FIRST LINE OF TITLE
SECOND LINE OF TITLE

Thèse n. 1234 2023
présentée le November 18, 2023
à la Faculté des sciences de base
laboratoire SuperScience
programme doctoral en SuperScience
École polytechnique fédérale de Lausanne

pour l'obtention du grade de Docteur ès Sciences
par

Paolino Paperino

acceptée sur proposition du jury:

Prof Name Surname, président du jury
Prof Name Surname, directeur de thèse
Prof Name Surname, rapporteur
Prof Name Surname, rapporteur
Prof Name Surname, rapporteur

Lausanne, EPFL, 2023

Wings are a constraint that makes
it possible to fly.
— Robert Bringhurst

To my parents…

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Lausanne, November 18, 2023 D. K.

In high-throughput material design, large databases of materials are searched for candidates with desirable characteristics. So far, searches based on experimental data have been limited in scope, due to the vast combinatorial space of materials, the heterogeneous quality of available data, and the difficulty in separating the intrinsic properties of a material from those that are contingent on the processing or the synthesis conditions. A viable alternative is to calculate material properties using computer simulations, that make it possible to exploit advances in parallel computing to construct databases with millions of entries, and to obtain results that are internally consistent. The quantitative accuracy of these predictions, however, is dependent on the quality of the reference electronic structure calculations, increasing the computational effort and reducing the breadth of the searches. Data-driven approaches have been applied to reduce the cost of accurate computational studies, by using only a small number of reference calculations for a representative subset of materials space, and using them to train surrogate models that predict inexpensively the outcome of such calculation on new materials. The way materials structures are processed into a numerical description as input of machine learning algorithms is crucial to obtain efficient and computationally inexpensive models. Recent advancements in the design of information-efficient representations have embedded novel types of information, such as neighborhood environments or pair descriptions. Despite the rapid development in offloading calculations to more dedicated hardware, these enhancements nevertheless substantially increase the cost of the representation that remains a crucial factor in simulations. It is therefore vital to delve deeper into the design space of representations to understand the type of information they encapsulate. Insights from such analyses aid in making more informed decisions regarding the trade-off between accuracy and performance. While a substantial amount of work has been undertaken to compare representations concerning their structure-property relationship, a thorough exploration into understanding the inherent nature of the information capacity of these representations remains mostly uncharted. This thesis introduces a set of measures that facilitate quantitative analysis concerning the relationship between features and datasets, thereby assisting in such decision-making processes and providing valuable insights to the academic community. Additionally, a considerable amount of effort has been dedicated to optimize the basis set involved in all representations, typically driven by heuristic considerations on the behavior of the regression target. This thesis showcases a scheme that utilizes splines to approximate the basis expansion coefficients, paving the way for expansive optimization methods to create more effective basis functions at no additional cost during simulation time. This is pivotal in simulations targeting materials encompassing a high variety of chemical species or relying on qualitative collective variables.

The discovery of new materials is one of the core pillars of technology, as every technology relies on a material and, needless to say, would not exist without it[**?**]. The search for new materials is bound by thermodynamic laws which tell what configurations are stable and can therefore be considered as potential material. methods provide approximate stability criteria which are in good agreement with experiments[**?**] making them a viable tool for the screening of new materials[**? ? ? ?**]. Due to the vast number of possible atomic structures to be considered, the efficiency of these methods is crucial.

Data-driven methods have become an efficient extension reducing expensive quantum chemistry calculations to a bare minimum while reaching close-to-*ab initio* accuracy over a wide configuration space[**?**], leading to the exploration of previously computationally intractable problems, such as the thermal conductivity of amorphous germanium telluride[**?**]. These methods are based on transforming geometrical, physical and chemical information into a vector representation, referred as descriptor, to then use it as features in a machine learning model. The development of expressive and computational inexpensive descriptors[**? ?**] has lead to applications in a wide range of areas[**? ? ?**]. Efficient descriptors are therefore essential for state of the art high-throughput material design application.

The efficient computation of expressive descriptors is a challenging problem which has seen a wide range of proposals[**? ? ? ?**]. When used to build an interatomic potential, or to predict other atomic-scale properties, representations are used together with different supervised learning schemes, so it is difficult to disentangle the interplay of descriptor, regression method, and target property that combine to determine the accuracy and computational cost of the different methods. [**?**] A deeper understanding of these descriptors is therefore essential, especially considering that the efficiency of accurate potentials is still a limiting factor for the research that can be conducted on materials.

The first part of this thesis presents a collection of measures that serve as toolkit to guide the choice of the descriptor and model. The second part discusses the implementation of machine learning model models as interatomic potentials, covering on one hand the efficient implementation of descriptors and data-driven models, and on the other hand their deployment into molecular dynamics software.

