O F	728 h	548 h	27 h

TABLE II. Training times for calculating the kernel matrix for 1000 molecules (forces and energies) of 7 molecules from the MD17 dataset with FCHL18 and FCHL19 with the GPR and OQML are given in seconds. Additionally, the time to calculate the prediction kernel for FCHL19/OQML is given in ms per atom. The numbers in this table are calculated as averages over 5 kernels using different random splits, run on a 24-core node equipped with two Intel Xeon E5-2680v3 @ 2.50 GHz CPUs.

Molecule	Atoms	FCHL18 OQML (s)	FCHL19 GPR (s)	FCHL19 OQML (s)	FCHL19 OQML (ms/atom)
Ethanol	9	387	2 252	66	7.3
Malonaldehyde	9	286	1 926	51	5.7
Naphthalene	18	7 886	11 782	455	25.3
Aspirin	21	10 067	101 451	527	25.1
Salicylic acid	16	3 940	6 836	249	15.6
Toluene	15	2 755	7 976	271	18.1
Uracil	12	N/A	2 576	87	7.3

Models based on FCHL19 with the GPR regressor are found to be substantially slower than OQML models. For the smallest molecules (ethanol, malonaldehyde, and uracil), the GPR kernel can be calculated in less than 1 h (between 1926 s and 2576 s), about 30–38 times slower than the corresponding OQML kernel. For the largest molecule, aspirin, the differences are even larger: the GPR kernel takes 101 s and 451 s, 192 times slower than the corresponding OQML kernel. Based on the observations in this section, a GPR model requires about half the amount of training data to reach the same accuracy as a model based on OQML. With the $\mathcal{O}(N^2)$ scaling of both GPR and OQML, this translates to a 4 times increase in prediction speed, and consequently, OQML models will be about 10–50 times faster than a GPR model if the models are trained to the same accuracy. This underlines how OQML is a favorable alternative to GPR, although the learning curve offsets are somewhat larger.

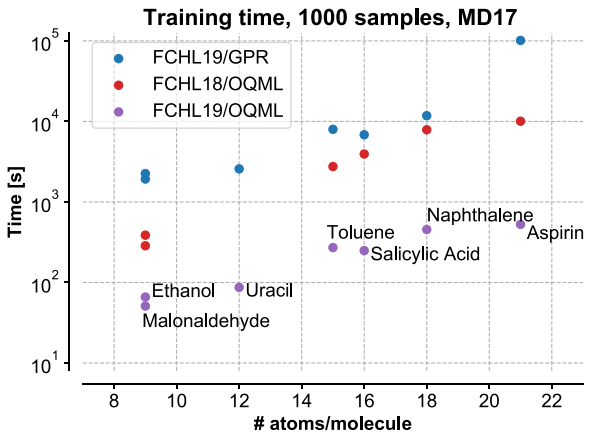

FIG. 8. The time to calculate the training kernel for 1000 training samples with three different methods is displayed for 7 molecules from the MD17 dataset, namely, ethanol, malonaldehyde, uracil, toluene, salicylic acid, naphthalene, and aspirin. Timings are displayed for the methods FCHL19/GPR, FCHL18/OQML, and FCHL19/OQML, and are calculated as averages over 5 kernels using different random splits, run on a 24-core node equipped with two Intel Xeon E5-2680v3 @ 2.50 GHz CPUs.

IV. METHODOLOGY

A. Datasets

This section contains a brief description of the datasets used to benchmark QML models trained with the revised FCHL19 representation.

1. QM7b

The QM7b dataset[48] is based on a subset of the GDB-13 database[50] and consists of 7211 molecules with up to 7 atoms of the elements CNOSCl, saturated with hydrogen atoms. For each molecule, the Perdew-Burke-Ernzerhof (PBE) equilibrium geometry is available along with 13 different properties also calculated at the DFT level.