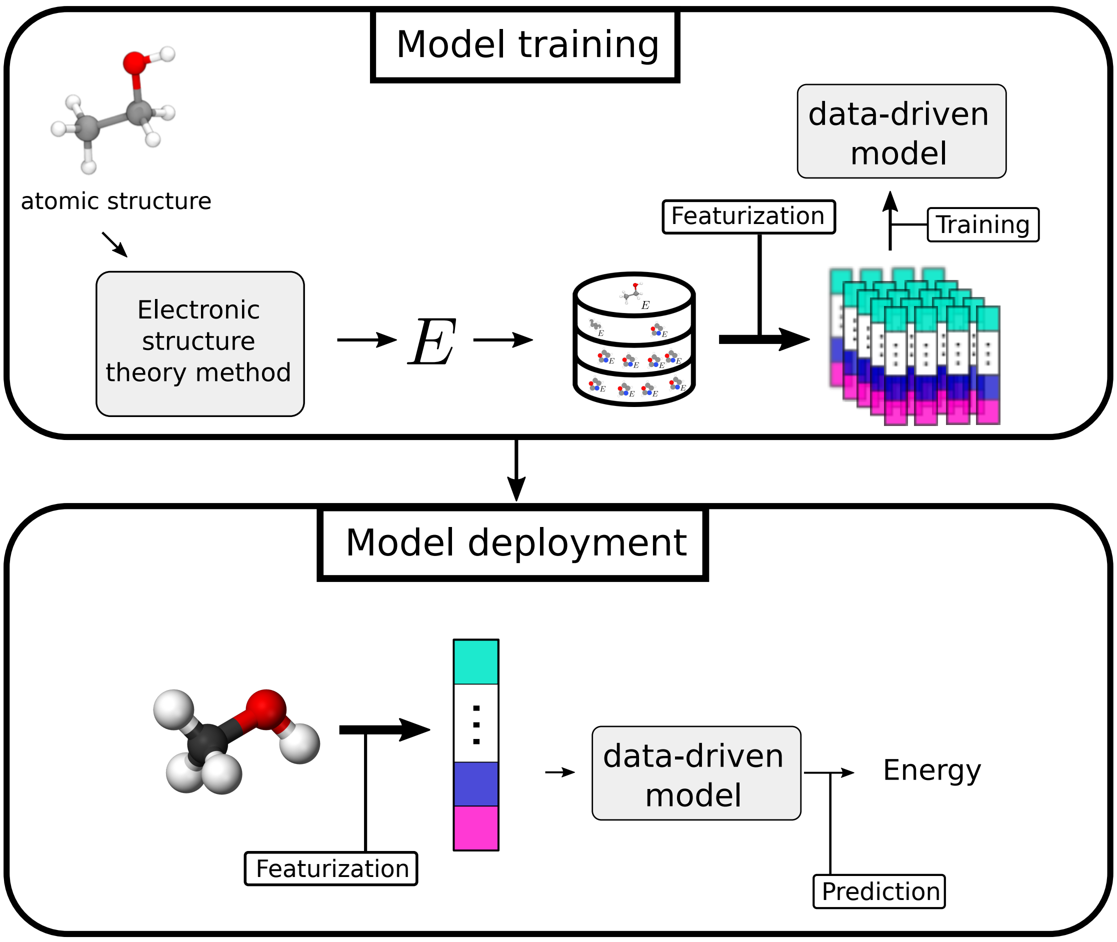

Figure 1: A schematic showing the idea of high-throughput calculations with data-driven model that serves as surrogate model to bypass the expensive electronic structure theory calculations after a training the model.

Interatomic potential have been long time used to approximate the potential energy in classical molecular dynamics simulation by decomposing the energy into many-body contributions

$$H(\mathbf{p},\mathbf{q}) = \frac{\mathbf{p}^2}{2m} + V(\mathbf{q}), \quad \frac{\partial \mathbf{p}}{\partial t} = \frac{\partial V(\mathbf{q})}{\partial \mathbf{q}}, \qquad \frac{\partial \mathbf{q}}{\partial t} = \frac{\mathbf{p}}{m} \quad \text{(classical Hamiltonian mechanics)}, \tag{1.1a}$$

$$V(\mathbf{q}) = \sum_{i=1} V_1(\mathbf{q}_i) + \sum_{i,j=1} V_2(\mathbf{q}_i,\mathbf{q}_j) + \sum_{i,j,k=1} V_3(\mathbf{q}_i,\mathbf{q}_j,\mathbf{q}_k) + \ldots \quad \text{(interatomic potential)}. \tag{1.1b}$$

Due to the limitation of computational power, the development of accurate interatomic potentials historically depended on the meticulous hand-tuning of parametric models called *empirical interatomic potential.* They were constrained by the chemical and phase space they could predict accurately [**?** **?**]. However, with increased computational power, the shift towards data-driven models emerged allowing a more automatized construction fitting models on data from empirical potential [**?**] or ab initio calculations [**?** **?** **?**]. Remarkably, current models have reached the capability to extensively cover chemical [**?**] and phase space [**?**] accurately.

The abundance of machine learning (ML) packages available today [**?**] prompts the question of the need for new ML packages for the development MLIP. Most existing packages, however, are not suited to the specific needs of MLIP, which include domain-specific featurization of 3D structures and gradient inference with respect to target properties. To ensure the code base remains manageable, strategic integration with existing software solutions is essential, wherever appropriate This requires not only a solid understanding of relevant mathematical methods but also proficiency in current software tools. This combination of requirements poses a significant challenge in the development of enduring and robust MLIP packages.

In this chapter I discuss my contributions to the software ecosystem that facilitates the deployment of MLIP into MD software. These contributions include my work to the `librascal` package [**?**], instrumental for constructing MLIP based on the SOAP featurization of atomic structures, the `scikit-matter` [**?**] package a toolkit for various data-driven preprocessing methods that facilitate MLIP construction as feature and sample selection. Additionally, I implemented an interface to `LAMMPS` [**?**] enabling the studies on ferroelectric phase transitions in barium titanate, as detailed in Ref. [**?**] and on the transport properties of lithium ortho-thiophosphate, as described in Ref. [**?**]. Finally, we briefly discuss how the packages `equisolve` [**?**] and `metatensor` [**?**] enable a more modular approach to build MLIPs.

1.1 Implementation of cubic splines for featurization

One of main contributions to `librascal` has been the implementation of a cublic spline to interpolate the radial expansion coefficients as expressed in Eq. (**??**). For each coefficient nl, the one dimensional function $f^{nl} : [0, r_c] \to \mathbb{R}$ is splined. The function f^{nl} maps the distance $r \in [0, r_c]$ to the neighbor contribution of the radial expansion coefficient as in Eq. (**??**). For the construction of the cubic spline the targeted interval $[0, r_c]$ is further partitioned into a set of subintervals $[r_1, r_2], \dots, [r_K, r_{K+1}]$ where $0 = r_1 < r_2 < \dots < r_K < r_{K+1} = r_c$. Then cubic splines are order 3 polynomials $p_k(r) = A_k + B_k r + C_k r^2 + D_k r^3$ on the interval $[0, 1]$ with the boundary conditions

$$p_k^{nl}(0) = f^{nl}(r_k) \text{ and } p_k^{nl}(1) = f^{nl}(r_{k+1}) \text{ for } k = 1, \dots, K, \tag{1.2a}$$

$$\left(\frac{\partial p_k^{nl}}{\partial r}\right)_{r=1} = \left(\frac{\partial p_{k+1}^{nl}}{\partial r}\right)_{r=0} \quad \text{for } k = 1, \dots, K-1, \tag{1.2b}$$

$$\left(\frac{\partial^2 p_k^{nl}}{\partial r^2}\right)_{r=1} = \left(\frac{\partial^2 p_{k+1}^{nl}}{\partial^2 r}\right)_{r=0} \quad \text{for } k = 1, \dots, K-1, \tag{1.2c}$$

$$\left(\frac{\partial^2 p_1^{nl}}{\partial r^2}\right)_{r=0} = 0 \text{ and } \left(\frac{\partial^2 p_K^{nl}}{\partial r^2}\right)_{r=1} = 0 \quad \text{(natural boundary conditions)}. \tag{1.2d}$$

From these $4K$ boundary conditions a tridiagonal linear system can be formed solving for the $4K$ unknowns [**?**]

$$\begin{pmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 2 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_{K-1} \\ B_K \end{pmatrix} = \begin{pmatrix} 3(f^{nl}(r_2) - f^{nl}(r_1)) \\ 3(f^{nl}(r_3) - f^{nl}(r_1)) \\ \vdots \\ 3(f^{nl}(r_K) - f^{nl}(r_{K-2})) \\ 3(f^{nl}(r_K) - f^{nl}(r_{K-1})) \end{pmatrix}. \tag{1.3a}$$

Such a systems can be solved in linear time with time complexity $O(2K)$ by iterating two times through the matrix following a Gaussian elimination scheme. The grid points $\{r_k\}_{k=0}^{K+1}$ are incrementally adjusted until the function approximation error falls beneath a user-specified error threshold. The approximation error can be estimated by sampling points in the intervals (r_k, r_{k+1}) and computing the difference between the function f^{nl} and the spline p^{nl}. During implementation, it has been found that conditioning on both the relative and absolute error yields the most robust in terms of interpolation accuracy and convergence of the grid size.

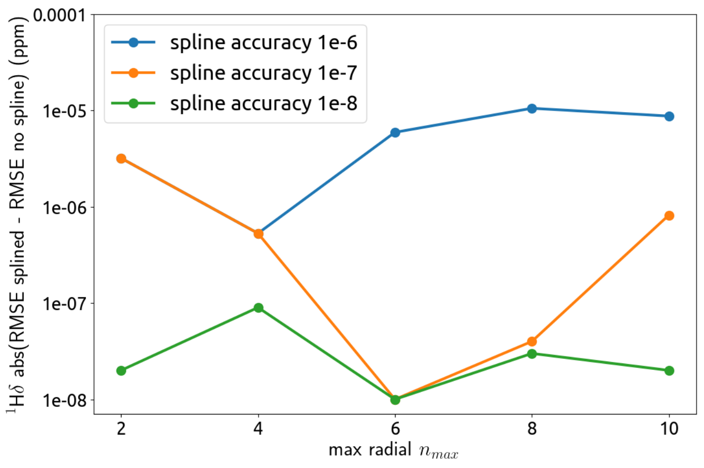

Figure 1.1: The relationship of the spline accuracy on the prediction difference for chemical shieldings of hydrogen environments as conducted in Ref. [**?**] for a linear model using SOAP descriptors with $l_{\max} = 9$. The impact of the spline accuracy on the difference in prediction of the chemical shifts is several magnitudes below the DFT accuracy, which is estimated to be on the order of 10^{-1}.

While algorithms for adaptive grids can be advantageous to control the grid size and thus reduce memory requirements, it has been found that using a uniform grid does not reach a memory intensive regime while having minimal impact on the prediction accuracy. This is evidenced by experiments on chemical shieldings of hydrogen environments [**?**] presented in Fig. **??**. Compared to an adaptive grid the uniform grid reduces the asymptotic complexity of determining the subinterval for evaluation from logarithmic $O(\log K)$ binary tree search to constant $O(1)$ lookup. Moreover, as the grid is constructed for each nl channel, the setup and evaluation of the spline opens the possibility to parallelize the spline evaluation over all channels. Viable forms of parallelization encompass multithreading, the use of Single Instruction Multiple Data (SIMD) instructions, or leveraging Graphics Processing Unit (GPU) acceleration.

1.2 Interfacing with molecular dynamics packages

Molecular dynamics (MD) packages, such as LAMMPS, GROMACS, CP2K, and i-PI [**? ? ? ?**], commonly adopt the strategy of segregating the computation of the potential energy into a distinct module such that different potentials can be used. As the computation of the potential energy only depends the atomic positions and type, and yields the energy and forces, it is a well-suited point for modularizing the code base. One significant benefit of this approach is the avoidance of reimplementing established thermostats, barostats, and time integrators. Although their implementation may appear straightforward, developing a robust code base transparently handles edge cases for the non-expert user is a time-consuming is nevertheless time-consuming task, demanding extensive documentation. Furthermore, well-established MD software packages, such as `GROMACS` [?] and `LAMMPS` [?], offer a variety of parallelization strategies that significantly improve the speed of interatomic potentials. These include MPI-based domain decomposition, CUDA- and OpenMP-based multithreading as well as SIMD abstraction modules for hardware-adaptive compute kernels. The embedding of hardware dependent parallelization strategies, such as customized CUDA and compute kernels, necessitates adapting the interatomic pontential code to these specific kernel routines. In contrast, MPI-based domain decomposition only requires dividing the potential into atomic contributions, as outlined in Eq. **??**. The forces then further evaluate to

$$-\frac{\partial E_A}{\partial \mathbf{r}_k} = -\sum_{i\in A}\frac{\partial E_i}{\partial \mathbf{r}_k} = -\sum_{i\in A}\sum_{j\in A_i}\frac{\partial E_i}{\partial \mathbf{r}_{ji}}\frac{\partial \mathbf{r}_{ji}}{\partial \mathbf{r}_k}\quad \text{, where } \frac{\partial \mathbf{r}_{ji}}{\partial \mathbf{r}_k} = \begin{cases} 1, & k=i \\ -1, & k=j \\ 0, & \text{else.} \end{cases} \tag{1.4}$$