2. QM9

The QM9[46] dataset is similar to QM7b, and it is only based on a subset of the GDB-17 database.[55] In contrast to QM7b, the QM9 dataset is much larger and contains 133 885 molecules with up to 9 atoms of the type CNOF saturated with hydrogen atoms. For each, the B3LYP equilibrium geometry is available, and the atomization energy is used to generate the learning curves in this study. Similar to previous studies, we leave out the "uncharacterized" subset of 3054 molecules that did not pass a geometry consistency check when the dataset was created.[56]

3. QM7b-T and GDB13-T

The QM7b-T and GDB13-T datasets[49] consist of non-equilibrium geometries sampled from *ab initio* molecular dynamics simulations at 350 K. QM7b-T contains non-equilibrium structures of molecules from QM7b, while GDB13-T contains non-equilibrium structures of a subset of the GDB-13 database[50] where each molecule contains 13 atoms of the type CNOSCl and saturated with hydrogen. For each molecule in the two sets, the MP2 correlation energy is given, i.e., the difference between the MP2 and the HF energy. This set is used to test the extrapolative powers of QML models by training on the QM7b-T dataset and predicting on the GDB13-T dataset which contains larger molecules.

4. Water40

The Water40 dataset[1] consists of 10 000 MD snapshots from a molecular dynamics simulation of a water cluster with 40 water

molecules sampled at 300 K. For each sample, a dispersion-corrected DFT single-point energy is calculated at the PBEh-3c level of theory.[57] In contrast to other datasets used in this study, reliable treatment of non-bonded interactions in the machine learning model is required to learn these energies accurately.

5. MD17

The MD17 dataset[28] contains snapshots from *ab inito* molecular dynamics on a number of small organic molecules for which reference force and energies are calculated at the DFT level. Out of the dataset, we benchmark our models on force and energy data from the molecules ethanol, salicylic acid, aspirin, malonaldehyde, toluene, naphthalene, and uracil.

B. Optimization of representation parameters

The optimal values of the parameters used to generate the FCHL19 representation for a given atomic environment are in principle hyperparameters of the model and must be re-fitted to each individual dataset to ensure optimal learning. However, in our experience, the variances in these parameters are relatively small and show substantial transferability from dataset to dataset. Since the number of parameters is relatively big (nine parameters in total), the amount of work required to ensure optimal learning for a specific dataset can be substantial.

Instead, we propose the use of two sets of "universal" default parameters that are fitted *a priori*. To fit these, we employed a random subset of 576 distorted geometries of small molecules with up to 5 atoms of the type CNO, saturated with hydrogen atoms, for which the forces and energies have been obtained from DFT calculations.[22] This dataset is publicly available (see Ref. 58).

The set was randomly divided into a training set (384 geometries) and a test set (192 geometries). A model was fitted on the training set, and predictions on the test set were used to minimize the following cost function with respect to the model parameters:

$$\mathscr{L} = 0.01 \sum_i \left(U_i - \hat{U}_i\right)^2 + \sum_i \frac{1}{n_i} \|\mathbf{F}_i - \hat{\mathbf{F}}_i\|^2, \tag{27}$$

where U_i is the energy of molecule i in the test set and $\mathbf{F_i}$ and n_i are the forces and number of atoms of the same molecule, respectively. The minimization was performed via Monte Carlo greedy optimization, where real-type parameters are optimized by multiplying by a factor randomly chosen from a normal distribution centered on 1 with the variance 0.05, and integer-type parameters are optimized by randomly adding +1 or −1.

Note that in order to reduce the number of free parameters, the hyperparameters are not fitted as element-specific parameters, but rather the same value of a hyperparameter is used to generate the representation for an atomic environment, regardless of the element type. The width of the Gaussian angular function, ζ, was fixed to π as this has shown to reduce the error from the Fourier expansion to be negligible. The distance cut-off for these "default" values was conservatively set to 8 Å.

In the end, we fit two different sets of model parameters: one for energies + forces and one for energies. For the latter parameter set, the term in Eq. (27) that includes forces was set to zero. The optimal values of all parameters can be found in Appendix A.