It is evident that the forces acting on atom i depend solely local energies within its neighborhood. Consequently, these forces can be embarrassingly parallelized by dividing the cells into domains, with each one assigned to an independent hardware component. Communication between the domains becomes necessary only when the partial forces $-\partial E_i/\partial \mathbf{r}_{ji}$ for atoms that fall within the neighborhoods of multiple domains are communicated instead of recomputed, see for example the parameter *newton* in the `LAMMPS` manual [?]. Global communication is required solely for aggregating the local energy contributions into the total energy, as specified by the *nstcalcenergy* parameter in the `GROMACS` manual [?]. In both cases, the implementation of these communication can abstracted out of the potential code, thereby enabling a modular design that does not need to account for these specific details. In the case of the study on barium titanate [?], the MPI parallelization of `LAMMPS` played a crucial role in investigating the impact of long-range dielectric correlations on the Curie temperature, due to the necessity of conducting simulations on large cell sizes.

1.3 Implementation of gradients in kernel models

We do not give a pedagological introduction to kernel methods, but only discuss the specificalities that arise when extending them for gradients prediction. For a comprehensive introduction to kernel models please refer to Ref. [?].

For a kernel k fitted on the samples $\{\mathbf{c}_t \in \mathbb{R}^M\}_{t=1}^N$ and targets $\{y_t \in \mathbb{R}\}_{t=1}^N$ the optimal weights $\boldsymbol{\alpha} \in \mathbb{R}^N$ are retrieved as solution of the minimization problem

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha}'\in\mathbb{R}^N}{\operatorname{argmin}}\, \ell(\boldsymbol{\alpha}') \tag{1.5a}$$

$$\ell(\boldsymbol{\alpha}) = \sum_{n}^{N} \| y_n - \sum_{t=1}^{N} \alpha_t k(\mathbf{c}_t, \mathbf{c}_n)\|^2 \tag{1.5b}$$

can be expressed in vector form as

$$\ell(\boldsymbol{\alpha}) = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 \tag{1.6a}$$

$$\tag{1.6b}$$

that can be easily solved setting the derivative to zero

$$0 = \frac{\partial \ell(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \tag{1.7a}$$

$$0 = 2\mathbf{K}^T\mathbf{K}\boldsymbol{\alpha} - 2\mathbf{K}\mathbf{y} \tag{1.7b}$$

$$\mathbf{K}\mathbf{y} = \mathbf{K}^T\mathbf{K}\boldsymbol{\alpha} \tag{1.7c}$$

$$\mathbf{K}\mathbf{y} = \mathbf{K}\mathbf{K}\boldsymbol{\alpha}, \text{ since } \mathbf{K} = \mathbf{K}^T \tag{1.7d}$$

which gives $\mathbf{K}^{-1}\mathbf{y}$ as solution for $\boldsymbol{\alpha}$. The solution can subsequently be used to evaluate on an arbitrary point i by the relationship

$$\sum_{t=1}^{N} \alpha_t k(\mathbf{c}_t, \mathbf{c}_i) = y_i. \tag{1.8}$$

Note that in principle the solution does not resolve in a exact solution, especially considering regularization, but we omitt this detail for simplicity of equations.

1.3.1 Kernel evaluation

Now to include the gradients wrt. to the atomic position $\mathbf{r}_k$ of atom k into the picture, we first note that the training points used to construct the kernel are independent from the points for which gradients are evaluated. In other words they are therefore independent to changes in $\mathbf{r}_k$ for any structure when evaluating **??**. We can therefore use the notation $k_t(\mathbf{c}_i)$ to denote $k(\mathbf{c}_t, \mathbf{c}_i)$ for easier readability of the derivatives. The partial force can then be expressed as

$$\frac{\partial E_i}{\partial \mathbf{r}_{ji}} = \sum_{t=1}^{N} \frac{\partial \alpha_t k_t(\mathbf{c}_i)}{\partial \mathbf{r}_{ji}} \tag{1.9a}$$

$$= \sum_{t=1}^{N} \alpha_t \frac{k_t(\mathbf{c}_i)}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \mathbf{r}_{ji}} \tag{1.9b}$$

$$\tag{1.9c}$$

which gives us the final expression

$$\frac{\partial E}{\partial \mathbf{r}_k} = \sum_{t=1}^{N} \alpha_t \underbrace{\sum_{i \in A} \sum_{j \in A_i} \frac{\partial k_t(\mathbf{c}_i)}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \mathbf{r}_{ji}} \frac{\partial \mathbf{r}_{ji}}{\partial \mathbf{r}_k}}_{\text{i,k kernel force component}} \tag{1.10}$$

One can include the gradients wrt. to the training points in the original expression and get

$$\sum_{t=1}^{N} \alpha_t k(\{\mathbf{c}_t\}_{t\in A}, \{\mathbf{c}\}_{n\in A}) = E. \tag{1.11}$$

and get

$$\sum_{t=1}^{N} \alpha_t k(\{\mathbf{c}_t\}_{t\in A}, \{\mathbf{c}\}_{n\in A}) + \sum_{t\in A'} \alpha_t \sum_{j'\in A'_t} \frac{\partial k(\mathbf{c}_t, \mathbf{c}_i)}{\partial \mathbf{c}_t} \frac{\partial \mathbf{c}_t}{\partial \mathbf{r}'_{j't}} \frac{\partial \mathbf{r}'_{jt}}{\partial \mathbf{r}'_{k'}} = E. \tag{1.12}$$

but this is not efficient

It can be seen that wa part that can be factored out of the weights is contributing to the We note that including gradients into the training also changes the evaluation of the kernel entries

$$k((t,k'), \mathbf{c}_i) = \sum_{t\in A'} \sum_{j'\in A'_t} \frac{\partial k(\mathbf{c}_t, \mathbf{c}_i)}{\partial \mathbf{c}_t} \frac{\partial \mathbf{c}_t}{\partial \mathbf{r}'_{j't}} \frac{\partial \mathbf{r}'_{jt}}{\partial \mathbf{r}'_{k'}} \tag{1.13a}$$

$$\frac{\partial k((t,k'), \mathbf{c}_i)}{\partial \mathbf{r}_k} = \sum_{i\in A} \sum_{j\in A_i} \sum_{t\in A'} \sum_{j'\in A'_t} \frac{\partial^2 k(\mathbf{c}_t, \mathbf{c}_i)}{\partial \mathbf{c}_t \partial \mathbf{c}_i} \frac{\partial \mathbf{c}_t}{\partial \mathbf{r}'_{jt}} \frac{\partial \mathbf{r}'_{jt}}{\partial \mathbf{r}'_{k'}} \frac{\partial \mathbf{c}_i}{\partial \mathbf{r}_{ji}} \frac{\partial \mathbf{r}_{ji}}{\partial \mathbf{r}_k} \tag{1.13b}$$

(1.13c)

This adds an additional complexity which we hide away by using the notation $k_t(\mathbf{c}_n)$ as we will see later that people typically ignore the gradient part as training points when using a low-rank approximation.

1.3.2 Kernel fitting

We can include the forces into the optimization problem using Eqs. **??** and **??**

...

$$\ell(\boldsymbol{\alpha}) = \sum_{n}^{N} \| E^{(n)} - \sum_{t=1}^{N} \alpha'_t k_t(\mathbf{c}_n) \|^2 + \sum_{k\in A^{(n)}} \| \frac{\partial E^{(n)}}{\partial \mathbf{r}_k} - \sum_{t=1}^{N} \alpha_t \sum_{i\in A^{(n)}} \sum_{j\in A_i^{(n)}} \frac{\partial k_t(\mathbf{c}_i)}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \mathbf{r}_{ji}} \frac{\partial \mathbf{r}_{ji}}{\partial \mathbf{r}_k} \|^2. \tag{1.14}$$

We use $A^{(n)}$ to index structures as we use A_n to index the atom within a structure. Unlike in Eq. **??** the α_t has been extracted out to see more clearly that the loss function can be translated into the vectorial form

$$\ell(\boldsymbol{\alpha}) = \|\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}\|^2, \tag{1.15a}$$

$$\text{with} \quad \mathbf{K} = \begin{bmatrix} \mathbf{K}_{N,N} & \mathbf{K}_{N,\partial N} \\ \mathbf{K}_{\partial N,N} & \mathbf{K}_{\partial N,\partial N} \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} \mathbf{E} \\ \frac{\partial \mathbf{E}}{\partial \mathbf{r}_k} \end{bmatrix}, \tag{1.15b}$$

$$\mathbf{K}_{NN} \in \mathbb{R}^{N,N}, \qquad \mathbf{K}_{\partial NN} \in \mathbb{R}^{\partial N,N}, \qquad \partial N = 3\sum_n |A^{(n)}|, \tag{1.15c}$$

$$\mathbf{E} \in \mathbb{R}^N, \qquad \frac{\partial \mathbf{E}}{\partial \mathbf{r}_k} \in \mathbb{R}^{\partial N} \tag{1.15d}$$

where $\mathbf{K}_{\nabla NN}$ is the matrix from the stacked sum-product of the feature and kernel gradients in the loss **??**, explicitely written where ∂N denotes the number stacked gradients for each Cartesian dimension $3\sum_n |A^{(n)}|$ and $\mathbf{K}_{\nabla NN}$ is the matrix from the stacked sum-product of the feature and kernel gradients in the loss **??**, explicitely written

$$[\mathbf{K}_{\nabla NN}]_{(n,k,p),t} = \sum_{i\in A^{(n)}} \sum_{j\in A_i^{(n)}} \frac{\partial k_t(\mathbf{c}_i)}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial r_{ji}^{(p)}} \frac{\partial r_{ji}^{(p)}}{\partial r_k^{(p)}}, \tag{1.16a}$$

where (n, k, p) is the flattened multi-index specifiyng the structure, the atom and the Cartesian dimension. The inclusion of the gradients explodes the size of the memory requirements for the kernel matrix, because the ∂N grows with total number of atoms instead of structures like the energies. A fruitful approach has been therefore to perform a low-rank approximation of the kernel matrix [**? ? ? ?**] by projecting the $3\sum_n N_n$ points onto a fixed number of representative points $\{\tilde{c}_t\}_{t=1}^T$, we refer as *pseudo points* as used in referencei [**?**]. In the Bayesian community the low-rank approximation has established the naming *subset of regressor* [**?**] while the frequentist community it is more commonly referred as Nyström approximation [**?**].

$$\tilde{\mathbf{K}} = \mathbf{K}_{TT} + \mathbf{K}_{T,N+\partial N} \Lambda^{-2} \mathbf{K}_{T,N+\partial N}^T, \tag{1.17a}$$

An established way has been to only use single envirnoments as pseudo points and to dismiss gradient contributions in them. One advantage is that this skips an expensive evaluation of kernel entries using gradients as training points. The additional advantage is that environmental information is often redundantly present within structures, thus a much smaller "represantative" set can be found reducing the evaluation time of one kernel entry from N_n^2 to N_n.

1.3.3 Moduler kernel computation

One disadvantage in the design of `librascal` in the entanglement of the featurization and the model building to the libraray. This was required as the construction of the kernel matrix requires understanding by the model building tool of the decomposition of the target property into local contributions Eq. (**??**) as well as what its gradients are, domain-agnostic packages as `scikit-learn` are not suitable for a direct application in this case.