C. Hyperparameter selection

For all learning results in Sec. III, the hyperparameters of the model (not including the representation) were optimized using nested 5-fold cross validation (CV). First, the dataset was randomized and split into 5 "outer" folds using the KFold class implemented in scikit-Learn.[59] Second, for each of the five folds, the training set was again randomized and split into 4 "inner" folds. Cross validation was performed on the inner folds to select optimal values for the kernel width and regularization. To select optimal kernel width and regularizer, a grid search was performed for $\sigma \in \{1, 2, 4, 8, 16, 32\}$ and $\lambda \in \{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}\}$. For OQML runs, instead of screening the parameter λ, the value of the lowest accepted singular value (in terms of the largest singular value) was screened in the range $\varepsilon_{\min} \in \{0, 10^{-12}, 10^{-11}, 10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}\}$. For datasets with energy labels, the set of $\{\lambda/\varepsilon_{\min}, \sigma\}$ with the lowest average MAE energy within the inner CV folds was selected to predict energies on the test set from the outer CV folds. Similarly, for datasets with both force and energy labels, the set with the lowest average $\mathscr{L}$ [see Eq. (27)] within the inner CV folds was selected.

D. Learning curves

Learning curves for models based on FCHL19 are presented as the average out-of-sample mean absolute error (MAE) over the five outer CV folds of the datasets. The leading term in this out-of-sample error is predicted to decay as $\frac{a}{N^b}$. To illustrate this effect, all learning curves are displayed on a log-log scale where this decay becomes linear, and all plotted learning curves thus contain a linear fit and the 95% confidence interval for the fit.[60–62] The 95% confidence interval is obtained using boot-strapping as implemented in the Python library Seaborn,[47] which is also used to generate the plots.

E. Timings

All timings were performed on a compute node equipped with two Intel Xeon E5-2680v3 @ 2.50 GHz CPUs (24 CPU cores in total) and 128 GB RAM. The OMP parallel kernel routines in the QML code were compiled with the GNU Fortran compiler version 4.8 and linked to Intel MKL. `QML` was installed using only the default settings, similar to those of a user installing `QML` directly from the Python Package Index (PyPI).

F. Software and software availability

All machine learning calculations were performed using the open source quantum machine learning package `QML`.[54] The code to reproduce the FCHL19 representation and several of the other models used in this paper as well as the relevant kernel and kernel derivative matrices can be found in the GitHub repository for `QML` at https://github.com/qmlcode/qml.

V. CONCLUSION AND OUTLOOK

We have presented a revised representation for chemical compounds which enables machine learning models that have state-of-the-art accuracy and much reduced computational cost in order to easily run on hardware that is accessible to most chemists.

The representation is built on a discretization of the previously published FCHL18 model.[1] Two sets of universal parameters for the representations were fitted to an initial training set and demonstrated to have a high degree of transferability.

Machine learning models trained with the revised FCHL19 representation show state-of-the-art prediction accuracy on several datasets. For models trained on atomization energies, such as QM7b and QM9, the accuracy is better than 0.5 kcal/mol and 0.25 kcal/mol at the largest training sizes, while the training times are reduced by 10–20 times compared to FCHL18. For QM7, it is possible to train a model on 7 K molecules in little over 3 min, while for the full set of 133 885 molecules in QM9, a model can be trained in roughly one day on a single node, compared to three weeks with our previous models. In general, we note that some other kernel-based models also perform very well on these datasets, namely, the SLATM- and SOAP-based models.

For the Water40 data, the revised FCHL19 model reduces the predicted binding energy error to below 0.1 kcal/mol/molecule, even with the representation being optimized solely for small molecules, demonstrating the transferability of the model.

Models trained on the MD17 dataset with the revised FCHL19 representation and the OQML or GPR regressors were found to yield models that reach state-of-the-art accuracy in force prediction while requiring 2–4 times less data compared to FCHL18 with the OQML regressor. The computational cost of these force predictions was found to be on the scale of milliseconds per atom. For energy prediction on the MD17 dataset, the predictive accuracy seems to be limited by noise in the dataset, but models based on FCHL19 were found to have low energy prediction errors, nevertheless.