The software package `metatensor` allows to attribute these structural characteristics to the object itself as metadata and offers `numpy`-like data manipulations that take advantage of the metadata. This allowed a disentanglement of the featurization that has reimplemented in a new package `rascaline` and model construction that has been moved to `equisolve`. Part of my contribution was to participate in the development of `metatensor` as well as heavily developing on the initial design of `equisolve` contributing module of shallow methods including standardizer, linear and the above-presented kernel model as well as the inital designs for neural network models based on the data formate created in `metatensor`.

While the term $\partial \mathbf{c}_i/\partial \mathbf{r}_k$ depends solely on the choice of featurization, the term $\partial k_t(\mathbf{c}_i)/\partial \mathbf{c}_i$ depends on the choice of kernel, thus the computation of the kernels and featurizations can be separated. A problem that arised in `librascal` is that the species introduce a high structural sparsity in the features that needs to be also considered in the kernel computation to be efficient. For that the kernel needs to support such a sparsity. While there exist multiple sparsity formats[], they are agnostic to the structure of the data and thus do not allow us to exploit structural sparsities that depend on specif information as the species. To solve was the implementation of metatensor.

1.4 A metadynamic software framework with LAMMPS, PLUMED and i-PI

To study phase transitions of barium titanite in Ref. [**?**] we needed to accelerate the sampling of the transition. One common technique that we use is metadynamics that adds a bias potential to the simulation that is later normalized out in the calculation of the free energy. While the forces of the MLIP were computed with `LAMMPS` and the forces of the bias potential were computed with `PLUMED`[**?**]. To consolidate both forces for the metadynamics the software-package `i-PI` was used, that implemented a custom protocol to each of these MD packages to allow communicatiof the forces to a python interface. A schematic of the interwork between the software packages can be seen in Fig. **??**.

As CV for the metadynamics $l = 1$ components of the spherical expansion coefficients computed with `librascal` were used. My implementation of the cubic spline helped to speed up the computation of the bias term. Since only the expansion coefficients centered for the oxygen neighbors centered on the titanite was sufficient for the CV, I further implemented an option to selectively computate the partial gradients for certain species. Note that a selectional computation of partial gradients also needs to consider dependencies of the gradient on the central energies of its neighbors as pointed out in Eq. (**??**).

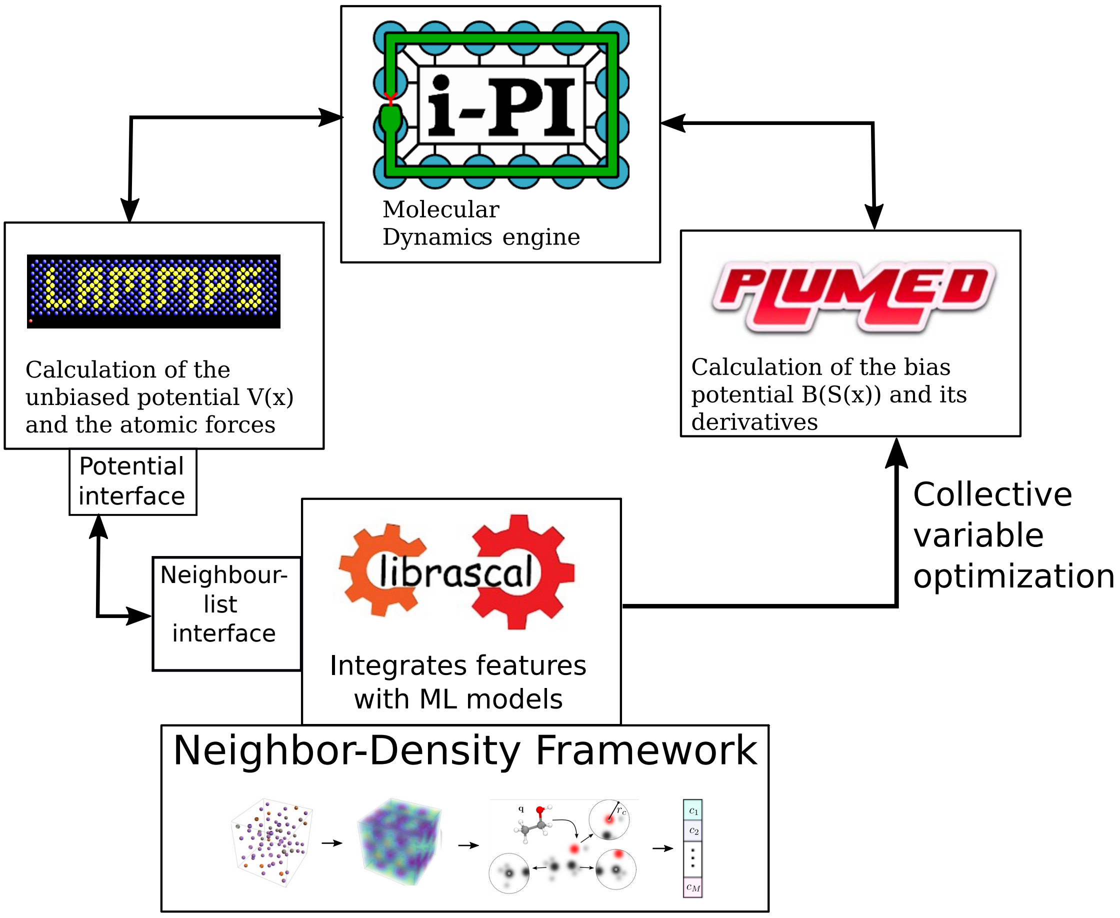
i-PI
Molecular Dynamics engine
LAMMPS
Calculation of the unbiased potential V(x) and the atomic forces
Potential interface
PLUMED
Calculation of the bias potential B(S(x)) and its derivatives
Collective variable optimization
Neighbour-list interface
librascal
Integrates features with ML models
Neighbor-Density Framework

Figure 1.2: A schematic showing the interwork of the software pieces to run metadynamic simulations to study interfacial effects of barium titanite.

1.5 Serialization of MLIP

In last decade, the rapid development of machine learning models has resulted in numerous interfaces for `LAMMPS`[?]. While the model accuracy is a good indicator for the usability of a model, it is still not reliable enough to be a guarantee that the model will work for MD software as undersampling of the phase space can always be an issue for any kind of model. One is therefore forced to retrain the model for a dataset covering a larger part of the phase pace or switch to a more accurate model during the analysis of an experiment. As all ML models show different trade-offs between accuracy, evaluation time, trainings cost and hyperparameter optimization the set of suitable models depend on the system of interest, the dataset size and given computational resources. As the dataset size changes over the course of analysis different models become more suitable candidates and thus a different model is more suitable for the analysis. The current software infrastructure of ML models for MD software causes a lot of friction when changing the model type as the MD software needs to be recompiled for a different interface.Even more crucial is the fact that the model development is often conducted in higher-level languages like `Python` or `Julia` due to their flexibility. This however restricts the usage of the model also to MD packages in the same higher-level langue or requires the implementation of a serialization for the model plus interface for an MD package. This work restricts the usage of a lot of developed ML models in low-level MD packages which are often required to conduct insightful research. For nearly five years following its initial publication, the widely-used SchNet model[?] lacked an interface with a low-level MD package. Similar problems exists in industry where models trained with different ML packages need to be shipped to devices with different hardware architectures and different software stacks making it hard to reliably provide the same version of the package on each device. The industry therefore developed an open standard for machine learning models open neural network exchange (ONNX). This standard can however not be used for MLIPs as they lack the support of the inference for gradients.

The Open Knowledgebase of Interatomic Models (OpenKIM)[?] tries to address the problem. They developed abstract representations of the data and processing directives necessary to perform a molecular simulation thereby unifying the interfaces of several MD packages to one interface, namely the KIM API[?]. While it reduces the cost of the number interfaces that are needed, they are far from covering comprehensively all relevant MD packages, as packages like GROMACS[?] and C2PK[?] are missing. Most of the supported MD packages are implemented in higher-level languages for which a custom MLIP interface can be easily implemented.

A solution we target with the software ecosystem developed in our lab is to use TorchScript as it supports the use of gradients and offers a usage of the model in C++ surpassing the high-level language barrier. It further supports advances model optimization utilities that can be used optimize complex models by kernel fusioning of the operation graph. Considering all that it seems like a promising candidate to standardize the landscape of MLIPs.

[1] Renzo Tomellini, Johan Veiga Benesch, and Aud Alming. Commentary: Fostering innovation in materials sciences and engineering. *APL Materials*, 1(1):011001, 2013.

[2] Martin Jansen. Conceptual inorganic materials discovery–a road map. *Advanced Materials*, 27(21):3229–3242, 2015.

[3] Gerbrand Ceder, Y-M Chiang, DR Sadoway, MK Aydinol, Y-I Jang, and Biying Huang. Identification of cathode materials for lithium batteries guided by first-principles calculations. *Nature*, 392(6677):694–696, 1998.

[4] Martin P Andersson, Thomas Bligaard, Arkady Kustov, Kasper E Larsen, Jeffrey Greeley, Tue Johannessen, Claus H Christensen, and Jens K Nørskov. Toward computational screening in heterogeneous catalysis: Pareto-optimal methanation catalysts. *Journal of Catalysis*, 239(2):501–506, 2006.

[5] Kesong Yang, Wahyu Setyawan, Shidong Wang, Marco Buongiorno Nardelli, and Stefano Curtarolo. A search model for topological insulators with high-throughput robustness descriptors. *Nature materials*, 11(7):614–619, 2012.

[6] Rafael Gómez-Bombarelli, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, David Duvenaud, Dougal Maclaurin, Martin A Blood-Forsythe, Hyun Sik Chae, Markus Einzinger, Dong-Gwang Ha, Tony Wu, et al. Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. *Nature materials*, 15(10):1120–1127, 2016.

[7] Albert P. Bartók, James Kermode, Noam Bernstein, and Gábor Csányi. Machine Learning a General-Purpose Interatomic Potential for Silicon. *Phys. Rev. X*, 8(4):041048, December 2018. https://link.aps.org/doi/10.1103/PhysRevX.8.041048.

[8] Gabriele C Sosso, Davide Donadio, Sebastiano Caravati, Jörg Behler, and Marco Bernasconi. Thermal transport in phase-change materials from atomistic simulations. *Physical Review B*, 86(10):104301, 2012.

[9] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *The Journal of chemical physics*, 134(7):074106, 2011.

[10] Albert P Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.

[11] Aria Mansouri Tehrani, Anton O Oliynyk, Marcus Parry, Zeshan Rizvi, Samantha Couper, Feng Lin, Lowell Miyagi, Taylor D Sparks, and Jakoah Brgoch. Machine learning directed search for ultraincompressible, superhard materials. *Journal of the American Chemical Society*, 140(31):9844–9853, 2018.

[12] Gabriele C Sosso, Volker L Deringer, Stephen R Elliott, and Gábor Csányi. Understanding the thermal properties of amorphous solids using machine-learning-based interatomic potentials. *Molecular Simulation*, 44(11):866–880, 2018.

[13] Yasemin Basdogan, Mitchell C Groenenboom, Ethan Henderson, Sandip De, Susan B Rempe, and John A Keith. Machine learning guided approach for studying solvation environments. *Journal of Chemical Theory and Computation*, 2019.

[14] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[15] Haoyan Huo and Matthias Rupp. Unified representation for machine learning of molecules and crystals. *arXiv preprint arXiv:1704.06439*, 13754, 2017.

[16] Yunxing Zuo, Chi Chen, Xiangguo Li, Zhi Deng, Yiming Chen, Jörg Behler, Gábor Csányi, Alexander V. Shapeev, Aidan P. Thompson, Mitchell A. Wood, and Shyue Ping Ong. Performance and Cost Assessment of Machine Learning Interatomic Potentials. *J. Phys. Chem. A*, page acs.jpca.9b08723, January 2020.