Our efforts are a substantial step toward both practical and transferable models that will allow the chemist to routinely train models and run molecular dynamics simulations with machine learned potentials throughout chemical space. These developments should be valuable for computational materials and molecular design campaigns, as well as for more interactive and immersive virtual reality simulation environments, which have recently been extended to enable users to manipulate real-time simulations of drug-ligand binding,[63] small molecule quantum chemistry,[64,65] and next generation digital education.[66,67] Future work will also deal with condensed-phase systems.

ACKNOWLEDGMENTS

The National Centre of Competence in Research (NCCR) Materials Revolution: Computational Design and Discovery of Novel Materials (MARVEL) of the Swiss National Science Foundation (SNSF) is acknowledged. This material is based upon work supported by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under Grant No. FA9550-15-1-0026. L.A.B. acknowledges the Alan Turing Institute under EPSRC Grant No. EP/N510129/1 as well as support of this work through EPSRC Grant No. EP/P021123/1. The authors thank David R. Glowacki and the Intangible Realities Laboratory, whose open-source VR-enabled real-time simulation framework Narupa helped inspire us to investigate faster machine learning algorithms. We further acknowledge the use of the following software: NumPy and the F2PY tool,[68] and OpenMP.[69]

TABLE III. Optimized representation parameters for FCHL19 for energy (E), and energy and forces (E + F). n_{Rs_2} and n_{Rs_3} are the number of bins for a pair or triplet of element types in the two- and three-body spectra, respectively. w and η_3 determine the width of the radial two- and three-body distribution functions, respectively. N_2 and N_3 determine the decay of the two- and three-body scaling functions, respectively. c_3 is a weight factor that determines the weight of the three-body part relative to the two-body part. ζ is the width of the Gaussian functions used in the Fourier series and fixed to π. r_{cut} is the distance cut-off, here fixed to 8.0 Å.

Parameter	E	E + F
n_{Rs_2}	22	24
n_{Rs_3}	17	20
w (Å^2)	0.41	0.32
η_3 (Å^{-2})	0.97	2.7
N_2	2.4	1.8
N_3	2.4	0.57
c_3 (Å^{N_3})	45.8	13.4
ζ	π	π
r_{cut} (Å)	8.0	8.0

APPENDIX A: OPTIMIZED REPRESENTATION PARAMETERS

The optimal representation parameters obtained through the Monte Carlo optimization are presented in Table III.

APPENDIX B: KERNEL DERIVATIVES

This section derives the first and second derivates of the kernel with respect to the coordinates. First, we define the signed difference between two representations,

$$\mathbf{d} = \mathbf{q} - \mathbf{q}^*. \tag{B1}$$

The derivative of representation with respect to a specific coordinate, r, typically the x-, y-, or z-coordinate of an atom in the chemical compound, is

$$\frac{\partial \mathbf{q}}{\partial r} = \left[\frac{\partial q_1}{\partial r}\ \frac{\partial q_2}{\partial r}\ \frac{\partial q_3}{\partial r} \cdots \frac{\partial q_n}{\partial r}\right]^{\top}. \tag{B2}$$

Defining a Gaussian kernel,

$$\mathcal{K}(\mathbf{q}, \mathbf{q}^*) = \exp\left(-\frac{\|\mathbf{d}\|_2^2}{2\sigma^2}\right). \tag{B3}$$

Defining a vector, $\mathbf{g}$, as the first derivative of the kernel with respect to q_i^*,

$$g_i \triangleq \frac{\partial}{\partial q_i^*}\mathcal{K}(\mathbf{q}, \mathbf{q}^*) = \frac{d_i}{\sigma^2}\exp\left(-\frac{\|\mathbf{d}\|_2^2}{2\sigma^2}\right). \tag{B4}$$

The kernel derivative with respect to coordinate r is

$$\frac{\partial}{\partial r}\mathcal{K}(\mathbf{q}, \mathbf{q}^*) = \mathbf{g} \cdot \left(\frac{\partial \mathbf{q}^*}{\partial r}\right). \tag{B5}$$

Defining a matrix, $\mathbf{H}$, as the second derivative with respect to q_i and q_j^*,

$$H_{ij} \triangleq \frac{\partial^2}{\partial q_i \partial q_j^*} \mathcal{K}(\mathbf{q}, \mathbf{q}^*) = \left(\delta_{ij} \frac{1}{\sigma^2} - \frac{d_i d_j}{\sigma^4}\right) \exp\left(-\frac{\|\mathbf{d}\|_2^2}{2\sigma^2}\right). \quad \text{(B6)}$$

The kernel derivative with respect to coordinates r_a and r_b is

$$\frac{\partial^2}{\partial r_a \partial r_b^*} \mathcal{K}(\mathbf{q}, \mathbf{q}^*) = \left(\frac{\partial \mathbf{q}}{\partial r_a}\right)^\top \mathbf{H} \left(\frac{\partial \mathbf{q}^*}{\partial r_b^*}\right). \quad \text{(B7)}$$

Analytical implementations of these derivatives with the kernel function defined in Eq. (26) are implemented in our QML code.[54]

REFERENCES

[1] 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).

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

[3] S. Lorenz, A. Gross, and M. Scheffler, "Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks," Chem. Phys. Lett. **395**, 210 (2004).

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

[5] J. Behler, "Perspective: Machine learning potentials for atomistic simulations," J. Chem. Phys. **145**, 170901 (2016).

[6] 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).

[7] J. S. Smith, O. Isayev, and A. E. Roitberg, "ANI-1, a data set of 20 million calculated off-equilibrium conformations for organic molecules," Sci. Data **4**, 170193 (2017).

[8] V. Botu and R. Ramprasad, "Learning scheme to predict atomic forces and accelerate materials simulations," Phys. Rev. B **92**, 094306 (2015).

[9] V. Botu and R. Ramprasad, "Adaptive machine learning framework to accelerate *ab initio* molecular dynamics," Int. J. Quantum Chem. **115**, 1074–1083 (2015).

[10] V. Botu, R. Batra, J. Chapman, and R. Ramprasad, "Machine learning force fields: Construction, validation, and outlook," J. Phys. Chem. C **121**, 511–522 (2016).

[11] T. D. Huan, R. Batra, J. Chapman, S. Krishnan, L. Chen, and R. Ramprasad, "A universal strategy for the creation of machine learning-based atomistic force fields," npj Comput. Mater. **3**, 37 (2017).

[12] Z. Li, J. R. Kermode, and A. De Vita, "Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces," Phys. Rev. Lett. **114**, 096405 (2015).

[13] K. Gubaev, E. V. Podryabinkin, and A. V. Shapeev, "Machine learning of molecular properties: Locality and active learning," J. Chem. Phys. **148**, 241727 (2018).

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

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

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

[17] 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).

[18] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, and K.-R. Müller, "SchNetPack: A deep learning toolbox for atomistic systems," J. Chem. Theory Comput. **15**, 448–455 (2019).

[19] 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).

[20] L. Zhang, J. Han, H. Wang, R. Car, and E.Weinan, "Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics," Phys. Rev. Lett. **120**, 143001 (2018).

[21] O. T. Unke and M. Meuwly, "PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges," J. Chem. Theory Comput. **15**, 3678–3693 (2019).

[22] 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).

[23] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," J. Chem. Inf. Model. **50**, 742–754 (2010).

[24] 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).

[25] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, "Assessment and validation of machine learning methods for predicting molecular atomization energies," J. Chem. Theory Comput. **9**, 3404–3419 (2013).

[26] C. R. Collins, G. J. Gordon, O. A. von Lilienfeld, and D. J. Yaron, "Constant size descriptors for accurate machine learning models of molecular properties," J. Chem. Phys. **148**, 241718 (2018).

[27] O. A. von Lilienfeld, R. Ramakrishnan, M. Rupp, and A. Knoll, "Fourier series of atomic radial distribution functions: A molecular fingerprint for machine learning models of quantum chemical properties," Int. J. Quantum Chem. **115**, 1084–1093 (2015).

[28] 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).

[29] 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).