[17] Frank H Stillinger and Thomas A Weber. Computer simulation of local order in condensed phases of silicon. *Physical review B*, 31(8):5262, 1985.

[18] Jerry Tersoff. Empirical interatomic potential for silicon with improved elastic properties. *Physical Review B*, 38(14):9902, 1988.

[19] Thomas B Blank, Steven D Brown, August W Calhoun, and Douglas J Doren. Neural network models of potential energy surfaces. *The Journal of chemical physics*, 103(10):4129–4137, 1995.

[20] Alex Brown, Bastiaan J Braams, Kurt Christoffel, Zhong Jin, and Joel M Bowman. Classical and quasiclassical spectral analysis of ch 5+ using an ab initio potential energy surface. *The Journal of chemical physics*, 119(17):8790–8793, 2003.

[21] Sönke Lorenz, Matthias Scheffler, and Axel Gross. Descriptions of surface chemical reactions using a neural network representation of the potential-energy surface. *Physical Review B*, 73(11):115431, 2006.

[22] Jörg Behler and Michele Parrinello. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.*, 98(14):146401, April 2007.

[23] Nataliya Lopanitsyna, Guillaume Fraux, Maximilian A Springer, Sandip De, and Michele Ceriotti. Modeling high-entropy transition metal alloys with alchemical compression. *Physical Review Materials*, 7(4):045802, 2023.

[24] Ibrahim Haddad. Artificial intelligence and data in open source. https://www.linuxfoundation.org/research/artificial-intelligence-and-data-in-open-source, mar 2022.

[25] Félix Musil, Max Veit, Till Junge, Markus Stricker, Alexander Goscinski, Guillaume Fraux, Rose Cersonsky, Michael J Willatt, Andrea Grisafi, and Michele Ceriotti. librascal – A scalable and versatile library to generate representations for atomic-scale learning. https://github.com/cosmo-epfl/librascal.

[26] Alexander Goscinski, Victor Paul Principe, Guillaume Fraux, Sergei Kliavinek, Benjamin Aaron Helfrecht, Philip Loche, Michele Ceriotti, and Rose Kathleen Cersonsky. scikit-matter: A suite of generalisable machine learning methods born out of chemistry and materials science. *Open Research Europe*, 3:81, 2023.

[27] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.*, 271:108171, 2022.

[28] Modeling the ferroelectric phase transition in barium titanate with dft accuracy and converged sampling.

[29] Lorenzo Gigli, Davide Tisi, Federico Grasselli, and Michele Ceriotti. Mechanism of charge transport in lithium thiophosphate. *arXiv preprint arXiv:2310.15679*, 2023.

[30] Philip Loche, Davide Tisi, Alexander Goscinski, Joe Abbott, and Guillaume Fraux. equisolve – A ML toolkit package to build models for the prediction of equivariant properties and gradients.

[31] Guillaume Fraux, Philip Loche, Davide Tisi, Filippo Bigi, Joe Abbott, Guillaume Fraux, Alexander Goscinski, and Michele Ceriotti. metatensor – Self-describing sparse tensor data format for atomistic machine learning and beyond.

[32] Richard H Bartels, John C Beatty, and Brian A Barsky. Hermite and cubic spline interpolation. *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*, pages 9–17, 1998.

[33] Federico M Paruzzo, Albert Hofstetter, Félix Musil, Sandip De, Michele Ceriotti, and Lyndon Emsley. Chemical shifts in molecular solids by machine learning. *Nature communications*, 9(1):1–10, 2018.

[34] B Hess, C Kutzner, D van der Spoel, and E Lindahl. {GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation}. *J Chem Theory Comput*, 4(3):435–447, 2008.

[35] Thomas D Kühne, Marcella Iannuzzi, Mauro Del Ben, Vladimir V Rybkin, Patrick Seewald, Frederick Stein, Teodoro Laino, Rustam Z Khaliullin, Ole Schütt, Florian Schiffmann, et al. Cp2k: An electronic structure and molecular dynamics software package-quickstep: Efficient and accurate electronic structure calculations. *The Journal of Chemical Physics*, 152(19), 2020.

[36] Venkat Kapil, Mariana Rossi, Ondrej Marsalek, Riccardo Petraglia, Yair Litman, Thomas Spura, Bingqing Cheng, Alice Cuzzocrea, Robert H Meißner, David M Wilkins, et al. i-pi 2.0: A universal force engine for advanced molecular simulations. *Computer Physics Communications*, 236:214–223, 2019.

[37] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C Smith, Berk Hess, and Erik Lindahl. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1:19–25, 2015.

[38] LAMMPS developers. LAMMPS Users Manual - newton command, 2023.

[39] GROMMACS developers. GROMACS Users Manual - nstcalcenergy, 2023.

[40] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[41] Bernhard W Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(1):1–21, 1985.

[42] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

[43] Alex Smola and Peter Bartlett. Sparse greedy gaussian process regression. *Advances in neural information processing systems*, 13, 2000.

[44] Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[45] Massimiliano Bonomi, Davide Branduardi, Giovanni Bussi, Carlo Camilloni, Davide Provasi, Paolo Raiteri, Davide Donadio, Fabrizio Marinelli, Fabio Pietrucci, Ricardo A Broglia, and Michele Parrinello. PLUMED: A portable plugin for free-energy calculations with molecular dynamics. *Comput. Phys. Commun.*, 180(10):1961–1972, October 2009.

[46] James R. Kermode, Albert P. Bartók, and Gábor Csányi. QUIP.

[47] 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(24):241722, June 2018.

[48] Daniel S Karls, Matthew Bierbaum, Alexander A Alemi, Ryan S Elliott, James P Sethna, and Ellad B Tadmor. The openkim processing pipeline: A cloud-based automatic material property computation engine. *The Journal of Chemical Physics*, 153(6), 2020.

[49] Ryan S. Elliott and Ellad B. Tadmor. Knowledgebase of interatomic models (kim) application programming interface (api), 2011.