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

[31] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, and P. Marquetand, "wACSF—Weighted atom-centered symmetry functions as descriptors in machine learning potentials," J. Chem. Phys. **148**, 241709 (2018).

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

[33] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld, "Prediction errors of molecular machine learning models lower than hybrid DFT error," J. Chem. Theory Comput. **13**, 5255–5264 (2017).

[34] S. De, A. P. Bartok, G. Csanyi, and M. Ceriotti, "Comparing molecules and solids across structural and alchemical space," Phys. Chem. Chem. Phys. **18**, 13754–13769 (2016).

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

[36] B. Huang and O. A. von Lilienfeld, "Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity," J. Chem. Phys. **145**, 161102 (2016).

[37] W. Pronobis, A. Tkatchenko, and K.-R. Müller, "Many-body descriptors for predicting molecular properties with machine learning: Analysis of pairwise and three-body interactions in molecules," J. Chem. Theory Comput. **14**, 2991 (2018).

[38] B. Huang and O. A. von Lilienfeld, "The "DNA" of chemistry: Scalable quantum machine learning with "amons"," arXiv:1707.04146 (2017).

[39] B. M. Axilrod and E. Teller, "Interaction of the van der Waals type between three atoms," J. Chem. Phys. **11**, 299 (1943).

[40] Y. Muto, "Force between nonpolar molecules," J. Phys. Math. Soc. Jpn. **17**, 629 (1943).

[41] C. E. Rasmussen and C. K. I. Williams, in *Gaussian Processes for Machine Learning* edited by T. Dietterich (MIT Press, Cambridge, 2006), www.GaussianProcess.org.

[42] S. Mathias, "A kernel-based learning method for an efficient approximation of the high-dimensional born-oppenheimer potential energy surface,"

M.Sc. thesis, Mathematisch-Naturwissenschaftliche Fakultät derRheinischen Friedrich-Wilhelms-Universität Bonn, Germany, 2015, http://wissrech.ins.uni-bonn.de/teaching/master/masterthesis_mathias_revised.pdf, accessed on July 2019.
[43]K. Hansen, F. Biegler, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, "Interaction potentials in molecules and non-local information in chemical space," J. Phys. Chem. Lett. **6**, 2326 (2015).
[44]A.-I. N. Tikhonov, *Solutions of Ill Posed Problems*, Scripta Series in Mathematics (Vh Winston, 1977).
[45]M. Rupp, R. Ramakrishnan, and O. A. von Lilienfeld, "Machine learning for quantum mechanical properties of atoms in molecules," J. Phys. Chem. Lett. **6**, 3309 (2015).
[46]R. Ramakrishnan, P. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," Sci. Data **1**, 140022 (2014).
[47]M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee, and A. Qalieh (2017). "mwaskom/seaborn," Zenodo, v0.8.1, https://doi.org/10.5281/zenodo.592845
[48]G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, "Machine learning of molecular electronic properties in chemical compound space," New J. Phys. **15**, 095003 (2013).
[49]L. Cheng, M. Welborn, A. S. Christensen, and T. F. Miller, "A universal density matrix functional from molecular orbital-based machine learning: Transferability across organic molecules," J. Chem. Phys. **150**, 131103 (2019).
[50]L. C. Blum and J.-L. Reymond, "970 million druglike small molecules for virtual screening in the chemical universe database GDB-13," J. Am. Chem. Soc. **131**, 8732 (2009).
[51]M. Welborn, L. Cheng, and T. F. Miller, "Transferability in machine learning for electronic structure via the molecular orbital basis," J. Chem. Theory Comput. **14**, 4772–4779 (2018).
[52]K. Lejaeghere, G. Bihlmayer, T. Björkman, P. Blaha, S. Blügel, 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. Dułak, O. Eriksson, J. A. Flores-Livas, K. F.Garrity, L. Genovese, P. Giannozzi, M. Giantomassi, S. Goedecker, X. Gonze, O. Grånäs, E. K. U. Gross, A. Gulans, F. Gygi, D. R. Hamann, P. J. Hasnip, N. A. W. Holzwarth, D. Iuşan, D. B. Jochym, F. Jollet, D. Jones, G. Kresse, K. Koepernik, E. Küçükbenli, Y. O. Kvashnin, I. L. M. Locht, S. Lubeck, M. Marsman, N. Marzari, U. Nitzsche, L. Nordström, 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. Thunström, 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).
[53]A. N. Bootsma and S. Wheeler, "Popular integration grids can result in large errors in dft-computed free energies," ChemRxiv:8864204.v5 (2019).
[54]A. S. Christensen, F. A. Faber, B. Huang, L. A. Bratholm, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, Qml: A python toolkit for quantum machine learning, https://github.com/qmlcode/qml, 2017.
[55]L. Ruddigkeit, R. van Deursen, L. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17," J. Chem. Inf. Model. **52**, 2864 (2012).
[56]R. Ramakrishnan, P. Dral, M. Rupp, and O. A. von Lilienfeld, "Uncharacterized: List of 3054 molecules which failed the geometry consistency check," FigShare (2014).
[57]S. Grimme, J. G. Brandenburg, C. Bannwarth, and A. Hansen, "Consistent structures and interactions by density functional theory with small atomic orbital basis sets," J. Chem. Phys. **143**, 054107 (2015).
[58]A. S. Christensen, F. A. Faber, and O. A. von Lilienfeld, "Training_data.tar.bz2," FigShare (2018).
[59]F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," J. Mach. Learn. Res. **12**, 2825–2830 (2011).
[60]C. Cortes, L. D. Jackel, S. A. Solla, V. Vapnik, and J. S. Denker, "Learning curves: Asymptotic values and rate of convergence," in *Advances in Neural Information Processing Systems* (Morgan-Kaufmann, 1994), pp. 327–334.
[61]K. R. Müller, M. Finke, N. Murata, K. Schulten, and S. Amari, "A numerical study on learning curves in stochastic multilayer feedforward networks," Neural Comput. **8**, 1085 (1996).
[62]O. A. von Lilienfeld, "Quantum machine learning in chemical compound space," Angew. Chem., Int. Ed. **57**, 4164 (2018).
[63]H. M. Deeks, R. K. Walters, S. R. Hare, M. B. O'Connor, A. J. Mulholland, and D. R. Glowacki, "Sampling protein-ligand binding pathways to recover crystallographic binding poses using interactive molecular dynamics in virtual reality," arXiv:1908.07395 (2019).
[64]S. Amabilino, L. A. Bratholm, S. J. Bennie, A. C. Vaucher, M. Reiher, and D. R. Glowacki, "Training neural nets to learn reactive potential energy surfaces using interactive quantum chemistry in virtual reality," J. Phys. Chem. A **123**, 4486–4499 (2019).
[65]M. O'Connor, H. M. Deeks, E. Dawn, O. Metatla, A. Roudaut, M. Sutton, L. M. Thomas, B. R. Glowacki, R. Sage, P. Tew, M. Wonnacott, P. Bates, A. J. Mulholland, and D. R. Glowacki, "Sampling molecular conformations and dynamics in a multiuser virtual reality framework," Sci. Adv. **4**, eaat2731 (2018).
[66]S. J. Bennie, K. E. Ranaghan, H. Deeks, H. E. Goldsmith, M. B. O'Connor, A. J. Mulholland, and D. R. Glowacki, "Teaching enzyme catalysis using interactive molecular dynamics in virtual reality," J. Chem. Educ. **96**, 2488 (2019).
[67]J. B. Ferrell, J. P. Campbell, D. R. McCarthy, K. T. McKay, M. Hensinger, R. Srinivasan, X. Zhao, A. Wurthmann, J. Li, and S. T. Schneebeli, "Chemical exploration with virtual reality in organic teaching laboratories," J. Chem. Educ. **96**, 1961 (2019).
[68]T. Oliphant, *NumPy: A guide to NumPy* (Trelgol Publishing, USA, 2006), accessed on July 2019.
[69]OpenMP Architecture Review Board, OpenMP application program interface version 3.0, 2008.